

GiNaC

1.8.6

Generated by Doxygen 1.9.6



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>7</b>
3.1 Class List	7
<b>4 File Index</b>	<b>15</b>
4.1 File List	15
<b>5 Namespace Documentation</b>	<b>19</b>
5.1 GiNaC Namespace Reference	19
5.1.1 Typedef Documentation	58
5.1.1.1 archive_node_id	58
5.1.1.2 archive_atom	58
5.1.1.3 synthesize_func	58
5.1.1.4 unarchive_map_t	58
5.1.1.5 exvector	59
5.1.1.6 exset	59
5.1.1.7 exmap	59
5.1.1.8 evalffunctype	59
5.1.1.9 FUNCP_1P	59
5.1.1.10 FUNCP_2P	59
5.1.1.11 FUNCP_CUBA	59
5.1.1.12 epvector	60
5.1.1.13 epp	60
5.1.1.14 exprseq	60
5.1.1.15 paramset	60
5.1.1.16 eval_funcp	60
5.1.1.17 evalf_funcp	60
5.1.1.18 conjugate_funcp	60
5.1.1.19 real_part_funcp	61
5.1.1.20 imag_part_funcp	61
5.1.1.21 expand_funcp	61
5.1.1.22 derivative_funcp	61
5.1.1.23 expl_derivative_funcp	61
5.1.1.24 power_funcp	61
5.1.1.25 series_funcp	61
5.1.1.26 print_funcp	61
5.1.1.27 info_funcp	62
5.1.1.28 eval_funcp_1	62
5.1.1.29 evalf_funcp_1	62

5.1.1.30 conjugate_funcp_1 . . . . .	62
5.1.1.31 real_part_funcp_1 . . . . .	62
5.1.1.32 imag_part_funcp_1 . . . . .	62
5.1.1.33 expand_funcp_1 . . . . .	62
5.1.1.34 derivative_funcp_1 . . . . .	62
5.1.1.35 expl_derivative_funcp_1 . . . . .	63
5.1.1.36 power_funcp_1 . . . . .	63
5.1.1.37 series_funcp_1 . . . . .	63
5.1.1.38 print_funcp_1 . . . . .	63
5.1.1.39 info_funcp_1 . . . . .	63
5.1.1.40 eval_funcp_2 . . . . .	63
5.1.1.41 evalf_funcp_2 . . . . .	63
5.1.1.42 conjugate_funcp_2 . . . . .	63
5.1.1.43 real_part_funcp_2 . . . . .	64
5.1.1.44 imag_part_funcp_2 . . . . .	64
5.1.1.45 expand_funcp_2 . . . . .	64
5.1.1.46 derivative_funcp_2 . . . . .	64
5.1.1.47 expl_derivative_funcp_2 . . . . .	64
5.1.1.48 power_funcp_2 . . . . .	64
5.1.1.49 series_funcp_2 . . . . .	64
5.1.1.50 print_funcp_2 . . . . .	64
5.1.1.51 info_funcp_2 . . . . .	65
5.1.1.52 eval_funcp_3 . . . . .	65
5.1.1.53 evalf_funcp_3 . . . . .	65
5.1.1.54 conjugate_funcp_3 . . . . .	65
5.1.1.55 real_part_funcp_3 . . . . .	65
5.1.1.56 imag_part_funcp_3 . . . . .	65
5.1.1.57 expand_funcp_3 . . . . .	65
5.1.1.58 derivative_funcp_3 . . . . .	65
5.1.1.59 expl_derivative_funcp_3 . . . . .	66
5.1.1.60 power_funcp_3 . . . . .	66
5.1.1.61 series_funcp_3 . . . . .	66
5.1.1.62 print_funcp_3 . . . . .	66
5.1.1.63 info_funcp_3 . . . . .	66
5.1.1.64 eval_funcp_4 . . . . .	66
5.1.1.65 evalf_funcp_4 . . . . .	66
5.1.1.66 conjugate_funcp_4 . . . . .	67
5.1.1.67 real_part_funcp_4 . . . . .	67
5.1.1.68 imag_part_funcp_4 . . . . .	67
5.1.1.69 expand_funcp_4 . . . . .	67
5.1.1.70 derivative_funcp_4 . . . . .	67
5.1.1.71 expl_derivative_funcp_4 . . . . .	67

---

5.1.1.72 power_funcp_4 . . . . .	67
5.1.1.73 series_funcp_4 . . . . .	68
5.1.1.74 print_funcp_4 . . . . .	68
5.1.1.75 info_funcp_4 . . . . .	68
5.1.1.76 eval_funcp_5 . . . . .	68
5.1.1.77 evalf_funcp_5 . . . . .	68
5.1.1.78 conjugate_funcp_5 . . . . .	68
5.1.1.79 real_part_funcp_5 . . . . .	68
5.1.1.80 imag_part_funcp_5 . . . . .	69
5.1.1.81 expand_funcp_5 . . . . .	69
5.1.1.82 derivative_funcp_5 . . . . .	69
5.1.1.83 expl_derivative_funcp_5 . . . . .	69
5.1.1.84 power_funcp_5 . . . . .	69
5.1.1.85 series_funcp_5 . . . . .	69
5.1.1.86 print_funcp_5 . . . . .	69
5.1.1.87 info_funcp_5 . . . . .	70
5.1.1.88 eval_funcp_6 . . . . .	70
5.1.1.89 evalf_funcp_6 . . . . .	70
5.1.1.90 conjugate_funcp_6 . . . . .	70
5.1.1.91 real_part_funcp_6 . . . . .	70
5.1.1.92 imag_part_funcp_6 . . . . .	70
5.1.1.93 expand_funcp_6 . . . . .	70
5.1.1.94 derivative_funcp_6 . . . . .	71
5.1.1.95 expl_derivative_funcp_6 . . . . .	71
5.1.1.96 power_funcp_6 . . . . .	71
5.1.1.97 series_funcp_6 . . . . .	71
5.1.1.98 print_funcp_6 . . . . .	71
5.1.1.99 info_funcp_6 . . . . .	71
5.1.1.100 eval_funcp_7 . . . . .	71
5.1.1.101 evalf_funcp_7 . . . . .	72
5.1.1.102 conjugate_funcp_7 . . . . .	72
5.1.1.103 real_part_funcp_7 . . . . .	72
5.1.1.104 imag_part_funcp_7 . . . . .	72
5.1.1.105 expand_funcp_7 . . . . .	72
5.1.1.106 derivative_funcp_7 . . . . .	72
5.1.1.107 expl_derivative_funcp_7 . . . . .	72
5.1.1.108 power_funcp_7 . . . . .	73
5.1.1.109 series_funcp_7 . . . . .	73
5.1.1.110 print_funcp_7 . . . . .	73
5.1.1.111 info_funcp_7 . . . . .	73
5.1.1.112 eval_funcp_8 . . . . .	73
5.1.1.113 evalf_funcp_8 . . . . .	73

5.1.1.114 conjugate_funcp_8 . . . . .	73
5.1.1.115 real_part_funcp_8 . . . . .	74
5.1.1.116 imag_part_funcp_8 . . . . .	74
5.1.1.117 expand_funcp_8 . . . . .	74
5.1.1.118 derivative_funcp_8 . . . . .	74
5.1.1.119 expl_derivative_funcp_8 . . . . .	74
5.1.1.120 power_funcp_8 . . . . .	74
5.1.1.121 series_funcp_8 . . . . .	74
5.1.1.122 print_funcp_8 . . . . .	75
5.1.1.123 info_funcp_8 . . . . .	75
5.1.1.124 eval_funcp_9 . . . . .	75
5.1.1.125 evalf_funcp_9 . . . . .	75
5.1.1.126 conjugate_funcp_9 . . . . .	75
5.1.1.127 real_part_funcp_9 . . . . .	75
5.1.1.128 imag_part_funcp_9 . . . . .	75
5.1.1.129 expand_funcp_9 . . . . .	76
5.1.1.130 derivative_funcp_9 . . . . .	76
5.1.1.131 expl_derivative_funcp_9 . . . . .	76
5.1.1.132 power_funcp_9 . . . . .	76
5.1.1.133 series_funcp_9 . . . . .	76
5.1.1.134 print_funcp_9 . . . . .	76
5.1.1.135 info_funcp_9 . . . . .	76
5.1.1.136 eval_funcp_10 . . . . .	77
5.1.1.137 evalf_funcp_10 . . . . .	77
5.1.1.138 conjugate_funcp_10 . . . . .	77
5.1.1.139 real_part_funcp_10 . . . . .	77
5.1.1.140 imag_part_funcp_10 . . . . .	77
5.1.1.141 expand_funcp_10 . . . . .	77
5.1.1.142 derivative_funcp_10 . . . . .	77
5.1.1.143 expl_derivative_funcp_10 . . . . .	78
5.1.1.144 power_funcp_10 . . . . .	78
5.1.1.145 series_funcp_10 . . . . .	78
5.1.1.146 print_funcp_10 . . . . .	78
5.1.1.147 info_funcp_10 . . . . .	78
5.1.1.148 eval_funcp_11 . . . . .	78
5.1.1.149 evalf_funcp_11 . . . . .	78
5.1.1.150 conjugate_funcp_11 . . . . .	79
5.1.1.151 real_part_funcp_11 . . . . .	79
5.1.1.152 imag_part_funcp_11 . . . . .	79
5.1.1.153 expand_funcp_11 . . . . .	79
5.1.1.154 derivative_funcp_11 . . . . .	79
5.1.1.155 expl_derivative_funcp_11 . . . . .	79

5.1.1.156 power_funcp_11 . . . . .	79
5.1.1.157 series_funcp_11 . . . . .	80
5.1.1.158 print_funcp_11 . . . . .	80
5.1.1.159 info_funcp_11 . . . . .	80
5.1.1.160 eval_funcp_12 . . . . .	80
5.1.1.161 evalf_funcp_12 . . . . .	80
5.1.1.162 conjugate_funcp_12 . . . . .	80
5.1.1.163 real_part_funcp_12 . . . . .	80
5.1.1.164 imag_part_funcp_12 . . . . .	81
5.1.1.165 expand_funcp_12 . . . . .	81
5.1.1.166 derivative_funcp_12 . . . . .	81
5.1.1.167 expl_derivative_funcp_12 . . . . .	81
5.1.1.168 power_funcp_12 . . . . .	81
5.1.1.169 series_funcp_12 . . . . .	81
5.1.1.170 print_funcp_12 . . . . .	82
5.1.1.171 info_funcp_12 . . . . .	82
5.1.1.172 eval_funcp_13 . . . . .	82
5.1.1.173 evalf_funcp_13 . . . . .	82
5.1.1.174 conjugate_funcp_13 . . . . .	82
5.1.1.175 real_part_funcp_13 . . . . .	82
5.1.1.176 imag_part_funcp_13 . . . . .	83
5.1.1.177 expand_funcp_13 . . . . .	83
5.1.1.178 derivative_funcp_13 . . . . .	83
5.1.1.179 expl_derivative_funcp_13 . . . . .	83
5.1.1.180 power_funcp_13 . . . . .	83
5.1.1.181 series_funcp_13 . . . . .	83
5.1.1.182 print_funcp_13 . . . . .	84
5.1.1.183 info_funcp_13 . . . . .	84
5.1.1.184 eval_funcp_14 . . . . .	84
5.1.1.185 evalf_funcp_14 . . . . .	84
5.1.1.186 conjugate_funcp_14 . . . . .	84
5.1.1.187 real_part_funcp_14 . . . . .	84
5.1.1.188 imag_part_funcp_14 . . . . .	85
5.1.1.189 expand_funcp_14 . . . . .	85
5.1.1.190 derivative_funcp_14 . . . . .	85
5.1.1.191 expl_derivative_funcp_14 . . . . .	85
5.1.1.192 power_funcp_14 . . . . .	85
5.1.1.193 series_funcp_14 . . . . .	85
5.1.1.194 print_funcp_14 . . . . .	86
5.1.1.195 info_funcp_14 . . . . .	86
5.1.1.196 eval_funcp_exvector . . . . .	86
5.1.1.197 evalf_funcp_exvector . . . . .	86

5.1.1.198 conjugate_funcp_exvector . . . . .	86
5.1.1.199 real_part_funcp_exvector . . . . .	86
5.1.1.200 imag_part_funcp_exvector . . . . .	86
5.1.1.201 expand_funcp_exvector . . . . .	87
5.1.1.202 derivative_funcp_exvector . . . . .	87
5.1.1.203 expl_derivative_funcp_exvector . . . . .	87
5.1.1.204 power_funcp_exvector . . . . .	87
5.1.1.205 series_funcp_exvector . . . . .	87
5.1.1.206 print_funcp_exvector . . . . .	87
5.1.1.207 info_funcp_exvector . . . . .	87
5.1.1.208 exhashmap . . . . .	88
5.1.1.209 spmap . . . . .	88
5.1.1.210 lookup_map . . . . .	88
5.1.1.211 lst . . . . .	88
5.1.1.212 uintvector . . . . .	88
5.1.1.213 unsignedvector . . . . .	88
5.1.1.214 exvectorvector . . . . .	88
5.1.1.215 sym_desc_vec . . . . .	89
5.1.1.216 digits_changed_callback . . . . .	89
5.1.1.217 print_context_class_info . . . . .	89
5.1.1.218 registered_class_info . . . . .	89
5.1.2 Enumeration Type Documentation . . . . .	89
5.1.2.1 anonymous enum . . . . .	89
5.1.3 Function Documentation . . . . .	89
5.1.3.1 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [1/33] . . . . .	89
5.1.3.2 GINAC_BIND_UNARCHIVER() [1/49] . . . . .	90
5.1.3.3 GINAC_DECLARE_UNARCHIVER() [1/51] . . . . .	90
5.1.3.4 write_unsigned() . . . . .	90
5.1.3.5 read_unsigned() . . . . .	90
5.1.3.6 operator<<() [1/16] . . . . .	90
5.1.3.7 operator<<() [2/16] . . . . .	91
5.1.3.8 operator>>() [1/3] . . . . .	91
5.1.3.9 operator>>() [2/3] . . . . .	91
5.1.3.10 find_factory_fcn() . . . . .	91
5.1.3.11 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [2/33] . . . . .	91
5.1.3.12 is_a() [1/3] . . . . .	92
5.1.3.13 is_exactly_a() [1/2] . . . . .	92
5.1.3.14 dynallocate() [1/2] . . . . .	92
5.1.3.15 dynallocate() [2/2] . . . . .	92
5.1.3.16 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [3/33] . . . . .	93
5.1.3.17 print_func< print_dflt >() [1/3] . . . . .	93
5.1.3.18 GINAC_BIND_UNARCHIVER() [2/49] . . . . .	93



5.1.3.19 GINAC_BIND_UNARCHIVER() [3/49]	93
5.1.3.20 GINAC_BIND_UNARCHIVER() [4/49]	93
5.1.3.21 GINAC_BIND_UNARCHIVER() [5/49]	93
5.1.3.22 GINAC_BIND_UNARCHIVER() [6/49]	94
5.1.3.23 GINAC_BIND_UNARCHIVER() [7/49]	94
5.1.3.24 GINAC_BIND_UNARCHIVER() [8/49]	94
5.1.3.25 is_dirac_slash()	94
5.1.3.26 base_and_index()	94
5.1.3.27 dirac_ONE()	94
5.1.3.28 get_dim_uint()	95
5.1.3.29 clifford_unit()	95
5.1.3.30 dirac_gamma()	95
5.1.3.31 dirac_gamma5()	96
5.1.3.32 dirac_gammaL()	96
5.1.3.33 dirac_gammaR()	97
5.1.3.34 dirac_slash()	97
5.1.3.35 get_representation_label() [1/2]	97
5.1.3.36 trace_string()	97
5.1.3.37 dirac_trace() [1/3]	98
5.1.3.38 dirac_trace() [2/3]	98
5.1.3.39 dirac_trace() [3/3]	98
5.1.3.40 canonicalize_clifford()	99
5.1.3.41 clifford_star_bar()	99
5.1.3.42 clifford_prime()	99
5.1.3.43 remove_dirac_ONE()	99
5.1.3.44 clifford_max_label()	100
5.1.3.45 clifford_norm()	100
5.1.3.46 clifford_inverse()	100
5.1.3.47 lst_to_clifford() [1/2]	100
5.1.3.48 lst_to_clifford() [2/2]	101
5.1.3.49 get_clifford_comp()	101
5.1.3.50 clifford_to_lst()	101
5.1.3.51 clifford_moebius_map() [1/2]	102
5.1.3.52 clifford_moebius_map() [2/2]	102
5.1.3.53 GINAC_DECLARE_UNARCHIVER() [2/51]	103
5.1.3.54 GINAC_DECLARE_UNARCHIVER() [3/51]	103
5.1.3.55 GINAC_DECLARE_UNARCHIVER() [4/51]	103
5.1.3.56 GINAC_DECLARE_UNARCHIVER() [5/51]	103
5.1.3.57 GINAC_DECLARE_UNARCHIVER() [6/51]	103
5.1.3.58 GINAC_DECLARE_UNARCHIVER() [7/51]	104
5.1.3.59 GINAC_DECLARE_UNARCHIVER() [8/51]	104
5.1.3.60 is_clifford_tinfo()	104

5.1.3.61 clifford_bar()	104
5.1.3.62 clifford_star()	104
5.1.3.63 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [4/33]	105
5.1.3.64 print_func< print_dflt >() [2/3]	105
5.1.3.65 GINAC_BIND_UNARCHIVER() [9/49]	105
5.1.3.66 GINAC_BIND_UNARCHIVER() [10/49]	105
5.1.3.67 GINAC_BIND_UNARCHIVER() [11/49]	105
5.1.3.68 GINAC_BIND_UNARCHIVER() [12/49]	105
5.1.3.69 GINAC_BIND_UNARCHIVER() [13/49]	106
5.1.3.70 permute_free_index_to_front()	106
5.1.3.71 color_ONE()	106
5.1.3.72 color_T()	107
5.1.3.73 color_f()	107
5.1.3.74 color_d()	108
5.1.3.75 color_h()	108
5.1.3.76 is_color_tinfo()	108
5.1.3.77 get_representation_label() [2/2]	109
5.1.3.78 color_trace() [1/3]	109
5.1.3.79 color_trace() [2/3]	109
5.1.3.80 color_trace() [3/3]	109
5.1.3.81 GINAC_DECLARE_UNARCHIVER() [9/51]	110
5.1.3.82 GINAC_DECLARE_UNARCHIVER() [10/51]	110
5.1.3.83 GINAC_DECLARE_UNARCHIVER() [11/51]	110
5.1.3.84 GINAC_DECLARE_UNARCHIVER() [12/51]	110
5.1.3.85 GINAC_DECLARE_UNARCHIVER() [13/51]	110
5.1.3.86 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [5/33]	111
5.1.3.87 GINAC_BIND_UNARCHIVER() [14/49]	111
5.1.3.88 GINAC_DECLARE_UNARCHIVER() [14/51]	111
5.1.3.89 crc32()	111
5.1.3.90 are_ex_trivially_equal()	111
5.1.3.91 operator<<() [3/16]	112
5.1.3.92 operator<<() [4/16]	112
5.1.3.93 operator<<() [5/16]	112
5.1.3.94 nops() [1/2]	112
5.1.3.95 expand() [1/2]	112
5.1.3.96 conjugate()	113
5.1.3.97 real_part()	113
5.1.3.98 imag_part()	113
5.1.3.99 has()	113
5.1.3.100 find()	114
5.1.3.101 is_polynomial()	114
5.1.3.102 degree()	114

5.1.3.103 ldegree()	114
5.1.3.104 coeff()	115
5.1.3.105 numer() [1/2]	115
5.1.3.106 denom() [1/2]	115
5.1.3.107 numer_denom()	115
5.1.3.108 normal()	116
5.1.3.109 to_rational()	116
5.1.3.110 to_polynomial()	116
5.1.3.111 collect()	116
5.1.3.112 eval()	116
5.1.3.113 evalf() [1/2]	117
5.1.3.114 evalm()	117
5.1.3.115 eval_integ()	117
5.1.3.116 diff()	117
5.1.3.117 series()	118
5.1.3.118 match()	118
5.1.3.119 simplify_indexed() [1/3]	118
5.1.3.120 simplify_indexed() [2/3]	118
5.1.3.121 symmetrize() [1/4]	119
5.1.3.122 symmetrize() [2/4]	119
5.1.3.123 antisymmetrize() [1/4]	119
5.1.3.124 antisymmetrize() [2/4]	119
5.1.3.125 symmetrize_cyclic() [1/4]	119
5.1.3.126 symmetrize_cyclic() [2/4]	120
5.1.3.127 op()	120
5.1.3.128 lhs()	120
5.1.3.129 rhs()	120
5.1.3.130 is_zero() [1/2]	120
5.1.3.131 swap() [1/2]	121
5.1.3.132 subs() [1/3]	121
5.1.3.133 subs() [2/3]	121
5.1.3.134 subs() [3/3]	121
5.1.3.135 is_a() [2/3]	122
5.1.3.136 is_exactly_a() [2/2]	122
5.1.3.137 ex_to()	122
5.1.3.138 compile_ex() [1/3]	122
5.1.3.139 compile_ex() [2/3]	123
5.1.3.140 compile_ex() [3/3]	123
5.1.3.141 link_ex() [1/3]	124
5.1.3.142 link_ex() [2/3]	124
5.1.3.143 link_ex() [3/3]	125
5.1.3.144 unlink_ex()	125

5.1.3.145 swap() [2/2]	125
5.1.3.146 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [6/33]	125
5.1.3.147 conjugateepvector()	126
5.1.3.148 factor()	126
5.1.3.149 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [7/33]	126
5.1.3.150 GINAC_DECLARE_UNARCHIVER() [15/51]	127
5.1.3.151 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [8/33]	127
5.1.3.152 GINAC_BIND_UNARCHIVER() [15/49]	127
5.1.3.153 GINAC_DECLARE_UNARCHIVER() [16/51]	127
5.1.3.154 GINAC_BIND_UNARCHIVER() [16/49]	127
5.1.3.155 GINAC_DECLARE_UNARCHIVER() [17/51]	128
5.1.3.156 is_the_function()	128
5.1.3.157 make_hash_seed()	128
5.1.3.158 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [9/33]	128
5.1.3.159 print_func< print_context >()	129
5.1.3.160 GINAC_BIND_UNARCHIVER() [17/49]	129
5.1.3.161 GINAC_BIND_UNARCHIVER() [18/49]	129
5.1.3.162 GINAC_BIND_UNARCHIVER() [19/49]	129
5.1.3.163 is_dummy_pair() [1/2]	129
5.1.3.164 is_dummy_pair() [2/2]	130
5.1.3.165 find_free_and_dummy() [1/2]	130
5.1.3.166 minimal_dim()	130
5.1.3.167 GINAC_DECLARE_UNARCHIVER() [18/51]	131
5.1.3.168 GINAC_DECLARE_UNARCHIVER() [19/51]	131
5.1.3.169 GINAC_DECLARE_UNARCHIVER() [20/51]	131
5.1.3.170 find_free_and_dummy() [2/2]	131
5.1.3.171 find_dummy_indices()	131
5.1.3.172 count_dummy_indices()	132
5.1.3.173 count_free_indices()	132
5.1.3.174 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [10/33]	132
5.1.3.175 GINAC_BIND_UNARCHIVER() [20/49]	132
5.1.3.176 indices_consistent()	133
5.1.3.177 number_of_type()	133
5.1.3.178 rename_dummy_indices()	133
5.1.3.179 find_variant_indices()	133
5.1.3.180 reposition_dummy_indices()	134
5.1.3.181 product_to_exvector()	134
5.1.3.182 idx_symmetrization()	134
5.1.3.183 simplify_indexed() [3/3]	134
5.1.3.184 simplify_indexed_product()	135
5.1.3.185 hasindex()	135
5.1.3.186 get_all_dummy_indices_safely()	135

5.1.3.187 <code>get_all_dummy_indices()</code> . . . . .	135
5.1.3.188 <code>rename_dummy_indices_uniquely()</code> [1/4] . . . . .	136
5.1.3.189 <code>rename_dummy_indices_uniquely()</code> [2/4] . . . . .	136
5.1.3.190 <code>rename_dummy_indices_uniquely()</code> [3/4] . . . . .	136
5.1.3.191 <code>rename_dummy_indices_uniquely()</code> [4/4] . . . . .	136
5.1.3.192 <code>expand_dummy_sum()</code> . . . . .	137
5.1.3.193 <code>GINAC_DECLARE_UNARCHIVER()</code> [21/51] . . . . .	137
5.1.3.194 <code>conjugate_evalf()</code> . . . . .	137
5.1.3.195 <code>conjugate_eval()</code> . . . . .	137
5.1.3.196 <code>conjugate_print_latex()</code> . . . . .	138
5.1.3.197 <code>conjugate_conjugate()</code> . . . . .	138
5.1.3.198 <code>conjugate_expl_derivative()</code> . . . . .	138
5.1.3.199 <code>conjugate_real_part()</code> . . . . .	138
5.1.3.200 <code>conjugate_imag_part()</code> . . . . .	138
5.1.3.201 <code>func_arg_info()</code> . . . . .	139
5.1.3.202 <code>conjugate_info()</code> . . . . .	139
5.1.3.203 <code>REGISTER_FUNCTION()</code> [1/36] . . . . .	139
5.1.3.204 <code>real_part_evalf()</code> . . . . .	139
5.1.3.205 <code>real_part_eval()</code> . . . . .	139
5.1.3.206 <code>real_part_print_latex()</code> . . . . .	140
5.1.3.207 <code>real_part_conjugate()</code> . . . . .	140
5.1.3.208 <code>real_part_real_part()</code> . . . . .	140
5.1.3.209 <code>real_part_imag_part()</code> . . . . .	140
5.1.3.210 <code>real_part_expl_derivative()</code> . . . . .	140
5.1.3.211 <code>REGISTER_FUNCTION()</code> [2/36] . . . . .	140
5.1.3.212 <code>imag_part_evalf()</code> . . . . .	141
5.1.3.213 <code>imag_part_eval()</code> . . . . .	141
5.1.3.214 <code>imag_part_print_latex()</code> . . . . .	141
5.1.3.215 <code>imag_part_conjugate()</code> . . . . .	141
5.1.3.216 <code>imag_part_real_part()</code> . . . . .	141
5.1.3.217 <code>imag_part_imag_part()</code> . . . . .	141
5.1.3.218 <code>imag_part_expl_derivative()</code> . . . . .	142
5.1.3.219 <code>REGISTER_FUNCTION()</code> [3/36] . . . . .	142
5.1.3.220 <code>abs_evalf()</code> . . . . .	142
5.1.3.221 <code>abs_eval()</code> . . . . .	142
5.1.3.222 <code>abs_expand()</code> . . . . .	142
5.1.3.223 <code>abs_expl_derivative()</code> . . . . .	143
5.1.3.224 <code>abs_print_latex()</code> . . . . .	143
5.1.3.225 <code>abs_print_csrc_float()</code> . . . . .	143
5.1.3.226 <code>abs_conjugate()</code> . . . . .	143
5.1.3.227 <code>abs_real_part()</code> . . . . .	143
5.1.3.228 <code>abs_imag_part()</code> . . . . .	144

5.1.3.229 <code>abs_power()</code> . . . . .	144
5.1.3.230 <code>abs_info()</code> . . . . .	144
5.1.3.231 <code>REGISTER_FUNCTION()</code> [4/36] . . . . .	144
5.1.3.232 <code>step_evalf()</code> . . . . .	144
5.1.3.233 <code>step_eval()</code> . . . . .	145
5.1.3.234 <code>step_series()</code> . . . . .	145
5.1.3.235 <code>step_conjugate()</code> . . . . .	145
5.1.3.236 <code>step_real_part()</code> . . . . .	145
5.1.3.237 <code>step_imag_part()</code> . . . . .	145
5.1.3.238 <code>REGISTER_FUNCTION()</code> [5/36] . . . . .	146
5.1.3.239 <code>csgn_evalf()</code> . . . . .	146
5.1.3.240 <code>csgn_eval()</code> . . . . .	146
5.1.3.241 <code>csgn_series()</code> . . . . .	146
5.1.3.242 <code>csgn_conjugate()</code> . . . . .	146
5.1.3.243 <code>csgn_real_part()</code> . . . . .	147
5.1.3.244 <code>csgn_imag_part()</code> . . . . .	147
5.1.3.245 <code>csgn_power()</code> . . . . .	147
5.1.3.246 <code>REGISTER_FUNCTION()</code> [6/36] . . . . .	147
5.1.3.247 <code>eta_evalf()</code> . . . . .	147
5.1.3.248 <code>eta_eval()</code> . . . . .	148
5.1.3.249 <code>eta_series()</code> . . . . .	148
5.1.3.250 <code>eta_conjugate()</code> . . . . .	148
5.1.3.251 <code>eta_real_part()</code> . . . . .	148
5.1.3.252 <code>eta_imag_part()</code> . . . . .	148
5.1.3.253 <code>REGISTER_FUNCTION()</code> [7/36] . . . . .	149
5.1.3.254 <code>Li2_evalf()</code> . . . . .	149
5.1.3.255 <code>Li2_eval()</code> . . . . .	149
5.1.3.256 <code>Li2_deriv()</code> . . . . .	149
5.1.3.257 <code>Li2_series()</code> [1/2] . . . . .	149
5.1.3.258 <code>Li2_conjugate()</code> . . . . .	150
5.1.3.259 <code>REGISTER_FUNCTION()</code> [8/36] . . . . .	150
5.1.3.260 <code>Li3_eval()</code> . . . . .	150
5.1.3.261 <code>REGISTER_FUNCTION()</code> [9/36] . . . . .	150
5.1.3.262 <code>zetaderiv_eval()</code> . . . . .	150
5.1.3.263 <code>zetaderiv_deriv()</code> . . . . .	151
5.1.3.264 <code>REGISTER_FUNCTION()</code> [10/36] . . . . .	151
5.1.3.265 <code>factorial_evalf()</code> . . . . .	151
5.1.3.266 <code>factorial_eval()</code> . . . . .	151
5.1.3.267 <code>factorial_print_dflit_latex()</code> . . . . .	151
5.1.3.268 <code>factorial_conjugate()</code> . . . . .	152
5.1.3.269 <code>factorial_real_part()</code> . . . . .	152
5.1.3.270 <code>factorial_imag_part()</code> . . . . .	152

5.1.3.271 REGISTER_FUNCTION() [11/36]	152
5.1.3.272 binomial_evalf()	152
5.1.3.273 binomial_sym()	153
5.1.3.274 binomial_eval()	153
5.1.3.275 binomial_conjugate()	153
5.1.3.276 binomial_real_part()	153
5.1.3.277 binomial_imag_part()	153
5.1.3.278 REGISTER_FUNCTION() [12/36]	154
5.1.3.279 Order_eval()	154
5.1.3.280 Order_series()	154
5.1.3.281 Order_conjugate()	154
5.1.3.282 Order_real_part()	154
5.1.3.283 Order_imag_part()	155
5.1.3.284 Order_expl_derivative()	155
5.1.3.285 REGISTER_FUNCTION() [13/36]	155
5.1.3.286 lsolve()	155
5.1.3.287 fsolve()	155
5.1.3.288 zeta() [1/3]	156
5.1.3.289 zeta() [2/3]	156
5.1.3.290 is_the_function< zeta_SERIAL >()	156
5.1.3.291 G() [1/2]	157
5.1.3.292 G() [2/2]	157
5.1.3.293 is_the_function< G_SERIAL >()	157
5.1.3.294 psi() [1/4]	157
5.1.3.295 psi() [2/4]	158
5.1.3.296 is_the_function< psi_SERIAL >()	158
5.1.3.297 iterated_integral() [1/2]	158
5.1.3.298 iterated_integral() [2/2]	158
5.1.3.299 is_the_function< iterated_integral_SERIAL >()	158
5.1.3.300 is_order_function()	159
5.1.3.301 convert_H_to_Li()	159
5.1.3.302 EllipticK_evalf()	159
5.1.3.303 EllipticK_eval()	159
5.1.3.304 EllipticK_deriv()	159
5.1.3.305 EllipticK_series()	160
5.1.3.306 EllipticK_print_latex()	160
5.1.3.307 REGISTER_FUNCTION() [14/36]	160
5.1.3.308 EllipticE_evalf()	160
5.1.3.309 EllipticE_eval()	160
5.1.3.310 EllipticE_deriv()	161
5.1.3.311 EllipticE_series()	161
5.1.3.312 EllipticE_print_latex()	161

5.1.3.313 REGISTER_FUNCTION() [15/36]	161
5.1.3.314 iterated_integral_evalf_impl()	161
5.1.3.315 iterated_integral2_evalf()	162
5.1.3.316 iterated_integral3_evalf()	162
5.1.3.317 iterated_integral2_eval()	162
5.1.3.318 iterated_integral3_eval()	162
5.1.3.319 lgamma_evalf()	162
5.1.3.320 lgamma_eval()	162
5.1.3.321 lgamma_deriv()	163
5.1.3.322 lgamma_series()	163
5.1.3.323 lgamma_conjugate()	163
5.1.3.324 REGISTER_FUNCTION() [16/36]	163
5.1.3.325 tgamma_evalf()	164
5.1.3.326 tgamma_eval()	164
5.1.3.327 tgamma_deriv()	164
5.1.3.328 tgamma_series()	164
5.1.3.329 tgamma_conjugate()	165
5.1.3.330 REGISTER_FUNCTION() [17/36]	165
5.1.3.331 beta_evalf()	165
5.1.3.332 beta_eval()	165
5.1.3.333 beta_deriv()	165
5.1.3.334 beta_series()	166
5.1.3.335 REGISTER_FUNCTION() [18/36]	166
5.1.3.336 psi1_evalf()	166
5.1.3.337 psi1_eval()	166
5.1.3.338 psi1_deriv()	167
5.1.3.339 psi1_series()	167
5.1.3.340 psi2_evalf()	167
5.1.3.341 psi2_eval()	167
5.1.3.342 psi2_deriv()	168
5.1.3.343 psi2_series()	168
5.1.3.344 G2_evalf()	168
5.1.3.345 G2_eval()	168
5.1.3.346 G3_evalf()	169
5.1.3.347 G3_eval()	169
5.1.3.348 Li_evalf()	169
5.1.3.349 Li_eval()	169
5.1.3.350 Li_series()	170
5.1.3.351 Li_deriv()	170
5.1.3.352 Li_print_latex()	170
5.1.3.353 REGISTER_FUNCTION() [19/36]	170
5.1.3.354 S_evalf()	170



5.1.3.355 S_eval()	171
5.1.3.356 S_series()	171
5.1.3.357 S_deriv()	171
5.1.3.358 S_print_latex()	171
5.1.3.359 REGISTER_FUNCTION() [20/36]	172
5.1.3.360 H_evalf()	172
5.1.3.361 H_eval()	172
5.1.3.362 H_series()	172
5.1.3.363 H_deriv()	172
5.1.3.364 H_print_latex()	173
5.1.3.365 REGISTER_FUNCTION() [21/36]	173
5.1.3.366 zeta1_evalf()	173
5.1.3.367 zeta1_eval()	173
5.1.3.368 zeta1_deriv()	173
5.1.3.369 zeta1_print_latex()	174
5.1.3.370 zeta2_evalf()	174
5.1.3.371 zeta2_eval()	174
5.1.3.372 zeta2_deriv()	174
5.1.3.373 zeta2_print_latex()	174
5.1.3.374 exp_evalf()	175
5.1.3.375 exp_eval()	175
5.1.3.376 exp_expand()	175
5.1.3.377 exp_deriv()	175
5.1.3.378 exp_real_part()	175
5.1.3.379 exp_imag_part()	176
5.1.3.380 exp_conjugate()	176
5.1.3.381 exp_power()	176
5.1.3.382 exp_info()	176
5.1.3.383 REGISTER_FUNCTION() [22/36]	176
5.1.3.384 log_evalf()	177
5.1.3.385 log_eval()	177
5.1.3.386 log_deriv()	177
5.1.3.387 log_series()	177
5.1.3.388 log_real_part()	177
5.1.3.389 log_imag_part()	178
5.1.3.390 log_expand()	178
5.1.3.391 log_conjugate()	178
5.1.3.392 log_info()	178
5.1.3.393 REGISTER_FUNCTION() [23/36]	178
5.1.3.394 sin_evalf()	179
5.1.3.395 sin_eval()	179
5.1.3.396 sin_deriv()	179

5.1.3.397 sin_real_part()	179
5.1.3.398 sin_imag_part()	179
5.1.3.399 sin_conjugate()	180
5.1.3.400 trig_info()	180
5.1.3.401 REGISTER_FUNCTION() [24/36]	180
5.1.3.402 cos_evalf()	180
5.1.3.403 cos_eval()	180
5.1.3.404 cos_deriv()	181
5.1.3.405 cos_real_part()	181
5.1.3.406 cos_imag_part()	181
5.1.3.407 cos_conjugate()	181
5.1.3.408 REGISTER_FUNCTION() [25/36]	181
5.1.3.409 tan_evalf()	182
5.1.3.410 tan_eval()	182
5.1.3.411 tan_deriv()	182
5.1.3.412 tan_real_part()	182
5.1.3.413 tan_imag_part()	182
5.1.3.414 tan_series()	183
5.1.3.415 tan_conjugate()	183
5.1.3.416 REGISTER_FUNCTION() [26/36]	183
5.1.3.417 asin_evalf()	183
5.1.3.418 asin_eval()	183
5.1.3.419 asin_deriv()	184
5.1.3.420 asin_conjugate()	184
5.1.3.421 asin_info()	184
5.1.3.422 REGISTER_FUNCTION() [27/36]	184
5.1.3.423 acos_evalf()	184
5.1.3.424 acos_eval()	185
5.1.3.425 acos_deriv()	185
5.1.3.426 acos_conjugate()	185
5.1.3.427 REGISTER_FUNCTION() [28/36]	185
5.1.3.428 atan_evalf()	185
5.1.3.429 atan_eval()	186
5.1.3.430 atan_deriv()	186
5.1.3.431 atan_series()	186
5.1.3.432 atan_conjugate()	186
5.1.3.433 atan_info()	186
5.1.3.434 REGISTER_FUNCTION() [29/36]	187
5.1.3.435 atan2_evalf()	187
5.1.3.436 atan2_eval()	187
5.1.3.437 atan2_deriv()	187
5.1.3.438 atan2_info()	187

5.1.3.439 REGISTER_FUNCTION() [30/36]	188
5.1.3.440 sinh_evalf()	188
5.1.3.441 sinh_eval()	188
5.1.3.442 sinh_deriv()	188
5.1.3.443 sinh_real_part()	188
5.1.3.444 sinh_imag_part()	189
5.1.3.445 sinh_conjugate()	189
5.1.3.446 REGISTER_FUNCTION() [31/36]	189
5.1.3.447 cosh_evalf()	189
5.1.3.448 cosh_eval()	189
5.1.3.449 cosh_deriv()	190
5.1.3.450 cosh_real_part()	190
5.1.3.451 cosh_imag_part()	190
5.1.3.452 cosh_conjugate()	190
5.1.3.453 REGISTER_FUNCTION() [32/36]	190
5.1.3.454 tanh_evalf()	191
5.1.3.455 tanh_eval()	191
5.1.3.456 tanh_deriv()	191
5.1.3.457 tanh_series()	191
5.1.3.458 tanh_real_part()	191
5.1.3.459 tanh_imag_part()	192
5.1.3.460 tanh_conjugate()	192
5.1.3.461 REGISTER_FUNCTION() [33/36]	192
5.1.3.462 asinh_evalf()	192
5.1.3.463 asinh_eval()	192
5.1.3.464 asinh_deriv()	193
5.1.3.465 asinh_conjugate()	193
5.1.3.466 REGISTER_FUNCTION() [34/36]	193
5.1.3.467 acosh_evalf()	193
5.1.3.468 acosh_eval()	193
5.1.3.469 acosh_deriv()	194
5.1.3.470 acosh_conjugate()	194
5.1.3.471 REGISTER_FUNCTION() [35/36]	194
5.1.3.472 atanh_evalf()	194
5.1.3.473 atanh_eval()	194
5.1.3.474 atanh_deriv()	195
5.1.3.475 atanh_series()	195
5.1.3.476 atanh_conjugate()	195
5.1.3.477 REGISTER_FUNCTION() [36/36]	195
5.1.3.478 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [11/33]	195
5.1.3.479 subsvalue()	196
5.1.3.480 adaptivesimpson()	196

5.1.3.481 GINAC_BIND_UNARCHIVER() [21/49]	196
5.1.3.482 GINAC_DECLARE_UNARCHIVER() [22/51]	196
5.1.3.483 ifactor()	197
5.1.3.484 is_discriminant_of_quadratic_number_field()	197
5.1.3.485 kronecker_symbol()	197
5.1.3.486 primitive_dirichlet_character()	198
5.1.3.487 dirichlet_character()	198
5.1.3.488 generalised_Bernoulli_number()	198
5.1.3.489 Bernoulli_polynomial()	199
5.1.3.490 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [12/33]	199
5.1.3.491 GINAC_BIND_UNARCHIVER() [22/49]	199
5.1.3.492 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [13/33]	199
5.1.3.493 GINAC_BIND_UNARCHIVER() [23/49]	199
5.1.3.494 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [14/33]	200
5.1.3.495 GINAC_BIND_UNARCHIVER() [24/49]	200
5.1.3.496 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [15/33]	200
5.1.3.497 GINAC_BIND_UNARCHIVER() [25/49]	200
5.1.3.498 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [16/33]	200
5.1.3.499 GINAC_BIND_UNARCHIVER() [26/49]	200
5.1.3.500 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [17/33]	201
5.1.3.501 GINAC_BIND_UNARCHIVER() [27/49]	201
5.1.3.502 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [18/33]	201
5.1.3.503 GINAC_BIND_UNARCHIVER() [28/49]	201
5.1.3.504 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [19/33]	201
5.1.3.505 GINAC_BIND_UNARCHIVER() [29/49]	201
5.1.3.506 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [20/33]	202
5.1.3.507 GINAC_BIND_UNARCHIVER() [30/49]	202
5.1.3.508 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [21/33]	202
5.1.3.509 GINAC_BIND_UNARCHIVER() [31/49]	202
5.1.3.510 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [22/33]	202
5.1.3.511 GINAC_BIND_UNARCHIVER() [32/49]	202
5.1.3.512 GINAC_DECLARE_UNARCHIVER() [23/51]	203
5.1.3.513 GINAC_DECLARE_UNARCHIVER() [24/51]	203
5.1.3.514 GINAC_DECLARE_UNARCHIVER() [25/51]	203
5.1.3.515 GINAC_DECLARE_UNARCHIVER() [26/51]	203
5.1.3.516 GINAC_DECLARE_UNARCHIVER() [27/51]	203
5.1.3.517 GINAC_DECLARE_UNARCHIVER() [28/51]	203
5.1.3.518 GINAC_DECLARE_UNARCHIVER() [29/51]	203
5.1.3.519 GINAC_DECLARE_UNARCHIVER() [30/51]	204
5.1.3.520 GINAC_DECLARE_UNARCHIVER() [31/51]	204
5.1.3.521 GINAC_DECLARE_UNARCHIVER() [32/51]	204
5.1.3.522 GINAC_DECLARE_UNARCHIVER() [33/51]	204

5.1.3.523 GINAC_DECLARE_UNARCHIVER() [34/51]	204
5.1.3.524 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [23/33]	204
5.1.3.525 GINAC_BIND_UNARCHIVER() [33/49]	205
5.1.3.526 lst_to_matrix()	205
5.1.3.527 diag_matrix() [1/2]	205
5.1.3.528 diag_matrix() [2/2]	205
5.1.3.529 unit_matrix() [1/2]	205
5.1.3.530 symbolic_matrix() [1/2]	206
5.1.3.531 reduced_matrix()	206
5.1.3.532 sub_matrix()	206
5.1.3.533 GINAC_DECLARE_UNARCHIVER() [35/51]	206
5.1.3.534 nops() [2/2]	207
5.1.3.535 expand() [2/2]	207
5.1.3.536 evalf() [2/2]	207
5.1.3.537 rows()	207
5.1.3.538 cols()	207
5.1.3.539 transpose()	208
5.1.3.540 determinant()	208
5.1.3.541 trace()	208
5.1.3.542 charpoly()	208
5.1.3.543 inverse() [1/3]	208
5.1.3.544 inverse() [2/3]	209
5.1.3.545 rank() [1/2]	209
5.1.3.546 rank() [2/2]	209
5.1.3.547 unit_matrix() [2/2]	209
5.1.3.548 symbolic_matrix() [2/2]	209
5.1.3.549 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [24/33]	210
5.1.3.550 tryfactsubs()	210
5.1.3.551 algebraic_match_mul_with_mul()	210
5.1.3.552 GINAC_BIND_UNARCHIVER() [34/49]	210
5.1.3.553 GINAC_DECLARE_UNARCHIVER() [36/51]	211
5.1.3.554 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [25/33]	211
5.1.3.555 reeval_ncmul()	211
5.1.3.556 hold_ncmul()	211
5.1.3.557 GINAC_BIND_UNARCHIVER() [35/49]	211
5.1.3.558 GINAC_DECLARE_UNARCHIVER() [37/51]	211
5.1.3.559 get_first_symbol()	211
5.1.3.560 add_symbol()	212
5.1.3.561 collect_symbols()	212
5.1.3.562 get_symbol_stats()	212
5.1.3.563 lcmcoeff()	213
5.1.3.564 lcm_of_coefficients_denominators()	213

5.1.3.565 multiply_lcm()	214
5.1.3.566 quo()	214
5.1.3.567 rem()	215
5.1.3.568 decomp_rational()	215
5.1.3.569 prem()	216
5.1.3.570 sprem()	216
5.1.3.571 divide()	217
5.1.3.572 divide_in_z()	217
5.1.3.573 sr_gcd()	218
5.1.3.574 interpolate()	219
5.1.3.575 heur_gcd_z()	219
5.1.3.576 heur_gcd()	220
5.1.3.577 gcd_pf_pow()	221
5.1.3.578 gcd_pf_mul()	221
5.1.3.579 gcd() <sup>[1/2]</sup>	221
5.1.3.580 gcd_pf_pow_pow()	222
5.1.3.581 lcm() <sup>[1/2]</sup>	222
5.1.3.582 sqrfree_yun()	223
5.1.3.583 sqrfree()	223
5.1.3.584 sqrfree_parfrac()	224
5.1.3.585 replace_with_symbol() <sup>[1/2]</sup>	224
5.1.3.586 replace_with_symbol() <sup>[2/2]</sup>	225
5.1.3.587 frac_cancel()	225
5.1.3.588 find_common_factor()	226
5.1.3.589 collect_common_factors()	226
5.1.3.590 resultant()	227
5.1.3.591 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() <sup>[26/33]</sup>	227
5.1.3.592 make_real_float()	227
5.1.3.593 read_real_float()	228
5.1.3.594 GINAC_BIND_UNARCHIVER() <sup>[36/49]</sup>	228
5.1.3.595 write_real_float()	228
5.1.3.596 print_real_number()	228
5.1.3.597 print_integer_csrc()	229
5.1.3.598 print_real_csrc()	229
5.1.3.599 coerce()	229
5.1.3.600 coerce< int, cln::cl_I >()	230
5.1.3.601 coerce< unsigned int, cln::cl_I >()	230
5.1.3.602 print_real_cl_N()	230
5.1.3.603 exp()	230
5.1.3.604 log()	231
5.1.3.605 sin()	231
5.1.3.606 cos()	232

5.1.3.607 tan()	232
5.1.3.608 asin()	232
5.1.3.609 acos()	233
5.1.3.610 atan() [1/2]	233
5.1.3.611 atan() [2/2]	233
5.1.3.612 sinh()	234
5.1.3.613 cosh()	234
5.1.3.614 tanh()	235
5.1.3.615 asinh()	235
5.1.3.616 acosh()	235
5.1.3.617 atanh()	236
5.1.3.618 Li2_series() [2/2]	236
5.1.3.619 Li2_projection()	236
5.1.3.620 Li2_()	237
5.1.3.621 Li2()	237
5.1.3.622 zeta() [3/3]	237
5.1.3.623 guess_precision()	237
5.1.3.624 lgamma() [1/2]	238
5.1.3.625 lgamma() [2/2]	238
5.1.3.626 tgamma() [1/2]	238
5.1.3.627 tgamma() [2/2]	238
5.1.3.628 psi() [3/4]	239
5.1.3.629 psi() [4/4]	239
5.1.3.630 factorial()	239
5.1.3.631 doublefactorial()	239
5.1.3.632 binomial()	240
5.1.3.633 bernoulli()	240
5.1.3.634 fibonacci()	241
5.1.3.635 abs()	241
5.1.3.636 mod()	242
5.1.3.637 smod()	242
5.1.3.638 irem() [1/2]	242
5.1.3.639 irem() [2/2]	243
5.1.3.640 iquo() [1/2]	243
5.1.3.641 iquo() [2/2]	244
5.1.3.642 gcd() [2/2]	244
5.1.3.643 lcm() [2/2]	245
5.1.3.644 sqrt() [1/2]	245
5.1.3.645 isqrt()	245
5.1.3.646 PiEvalf()	246
5.1.3.647 EulerEvalf()	246
5.1.3.648 CatalanEvalf()	246

5.1.3.649 operator<<() [6/16]	246
5.1.3.650 GINAC_DECLARE_UNARCHIVER() [38/51]	246
5.1.3.651 pow() [1/3]	247
5.1.3.652 inverse() [3/3]	247
5.1.3.653 step()	247
5.1.3.654 csgn()	247
5.1.3.655 is_zero() [2/2]	248
5.1.3.656 is_positive()	248
5.1.3.657 is_negative()	248
5.1.3.658 is_integer()	248
5.1.3.659 is_pos_integer()	248
5.1.3.660 is_nonneg_integer()	249
5.1.3.661 is_even()	249
5.1.3.662 is_odd()	249
5.1.3.663 is_prime()	249
5.1.3.664 is_rational()	249
5.1.3.665 is_real()	250
5.1.3.666 is_cinteger()	250
5.1.3.667 is_crational()	250
5.1.3.668 to_int()	250
5.1.3.669 to_long()	250
5.1.3.670 to_double()	251
5.1.3.671 real()	251
5.1.3.672 imag()	251
5.1.3.673 numer() [2/2]	251
5.1.3.674 denom() [2/2]	251
5.1.3.675 exadd()	252
5.1.3.676 exmul()	252
5.1.3.677 exminus()	252
5.1.3.678 operator+() [1/4]	252
5.1.3.679 operator-() [1/4]	253
5.1.3.680 operator*() [1/2]	253
5.1.3.681 operator/() [1/2]	253
5.1.3.682 operator+() [2/4]	253
5.1.3.683 operator-() [2/4]	253
5.1.3.684 operator*() [2/2]	254
5.1.3.685 operator/() [2/2]	254
5.1.3.686 operator+=() [1/2]	254
5.1.3.687 operator-=() [1/2]	254
5.1.3.688 operator*=() [1/2]	254
5.1.3.689 operator/=() [1/2]	255
5.1.3.690 operator+=() [2/2]	255



5.1.3.691 operator==( ) [ 2/2 ] . . . . .	255
5.1.3.692 operator*==( ) [ 2/2 ] . . . . .	255
5.1.3.693 operator/==( ) [ 2/2 ] . . . . .	255
5.1.3.694 operator+( ) [ 3/4 ] . . . . .	256
5.1.3.695 operator-( ) [ 3/4 ] . . . . .	256
5.1.3.696 operator+( ) [ 4/4 ] . . . . .	256
5.1.3.697 operator-( ) [ 4/4 ] . . . . .	256
5.1.3.698 operator++( ) [ 1/4 ] . . . . .	256
5.1.3.699 operator--( ) [ 1/4 ] . . . . .	257
5.1.3.700 operator++( ) [ 2/4 ] . . . . .	257
5.1.3.701 operator--( ) [ 2/4 ] . . . . .	257
5.1.3.702 operator++( ) [ 3/4 ] . . . . .	257
5.1.3.703 operator--( ) [ 3/4 ] . . . . .	258
5.1.3.704 operator++( ) [ 4/4 ] . . . . .	258
5.1.3.705 operator--( ) [ 4/4 ] . . . . .	258
5.1.3.706 operator==( ) . . . . .	258
5.1.3.707 operator!=( ) . . . . .	259
5.1.3.708 operator<( ) . . . . .	259
5.1.3.709 operator<=( ) . . . . .	259
5.1.3.710 operator>( ) . . . . .	259
5.1.3.711 operator>=( ) . . . . .	259
5.1.3.712 my_ios_index( ) . . . . .	260
5.1.3.713 my_ios_callback( ) . . . . .	260
5.1.3.714 get_print_context( ) . . . . .	260
5.1.3.715 set_print_context( ) . . . . .	260
5.1.3.716 get_print_options( ) . . . . .	260
5.1.3.717 set_print_options( ) . . . . .	261
5.1.3.718 operator<<( ) [ 7/16 ] . . . . .	261
5.1.3.719 operator>>( ) [ 3/3 ] . . . . .	261
5.1.3.720 dflt( ) . . . . .	261
5.1.3.721 latex( ) . . . . .	261
5.1.3.722 python( ) . . . . .	262
5.1.3.723 python_repr( ) . . . . .	262
5.1.3.724 tree( ) . . . . .	262
5.1.3.725 csrc( ) . . . . .	262
5.1.3.726 csrc_float( ) . . . . .	262
5.1.3.727 csrc_double( ) . . . . .	263
5.1.3.728 csrc_cl_N( ) . . . . .	263
5.1.3.729 index_dimensions( ) . . . . .	263
5.1.3.730 no_index_dimensions( ) . . . . .	263
5.1.3.731 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT( ) [ 27/33 ] . . . . .	263
5.1.3.732 print_sym_pow( ) . . . . .	264

5.1.3.733 GINAC_BIND_UNARCHIVER() [37/49]	264
5.1.3.734 GINAC_DECLARE_UNARCHIVER() [39/51]	264
5.1.3.735 pow() [2/3]	264
5.1.3.736 pow() [3/3]	264
5.1.3.737 sqrt() [2/2]	265
5.1.3.738 is_a() [3/3]	265
5.1.3.739 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [28/33]	265
5.1.3.740 GINAC_BIND_UNARCHIVER() [38/49]	265
5.1.3.741 GINAC_DECLARE_UNARCHIVER() [40/51]	265
5.1.3.742 series_to_poly()	265
5.1.3.743 is_terminating()	266
5.1.3.744 make_return_type_t()	266
5.1.3.745 set_print_func() [1/2]	266
5.1.3.746 set_print_func() [2/2]	267
5.1.3.747 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [29/33]	267
5.1.3.748 GINAC_BIND_UNARCHIVER() [39/49]	267
5.1.3.749 print_operator()	267
5.1.3.750 GINAC_DECLARE_UNARCHIVER() [41/51]	267
5.1.3.751 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [30/33]	268
5.1.3.752 get_default_TeX_name()	268
5.1.3.753 GINAC_BIND_UNARCHIVER() [40/49]	268
5.1.3.754 GINAC_BIND_UNARCHIVER() [41/49]	268
5.1.3.755 GINAC_BIND_UNARCHIVER() [42/49]	268
5.1.3.756 GINAC_DECLARE_UNARCHIVER() [42/51]	269
5.1.3.757 GINAC_DECLARE_UNARCHIVER() [43/51]	269
5.1.3.758 GINAC_DECLARE_UNARCHIVER() [44/51]	269
5.1.3.759 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [31/33]	269
5.1.3.760 GINAC_BIND_UNARCHIVER() [43/49]	269
5.1.3.761 index0()	269
5.1.3.762 index1()	270
5.1.3.763 index2()	270
5.1.3.764 index3()	270
5.1.3.765 not_symmetric()	270
5.1.3.766 symmetric2()	270
5.1.3.767 symmetric3()	270
5.1.3.768 symmetric4()	271
5.1.3.769 antisymmetric2()	271
5.1.3.770 antisymmetric3()	271
5.1.3.771 antisymmetric4()	271
5.1.3.772 canonicalize()	271
5.1.3.773 symm()	272
5.1.3.774 symmetrize() [3/4]	272

5.1.3.775 antisymmetrize() [3/4]	272
5.1.3.776 symmetrize_cyclic() [3/4]	273
5.1.3.777 GINAC_DECLARE_UNARCHIVER() [45/51]	273
5.1.3.778 sy_none() [1/4]	273
5.1.3.779 sy_none() [2/4]	273
5.1.3.780 sy_none() [3/4]	273
5.1.3.781 sy_none() [4/4]	274
5.1.3.782 sy_symm() [1/4]	274
5.1.3.783 sy_symm() [2/4]	274
5.1.3.784 sy_symm() [3/4]	274
5.1.3.785 sy_symm() [4/4]	274
5.1.3.786 sy_anti() [1/4]	275
5.1.3.787 sy_anti() [2/4]	275
5.1.3.788 sy_anti() [3/4]	275
5.1.3.789 sy_anti() [4/4]	275
5.1.3.790 sy_cycl() [1/4]	275
5.1.3.791 sy_cycl() [2/4]	276
5.1.3.792 sy_cycl() [3/4]	276
5.1.3.793 sy_cycl() [4/4]	276
5.1.3.794 symmetrize() [4/4]	276
5.1.3.795 antisymmetrize() [4/4]	276
5.1.3.796 symmetrize_cyclic() [4/4]	277
5.1.3.797 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [32/33]	277
5.1.3.798 print_func< print_dflt >() [3/3]	277
5.1.3.799 GINAC_BIND_UNARCHIVER() [44/49]	277
5.1.3.800 GINAC_BIND_UNARCHIVER() [45/49]	277
5.1.3.801 GINAC_BIND_UNARCHIVER() [46/49]	277
5.1.3.802 GINAC_BIND_UNARCHIVER() [47/49]	278
5.1.3.803 GINAC_BIND_UNARCHIVER() [48/49]	278
5.1.3.804 delta_tensor()	278
5.1.3.805 metric_tensor()	278
5.1.3.806 lorentz_g()	279
5.1.3.807 spinor_metric()	279
5.1.3.808 epsilon_tensor() [1/2]	280
5.1.3.809 epsilon_tensor() [2/2]	280
5.1.3.810 lorentz_eps()	281
5.1.3.811 GINAC_DECLARE_UNARCHIVER() [46/51]	281
5.1.3.812 GINAC_DECLARE_UNARCHIVER() [47/51]	281
5.1.3.813 GINAC_DECLARE_UNARCHIVER() [48/51]	282
5.1.3.814 GINAC_DECLARE_UNARCHIVER() [49/51]	282
5.1.3.815 GINAC_DECLARE_UNARCHIVER() [50/51]	282
5.1.3.816 log2()	282

5.1.3.817 multinomial_coefficient()	282
5.1.3.818 rotate_left()	283
5.1.3.819 compare_pointers()	283
5.1.3.820 golden_ratio_hash()	283
5.1.3.821 permutation_sign() [1/2]	284
5.1.3.822 permutation_sign() [2/2]	284
5.1.3.823 shaker_sort()	284
5.1.3.824 cyclic_permutation()	284
5.1.3.825 format_index_value() [1/2]	285
5.1.3.826 format_index_value() [2/2]	285
5.1.3.827 operator<<() [8/16]	285
5.1.3.828 operator<<() [9/16]	285
5.1.3.829 operator<<() [10/16]	286
5.1.3.830 operator<<() [11/16]	286
5.1.3.831 operator<<() [12/16]	286
5.1.3.832 operator<<() [13/16]	286
5.1.3.833 operator<<() [14/16]	287
5.1.3.834 operator<<() [15/16]	287
5.1.3.835 operator<<() [16/16]	287
5.1.3.836 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [33/33]	287
5.1.3.837 GINAC_BIND_UNARCHIVER() [49/49]	288
5.1.3.838 haswild()	288
5.1.3.839 GINAC_DECLARE_UNARCHIVER() [51/51]	288
5.1.3.840 wild()	288
5.1.4 Variable Documentation	288
5.1.4.1 unarch_table_instance	288
5.1.4.2 map_evalm	289
5.1.4.3 map_eval_integ	289
5.1.4.4 tensor	289
5.1.4.5 Pi	289
5.1.4.6 Euler	289
5.1.4.7 Catalan	290
5.1.4.8 crctab	290
5.1.4.9 library_initializer	290
5.1.4.10 _num0_bp	290
5.1.4.11 idx	290
5.1.4.12 force_include_tgamma	290
5.1.4.13 force_include_zeta1	291
5.1.4.14 GINAC_BIND_UNARCHIVER	291
5.1.4.15 l	291
5.1.4.16 Digits	291
5.1.4.17 next_print_context_id	292

5.1.4.18 version_major . . . . .	292
5.1.4.19 version_minor . . . . .	292
5.1.4.20 version_micro . . . . .	292
5.1.4.21 _num_120_p . . . . .	292
5.1.4.22 _ex_120 . . . . .	292
5.1.4.23 _num_60_p . . . . .	292
5.1.4.24 _ex_60 . . . . .	293
5.1.4.25 _num_48_p . . . . .	293
5.1.4.26 _ex_48 . . . . .	293
5.1.4.27 _num_30_p . . . . .	293
5.1.4.28 _ex_30 . . . . .	293
5.1.4.29 _num_25_p . . . . .	293
5.1.4.30 _ex_25 . . . . .	294
5.1.4.31 _num_24_p . . . . .	294
5.1.4.32 _ex_24 . . . . .	294
5.1.4.33 _num_20_p . . . . .	294
5.1.4.34 _ex_20 . . . . .	294
5.1.4.35 _num_18_p . . . . .	294
5.1.4.36 _ex_18 . . . . .	295
5.1.4.37 _num_15_p . . . . .	295
5.1.4.38 _ex_15 . . . . .	295
5.1.4.39 _num_12_p . . . . .	295
5.1.4.40 _ex_12 . . . . .	295
5.1.4.41 _num_11_p . . . . .	295
5.1.4.42 _ex_11 . . . . .	296
5.1.4.43 _num_10_p . . . . .	296
5.1.4.44 _ex_10 . . . . .	296
5.1.4.45 _num_9_p . . . . .	296
5.1.4.46 _ex_9 . . . . .	296
5.1.4.47 _num_8_p . . . . .	296
5.1.4.48 _ex_8 . . . . .	297
5.1.4.49 _num_7_p . . . . .	297
5.1.4.50 _ex_7 . . . . .	297
5.1.4.51 _num_6_p . . . . .	297
5.1.4.52 _ex_6 . . . . .	297
5.1.4.53 _num_5_p . . . . .	297
5.1.4.54 _ex_5 . . . . .	298
5.1.4.55 _num_4_p . . . . .	298
5.1.4.56 _ex_4 . . . . .	298
5.1.4.57 _num_3_p . . . . .	298
5.1.4.58 _ex_3 . . . . .	298
5.1.4.59 _num_2_p . . . . .	298

5.1.4.60 _ex_2 . . . . .	299
5.1.4.61 _num_1_p . . . . .	299
5.1.4.62 _ex_1 . . . . .	299
5.1.4.63 _num_1_2_p . . . . .	299
5.1.4.64 _ex_1_2 . . . . .	299
5.1.4.65 _num_1_3_p . . . . .	300
5.1.4.66 _ex_1_3 . . . . .	300
5.1.4.67 _num_1_4_p . . . . .	300
5.1.4.68 _ex_1_4 . . . . .	300
5.1.4.69 _num0_p . . . . .	300
5.1.4.70 _ex0 . . . . .	301
5.1.4.71 _num1_4_p . . . . .	301
5.1.4.72 _ex1_4 . . . . .	301
5.1.4.73 _num1_3_p . . . . .	301
5.1.4.74 _ex1_3 . . . . .	302
5.1.4.75 _num1_2_p . . . . .	302
5.1.4.76 _ex1_2 . . . . .	302
5.1.4.77 _num1_p . . . . .	302
5.1.4.78 _ex1 . . . . .	303
5.1.4.79 _num2_p . . . . .	303
5.1.4.80 _ex2 . . . . .	303
5.1.4.81 _num3_p . . . . .	304
5.1.4.82 _ex3 . . . . .	304
5.1.4.83 _num4_p . . . . .	304
5.1.4.84 _ex4 . . . . .	304
5.1.4.85 _num5_p . . . . .	304
5.1.4.86 _ex5 . . . . .	304
5.1.4.87 _num6_p . . . . .	305
5.1.4.88 _ex6 . . . . .	305
5.1.4.89 _num7_p . . . . .	305
5.1.4.90 _ex7 . . . . .	305
5.1.4.91 _num8_p . . . . .	305
5.1.4.92 _ex8 . . . . .	305
5.1.4.93 _num9_p . . . . .	306
5.1.4.94 _ex9 . . . . .	306
5.1.4.95 _num10_p . . . . .	306
5.1.4.96 _ex10 . . . . .	306
5.1.4.97 _num11_p . . . . .	306
5.1.4.98 _ex11 . . . . .	306
5.1.4.99 _num12_p . . . . .	307
5.1.4.100 _ex12 . . . . .	307
5.1.4.101 _num15_p . . . . .	307

5.1.4.102 _ex15 . . . . .	307
5.1.4.103 _num18_p . . . . .	307
5.1.4.104 _ex18 . . . . .	307
5.1.4.105 _num20_p . . . . .	308
5.1.4.106 _ex20 . . . . .	308
5.1.4.107 _num24_p . . . . .	308
5.1.4.108 _ex24 . . . . .	308
5.1.4.109 _num25_p . . . . .	308
5.1.4.110 _ex25 . . . . .	308
5.1.4.111 _num30_p . . . . .	309
5.1.4.112 _ex30 . . . . .	309
5.1.4.113 _num48_p . . . . .	309
5.1.4.114 _ex48 . . . . .	309
5.1.4.115 _num60_p . . . . .	309
5.1.4.116 _ex60 . . . . .	309
5.1.4.117 _num120_p . . . . .	310
5.1.4.118 _ex120 . . . . .	310
5.2 GiNaC::internal Namespace Reference . . . . .	310
5.3 std Namespace Reference . . . . .	310
5.3.1 Function Documentation . . . . .	310
5.3.1.1 swap() . . . . .	310
<b>6 Class Documentation</b> . . . . .	<b>311</b>
6.1 GiNaC::internal::_iter_rep Struct Reference . . . . .	311
6.1.1 Constructor & Destructor Documentation . . . . .	312
6.1.1.1 _iter_rep() . . . . .	312
6.1.2 Member Function Documentation . . . . .	312
6.1.2.1 operator==( ) . . . . .	312
6.1.2.2 operator!=( ) . . . . .	312
6.1.3 Member Data Documentation . . . . .	312
6.1.3.1 e . . . . .	313
6.1.3.2 i . . . . .	313
6.1.3.3 i_end . . . . .	313
6.2 GiNaC::_numeric_digits Class Reference . . . . .	313
6.2.1 Detailed Description . . . . .	314
6.2.2 Constructor & Destructor Documentation . . . . .	314
6.2.2.1 _numeric_digits() . . . . .	314
6.2.3 Member Function Documentation . . . . .	314
6.2.3.1 operator=( ) . . . . .	314
6.2.3.2 operator long() . . . . .	315
6.2.3.3 print() . . . . .	315
6.2.3.4 add_callback() . . . . .	315

6.2.4 Member Data Documentation	315
6.2.4.1 digits	315
6.2.4.2 too_late	315
6.2.4.3 callbacklist	316
6.3 GiNaC::add Class Reference	316
6.3.1 Detailed Description	323
6.3.2 Constructor & Destructor Documentation	323
6.3.2.1 add() [1/6]	323
6.3.2.2 add() [2/6]	324
6.3.2.3 add() [3/6]	324
6.3.2.4 add() [4/6]	324
6.3.2.5 add() [5/6]	324
6.3.2.6 add() [6/6]	324
6.3.3 Member Function Documentation	325
6.3.3.1 precedence()	325
6.3.3.2 info()	325
6.3.3.3 is_polynomial()	325
6.3.3.4 degree()	326
6.3.3.5 ldegree()	326
6.3.3.6 coeff()	326
6.3.3.7 eval()	326
6.3.3.8 evalm()	327
6.3.3.9 series()	327
6.3.3.10 normal()	327
6.3.3.11 integer_content()	328
6.3.3.12 smod()	328
6.3.3.13 max_coefficient()	328
6.3.3.14 conjugate()	329
6.3.3.15 real_part()	329
6.3.3.16 imag_part()	329
6.3.3.17 get_free_indices()	329
6.3.3.18 eval_ncmul()	329
6.3.3.19 derivative()	330
6.3.3.20 return_type()	330
6.3.3.21 return_type_tinfo()	330
6.3.3.22 thisexpairseq() [1/2]	330
6.3.3.23 thisexpairseq() [2/2]	331
6.3.3.24 split_ex_to_pair()	331
6.3.3.25 combine_ex_with_coeff_to_pair()	331
6.3.3.26 combine_pair_with_coeff_to_pair()	331
6.3.3.27 recombine_pair_to_ex()	332
6.3.3.28 expand()	332



6.3.3.29 print_add()	332
6.3.3.30 do_print()	333
6.3.3.31 do_print_latex()	333
6.3.3.32 do_print_csrc()	333
6.3.3.33 do_print_python_repr()	333
6.3.4 Friends And Related Function Documentation	333
6.3.4.1 mul	333
6.3.4.2 power	334
6.4 GiNaC::archive Class Reference	334
6.4.1 Detailed Description	335
6.4.2 Member Typedef Documentation	335
6.4.2.1 inv_at_cit	336
6.4.3 Constructor & Destructor Documentation	336
6.4.3.1 archive() [1/3]	336
6.4.3.2 ~archive()	336
6.4.3.3 archive() [2/3]	336
6.4.3.4 archive() [3/3]	336
6.4.4 Member Function Documentation	336
6.4.4.1 archive_ex()	336
6.4.4.2 unarchive_ex() [1/3]	337
6.4.4.3 unarchive_ex() [2/3]	337
6.4.4.4 unarchive_ex() [3/3]	338
6.4.4.5 num_expressions()	338
6.4.4.6 get_top_node()	338
6.4.4.7 clear()	338
6.4.4.8 add_node()	339
6.4.4.9 get_node()	339
6.4.4.10 forget()	339
6.4.4.11 printraw()	339
6.4.4.12 atomize()	340
6.4.4.13 unatomize()	340
6.4.5 Friends And Related Function Documentation	340
6.4.5.1 operator<<	340
6.4.5.2 operator>>	340
6.4.6 Member Data Documentation	341
6.4.6.1 nodes	341
6.4.6.2 exprs	341
6.4.6.3 atoms	341
6.4.6.4 inverse_atoms	341
6.4.6.5 exprtable	341
6.5 GiNaC::archive_node Class Reference	342
6.5.1 Detailed Description	344

6.5.2 Member Typedef Documentation	344
6.5.2.1 propinfovector	344
6.5.2.2 archive_node_cit	344
6.5.3 Member Enumeration Documentation	344
6.5.3.1 property_type	344
6.5.4 Constructor & Destructor Documentation	345
6.5.4.1 archive_node() [1/2]	345
6.5.4.2 archive_node() [2/2]	345
6.5.5 Member Function Documentation	345
6.5.5.1 operator=()	345
6.5.5.2 add_bool()	346
6.5.5.3 add_unsigned()	346
6.5.5.4 add_string()	346
6.5.5.5 add_ex()	346
6.5.5.6 find_bool()	347
6.5.5.7 find_unsigned()	347
6.5.5.8 find_string()	347
6.5.5.9 find_first()	348
6.5.5.10 find_last()	348
6.5.5.11 find_property_range()	348
6.5.5.12 find_ex()	348
6.5.5.13 find_ex_by_loc()	349
6.5.5.14 find_ex_node()	349
6.5.5.15 get_properties()	349
6.5.5.16 unarchive()	349
6.5.5.17 has_same_ex_as()	350
6.5.5.18 has_ex()	350
6.5.5.19 get_ex()	350
6.5.5.20 forget()	350
6.5.5.21 printraw()	350
6.5.6 Friends And Related Function Documentation	351
6.5.6.1 operator<<	351
6.5.6.2 operator>>	351
6.5.7 Member Data Documentation	351
6.5.7.1 a	351
6.5.7.2 props	351
6.5.7.3 has_expression	352
6.5.7.4 e	352
6.6 GiNaC::archive_node::archive_node_cit_range Struct Reference	352
6.6.1 Member Data Documentation	353
6.6.1.1 begin	353
6.6.1.2 end	354

6.7 GiNaC::archive::archived_ex Struct Reference	354
6.7.1 Detailed Description	354
6.7.2 Constructor & Destructor Documentation	354
6.7.2.1 archived_ex() [1/2]	354
6.7.2.2 archived_ex() [2/2]	354
6.7.3 Member Data Documentation	355
6.7.3.1 name	355
6.7.3.2 root	355
6.8 GiNaC::basic Class Reference	355
6.8.1 Detailed Description	360
6.8.2 Constructor & Destructor Documentation	360
6.8.2.1 basic() [1/2]	360
6.8.2.2 ~basic()	360
6.8.2.3 basic() [2/2]	360
6.8.3 Member Function Documentation	360
6.8.3.1 operator=()	361
6.8.3.2 duplicate()	361
6.8.3.3 eval()	361
6.8.3.4 evalf()	361
6.8.3.5 evalm()	362
6.8.3.6 eval_integ()	362
6.8.3.7 eval_ncmul()	362
6.8.3.8 eval_indexed()	362
6.8.3.9 print()	362
6.8.3.10 dbgprint()	363
6.8.3.11 dbgprinttree()	363
6.8.3.12 precedence()	363
6.8.3.13 info()	364
6.8.3.14 nops()	364
6.8.3.15 op()	364
6.8.3.16 operator[]() [1/4]	365
6.8.3.17 operator[]() [2/4]	365
6.8.3.18 let_op()	365
6.8.3.19 operator[]() [3/4]	365
6.8.3.20 operator[]() [4/4]	366
6.8.3.21 has()	366
6.8.3.22 match()	366
6.8.3.23 match_same_type()	367
6.8.3.24 subs()	367
6.8.3.25 map()	367
6.8.3.26 accept()	368
6.8.3.27 is_polynomial()	368

6.8.3.28 degree()	368
6.8.3.29 ldegree()	368
6.8.3.30 coeff()	369
6.8.3.31 expand()	369
6.8.3.32 collect()	369
6.8.3.33 derivative()	370
6.8.3.34 series()	370
6.8.3.35 normal()	371
6.8.3.36 to_rational()	371
6.8.3.37 to_polynomial()	371
6.8.3.38 integer_content()	372
6.8.3.39 smod()	372
6.8.3.40 max_coefficient()	372
6.8.3.41 get_free_indices()	373
6.8.3.42 add_indexed()	373
6.8.3.43 scalar_mul_indexed()	373
6.8.3.44 contract_with()	374
6.8.3.45 return_type()	375
6.8.3.46 return_type_tinfo()	375
6.8.3.47 conjugate()	375
6.8.3.48 real_part()	375
6.8.3.49 imag_part()	375
6.8.3.50 compare_same_type()	376
6.8.3.51 is_equal_same_type()	376
6.8.3.52 calchash()	376
6.8.3.53 print_dispatch() [1/2]	377
6.8.3.54 print_dispatch() [2/2]	377
6.8.3.55 archive()	377
6.8.3.56 read_archive()	378
6.8.3.57 subs_one_level()	378
6.8.3.58 diff()	378
6.8.3.59 compare()	379
6.8.3.60 is_equal()	379
6.8.3.61 hold()	380
6.8.3.62 gethash()	380
6.8.3.63 setflag()	381
6.8.3.64 clearflag()	381
6.8.3.65 ensure_if_modifiable()	381
6.8.3.66 do_print()	382
6.8.3.67 do_print_tree()	382
6.8.3.68 do_print_python_repr()	382
6.8.4 Friends And Related Function Documentation	382

6.8.4.1 ex	382
6.8.5 Member Data Documentation	382
6.8.5.1 flags	383
6.8.5.2 hashvalue	383
6.9 GiNaC::basic_log_kernel Class Reference	383
6.9.1 Detailed Description	388
6.9.2 Member Function Documentation	389
6.9.2.1 series_coeff_impl()	389
6.9.2.2 do_print()	389
6.10 GiNaC::basic_multi_iterator< T > Class Template Reference	389
6.10.1 Detailed Description	391
6.10.2 Constructor & Destructor Documentation	391
6.10.2.1 basic_multi_iterator() [1/3]	391
6.10.2.2 basic_multi_iterator() [2/3]	391
6.10.2.3 basic_multi_iterator() [3/3]	392
6.10.2.4 ~basic_multi_iterator()	392
6.10.3 Member Function Documentation	392
6.10.3.1 size()	392
6.10.3.2 overflow()	392
6.10.3.3 get_vector()	393
6.10.3.4 operator[]() [1/2]	393
6.10.3.5 operator[]() [2/2]	393
6.10.3.6 operator()() [1/2]	393
6.10.3.7 operator()() [2/2]	393
6.10.3.8 init()	394
6.10.3.9 operator++()	394
6.10.4 Friends And Related Function Documentation	394
6.10.4.1 operator<<	394
6.10.5 Member Data Documentation	394
6.10.5.1 N	394
6.10.5.2 B	395
6.10.5.3 v	395
6.10.5.4 flag_overflow	395
6.11 GiNaC::basic_partition_generator Class Reference	395
6.11.1 Detailed Description	396
6.11.2 Constructor & Destructor Documentation	396
6.11.2.1 basic_partition_generator()	396
6.11.3 Member Data Documentation	397
6.11.3.1 mpgen	397
6.12 GiNaC::class_info< OPT > Class Template Reference	397
6.12.1 Constructor & Destructor Documentation	398
6.12.1.1 class_info()	398

6.12.2 Member Function Documentation	398
6.12.2.1 get_parent()	399
6.12.2.2 find()	399
6.12.2.3 dump_hierarchy()	399
6.12.2.4 dump_tree()	399
6.12.2.5 identify_parents()	400
6.12.3 Member Data Documentation	400
6.12.3.1 options	400
6.12.3.2 first	400
6.12.3.3 next	400
6.12.3.4 parent	400
6.12.3.5 parents_identified	401
6.13 GiNaC::clifford Class Reference	401
6.13.1 Detailed Description	410
6.13.2 Constructor & Destructor Documentation	410
6.13.2.1 clifford() [1/4]	410
6.13.2.2 clifford() [2/4]	410
6.13.2.3 clifford() [3/4]	411
6.13.2.4 clifford() [4/4]	411
6.13.3 Member Function Documentation	411
6.13.3.1 precedence()	411
6.13.3.2 archive()	411
6.13.3.3 read_archive()	412
6.13.3.4 eval_ncmul()	412
6.13.3.5 match_same_type()	412
6.13.3.6 thiscontainer() [1/2]	413
6.13.3.7 thiscontainer() [2/2]	413
6.13.3.8 return_type()	413
6.13.3.9 return_type_tinfo()	413
6.13.3.10 get_representation_label()	413
6.13.3.11 get_metric() [1/2]	413
6.13.3.12 get_metric() [2/2]	414
6.13.3.13 same_metric()	414
6.13.3.14 get_commutator_sign()	414
6.13.3.15 nops()	414
6.13.3.16 op()	415
6.13.3.17 let_op()	415
6.13.3.18 subs()	415
6.13.3.19 do_print_dflt()	415
6.13.3.20 do_print_latex()	416
6.13.3.21 do_print_tree()	416
6.13.4 Member Data Documentation	416

6.13.4.1 representation_label . . . . .	416
6.13.4.2 metric . . . . .	416
6.13.4.3 commutator_sign . . . . .	417
6.14 GiNaC::cliffordunit Class Reference . . . . .	417
6.14.1 Detailed Description . . . . .	421
6.14.2 Member Function Documentation . . . . .	421
6.14.2.1 contract_with() . . . . .	422
6.14.2.2 do_print() . . . . .	422
6.14.2.3 do_print_latex() . . . . .	422
6.15 GiNaC::color Class Reference . . . . .	422
6.15.1 Detailed Description . . . . .	431
6.15.2 Constructor & Destructor Documentation . . . . .	431
6.15.2.1 color() [1/4] . . . . .	432
6.15.2.2 color() [2/4] . . . . .	432
6.15.2.3 color() [3/4] . . . . .	432
6.15.2.4 color() [4/4] . . . . .	432
6.15.3 Member Function Documentation . . . . .	432
6.15.3.1 archive() . . . . .	433
6.15.3.2 read_archive() . . . . .	433
6.15.3.3 eval_ncmul() . . . . .	433
6.15.3.4 match_same_type() . . . . .	434
6.15.3.5 thiscontainer() [1/2] . . . . .	434
6.15.3.6 thiscontainer() [2/2] . . . . .	434
6.15.3.7 return_type() . . . . .	434
6.15.3.8 return_type_tinfo() . . . . .	435
6.15.3.9 get_representation_label() . . . . .	435
6.15.4 Member Data Documentation . . . . .	435
6.15.4.1 representation_label . . . . .	435
6.16 GiNaC::compare_all_equal< T > Class Template Reference . . . . .	435
6.16.1 Detailed Description . . . . .	436
6.16.2 Constructor & Destructor Documentation . . . . .	436
6.16.2.1 ~compare_all_equal() . . . . .	436
6.16.3 Member Function Documentation . . . . .	436
6.16.3.1 struct_is_equal() . . . . .	436
6.16.3.2 struct_compare() . . . . .	436
6.17 GiNaC::compare_bitwise< T > Class Template Reference . . . . .	437
6.17.1 Detailed Description . . . . .	437
6.17.2 Constructor & Destructor Documentation . . . . .	437
6.17.2.1 ~compare_bitwise() . . . . .	437
6.17.3 Member Function Documentation . . . . .	437
6.17.3.1 struct_is_equal() . . . . .	437
6.17.3.2 struct_compare() . . . . .	438

6.18 GiNaC::compare_std_less< T > Class Template Reference . . . . .	438
6.18.1 Detailed Description . . . . .	438
6.18.2 Constructor & Destructor Documentation . . . . .	438
6.18.2.1 ~compare_std_less() . . . . .	438
6.18.3 Member Function Documentation . . . . .	438
6.18.3.1 struct_is_equal() . . . . .	439
6.18.3.2 struct_compare() . . . . .	439
6.19 GiNaC::composition_generator Class Reference . . . . .	439
6.19.1 Detailed Description . . . . .	440
6.19.2 Constructor & Destructor Documentation . . . . .	440
6.19.2.1 composition_generator() . . . . .	440
6.19.3 Member Function Documentation . . . . .	440
6.19.3.1 get() . . . . .	440
6.19.3.2 next() . . . . .	441
6.19.4 Member Data Documentation . . . . .	441
6.19.4.1 cmgen . . . . .	441
6.19.4.2 atend . . . . .	441
6.19.4.3 trivial . . . . .	441
6.19.4.4 composition . . . . .	441
6.19.4.5 current_updated . . . . .	442
6.20 GiNaC::const_iterator Class Reference . . . . .	442
6.20.1 Member Typedef Documentation . . . . .	443
6.20.1.1 iterator_category . . . . .	443
6.20.1.2 value_type . . . . .	444
6.20.1.3 difference_type . . . . .	444
6.20.1.4 pointer . . . . .	444
6.20.1.5 reference . . . . .	444
6.20.2 Constructor & Destructor Documentation . . . . .	444
6.20.2.1 const_iterator() [1/2] . . . . .	444
6.20.2.2 const_iterator() [2/2] . . . . .	444
6.20.3 Member Function Documentation . . . . .	444
6.20.3.1 operator*() . . . . .	445
6.20.3.2 operator->() . . . . .	445
6.20.3.3 operator[]() . . . . .	445
6.20.3.4 operator++() [1/2] . . . . .	445
6.20.3.5 operator++() [2/2] . . . . .	445
6.20.3.6 operator+=() . . . . .	445
6.20.3.7 operator+() . . . . .	446
6.20.3.8 operator--() [1/2] . . . . .	446
6.20.3.9 operator--() [2/2] . . . . .	446
6.20.3.10 operator-=() . . . . .	446
6.20.3.11 operator-() . . . . .	446



6.20.3.12 operator==( )	446
6.20.3.13 operator!=( )	447
6.20.3.14 operator<( )	447
6.20.3.15 operator>( )	447
6.20.3.16 operator<=( )	447
6.20.3.17 operator>=( )	447
6.20.4 Friends And Related Function Documentation	447
6.20.4.1 ex	447
6.20.4.2 const_preorder_iterator	448
6.20.4.3 const_postorder_iterator	448
6.20.4.4 operator+	448
6.20.4.5 operator-	448
6.20.5 Member Data Documentation	448
6.20.5.1 e	448
6.20.5.2 i	448
6.21 GiNaC::const_postorder_iterator Class Reference	449
6.21.1 Member Typedef Documentation	449
6.21.1.1 iterator_category	449
6.21.1.2 value_type	449
6.21.1.3 difference_type	450
6.21.1.4 pointer	450
6.21.1.5 reference	450
6.21.2 Constructor & Destructor Documentation	450
6.21.2.1 const_postorder_iterator() [1/2]	450
6.21.2.2 const_postorder_iterator() [2/2]	450
6.21.3 Member Function Documentation	450
6.21.3.1 operator*( )	450
6.21.3.2 operator->( )	451
6.21.3.3 operator++( ) [1/2]	451
6.21.3.4 operator++( ) [2/2]	451
6.21.3.5 operator==( )	451
6.21.3.6 operator!=( )	451
6.21.3.7 descend( )	451
6.21.3.8 increment( )	452
6.21.4 Member Data Documentation	452
6.21.4.1 s	452
6.22 GiNaC::const_preorder_iterator Class Reference	452
6.22.1 Member Typedef Documentation	453
6.22.1.1 iterator_category	453
6.22.1.2 value_type	453
6.22.1.3 difference_type	453
6.22.1.4 pointer	453

6.22.1.5 reference	453
6.22.2 Constructor & Destructor Documentation	453
6.22.2.1 const_preorder_iterator() [1/2]	454
6.22.2.2 const_preorder_iterator() [2/2]	454
6.22.3 Member Function Documentation	454
6.22.3.1 operator*()	454
6.22.3.2 operator->()	454
6.22.3.3 operator++() [1/2]	454
6.22.3.4 operator++() [2/2]	454
6.22.3.5 operator==()	455
6.22.3.6 operator!=()	455
6.22.3.7 increment()	455
6.22.4 Member Data Documentation	455
6.22.4.1 s	455
6.23 GiNaC::constant Class Reference	456
6.23.1 Detailed Description	461
6.23.2 Constructor & Destructor Documentation	461
6.23.2.1 constant() [1/2]	461
6.23.2.2 constant() [2/2]	462
6.23.3 Member Function Documentation	462
6.23.3.1 info()	462
6.23.3.2 evalf()	462
6.23.3.3 is_polynomial()	463
6.23.3.4 conjugate()	463
6.23.3.5 real_part()	463
6.23.3.6 imag_part()	463
6.23.3.7 archive()	463
6.23.3.8 read_archive()	464
6.23.3.9 derivative()	464
6.23.3.10 is_equal_same_type()	464
6.23.3.11 calchash()	465
6.23.3.12 do_print()	465
6.23.3.13 do_print_tree()	465
6.23.3.14 do_print_latex()	465
6.23.3.15 do_print_python_repr()	465
6.23.4 Member Data Documentation	466
6.23.4.1 name	466
6.23.4.2 TeX_name	466
6.23.4.3 ef	466
6.23.4.4 number	466
6.23.4.5 serial	466
6.23.4.6 next_serial	467

6.23.4.7 domain	467
6.24 GiNaC::container< C > Class Template Reference	467
6.24.1 Detailed Description	473
6.24.2 Member Typedef Documentation	473
6.24.2.1 STLT	474
6.24.2.2 const_iterator	474
6.24.2.3 const_reverse_iterator	474
6.24.3 Constructor & Destructor Documentation	474
6.24.3.1 container() [1/4]	474
6.24.3.2 container() [2/4]	474
6.24.3.3 container() [3/4]	475
6.24.3.4 container() [4/4]	475
6.24.4 Member Function Documentation	475
6.24.4.1 get_default_flags()	475
6.24.4.2 get_open_delim()	475
6.24.4.3 get_close_delim()	475
6.24.4.4 info()	476
6.24.4.5 precedence()	476
6.24.4.6 nops()	476
6.24.4.7 op()	477
6.24.4.8 let_op()	477
6.24.4.9 subs()	477
6.24.4.10 read_archive()	478
6.24.4.11 archive()	478
6.24.4.12 conjugate()	478
6.24.4.13 real_part()	479
6.24.4.14 imag_part()	479
6.24.4.15 is_equal_same_type()	479
6.24.4.16 thiscontainer() [1/2]	480
6.24.4.17 thiscontainer() [2/2]	480
6.24.4.18 printseq()	480
6.24.4.19 sort_() [1/2]	480
6.24.4.20 sort_() [2/2]	481
6.24.4.21 unique_() [1/2]	481
6.24.4.22 prepend()	481
6.24.4.23 append()	481
6.24.4.24 remove_first()	481
6.24.4.25 remove_last()	482
6.24.4.26 remove_all()	482
6.24.4.27 sort()	482
6.24.4.28 unique()	482
6.24.4.29 begin()	482

6.24.4.30 end()	483
6.24.4.31 rbegin()	483
6.24.4.32 rend()	483
6.24.4.33 do_print()	483
6.24.4.34 do_print_tree()	483
6.24.4.35 do_print_python()	484
6.24.4.36 do_print_python_repr()	484
6.24.4.37 subschildren()	484
6.24.4.38 unique_() [2/2]	484
6.25 GiNaC::container_storage< C > Class Template Reference	485
6.25.1 Detailed Description	486
6.25.2 Member Typedef Documentation	486
6.25.2.1 STLT	486
6.25.3 Constructor & Destructor Documentation	486
6.25.3.1 container_storage() [1/4]	486
6.25.3.2 container_storage() [2/4]	486
6.25.3.3 container_storage() [3/4]	486
6.25.3.4 container_storage() [4/4]	487
6.25.3.5 ~container_storage()	487
6.25.4 Member Function Documentation	487
6.25.4.1 reserve() [1/4]	487
6.25.4.2 reserve() [2/4]	487
6.25.4.3 reserve() [3/4]	487
6.25.4.4 reserve() [4/4]	488
6.25.5 Member Data Documentation	488
6.25.5.1 seq	488
6.26 GiNaC::composition_generator::coolmulti Struct Reference	489
6.26.1 Constructor & Destructor Documentation	489
6.26.1.1 coolmulti()	489
6.26.1.2 ~coolmulti()	490
6.26.2 Member Function Documentation	490
6.26.2.1 next_permutation()	490
6.26.2.2 finished()	490
6.26.3 Member Data Documentation	490
6.26.3.1 head	490
6.26.3.2 i	490
6.26.3.3 after_i	491
6.27 GiNaC::derivative_map_function Struct Reference	491
6.27.1 Detailed Description	492
6.27.2 Constructor & Destructor Documentation	492
6.27.2.1 derivative_map_function()	492
6.27.3 Member Function Documentation	492

6.27.3.1 operator()	492
6.27.4 Member Data Documentation	493
6.27.4.1 s	493
6.28 GiNaC::determinant_algo Class Reference	493
6.28.1 Detailed Description	493
6.28.2 Member Enumeration Documentation	493
6.28.2.1 anonymous enum	494
6.29 GiNaC::diracgamma Class Reference	494
6.29.1 Detailed Description	500
6.29.2 Member Function Documentation	500
6.29.2.1 contract_with()	500
6.29.2.2 do_print()	500
6.29.2.3 do_print_latex()	500
6.30 GiNaC::diracgamma5 Class Reference	501
6.30.1 Detailed Description	505
6.30.2 Member Function Documentation	505
6.30.2.1 conjugate()	505
6.30.2.2 do_print()	505
6.30.2.3 do_print_latex()	506
6.31 GiNaC::diracgammaL Class Reference	506
6.31.1 Detailed Description	510
6.31.2 Member Function Documentation	510
6.31.2.1 conjugate()	511
6.31.2.2 do_print()	511
6.31.2.3 do_print_latex()	511
6.32 GiNaC::diracgammaR Class Reference	511
6.32.1 Detailed Description	515
6.32.2 Member Function Documentation	515
6.32.2.1 conjugate()	516
6.32.2.2 do_print()	516
6.32.2.3 do_print_latex()	516
6.33 GiNaC::diracone Class Reference	516
6.33.1 Detailed Description	520
6.33.2 Member Function Documentation	520
6.33.2.1 do_print()	520
6.33.2.2 do_print_latex()	521
6.34 GiNaC::do_taylor Class Reference	521
6.34.1 Detailed Description	521
6.35 GiNaC::domain Class Reference	521
6.35.1 Detailed Description	521
6.35.2 Member Enumeration Documentation	521
6.35.2.1 anonymous enum	521

6.36 GiNaC::dunno Class Reference	522
6.36.1 Detailed Description	522
6.37 GiNaC::Ebar_kernel Class Reference	522
6.37.1 Detailed Description	527
6.37.2 Constructor & Destructor Documentation	528
6.37.2.1 Ebar_kernel()	528
6.37.3 Member Function Documentation	528
6.37.3.1 nops()	528
6.37.3.2 op()	528
6.37.3.3 let_op()	528
6.37.3.4 is_numeric()	529
6.37.3.5 get_numerical_value()	529
6.37.3.6 series_coeff_impl()	529
6.37.3.7 do_print()	529
6.37.4 Member Data Documentation	530
6.37.4.1 n	530
6.37.4.2 m	530
6.37.4.3 x	530
6.37.4.4 y	530
6.38 GiNaC::Eisenstein_h_kernel Class Reference	531
6.38.1 Detailed Description	537
6.38.2 Constructor & Destructor Documentation	537
6.38.2.1 Eisenstein_h_kernel()	537
6.38.3 Member Function Documentation	537
6.38.3.1 series()	537
6.38.3.2 nops()	538
6.38.3.3 op()	538
6.38.3.4 let_op()	538
6.38.3.5 is_numeric()	538
6.38.3.6 Laurent_series()	539
6.38.3.7 get_numerical_value()	539
6.38.3.8 uses_Laurent_series()	539
6.38.3.9 coefficient_a0()	539
6.38.3.10 coefficient_an()	540
6.38.3.11 q_expansion_modular_form()	540
6.38.3.12 do_print()	540
6.38.4 Member Data Documentation	540
6.38.4.1 k	540
6.38.4.2 N	541
6.38.4.3 r	541
6.38.4.4 s	541
6.38.4.5 C_norm	541

6.39 GiNaC::Eisenstein_kernel Class Reference	542
6.39.1 Detailed Description	548
6.39.2 Constructor & Destructor Documentation	548
6.39.2.1 Eisenstein_kernel()	548
6.39.3 Member Function Documentation	548
6.39.3.1 series()	549
6.39.3.2 nops()	549
6.39.3.3 op()	549
6.39.3.4 let_op()	549
6.39.3.5 is_numeric()	550
6.39.3.6 Laurent_series()	550
6.39.3.7 get_numerical_value()	550
6.39.3.8 uses_Laurent_series()	550
6.39.3.9 q_expansion_modular_form()	551
6.39.3.10 do_print()	551
6.39.4 Member Data Documentation	551
6.39.4.1 k	551
6.39.4.2 N	551
6.39.4.3 a	551
6.39.4.4 b	552
6.39.4.5 K	552
6.39.4.6 C_norm	552
6.40 GiNaC::composition_generator::coolmulti::element Struct Reference	552
6.40.1 Constructor & Destructor Documentation	553
6.40.1.1 element()	553
6.40.1.2 ~element()	553
6.40.2 Member Data Documentation	553
6.40.2.1 value	553
6.40.2.2 next	553
6.41 GiNaC::ELi_kernel Class Reference	554
6.41.1 Detailed Description	559
6.41.2 Constructor & Destructor Documentation	560
6.41.2.1 ELi_kernel()	560
6.41.3 Member Function Documentation	560
6.41.3.1 nops()	560
6.41.3.2 op()	560
6.41.3.3 let_op()	560
6.41.3.4 is_numeric()	561
6.41.3.5 get_numerical_value()	561
6.41.3.6 series_coeff_impl()	561
6.41.3.7 do_print()	561
6.41.4 Member Data Documentation	562

6.41.4.1 n	562
6.41.4.2 m	562
6.41.4.3 x	562
6.41.4.4 y	562
6.42 std::equal_to< GiNaC::ex > Struct Reference	562
6.42.1 Detailed Description	563
6.42.2 Member Function Documentation	563
6.42.2.1 operator()	563
6.43 GiNaC::error_and_integral Struct Reference	564
6.43.1 Constructor & Destructor Documentation	564
6.43.1.1 error_and_integral()	565
6.43.2 Member Data Documentation	565
6.43.2.1 error	565
6.43.2.2 integral	565
6.44 GiNaC::error_and_integral_is_less Struct Reference	565
6.44.1 Member Function Documentation	565
6.44.1.1 operator()	565
6.45 GiNaC::eval_integ_map_function Struct Reference	566
6.45.1 Detailed Description	567
6.45.2 Member Function Documentation	567
6.45.2.1 operator()	567
6.46 GiNaC::evalf_map_function Struct Reference	567
6.46.1 Detailed Description	568
6.46.2 Member Function Documentation	568
6.46.2.1 operator()	568
6.47 GiNaC::evalm_map_function Struct Reference	569
6.47.1 Detailed Description	570
6.47.2 Member Function Documentation	570
6.47.2.1 operator()	570
6.48 GiNaC::ex Class Reference	570
6.48.1 Detailed Description	574
6.48.2 Constructor & Destructor Documentation	574
6.48.2.1 ex() [1/10]	574
6.48.2.2 ex() [2/10]	574
6.48.2.3 ex() [3/10]	575
6.48.2.4 ex() [4/10]	575
6.48.2.5 ex() [5/10]	575
6.48.2.6 ex() [6/10]	575
6.48.2.7 ex() [7/10]	575
6.48.2.8 ex() [8/10]	575
6.48.2.9 ex() [9/10]	576
6.48.2.10 ex() [10/10]	576



6.48.3 Member Function Documentation	576
6.48.3.1 swap()	576
6.48.3.2 begin()	576
6.48.3.3 end()	577
6.48.3.4 preorder_begin()	577
6.48.3.5 preorder_end()	577
6.48.3.6 postorder_begin()	577
6.48.3.7 postorder_end()	577
6.48.3.8 eval()	577
6.48.3.9 evalf()	578
6.48.3.10 evalm()	578
6.48.3.11 eval_ncmul()	578
6.48.3.12 eval_integ()	578
6.48.3.13 print()	579
6.48.3.14 dbgprint()	579
6.48.3.15 dbgprinttree()	579
6.48.3.16 info()	580
6.48.3.17 nops()	580
6.48.3.18 op()	581
6.48.3.19 operator[]() [1/4]	581
6.48.3.20 operator[]() [2/4]	581
6.48.3.21 let_op()	581
6.48.3.22 operator[]() [3/4]	582
6.48.3.23 operator[]() [4/4]	582
6.48.3.24 lhs()	582
6.48.3.25 rhs()	582
6.48.3.26 conjugate()	582
6.48.3.27 real_part()	583
6.48.3.28 imag_part()	583
6.48.3.29 has()	583
6.48.3.30 find()	583
6.48.3.31 match() [1/2]	584
6.48.3.32 match() [2/2]	584
6.48.3.33 subs() [1/3]	584
6.48.3.34 subs() [2/3]	585
6.48.3.35 subs() [3/3]	585
6.48.3.36 map() [1/2]	585
6.48.3.37 map() [2/2]	585
6.48.3.38 accept()	586
6.48.3.39 traverse_preorder()	586
6.48.3.40 traverse_postorder()	586
6.48.3.41 traverse()	586

6.48.3.42 <code>is_polynomial()</code>	587
6.48.3.43 <code>degree()</code>	587
6.48.3.44 <code>ldegree()</code>	587
6.48.3.45 <code>coeff()</code>	587
6.48.3.46 <code>lcoeff()</code>	588
6.48.3.47 <code>tcoeff()</code>	588
6.48.3.48 <code>expand()</code>	588
6.48.3.49 <code>collect()</code>	588
6.48.3.50 <code>diff()</code>	588
6.48.3.51 <code>series()</code>	589
6.48.3.52 <code>normal()</code>	590
6.48.3.53 <code>to_rational()</code>	590
6.48.3.54 <code>to_polynomial()</code>	591
6.48.3.55 <code>numer()</code>	591
6.48.3.56 <code>denom()</code>	591
6.48.3.57 <code>numer_denom()</code>	592
6.48.3.58 <code>unit()</code>	592
6.48.3.59 <code>content()</code>	593
6.48.3.60 <code>integer_content()</code>	593
6.48.3.61 <code>primpart()</code> [1/2]	593
6.48.3.62 <code>primpart()</code> [2/2]	594
6.48.3.63 <code>unitcontprim()</code>	594
6.48.3.64 <code>smod()</code>	595
6.48.3.65 <code>max_coefficient()</code>	595
6.48.3.66 <code>get_free_indices()</code>	596
6.48.3.67 <code>simplify_indexed()</code> [1/2]	596
6.48.3.68 <code>simplify_indexed()</code> [2/2]	596
6.48.3.69 <code>compare()</code>	597
6.48.3.70 <code>is_equal()</code>	597
6.48.3.71 <code>is_zero()</code>	598
6.48.3.72 <code>is_zero_matrix()</code>	598
6.48.3.73 <code>symmetrize()</code> [1/2]	598
6.48.3.74 <code>symmetrize()</code> [2/2]	598
6.48.3.75 <code>antisymmetrize()</code> [1/2]	599
6.48.3.76 <code>antisymmetrize()</code> [2/2]	599
6.48.3.77 <code>symmetrize_cyclic()</code> [1/2]	599
6.48.3.78 <code>symmetrize_cyclic()</code> [2/2]	599
6.48.3.79 <code>return_type()</code>	600
6.48.3.80 <code>return_type_tinfo()</code>	600
6.48.3.81 <code>gethash()</code>	600
6.48.3.82 <code>construct_from_basic()</code>	600
6.48.3.83 <code>construct_from_int()</code>	601

6.48.3.84 construct_from_uint()	601
6.48.3.85 construct_from_long()	601
6.48.3.86 construct_from_ulong()	601
6.48.3.87 construct_from_longlong()	602
6.48.3.88 construct_from_ulonglong()	602
6.48.3.89 construct_from_double()	602
6.48.3.90 construct_from_string_and_lst()	602
6.48.3.91 makewriteable()	602
6.48.3.92 share()	603
6.48.4 Friends And Related Function Documentation	603
6.48.4.1 archive_node	603
6.48.4.2 are_ex_trivially_equal	603
6.48.4.3 ex_to	603
6.48.4.4 is_a	604
6.48.4.5 is_exactly_a	604
6.48.5 Member Data Documentation	604
6.48.5.1 bp	604
6.49 GiNaC::ex_base_is_less Struct Reference	605
6.49.1 Member Function Documentation	605
6.49.1.1 operator()()	605
6.50 GiNaC::ex_is_equal Struct Reference	605
6.50.1 Member Function Documentation	605
6.50.1.1 operator()()	605
6.51 GiNaC::ex_is_less Struct Reference	606
6.51.1 Member Function Documentation	606
6.51.1.1 operator()()	606
6.52 GiNaC::ex_swap Struct Reference	606
6.52.1 Member Function Documentation	606
6.52.1.1 operator()()	606
6.53 GiNaC::expair Class Reference	607
6.53.1 Detailed Description	608
6.53.2 Constructor & Destructor Documentation	608
6.53.2.1 expair() [1/2]	608
6.53.2.2 expair() [2/2]	608
6.53.3 Member Function Documentation	608
6.53.3.1 is_equal()	608
6.53.3.2 is_less()	609
6.53.3.3 compare()	609
6.53.3.4 print()	609
6.53.3.5 is_canonical_numeric()	609
6.53.3.6 swap()	609
6.53.3.7 conjugate()	610

6.53.4 Member Data Documentation	610
6.53.4.1 rest	610
6.53.4.2 coeff	610
6.54 GiNaC::expair_is_less Struct Reference	610
6.54.1 Detailed Description	611
6.54.2 Member Function Documentation	611
6.54.2.1 operator()	611
6.55 GiNaC::expair_rest_is_less Struct Reference	611
6.55.1 Detailed Description	611
6.55.2 Member Function Documentation	611
6.55.2.1 operator()	612
6.56 GiNaC::expair_swap Struct Reference	612
6.56.1 Member Function Documentation	612
6.56.1.1 operator()	612
6.57 GiNaC::expairseq Class Reference	613
6.57.1 Detailed Description	619
6.57.2 Constructor & Destructor Documentation	619
6.57.2.1 expairseq() [1/4]	619
6.57.2.2 expairseq() [2/4]	619
6.57.2.3 expairseq() [3/4]	620
6.57.2.4 expairseq() [4/4]	620
6.57.3 Member Function Documentation	620
6.57.3.1 precedence()	620
6.57.3.2 info()	620
6.57.3.3 nops()	621
6.57.3.4 op()	621
6.57.3.5 map()	621
6.57.3.6 eval()	622
6.57.3.7 to_rational()	622
6.57.3.8 to_polynomial()	622
6.57.3.9 match()	623
6.57.3.10 subs()	623
6.57.3.11 conjugate()	623
6.57.3.12 archive()	624
6.57.3.13 read_archive()	624
6.57.3.14 is_equal_same_type()	624
6.57.3.15 return_type()	625
6.57.3.16 calchash()	625
6.57.3.17 expand()	625
6.57.3.18 thisexpairseq() [1/2]	626
6.57.3.19 thisexpairseq() [2/2]	626
6.57.3.20 printseq()	626

6.57.3.21 printpair()	626
6.57.3.22 split_ex_to_pair()	627
6.57.3.23 combine_ex_with_coeff_to_pair()	627
6.57.3.24 combine_pair_with_coeff_to_pair()	627
6.57.3.25 recombine_pair_to_ex()	628
6.57.3.26 expair_needs_further_processing()	628
6.57.3.27 default_overall_coeff()	628
6.57.3.28 combine_overall_coeff() [1/2]	628
6.57.3.29 combine_overall_coeff() [2/2]	629
6.57.3.30 can_make_flat()	629
6.57.3.31 do_print()	629
6.57.3.32 do_print_tree()	629
6.57.3.33 construct_from_2_ex()	629
6.57.3.34 construct_from_2_expairseq()	630
6.57.3.35 construct_from_expairseq_ex()	630
6.57.3.36 construct_from_exvector()	630
6.57.3.37 construct_from_epvector() [1/2]	630
6.57.3.38 construct_from_epvector() [2/2]	631
6.57.3.39 make_flat() [1/2]	631
6.57.3.40 make_flat() [2/2]	631
6.57.3.41 canonicalize()	631
6.57.3.42 combine_same_terms_sorted_seq()	632
6.57.3.43 is_canonical()	632
6.57.3.44 expandchildren()	632
6.57.3.45 evalchildren()	633
6.57.3.46 subschildren()	633
6.57.4 Member Data Documentation	633
6.57.4.1 seq	634
6.57.4.2 overall_coeff	634
6.58 GiNaC::expand_map_function Struct Reference	634
6.58.1 Detailed Description	636
6.58.2 Constructor & Destructor Documentation	636
6.58.2.1 expand_map_function()	636
6.58.3 Member Function Documentation	636
6.58.3.1 operator()()	636
6.58.4 Member Data Documentation	636
6.58.4.1 options	636
6.59 GiNaC::expand_options Class Reference	636
6.59.1 Detailed Description	637
6.59.2 Member Enumeration Documentation	637
6.59.2.1 anonymous enum	637
6.60 GiNaC::factor_options Class Reference	637

6.60.1 Detailed Description	637
6.60.2 Member Enumeration Documentation	638
6.60.2.1 anonymous enum	638
6.61 GiNaC::fail Class Reference	638
6.61.1 Member Function Documentation	642
6.61.1.1 return_type()	642
6.61.1.2 do_print()	642
6.62 GiNaC::fderivative Class Reference	643
6.62.1 Detailed Description	652
6.62.2 Constructor & Destructor Documentation	652
6.62.2.1 fderivative() [1/3]	652
6.62.2.2 fderivative() [2/3]	653
6.62.2.3 fderivative() [3/3]	653
6.62.3 Member Function Documentation	653
6.62.3.1 print()	653
6.62.3.2 eval()	654
6.62.3.3 series()	654
6.62.3.4 thiscontainer() [1/2]	654
6.62.3.5 thiscontainer() [2/2]	654
6.62.3.6 archive()	655
6.62.3.7 read_archive()	655
6.62.3.8 derivative()	655
6.62.3.9 is_equal_same_type()	656
6.62.3.10 match_same_type()	656
6.62.3.11 derivatives()	656
6.62.3.12 do_print()	657
6.62.3.13 do_print_latex()	657
6.62.3.14 do_print_csrc()	657
6.62.3.15 do_print_tree()	657
6.62.4 Member Data Documentation	657
6.62.4.1 parameter_set	658
6.63 GiNaC::function Class Reference	658
6.63.1 Detailed Description	666
6.63.2 Constructor & Destructor Documentation	666
6.63.2.1 function() [1/18]	666
6.63.2.2 function() [2/18]	667
6.63.2.3 function() [3/18]	667
6.63.2.4 function() [4/18]	667
6.63.2.5 function() [5/18]	667
6.63.2.6 function() [6/18]	667
6.63.2.7 function() [7/18]	668
6.63.2.8 function() [8/18]	668

6.63.2.9 function() [9/18]	668
6.63.2.10 function() [10/18]	668
6.63.2.11 function() [11/18]	669
6.63.2.12 function() [12/18]	669
6.63.2.13 function() [13/18]	669
6.63.2.14 function() [14/18]	670
6.63.2.15 function() [15/18]	670
6.63.2.16 function() [16/18]	670
6.63.2.17 function() [17/18]	671
6.63.2.18 function() [18/18]	671
6.63.3 Member Function Documentation	671
6.63.3.1 print()	671
6.63.3.2 precedence()	671
6.63.3.3 expand()	672
6.63.3.4 eval()	672
6.63.3.5 evalf()	672
6.63.3.6 eval_ncmul()	673
6.63.3.7 calchash()	673
6.63.3.8 series()	673
6.63.3.9 thiscontainer() [1/2]	674
6.63.3.10 thiscontainer() [2/2]	674
6.63.3.11 conjugate()	674
6.63.3.12 real_part()	674
6.63.3.13 imag_part()	675
6.63.3.14 archive()	675
6.63.3.15 read_archive()	675
6.63.3.16 info()	675
6.63.3.17 derivative()	676
6.63.3.18 is_equal_same_type()	676
6.63.3.19 match_same_type()	676
6.63.3.20 return_type()	677
6.63.3.21 return_type_tinfo()	677
6.63.3.22 pderivative()	677
6.63.3.23 expl_derivative()	677
6.63.3.24 registered_functions()	678
6.63.3.25 lookup_remember_table()	678
6.63.3.26 store_remember_table()	678
6.63.3.27 power()	678
6.63.3.28 register_new()	678
6.63.3.29 find_function()	679
6.63.3.30 get_registered_functions()	679
6.63.3.31 get_serial()	679

6.63.3.32 <code>get_name()</code>	679
6.63.4 Friends And Related Function Documentation	679
6.63.4.1 <code>remember_table_entry</code>	679
6.63.5 Member Data Documentation	680
6.63.5.1 <code>current_serial</code>	680
6.63.5.2 <code>serial</code>	680
6.64 GiNaC::function_options Class Reference	680
6.64.1 Constructor & Destructor Documentation	686
6.64.1.1 <code>function_options()</code> [1/3]	686
6.64.1.2 <code>function_options()</code> [2/3]	686
6.64.1.3 <code>function_options()</code> [3/3]	687
6.64.1.4 <code>~function_options()</code>	687
6.64.2 Member Function Documentation	687
6.64.2.1 <code>initialize()</code>	687
6.64.2.2 <code>dummy()</code>	687
6.64.2.3 <code>set_name()</code>	687
6.64.2.4 <code>latex_name()</code>	688
6.64.2.5 <code>eval_func()</code> [1/15]	688
6.64.2.6 <code>eval_func()</code> [2/15]	688
6.64.2.7 <code>eval_func()</code> [3/15]	688
6.64.2.8 <code>eval_func()</code> [4/15]	688
6.64.2.9 <code>eval_func()</code> [5/15]	688
6.64.2.10 <code>eval_func()</code> [6/15]	689
6.64.2.11 <code>eval_func()</code> [7/15]	689
6.64.2.12 <code>eval_func()</code> [8/15]	689
6.64.2.13 <code>eval_func()</code> [9/15]	689
6.64.2.14 <code>eval_func()</code> [10/15]	689
6.64.2.15 <code>eval_func()</code> [11/15]	689
6.64.2.16 <code>eval_func()</code> [12/15]	690
6.64.2.17 <code>eval_func()</code> [13/15]	690
6.64.2.18 <code>eval_func()</code> [14/15]	690
6.64.2.19 <code>evalf_func()</code> [1/15]	690
6.64.2.20 <code>evalf_func()</code> [2/15]	690
6.64.2.21 <code>evalf_func()</code> [3/15]	690
6.64.2.22 <code>evalf_func()</code> [4/15]	691
6.64.2.23 <code>evalf_func()</code> [5/15]	691
6.64.2.24 <code>evalf_func()</code> [6/15]	691
6.64.2.25 <code>evalf_func()</code> [7/15]	691
6.64.2.26 <code>evalf_func()</code> [8/15]	691
6.64.2.27 <code>evalf_func()</code> [9/15]	691
6.64.2.28 <code>evalf_func()</code> [10/15]	692
6.64.2.29 <code>evalf_func()</code> [11/15]	692



6.64.2.30 evalf_func() [12/15]	692
6.64.2.31 evalf_func() [13/15]	692
6.64.2.32 evalf_func() [14/15]	692
6.64.2.33 conjugate_func() [1/15]	692
6.64.2.34 conjugate_func() [2/15]	693
6.64.2.35 conjugate_func() [3/15]	693
6.64.2.36 conjugate_func() [4/15]	693
6.64.2.37 conjugate_func() [5/15]	693
6.64.2.38 conjugate_func() [6/15]	693
6.64.2.39 conjugate_func() [7/15]	693
6.64.2.40 conjugate_func() [8/15]	694
6.64.2.41 conjugate_func() [9/15]	694
6.64.2.42 conjugate_func() [10/15]	694
6.64.2.43 conjugate_func() [11/15]	694
6.64.2.44 conjugate_func() [12/15]	694
6.64.2.45 conjugate_func() [13/15]	694
6.64.2.46 conjugate_func() [14/15]	695
6.64.2.47 real_part_func() [1/15]	695
6.64.2.48 real_part_func() [2/15]	695
6.64.2.49 real_part_func() [3/15]	695
6.64.2.50 real_part_func() [4/15]	695
6.64.2.51 real_part_func() [5/15]	695
6.64.2.52 real_part_func() [6/15]	696
6.64.2.53 real_part_func() [7/15]	696
6.64.2.54 real_part_func() [8/15]	696
6.64.2.55 real_part_func() [9/15]	696
6.64.2.56 real_part_func() [10/15]	696
6.64.2.57 real_part_func() [11/15]	696
6.64.2.58 real_part_func() [12/15]	697
6.64.2.59 real_part_func() [13/15]	697
6.64.2.60 real_part_func() [14/15]	697
6.64.2.61 imag_part_func() [1/15]	697
6.64.2.62 imag_part_func() [2/15]	697
6.64.2.63 imag_part_func() [3/15]	697
6.64.2.64 imag_part_func() [4/15]	698
6.64.2.65 imag_part_func() [5/15]	698
6.64.2.66 imag_part_func() [6/15]	698
6.64.2.67 imag_part_func() [7/15]	698
6.64.2.68 imag_part_func() [8/15]	698
6.64.2.69 imag_part_func() [9/15]	698
6.64.2.70 imag_part_func() [10/15]	699
6.64.2.71 imag_part_func() [11/15]	699

6.64.2.72 <a href="#">imag_part_func()</a> [12/15]	699
6.64.2.73 <a href="#">imag_part_func()</a> [13/15]	699
6.64.2.74 <a href="#">imag_part_func()</a> [14/15]	699
6.64.2.75 <a href="#">expand_func()</a> [1/15]	699
6.64.2.76 <a href="#">expand_func()</a> [2/15]	700
6.64.2.77 <a href="#">expand_func()</a> [3/15]	700
6.64.2.78 <a href="#">expand_func()</a> [4/15]	700
6.64.2.79 <a href="#">expand_func()</a> [5/15]	700
6.64.2.80 <a href="#">expand_func()</a> [6/15]	700
6.64.2.81 <a href="#">expand_func()</a> [7/15]	700
6.64.2.82 <a href="#">expand_func()</a> [8/15]	701
6.64.2.83 <a href="#">expand_func()</a> [9/15]	701
6.64.2.84 <a href="#">expand_func()</a> [10/15]	701
6.64.2.85 <a href="#">expand_func()</a> [11/15]	701
6.64.2.86 <a href="#">expand_func()</a> [12/15]	701
6.64.2.87 <a href="#">expand_func()</a> [13/15]	701
6.64.2.88 <a href="#">expand_func()</a> [14/15]	702
6.64.2.89 <a href="#">derivative_func()</a> [1/15]	702
6.64.2.90 <a href="#">derivative_func()</a> [2/15]	702
6.64.2.91 <a href="#">derivative_func()</a> [3/15]	702
6.64.2.92 <a href="#">derivative_func()</a> [4/15]	702
6.64.2.93 <a href="#">derivative_func()</a> [5/15]	702
6.64.2.94 <a href="#">derivative_func()</a> [6/15]	703
6.64.2.95 <a href="#">derivative_func()</a> [7/15]	703
6.64.2.96 <a href="#">derivative_func()</a> [8/15]	703
6.64.2.97 <a href="#">derivative_func()</a> [9/15]	703
6.64.2.98 <a href="#">derivative_func()</a> [10/15]	703
6.64.2.99 <a href="#">derivative_func()</a> [11/15]	703
6.64.2.100 <a href="#">derivative_func()</a> [12/15]	704
6.64.2.101 <a href="#">derivative_func()</a> [13/15]	704
6.64.2.102 <a href="#">derivative_func()</a> [14/15]	704
6.64.2.103 <a href="#">expl_derivative_func()</a> [1/15]	704
6.64.2.104 <a href="#">expl_derivative_func()</a> [2/15]	704
6.64.2.105 <a href="#">expl_derivative_func()</a> [3/15]	704
6.64.2.106 <a href="#">expl_derivative_func()</a> [4/15]	705
6.64.2.107 <a href="#">expl_derivative_func()</a> [5/15]	705
6.64.2.108 <a href="#">expl_derivative_func()</a> [6/15]	705
6.64.2.109 <a href="#">expl_derivative_func()</a> [7/15]	705
6.64.2.110 <a href="#">expl_derivative_func()</a> [8/15]	705
6.64.2.111 <a href="#">expl_derivative_func()</a> [9/15]	705
6.64.2.112 <a href="#">expl_derivative_func()</a> [10/15]	706
6.64.2.113 <a href="#">expl_derivative_func()</a> [11/15]	706

6.64.2.114 expl_derivative_func() [12/15]	706
6.64.2.115 expl_derivative_func() [13/15]	706
6.64.2.116 expl_derivative_func() [14/15]	706
6.64.2.117 power_func() [1/15]	706
6.64.2.118 power_func() [2/15]	707
6.64.2.119 power_func() [3/15]	707
6.64.2.120 power_func() [4/15]	707
6.64.2.121 power_func() [5/15]	707
6.64.2.122 power_func() [6/15]	707
6.64.2.123 power_func() [7/15]	707
6.64.2.124 power_func() [8/15]	708
6.64.2.125 power_func() [9/15]	708
6.64.2.126 power_func() [10/15]	708
6.64.2.127 power_func() [11/15]	708
6.64.2.128 power_func() [12/15]	708
6.64.2.129 power_func() [13/15]	708
6.64.2.130 power_func() [14/15]	709
6.64.2.131 series_func() [1/15]	709
6.64.2.132 series_func() [2/15]	709
6.64.2.133 series_func() [3/15]	709
6.64.2.134 series_func() [4/15]	709
6.64.2.135 series_func() [5/15]	709
6.64.2.136 series_func() [6/15]	710
6.64.2.137 series_func() [7/15]	710
6.64.2.138 series_func() [8/15]	710
6.64.2.139 series_func() [9/15]	710
6.64.2.140 series_func() [10/15]	710
6.64.2.141 series_func() [11/15]	710
6.64.2.142 series_func() [12/15]	711
6.64.2.143 series_func() [13/15]	711
6.64.2.144 series_func() [14/15]	711
6.64.2.145 info_func() [1/15]	711
6.64.2.146 info_func() [2/15]	711
6.64.2.147 info_func() [3/15]	711
6.64.2.148 info_func() [4/15]	712
6.64.2.149 info_func() [5/15]	712
6.64.2.150 info_func() [6/15]	712
6.64.2.151 info_func() [7/15]	712
6.64.2.152 info_func() [8/15]	712
6.64.2.153 info_func() [9/15]	712
6.64.2.154 info_func() [10/15]	713
6.64.2.155 info_func() [11/15]	713

6.64.2.156 info_func() [12/15]	713
6.64.2.157 info_func() [13/15]	713
6.64.2.158 info_func() [14/15]	713
6.64.2.159 eval_func() [15/15]	713
6.64.2.160 evalf_func() [15/15]	714
6.64.2.161 conjugate_func() [15/15]	714
6.64.2.162 real_part_func() [15/15]	714
6.64.2.163 imag_part_func() [15/15]	714
6.64.2.164 expand_func() [15/15]	714
6.64.2.165 derivative_func() [15/15]	714
6.64.2.166 expl_derivative_func() [15/15]	715
6.64.2.167 power_func() [15/15]	715
6.64.2.168 series_func() [15/15]	715
6.64.2.169 info_func() [15/15]	715
6.64.2.170 print_func() [1/15]	715
6.64.2.171 print_func() [2/15]	716
6.64.2.172 print_func() [3/15]	716
6.64.2.173 print_func() [4/15]	716
6.64.2.174 print_func() [5/15]	716
6.64.2.175 print_func() [6/15]	716
6.64.2.176 print_func() [7/15]	717
6.64.2.177 print_func() [8/15]	717
6.64.2.178 print_func() [9/15]	717
6.64.2.179 print_func() [10/15]	717
6.64.2.180 print_func() [11/15]	717
6.64.2.181 print_func() [12/15]	718
6.64.2.182 print_func() [13/15]	718
6.64.2.183 print_func() [14/15]	718
6.64.2.184 print_func() [15/15]	718
6.64.2.185 set_return_type()	718
6.64.2.186 do_not_evalf_params()	719
6.64.2.187 remember()	719
6.64.2.188 overloaded()	719
6.64.2.189 set_symmetry()	719
6.64.2.190 get_name()	719
6.64.2.191 get_nparams()	719
6.64.2.192 has_derivative()	720
6.64.2.193 has_power()	720
6.64.2.194 test_and_set_nparams()	720
6.64.2.195 set_print_func()	720
6.64.3 Friends And Related Function Documentation	720
6.64.3.1 function	720

6.64.3.2 fderivative . . . . .	721
6.64.4 Member Data Documentation . . . . .	721
6.64.4.1 name . . . . .	721
6.64.4.2 TeX_name . . . . .	721
6.64.4.3 nparams . . . . .	721
6.64.4.4 eval_f . . . . .	721
6.64.4.5 evalf_f . . . . .	721
6.64.4.6 conjugate_f . . . . .	722
6.64.4.7 real_part_f . . . . .	722
6.64.4.8 imag_part_f . . . . .	722
6.64.4.9 expand_f . . . . .	722
6.64.4.10 derivative_f . . . . .	722
6.64.4.11 expl_derivative_f . . . . .	722
6.64.4.12 power_f . . . . .	723
6.64.4.13 series_f . . . . .	723
6.64.4.14 print_dispatch_table . . . . .	723
6.64.4.15 info_f . . . . .	723
6.64.4.16 evalf_params_first . . . . .	723
6.64.4.17 use_return_type . . . . .	723
6.64.4.18 return_type . . . . .	724
6.64.4.19 return_type_tinfo . . . . .	724
6.64.4.20 use_remember . . . . .	724
6.64.4.21 remember_size . . . . .	724
6.64.4.22 remember_assoc_size . . . . .	724
6.64.4.23 remember_strategy . . . . .	724
6.64.4.24 eval_use_exvector_args . . . . .	725
6.64.4.25 evalf_use_exvector_args . . . . .	725
6.64.4.26 conjugate_use_exvector_args . . . . .	725
6.64.4.27 real_part_use_exvector_args . . . . .	725
6.64.4.28 imag_part_use_exvector_args . . . . .	725
6.64.4.29 expand_use_exvector_args . . . . .	725
6.64.4.30 derivative_use_exvector_args . . . . .	726
6.64.4.31 expl_derivative_use_exvector_args . . . . .	726
6.64.4.32 power_use_exvector_args . . . . .	726
6.64.4.33 series_use_exvector_args . . . . .	726
6.64.4.34 print_use_exvector_args . . . . .	726
6.64.4.35 info_use_exvector_args . . . . .	726
6.64.4.36 functions_with_same_name . . . . .	727
6.64.4.37 symtree . . . . .	727
6.65 GiNaC::G2_SERIAL Class Reference . . . . .	727
6.65.1 Detailed Description . . . . .	727
6.65.2 Member Data Documentation . . . . .	727

6.65.2.1 serial . . . . .	727
6.66 GiNaC::G3_SERIAL Class Reference . . . . .	728
6.66.1 Detailed Description . . . . .	728
6.66.2 Member Data Documentation . . . . .	728
6.66.2.1 serial . . . . .	728
6.67 GiNaC::gcd_options Struct Reference . . . . .	728
6.67.1 Detailed Description . . . . .	728
6.67.2 Member Enumeration Documentation . . . . .	728
6.67.2.1 anonymous enum . . . . .	728
6.68 GiNaC::gcdheu_failed Class Reference . . . . .	729
6.68.1 Detailed Description . . . . .	729
6.69 GiNaC::has_distance< T > Class Template Reference . . . . .	729
6.69.1 Detailed Description . . . . .	730
6.69.2 Member Typedef Documentation . . . . .	730
6.69.2.1 yes_type . . . . .	730
6.69.2.2 no_type . . . . .	730
6.69.3 Member Enumeration Documentation . . . . .	730
6.69.3.1 anonymous enum . . . . .	730
6.69.4 Member Function Documentation . . . . .	731
6.69.4.1 test() [1/2] . . . . .	731
6.69.4.2 test() [2/2] . . . . .	731
6.70 GiNaC::has_options Class Reference . . . . .	731
6.70.1 Detailed Description . . . . .	731
6.70.2 Member Enumeration Documentation . . . . .	731
6.70.2.1 anonymous enum . . . . .	731
6.71 std::hash< GiNaC::ex > Struct Reference . . . . .	732
6.71.1 Detailed Description . . . . .	732
6.71.2 Member Function Documentation . . . . .	732
6.71.2.1 operator()() . . . . .	732
6.72 GiNaC::idx Class Reference . . . . .	733
6.72.1 Detailed Description . . . . .	738
6.72.2 Constructor & Destructor Documentation . . . . .	738
6.72.2.1 idx() . . . . .	738
6.72.3 Member Function Documentation . . . . .	739
6.72.3.1 info() . . . . .	739
6.72.3.2 nops() . . . . .	739
6.72.3.3 op() . . . . .	739
6.72.3.4 map() . . . . .	740
6.72.3.5 evalf() . . . . .	740
6.72.3.6 subs() . . . . .	740
6.72.3.7 archive() . . . . .	740
6.72.3.8 read_archive() . . . . .	741

6.72.3.9 derivative()	741
6.72.3.10 match_same_type()	741
6.72.3.11 calchash()	742
6.72.3.12 is_dummy_pair_same_type()	742
6.72.3.13 get_value()	742
6.72.3.14 is_numeric()	742
6.72.3.15 is_symbolic()	743
6.72.3.16 get_dim()	743
6.72.3.17 is_dim_numeric()	743
6.72.3.18 is_dim_symbolic()	743
6.72.3.19 replace_dim()	744
6.72.3.20 minimal_dim()	744
6.72.3.21 print_index()	744
6.72.3.22 do_print()	744
6.72.3.23 do_print_csrc()	745
6.72.3.24 do_print_latex()	745
6.72.3.25 do_print_tree()	745
6.72.4 Member Data Documentation	745
6.72.4.1 value	745
6.72.4.2 dim	746
6.73 GiNaC::idx_is_equal_ignore_dim Struct Reference	746
6.73.1 Member Function Documentation	746
6.73.1.1 operator>()	746
6.74 GiNaC::indexed Class Reference	747
6.74.1 Detailed Description	755
6.74.2 Constructor & Destructor Documentation	755
6.74.2.1 indexed() [1/13]	755
6.74.2.2 indexed() [2/13]	755
6.74.2.3 indexed() [3/13]	756
6.74.2.4 indexed() [4/13]	756
6.74.2.5 indexed() [5/13]	757
6.74.2.6 indexed() [6/13]	757
6.74.2.7 indexed() [7/13]	759
6.74.2.8 indexed() [8/13]	759
6.74.2.9 indexed() [9/13]	760
6.74.2.10 indexed() [10/13]	761
6.74.2.11 indexed() [11/13]	761
6.74.2.12 indexed() [12/13]	761
6.74.2.13 indexed() [13/13]	761
6.74.3 Member Function Documentation	761
6.74.3.1 precedence()	762
6.74.3.2 info()	762

6.74.3.3 eval()	762
6.74.3.4 real_part()	762
6.74.3.5 imag_part()	763
6.74.3.6 get_free_indices()	763
6.74.3.7 archive()	763
6.74.3.8 read_archive()	763
6.74.3.9 derivative()	764
6.74.3.10 thiscontainer() [1/2]	764
6.74.3.11 thiscontainer() [2/2]	764
6.74.3.12 return_type()	764
6.74.3.13 return_type_tinfo()	765
6.74.3.14 expand()	765
6.74.3.15 all_index_values_are()	765
6.74.3.16 get_indices()	765
6.74.3.17 get_dummy_indices() [1/2]	766
6.74.3.18 get_dummy_indices() [2/2]	766
6.74.3.19 has_dummy_index_for()	766
6.74.3.20 get_symmetry()	766
6.74.3.21 printindices()	767
6.74.3.22 print_indexed()	767
6.74.3.23 do_print()	767
6.74.3.24 do_print_latex()	767
6.74.3.25 do_print_tree()	767
6.74.3.26 validate()	768
6.74.4 Friends And Related Function Documentation	768
6.74.4.1 simplify_indexed	768
6.74.4.2 simplify_indexed_product	768
6.74.4.3 reposition_dummy_indices	768
6.74.5 Member Data Documentation	769
6.74.5.1 symtree	769
6.75 GiNaC::info_flags Class Reference	769
6.75.1 Detailed Description	770
6.75.2 Member Enumeration Documentation	770
6.75.2.1 anonymous enum	770
6.76 GiNaC::integral Class Reference	771
6.76.1 Detailed Description	776
6.76.2 Constructor & Destructor Documentation	776
6.76.2.1 integral()	777
6.76.3 Member Function Documentation	777
6.76.3.1 precedence()	777
6.76.3.2 eval()	777
6.76.3.3 evalf()	777



6.76.3.4 degree()	778
6.76.3.5 ldegree()	778
6.76.3.6 eval_ncmul()	778
6.76.3.7 nops()	778
6.76.3.8 op()	779
6.76.3.9 let_op()	779
6.76.3.10 expand()	779
6.76.3.11 get_free_indices()	779
6.76.3.12 return_type()	780
6.76.3.13 return_type_tinfo()	780
6.76.3.14 conjugate()	780
6.76.3.15 eval_integ()	780
6.76.3.16 archive()	780
6.76.3.17 read_archive()	781
6.76.3.18 derivative()	781
6.76.3.19 series()	781
6.76.3.20 do_print()	782
6.76.3.21 do_print_latex()	782
6.76.4 Member Data Documentation	782
6.76.4.1 max_integration_level	782
6.76.4.2 relative_integration_error	782
6.76.4.3 x	782
6.76.4.4 a	783
6.76.4.5 b	783
6.76.4.6 f	783
6.77 GiNaC::integration_kernel Class Reference	783
6.77.1 Detailed Description	788
6.77.2 Member Function Documentation	788
6.77.2.1 series()	789
6.77.2.2 has_trailing_zero()	789
6.77.2.3 is_numeric()	789
6.77.2.4 Laurent_series()	790
6.77.2.5 get_numerical_value()	790
6.77.2.6 uses_Laurent_series()	790
6.77.2.7 series_coeff_impl()	790
6.77.2.8 get_cache_size()	791
6.77.2.9 set_cache_step()	791
6.77.2.10 get_series_coeff()	791
6.77.2.11 series_coeff()	791
6.77.2.12 get_numerical_value_impl()	792
6.77.2.13 do_print()	792
6.77.3 Member Data Documentation	792

6.77.3.1	cache_step_size	792
6.77.3.2	series_vec	792
6.78	GiNaC::is_not_a_clifford Struct Reference	792
6.78.1	Detailed Description	793
6.78.2	Member Function Documentation	793
6.78.2.1	operator()	793
6.79	GiNaC::is_summation_idx Struct Reference	793
6.79.1	Member Function Documentation	793
6.79.1.1	operator()	793
6.80	GiNaC::iterated_integral2_SERIAL Class Reference	794
6.80.1	Detailed Description	794
6.80.2	Member Data Documentation	794
6.80.2.1	serial	794
6.81	GiNaC::iterated_integral3_SERIAL Class Reference	794
6.81.1	Detailed Description	795
6.81.2	Member Data Documentation	795
6.81.2.1	serial	795
6.82	GiNaC::Kronecker_dtau_kernel Class Reference	795
6.82.1	Detailed Description	800
6.82.2	Constructor & Destructor Documentation	801
6.82.2.1	Kronecker_dtau_kernel()	801
6.82.3	Member Function Documentation	801
6.82.3.1	nops()	801
6.82.3.2	op()	801
6.82.3.3	let_op()	801
6.82.3.4	is_numeric()	802
6.82.3.5	get_numerical_value()	802
6.82.3.6	series_coeff_impl()	802
6.82.3.7	do_print()	802
6.82.4	Member Data Documentation	803
6.82.4.1	n	803
6.82.4.2	z	803
6.82.4.3	K	803
6.82.4.4	C_norm	803
6.83	GiNaC::Kronecker_dz_kernel Class Reference	804
6.83.1	Detailed Description	809
6.83.2	Constructor & Destructor Documentation	810
6.83.2.1	Kronecker_dz_kernel()	810
6.83.3	Member Function Documentation	810
6.83.3.1	nops()	810
6.83.3.2	op()	810
6.83.3.3	let_op()	810

6.83.3.4 is_numeric()	811
6.83.3.5 get_numerical_value()	811
6.83.3.6 series_coeff_impl()	811
6.83.3.7 do_print()	811
6.83.4 Member Data Documentation	812
6.83.4.1 n	812
6.83.4.2 z_j	812
6.83.4.3 tau	812
6.83.4.4 K	812
6.83.4.5 C_norm	812
6.84 GiNaC::lanczos_coeffs Class Reference	813
6.84.1 Constructor & Destructor Documentation	813
6.84.1.1 lanczos_coeffs()	813
6.84.2 Member Function Documentation	813
6.84.2.1 sufficiently_accurate()	813
6.84.2.2 get_order()	814
6.84.2.3 calc_lanczos_A()	814
6.84.3 Member Data Documentation	814
6.84.3.1 coeffs	814
6.84.3.2 current_vector	814
6.85 std::less< GiNaC::ptr< T > > Struct Template Reference	814
6.85.1 Detailed Description	815
6.85.2 Member Function Documentation	815
6.85.2.1 operator()()	815
6.86 GiNaC::library_init Class Reference	815
6.86.1 Detailed Description	816
6.86.2 Constructor & Destructor Documentation	816
6.86.2.1 library_init()	816
6.86.2.2 ~library_init()	817
6.86.3 Member Function Documentation	817
6.86.3.1 init_unarchivers()	817
6.86.4 Member Data Documentation	817
6.86.4.1 count	817
6.87 GiNaC::make_flat_inserter Class Reference	817
6.87.1 Detailed Description	818
6.87.2 Constructor & Destructor Documentation	818
6.87.2.1 make_flat_inserter() [1/2]	818
6.87.2.2 make_flat_inserter() [2/2]	818
6.87.3 Member Function Documentation	818
6.87.3.1 handle_factor()	819
6.87.3.2 combine_indices()	819
6.87.4 Member Data Documentation	819

6.87.4.1 do_renaming . . . . .	819
6.87.4.2 used_indices . . . . .	819
6.88 GiNaC::map_function Struct Reference . . . . .	820
6.88.1 Detailed Description . . . . .	821
6.88.2 Member Typedef Documentation . . . . .	821
6.88.2.1 argument_type . . . . .	821
6.88.2.2 result_type . . . . .	821
6.88.3 Constructor & Destructor Documentation . . . . .	821
6.88.3.1 ~map_function() . . . . .	821
6.88.4 Member Function Documentation . . . . .	821
6.88.4.1 operator()() . . . . .	822
6.89 GiNaC::matrix Class Reference . . . . .	822
6.89.1 Detailed Description . . . . .	828
6.89.2 Constructor & Destructor Documentation . . . . .	828
6.89.2.1 matrix() [1/5] . . . . .	829
6.89.2.2 matrix() [2/5] . . . . .	830
6.89.2.3 matrix() [3/5] . . . . .	830
6.89.2.4 matrix() [4/5] . . . . .	830
6.89.2.5 matrix() [5/5] . . . . .	831
6.89.3 Member Function Documentation . . . . .	831
6.89.3.1 nops() . . . . .	831
6.89.3.2 op() . . . . .	831
6.89.3.3 let_op() . . . . .	831
6.89.3.4 evalm() . . . . .	832
6.89.3.5 subs() . . . . .	832
6.89.3.6 eval_indexed() . . . . .	832
6.89.3.7 add_indexed() . . . . .	832
6.89.3.8 scalar_mul_indexed() . . . . .	833
6.89.3.9 contract_with() . . . . .	833
6.89.3.10 conjugate() . . . . .	833
6.89.3.11 real_part() . . . . .	833
6.89.3.12 imag_part() . . . . .	834
6.89.3.13 archive() . . . . .	834
6.89.3.14 read_archive() . . . . .	834
6.89.3.15 match_same_type() . . . . .	834
6.89.3.16 return_type() . . . . .	835
6.89.3.17 rows() . . . . .	835
6.89.3.18 cols() . . . . .	835
6.89.3.19 add() . . . . .	835
6.89.3.20 sub() . . . . .	836
6.89.3.21 mul() [1/2] . . . . .	836
6.89.3.22 mul() [2/2] . . . . .	836

6.89.3.23 mul_scalar()	836
6.89.3.24 pow()	837
6.89.3.25 operator>() [1/2]	837
6.89.3.26 operator>() [2/2]	837
6.89.3.27 set()	838
6.89.3.28 transpose()	838
6.89.3.29 determinant()	838
6.89.3.30 trace()	839
6.89.3.31 charpoly()	839
6.89.3.32 inverse() [1/2]	840
6.89.3.33 inverse() [2/2]	840
6.89.3.34 solve()	841
6.89.3.35 rank() [1/2]	842
6.89.3.36 rank() [2/2]	842
6.89.3.37 is_zero_matrix()	842
6.89.3.38 determinant_minor()	842
6.89.3.39 echelon_form()	843
6.89.3.40 gauss_elimination()	843
6.89.3.41 division_free_elimination()	843
6.89.3.42 fraction_free_elimination()	844
6.89.3.43 markowitz_elimination()	844
6.89.3.44 pivot()	845
6.89.3.45 print_elements()	845
6.89.3.46 do_print()	845
6.89.3.47 do_print_latex()	846
6.89.3.48 do_print_python_repr()	846
6.89.4 Member Data Documentation	846
6.89.4.1 row	846
6.89.4.2 col	846
6.89.4.3 m	847
6.90 GiNaC::minkmetric Class Reference	847
6.90.1 Detailed Description	852
6.90.2 Constructor & Destructor Documentation	852
6.90.2.1 minkmetric()	852
6.90.3 Member Function Documentation	852
6.90.3.1 info()	853
6.90.3.2 eval_indexed()	853
6.90.3.3 archive()	853
6.90.3.4 read_archive()	854
6.90.3.5 return_type()	854
6.90.3.6 do_print()	854
6.90.3.7 do_print_latex()	854

6.90.4 Member Data Documentation	854
6.90.4.1 pos_sig	855
6.91 GiNaC::modular_form_kernel Class Reference	855
6.91.1 Detailed Description	861
6.91.2 Constructor & Destructor Documentation	861
6.91.2.1 modular_form_kernel()	861
6.91.3 Member Function Documentation	861
6.91.3.1 series()	861
6.91.3.2 nops()	861
6.91.3.3 op()	862
6.91.3.4 let_op()	862
6.91.3.5 is_numeric()	862
6.91.3.6 Laurent_series()	862
6.91.3.7 get_numerical_value()	863
6.91.3.8 uses_Laurent_series()	863
6.91.3.9 q_expansion_modular_form()	863
6.91.3.10 do_print()	863
6.91.4 Member Data Documentation	863
6.91.4.1 k	864
6.91.4.2 P	864
6.91.4.3 C_norm	864
6.92 GiNaC::basic_partition_generator::mpartition2 Struct Reference	864
6.92.1 Constructor & Destructor Documentation	864
6.92.1.1 mpartition2()	865
6.92.2 Member Function Documentation	865
6.92.2.1 next_partition()	865
6.92.3 Member Data Documentation	865
6.92.3.1 x	865
6.92.3.2 n	865
6.92.3.3 m	865
6.93 GiNaC::mul Class Reference	866
6.93.1 Detailed Description	874
6.93.2 Constructor & Destructor Documentation	874
6.93.2.1 mul() [1/7]	874
6.93.2.2 mul() [2/7]	874
6.93.2.3 mul() [3/7]	874
6.93.2.4 mul() [4/7]	875
6.93.2.5 mul() [5/7]	875
6.93.2.6 mul() [6/7]	875
6.93.2.7 mul() [7/7]	875
6.93.3 Member Function Documentation	875
6.93.3.1 precedence()	876

6.93.3.2 info()	876
6.93.3.3 is_polynomial()	876
6.93.3.4 degree()	877
6.93.3.5 ldegree()	877
6.93.3.6 coeff()	877
6.93.3.7 has()	877
6.93.3.8 eval()	878
6.93.3.9 evalf()	878
6.93.3.10 real_part()	878
6.93.3.11 imag_part()	879
6.93.3.12 evalm()	879
6.93.3.13 series()	879
6.93.3.14 normal()	880
6.93.3.15 integer_content()	880
6.93.3.16 smod()	880
6.93.3.17 max_coefficient()	881
6.93.3.18 get_free_indices()	881
6.93.3.19 conjugate()	881
6.93.3.20 derivative()	882
6.93.3.21 eval_ncmul()	882
6.93.3.22 return_type()	882
6.93.3.23 return_type_tinfo()	882
6.93.3.24 thisexpairseq() [1/2]	883
6.93.3.25 thisexpairseq() [2/2]	883
6.93.3.26 split_ex_to_pair()	883
6.93.3.27 combine_ex_with_coeff_to_pair()	884
6.93.3.28 combine_pair_with_coeff_to_pair()	884
6.93.3.29 recombine_pair_to_ex()	884
6.93.3.30 expair_needs_further_processing()	884
6.93.3.31 default_overall_coeff()	885
6.93.3.32 combine_overall_coeff() [1/2]	885
6.93.3.33 combine_overall_coeff() [2/2]	885
6.93.3.34 can_make_flat()	885
6.93.3.35 expand()	886
6.93.3.36 algebraic_subs_mul()	886
6.93.3.37 find_real_imag()	886
6.93.3.38 print_overall_coeff()	886
6.93.3.39 do_print()	887
6.93.3.40 do_print_latex()	887
6.93.3.41 do_print_csrc()	887
6.93.3.42 do_print_python_repr()	887
6.93.3.43 can_be_further_expanded()	887

6.93.3.44 expandchildren()	888
6.93.4 Friends And Related Function Documentation	888
6.93.4.1 add	888
6.93.4.2 ncmul	888
6.93.4.3 power	888
6.94 GiNaC::multi_iterator_counter< T > Class Template Reference	889
6.94.1 Detailed Description	890
6.94.2 Constructor & Destructor Documentation	891
6.94.2.1 multi_iterator_counter() [1/3]	891
6.94.2.2 multi_iterator_counter() [2/3]	891
6.94.2.3 multi_iterator_counter() [3/3]	891
6.94.3 Member Function Documentation	891
6.94.3.1 init()	891
6.94.3.2 operator++()	892
6.94.4 Friends And Related Function Documentation	892
6.94.4.1 operator<<	892
6.95 GiNaC::multi_iterator_counter_indv< T > Class Template Reference	892
6.95.1 Detailed Description	894
6.95.2 Constructor & Destructor Documentation	894
6.95.2.1 multi_iterator_counter_indv() [1/3]	894
6.95.2.2 multi_iterator_counter_indv() [2/3]	895
6.95.2.3 multi_iterator_counter_indv() [3/3]	895
6.95.3 Member Function Documentation	895
6.95.3.1 init()	895
6.95.3.2 operator++()	895
6.95.4 Friends And Related Function Documentation	896
6.95.4.1 operator<<	896
6.95.5 Member Data Documentation	896
6.95.5.1 Nv	896
6.96 GiNaC::multi_iterator_ordered< T > Class Template Reference	896
6.96.1 Detailed Description	898
6.96.2 Constructor & Destructor Documentation	898
6.96.2.1 multi_iterator_ordered() [1/3]	898
6.96.2.2 multi_iterator_ordered() [2/3]	899
6.96.2.3 multi_iterator_ordered() [3/3]	899
6.96.3 Member Function Documentation	899
6.96.3.1 init()	899
6.96.3.2 operator++()	899
6.96.4 Friends And Related Function Documentation	900
6.96.4.1 operator<<	900
6.97 GiNaC::multi_iterator_ordered_eq< T > Class Template Reference	900
6.97.1 Detailed Description	902



6.97.2 Constructor & Destructor Documentation	902
6.97.2.1 multi_iterator_ordered_eq() [1/3]	902
6.97.2.2 multi_iterator_ordered_eq() [2/3]	902
6.97.2.3 multi_iterator_ordered_eq() [3/3]	903
6.97.3 Member Function Documentation	903
6.97.3.1 init()	903
6.97.3.2 operator++()	903
6.97.4 Friends And Related Function Documentation	903
6.97.4.1 operator<<	904
6.98 GiNaC::multi_iterator_ordered_eq_indv< T > Class Template Reference	904
6.98.1 Detailed Description	906
6.98.2 Constructor & Destructor Documentation	906
6.98.2.1 multi_iterator_ordered_eq_indv() [1/3]	906
6.98.2.2 multi_iterator_ordered_eq_indv() [2/3]	906
6.98.2.3 multi_iterator_ordered_eq_indv() [3/3]	906
6.98.3 Member Function Documentation	907
6.98.3.1 init()	907
6.98.3.2 operator++()	907
6.98.4 Friends And Related Function Documentation	907
6.98.4.1 operator<<	907
6.98.5 Member Data Documentation	907
6.98.5.1 Nv	908
6.99 GiNaC::multi_iterator_permutation< T > Class Template Reference	908
6.99.1 Detailed Description	910
6.99.2 Constructor & Destructor Documentation	910
6.99.2.1 multi_iterator_permutation() [1/3]	910
6.99.2.2 multi_iterator_permutation() [2/3]	910
6.99.2.3 multi_iterator_permutation() [3/3]	911
6.99.3 Member Function Documentation	911
6.99.3.1 init()	911
6.99.3.2 operator++()	911
6.99.3.3 get_sign()	912
6.99.4 Friends And Related Function Documentation	912
6.99.4.1 operator<<	912
6.100 GiNaC::multi_iterator_shuffle< T > Class Template Reference	912
6.100.1 Detailed Description	915
6.100.2 Constructor & Destructor Documentation	915
6.100.2.1 multi_iterator_shuffle() [1/2]	915
6.100.2.2 multi_iterator_shuffle() [2/2]	915
6.100.3 Member Function Documentation	915
6.100.3.1 init()	915
6.100.3.2 operator++()	916

6.100.4 Friends And Related Function Documentation	916
6.100.4.1 operator<<	916
6.100.5 Member Data Documentation	916
6.100.5.1 N_internal	916
6.100.5.2 v_internal	916
6.100.5.3 v_orig	917
6.101 GiNaC::multi_iterator_shuffle_prime< T > Class Template Reference	917
6.101.1 Detailed Description	919
6.101.2 Constructor & Destructor Documentation	920
6.101.2.1 multi_iterator_shuffle_prime() [1/2]	920
6.101.2.2 multi_iterator_shuffle_prime() [2/2]	920
6.101.3 Member Function Documentation	920
6.101.3.1 init()	920
6.101.4 Friends And Related Function Documentation	920
6.101.4.1 operator<<	920
6.102 GiNaC::multiple_polylog_kernel Class Reference	921
6.102.1 Detailed Description	926
6.102.2 Constructor & Destructor Documentation	927
6.102.2.1 multiple_polylog_kernel()	927
6.102.3 Member Function Documentation	927
6.102.3.1 nops()	927
6.102.3.2 op()	927
6.102.3.3 let_op()	927
6.102.3.4 is_numeric()	928
6.102.3.5 series_coeff_impl()	928
6.102.3.6 do_print()	928
6.102.4 Member Data Documentation	928
6.102.4.1 z	928
6.103 GiNaC::ncmul Class Reference	929
6.103.1 Detailed Description	936
6.103.2 Constructor & Destructor Documentation	936
6.103.2.1 ncmul() [1/7]	936
6.103.2.2 ncmul() [2/7]	936
6.103.2.3 ncmul() [3/7]	936
6.103.2.4 ncmul() [4/7]	936
6.103.2.5 ncmul() [5/7]	937
6.103.2.6 ncmul() [6/7]	937
6.103.2.7 ncmul() [7/7]	937
6.103.3 Member Function Documentation	937
6.103.3.1 precedence()	937
6.103.3.2 info()	937
6.103.3.3 degree()	938

6.103.3.4 ldegree()	938
6.103.3.5 expand()	938
6.103.3.6 coeff()	938
6.103.3.7 eval()	939
6.103.3.8 evalm()	939
6.103.3.9 get_free_indices()	939
6.103.3.10 thiscontainer() [1/2]	940
6.103.3.11 thiscontainer() [2/2]	940
6.103.3.12 conjugate()	940
6.103.3.13 real_part()	940
6.103.3.14 imag_part()	940
6.103.3.15 derivative()	941
6.103.3.16 return_type()	941
6.103.3.17 return_type_tinfo()	941
6.103.3.18 do_print()	941
6.103.3.19 do_print_csrc()	942
6.103.3.20 count_factors()	942
6.103.3.21 append_factors()	942
6.103.3.22 expandchildren()	942
6.103.3.23 get_factors()	942
6.103.4 Friends And Related Function Documentation	943
6.103.4.1 power	943
6.103.4.2 reeval_ncmul	943
6.103.4.3 hold_ncmul	943
6.104 GiNaC::normal_map_function Struct Reference	943
6.104.1 Detailed Description	944
6.104.2 Member Function Documentation	944
6.104.2.1 operator()()	944
6.105 GiNaC::numeric Class Reference	945
6.105.1 Detailed Description	952
6.105.2 Constructor & Destructor Documentation	952
6.105.2.1 numeric() [1/10]	952
6.105.2.2 numeric() [2/10]	952
6.105.2.3 numeric() [3/10]	952
6.105.2.4 numeric() [4/10]	952
6.105.2.5 numeric() [5/10]	953
6.105.2.6 numeric() [6/10]	953
6.105.2.7 numeric() [7/10]	953
6.105.2.8 numeric() [8/10]	953
6.105.2.9 numeric() [9/10]	954
6.105.2.10 numeric() [10/10]	954
6.105.3 Member Function Documentation	954

6.105.3.1 precedence()	954
6.105.3.2 info()	955
6.105.3.3 is_polynomial()	955
6.105.3.4 degree()	955
6.105.3.5 ldegree()	956
6.105.3.6 coeff()	956
6.105.3.7 has()	956
6.105.3.8 eval()	956
6.105.3.9 evalf()	957
6.105.3.10 subs()	957
6.105.3.11 normal()	957
6.105.3.12 to_rational()	958
6.105.3.13 to_polynomial()	958
6.105.3.14 integer_content()	958
6.105.3.15 smod()	958
6.105.3.16 max_coefficient()	959
6.105.3.17 conjugate()	959
6.105.3.18 real_part()	959
6.105.3.19 imag_part()	960
6.105.3.20 archive()	960
6.105.3.21 read_archive()	960
6.105.3.22 derivative()	960
6.105.3.23 is_equal_same_type()	961
6.105.3.24 calchash()	961
6.105.3.25 add()	961
6.105.3.26 sub()	962
6.105.3.27 mul()	962
6.105.3.28 div()	962
6.105.3.29 power()	963
6.105.3.30 add_dyn()	963
6.105.3.31 sub_dyn()	963
6.105.3.32 mul_dyn()	963
6.105.3.33 div_dyn()	963
6.105.3.34 power_dyn()	964
6.105.3.35 operator=() [1/6]	964
6.105.3.36 operator=() [2/6]	964
6.105.3.37 operator=() [3/6]	964
6.105.3.38 operator=() [4/6]	965
6.105.3.39 operator=() [5/6]	965
6.105.3.40 operator=() [6/6]	965
6.105.3.41 inverse()	965
6.105.3.42 step()	965

6.105.3.43	<a href="#">csgn()</a>	966
6.105.3.44	<a href="#">compare()</a>	966
6.105.3.45	<a href="#">is_equal()</a>	966
6.105.3.46	<a href="#">is_zero()</a>	967
6.105.3.47	<a href="#">is_positive()</a>	967
6.105.3.48	<a href="#">is_negative()</a>	967
6.105.3.49	<a href="#">is_integer()</a>	967
6.105.3.50	<a href="#">is_pos_integer()</a>	968
6.105.3.51	<a href="#">is_nonneg_integer()</a>	968
6.105.3.52	<a href="#">is_even()</a>	968
6.105.3.53	<a href="#">is_odd()</a>	968
6.105.3.54	<a href="#">is_prime()</a>	969
6.105.3.55	<a href="#">is_rational()</a>	969
6.105.3.56	<a href="#">is_real()</a>	969
6.105.3.57	<a href="#">is_cinteger()</a>	969
6.105.3.58	<a href="#">is_crational()</a>	970
6.105.3.59	<a href="#">operator==(())</a>	970
6.105.3.60	<a href="#">operator!=(())</a>	970
6.105.3.61	<a href="#">operator&lt;()</a>	970
6.105.3.62	<a href="#">operator&lt;=()</a>	970
6.105.3.63	<a href="#">operator&gt;()</a>	972
6.105.3.64	<a href="#">operator&gt;=()</a>	972
6.105.3.65	<a href="#">to_int()</a>	972
6.105.3.66	<a href="#">to_long()</a>	973
6.105.3.67	<a href="#">to_double()</a>	973
6.105.3.68	<a href="#">to_cl_N()</a>	973
6.105.3.69	<a href="#">real()</a>	973
6.105.3.70	<a href="#">imag()</a>	974
6.105.3.71	<a href="#">numer()</a>	974
6.105.3.72	<a href="#">denom()</a>	974
6.105.3.73	<a href="#">int_length()</a>	974
6.105.3.74	<a href="#">print_numeric()</a>	975
6.105.3.75	<a href="#">do_print()</a>	975
6.105.3.76	<a href="#">do_print_latex()</a>	975
6.105.3.77	<a href="#">do_print_csrc()</a>	975
6.105.3.78	<a href="#">do_print_csrc_cl_N()</a>	976
6.105.3.79	<a href="#">do_print_tree()</a>	976
6.105.3.80	<a href="#">do_print_python_repr()</a>	976
6.105.4	<a href="#">Member Data Documentation</a>	976
6.105.4.1	<a href="#">value</a>	976
6.106	<a href="#">GiNaC::op0_is_equal Struct Reference</a>	977
6.106.1	<a href="#">Member Function Documentation</a>	977

6.106.1.1 operator()()	977
6.107 GiNaC::partition_generator Class Reference	977
6.107.1 Detailed Description	978
6.107.2 Constructor & Destructor Documentation	979
6.107.2.1 partition_generator()	979
6.107.3 Member Function Documentation	979
6.107.3.1 get()	979
6.107.3.2 next()	979
6.107.4 Member Data Documentation	979
6.107.4.1 partition	979
6.107.4.2 current_updated	980
6.108 GiNaC::partition_with_zero_parts_generator Class Reference	980
6.108.1 Detailed Description	981
6.108.2 Constructor & Destructor Documentation	981
6.108.2.1 partition_with_zero_parts_generator()	981
6.108.3 Member Function Documentation	981
6.108.3.1 get()	982
6.108.3.2 next()	982
6.108.4 Member Data Documentation	982
6.108.4.1 m	982
6.108.4.2 partition	982
6.108.4.3 current_updated	982
6.109 GiNaC::pointer_to_map_function Class Reference	983
6.109.1 Constructor & Destructor Documentation	984
6.109.1.1 pointer_to_map_function()	984
6.109.2 Member Function Documentation	984
6.109.2.1 operator()()	984
6.109.3 Member Data Documentation	985
6.109.3.1 ptr	985
6.110 GiNaC::pointer_to_map_function_1arg< T1 > Class Template Reference	985
6.110.1 Constructor & Destructor Documentation	987
6.110.1.1 pointer_to_map_function_1arg()	987
6.110.2 Member Function Documentation	987
6.110.2.1 operator()()	987
6.110.3 Member Data Documentation	987
6.110.3.1 ptr	987
6.110.3.2 arg1	987
6.111 GiNaC::pointer_to_map_function_2args< T1, T2 > Class Template Reference	988
6.111.1 Constructor & Destructor Documentation	989
6.111.1.1 pointer_to_map_function_2args()	989
6.111.2 Member Function Documentation	989
6.111.2.1 operator()()	989

6.111.3 Member Data Documentation	990
6.111.3.1 ptr	990
6.111.3.2 arg1	990
6.111.3.3 arg2	990
6.112 GiNaC::pointer_to_map_function_3args< T1, T2, T3 > Class Template Reference	990
6.112.1 Constructor & Destructor Documentation	992
6.112.1.1 pointer_to_map_function_3args()	992
6.112.2 Member Function Documentation	992
6.112.2.1 operator()()	992
6.112.3 Member Data Documentation	992
6.112.3.1 ptr	992
6.112.3.2 arg1	993
6.112.3.3 arg2	993
6.112.3.4 arg3	993
6.113 GiNaC::pointer_to_member_to_map_function< C > Class Template Reference	993
6.113.1 Constructor & Destructor Documentation	995
6.113.1.1 pointer_to_member_to_map_function()	995
6.113.2 Member Function Documentation	995
6.113.2.1 operator()()	995
6.113.3 Member Data Documentation	995
6.113.3.1 ptr	995
6.113.3.2 c	995
6.114 GiNaC::pointer_to_member_to_map_function_1arg< C, T1 > Class Template Reference	996
6.114.1 Constructor & Destructor Documentation	997
6.114.1.1 pointer_to_member_to_map_function_1arg()	997
6.114.2 Member Function Documentation	997
6.114.2.1 operator()()	997
6.114.3 Member Data Documentation	998
6.114.3.1 ptr	998
6.114.3.2 c	998
6.114.3.3 arg1	998
6.115 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 > Class Template Reference	998
6.115.1 Constructor & Destructor Documentation	1000
6.115.1.1 pointer_to_member_to_map_function_2args()	1000
6.115.2 Member Function Documentation	1000
6.115.2.1 operator()()	1000
6.115.3 Member Data Documentation	1000
6.115.3.1 ptr	1000
6.115.3.2 c	1001
6.115.3.3 arg1	1001
6.115.3.4 arg2	1001
6.116 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 > Class Template Reference	1001

6.116.1 Constructor & Destructor Documentation	1003
6.116.1.1 pointer_to_member_to_map_function_3args()	1003
6.116.2 Member Function Documentation	1003
6.116.2.1 operator()()	1003
6.116.3 Member Data Documentation	1003
6.116.3.1 ptr	1003
6.116.3.2 c	1004
6.116.3.3 arg1	1004
6.116.3.4 arg2	1004
6.116.3.5 arg3	1004
6.117 GiNaC::pole_error Class Reference	1005
6.117.1 Detailed Description	1005
6.117.2 Constructor & Destructor Documentation	1006
6.117.2.1 pole_error()	1006
6.117.3 Member Function Documentation	1006
6.117.3.1 degree()	1006
6.117.4 Member Data Documentation	1006
6.117.4.1 deg	1006
6.118 GiNaC::possymbol Class Reference	1007
6.118.1 Detailed Description	1013
6.118.2 Constructor & Destructor Documentation	1013
6.118.2.1 possymbol() [1/3]	1013
6.118.2.2 possymbol() [2/3]	1013
6.118.2.3 possymbol() [3/3]	1013
6.118.3 Member Function Documentation	1013
6.118.3.1 get_domain()	1013
6.118.3.2 duplicate()	1014
6.119 GiNaC::power Class Reference	1014
6.119.1 Detailed Description	1020
6.119.2 Constructor & Destructor Documentation	1020
6.119.2.1 power() [1/2]	1020
6.119.2.2 power() [2/2]	1020
6.119.3 Member Function Documentation	1020
6.119.3.1 precedence()	1021
6.119.3.2 info()	1021
6.119.3.3 nops()	1021
6.119.3.4 op()	1021
6.119.3.5 map()	1022
6.119.3.6 is_polynomial()	1022
6.119.3.7 degree()	1022
6.119.3.8 ldegree()	1022
6.119.3.9 coeff()	1023



6.119.3.10 eval()	1023
6.119.3.11 evalf()	1024
6.119.3.12 evalm()	1024
6.119.3.13 series()	1024
6.119.3.14 subs()	1025
6.119.3.15 has()	1025
6.119.3.16 normal()	1025
6.119.3.17 to_rational()	1026
6.119.3.18 to_polynomial()	1026
6.119.3.19 conjugate()	1026
6.119.3.20 real_part()	1026
6.119.3.21 imag_part()	1027
6.119.3.22 archive()	1027
6.119.3.23 read_archive()	1027
6.119.3.24 derivative()	1027
6.119.3.25 eval_ncmul()	1028
6.119.3.26 return_type()	1028
6.119.3.27 return_type_tinfo()	1028
6.119.3.28 expand()	1028
6.119.3.29 print_power()	1029
6.119.3.30 do_print_dfft()	1029
6.119.3.31 do_print_latex()	1029
6.119.3.32 do_print_csrc()	1029
6.119.3.33 do_print_python()	1030
6.119.3.34 do_print_python_repr()	1030
6.119.3.35 do_print_csrc_cl_N()	1030
6.119.3.36 expand_add()	1030
6.119.3.37 expand_add_2()	1031
6.119.3.38 expand_mul()	1031
6.119.4 Friends And Related Function Documentation	1031
6.119.4.1 mul	1031
6.119.5 Member Data Documentation	1032
6.119.5.1 basis	1032
6.119.5.2 exponent	1032
6.120 GiNaC::print_context Class Reference	1032
6.120.1 Detailed Description	1033
6.120.2 Constructor & Destructor Documentation	1033
6.120.2.1 print_context()	1033
6.120.2.2 ~print_context()	1033
6.120.3 Member Data Documentation	1033
6.120.3.1 s	1033
6.120.3.2 options	1034

6.121 GiNaC::print_context_options Class Reference	1034
6.121.1 Detailed Description	1034
6.121.2 Constructor & Destructor Documentation	1034
6.121.2.1 print_context_options()	1035
6.121.3 Member Function Documentation	1035
6.121.3.1 get_name()	1035
6.121.3.2 get_parent_name()	1035
6.121.3.3 get_id()	1035
6.121.4 Member Data Documentation	1035
6.121.4.1 name	1035
6.121.4.2 parent_name	1036
6.121.4.3 id	1036
6.122 GiNaC::print_csrc Class Reference	1036
6.122.1 Detailed Description	1037
6.122.2 Constructor & Destructor Documentation	1037
6.122.2.1 print_csrc()	1037
6.123 GiNaC::print_csrc_cl_N Class Reference	1038
6.123.1 Detailed Description	1039
6.123.2 Constructor & Destructor Documentation	1039
6.123.2.1 print_csrc_cl_N()	1039
6.124 GiNaC::print_csrc_double Class Reference	1040
6.124.1 Detailed Description	1041
6.124.2 Constructor & Destructor Documentation	1041
6.124.2.1 print_csrc_double()	1041
6.125 GiNaC::print_csrc_float Class Reference	1042
6.125.1 Detailed Description	1043
6.125.2 Constructor & Destructor Documentation	1043
6.125.2.1 print_csrc_float()	1043
6.126 GiNaC::print_dflt Class Reference	1044
6.126.1 Detailed Description	1045
6.126.2 Constructor & Destructor Documentation	1045
6.126.2.1 print_dflt()	1045
6.127 GiNaC::print_functor Class Reference	1045
6.127.1 Detailed Description	1045
6.127.2 Constructor & Destructor Documentation	1046
6.127.2.1 print_functor() [1/5]	1046
6.127.2.2 print_functor() [2/5]	1046
6.127.2.3 print_functor() [3/5]	1046
6.127.2.4 print_functor() [4/5]	1046
6.127.2.5 print_functor() [5/5]	1046
6.127.3 Member Function Documentation	1046
6.127.3.1 operator=()	1047

6.127.3.2 operator()()	1047
6.127.3.3 is_valid()	1047
6.127.4 Member Data Documentation	1047
6.127.4.1 impl	1047
6.128 GiNaC::print_functor_impl Class Reference	1048
6.128.1 Detailed Description	1048
6.128.2 Constructor & Destructor Documentation	1048
6.128.2.1 ~print_functor_impl()	1048
6.128.3 Member Function Documentation	1048
6.128.3.1 duplicate()	1049
6.128.3.2 operator()()	1049
6.129 GiNaC::print_latex Class Reference	1049
6.129.1 Detailed Description	1050
6.129.2 Constructor & Destructor Documentation	1050
6.129.2.1 print_latex()	1050
6.130 GiNaC::print_memfun_handler< T, C > Class Template Reference	1051
6.130.1 Detailed Description	1052
6.130.2 Member Typedef Documentation	1052
6.130.2.1 F	1052
6.130.3 Constructor & Destructor Documentation	1052
6.130.3.1 print_memfun_handler()	1052
6.130.4 Member Function Documentation	1052
6.130.4.1 duplicate()	1053
6.130.4.2 operator()()	1053
6.130.5 Member Data Documentation	1053
6.130.5.1 f	1053
6.131 GiNaC::print_options Class Reference	1053
6.131.1 Detailed Description	1054
6.131.2 Member Enumeration Documentation	1054
6.131.2.1 anonymous enum	1054
6.132 GiNaC::print_ptrfun_handler< T, C > Class Template Reference	1055
6.132.1 Detailed Description	1056
6.132.2 Member Typedef Documentation	1056
6.132.2.1 F	1056
6.132.3 Constructor & Destructor Documentation	1056
6.132.3.1 print_ptrfun_handler()	1056
6.132.4 Member Function Documentation	1056
6.132.4.1 duplicate()	1057
6.132.4.2 operator()()	1057
6.132.5 Member Data Documentation	1057
6.132.5.1 f	1057
6.133 GiNaC::print_python Class Reference	1058

6.133.1 Detailed Description	1059
6.133.2 Constructor & Destructor Documentation	1059
6.133.2.1 print_python()	1059
6.134 GiNaC::print_python_repr Class Reference	1059
6.134.1 Detailed Description	1060
6.134.2 Constructor & Destructor Documentation	1060
6.134.2.1 print_python_repr()	1060
6.135 GiNaC::print_tree Class Reference	1061
6.135.1 Detailed Description	1062
6.135.2 Constructor & Destructor Documentation	1062
6.135.2.1 print_tree() [1/2]	1062
6.135.2.2 print_tree() [2/2]	1062
6.135.3 Member Data Documentation	1062
6.135.3.1 delta_indent	1062
6.136 GiNaC::archive_node::property Struct Reference	1063
6.136.1 Detailed Description	1063
6.136.2 Constructor & Destructor Documentation	1063
6.136.2.1 property() [1/2]	1063
6.136.2.2 property() [2/2]	1063
6.136.3 Member Data Documentation	1063
6.136.3.1 type	1064
6.136.3.2 name	1064
6.136.3.3 value	1064
6.137 GiNaC::archive_node::property_info Struct Reference	1064
6.137.1 Detailed Description	1065
6.137.2 Constructor & Destructor Documentation	1065
6.137.2.1 property_info() [1/2]	1065
6.137.2.2 property_info() [2/2]	1065
6.137.3 Member Data Documentation	1065
6.137.3.1 type	1065
6.137.3.2 name	1065
6.137.3.3 count	1066
6.138 GiNaC::pseries Class Reference	1066
6.138.1 Detailed Description	1072
6.138.2 Constructor & Destructor Documentation	1072
6.138.2.1 pseries() [1/2]	1072
6.138.2.2 pseries() [2/2]	1073
6.138.3 Member Function Documentation	1073
6.138.3.1 precedence()	1073
6.138.3.2 nops()	1073
6.138.3.3 op()	1074
6.138.3.4 degree()	1074

6.138.3.5 ldegree()	1074
6.138.3.6 coeff()	1075
6.138.3.7 collect()	1075
6.138.3.8 eval()	1075
6.138.3.9 evalf()	1075
6.138.3.10 series()	1076
6.138.3.11 subs()	1076
6.138.3.12 normal()	1076
6.138.3.13 expand()	1077
6.138.3.14 conjugate()	1077
6.138.3.15 real_part()	1077
6.138.3.16 imag_part()	1077
6.138.3.17 eval_integ()	1078
6.138.3.18 evalm()	1078
6.138.3.19 archive()	1078
6.138.3.20 read_archive()	1078
6.138.3.21 derivative()	1079
6.138.3.22 get_var()	1079
6.138.3.23 get_point()	1079
6.138.3.24 convert_to_poly()	1079
6.138.3.25 is_compatible_to()	1080
6.138.3.26 is_zero()	1080
6.138.3.27 is_terminating()	1080
6.138.3.28 coeffop()	1080
6.138.3.29 exponop()	1081
6.138.3.30 add_series()	1081
6.138.3.31 mul_const()	1081
6.138.3.32 mul_series()	1082
6.138.3.33 power_const()	1082
6.138.3.34 shift_exponents()	1082
6.138.3.35 print_series()	1083
6.138.3.36 do_print()	1083
6.138.3.37 do_print_latex()	1083
6.138.3.38 do_print_tree()	1083
6.138.3.39 do_print_python()	1084
6.138.3.40 do_print_python_repr()	1084
6.138.4 Member Data Documentation	1084
6.138.4.1 seq	1084
6.138.4.2 var	1084
6.138.4.3 point	1085
6.139 GiNaC::psi1_SERIAL Class Reference	1085
6.139.1 Detailed Description	1085

6.139.2 Member Data Documentation	1085
6.139.2.1 serial	1085
6.140 GiNaC::psi2_SERIAL Class Reference	1086
6.140.1 Detailed Description	1086
6.140.2 Member Data Documentation	1086
6.140.2.1 serial	1086
6.141 GiNaC::ptr< T > Class Template Reference	1087
6.141.1 Detailed Description	1088
6.141.2 Constructor & Destructor Documentation	1088
6.141.2.1 ptr() [1/3]	1088
6.141.2.2 ptr() [2/3]	1088
6.141.2.3 ptr() [3/3]	1089
6.141.2.4 ~ptr()	1089
6.141.3 Member Function Documentation	1089
6.141.3.1 operator=()	1089
6.141.3.2 operator*()	1089
6.141.3.3 operator->()	1089
6.141.3.4 makewritable()	1090
6.141.3.5 swap()	1090
6.141.3.6 operator==()	1090
6.141.3.7 operator"!=()	1090
6.141.4 Friends And Related Function Documentation	1090
6.141.4.1 std::less< ptr< T > >	1091
6.141.4.2 get_pointer	1091
6.141.4.3 operator== [1/2]	1091
6.141.4.4 operator"!= [1/2]	1091
6.141.4.5 operator== [2/2]	1091
6.141.4.6 operator"!= [2/2]	1092
6.141.4.7 operator<<	1092
6.141.5 Member Data Documentation	1092
6.141.5.1 p	1092
6.142 GiNaC::realsymbol Class Reference	1092
6.142.1 Detailed Description	1098
6.142.2 Constructor & Destructor Documentation	1098
6.142.2.1 realsymbol() [1/3]	1098
6.142.2.2 realsymbol() [2/3]	1098
6.142.2.3 realsymbol() [3/3]	1098
6.142.3 Member Function Documentation	1099
6.142.3.1 get_domain()	1099
6.142.3.2 conjugate()	1099
6.142.3.3 real_part()	1099
6.142.3.4 imag_part()	1099

6.142.3.5 duplicate()	1099
6.143 GiNaC::refcounted Class Reference	1100
6.143.1 Detailed Description	1101
6.143.2 Constructor & Destructor Documentation	1101
6.143.2.1 refcounted()	1101
6.143.3 Member Function Documentation	1101
6.143.3.1 add_reference()	1101
6.143.3.2 remove_reference()	1101
6.143.3.3 get_refcount()	1102
6.143.3.4 set_refcount()	1102
6.143.4 Member Data Documentation	1102
6.143.4.1 refcount	1102
6.144 GiNaC::registered_class_options Class Reference	1102
6.144.1 Detailed Description	1103
6.144.2 Constructor & Destructor Documentation	1103
6.144.2.1 registered_class_options()	1103
6.144.3 Member Function Documentation	1103
6.144.3.1 get_name()	1103
6.144.3.2 get_parent_name()	1103
6.144.3.3 get_id()	1104
6.144.3.4 get_print_dispatch_table()	1104
6.144.3.5 print_func() [1/3]	1104
6.144.3.6 print_func() [2/3]	1104
6.144.3.7 print_func() [3/3]	1104
6.144.3.8 set_print_func()	1105
6.144.4 Member Data Documentation	1105
6.144.4.1 name	1105
6.144.4.2 parent_name	1105
6.144.4.3 tinfo_key	1105
6.144.4.4 print_dispatch_table	1106
6.145 GiNaC::relational Class Reference	1106
6.145.1 Detailed Description	1111
6.145.2 Member Typedef Documentation	1112
6.145.2.1 safe_bool	1112
6.145.3 Member Enumeration Documentation	1112
6.145.3.1 operators	1112
6.145.4 Constructor & Destructor Documentation	1112
6.145.4.1 relational()	1112
6.145.5 Member Function Documentation	1112
6.145.5.1 precedence()	1113
6.145.5.2 info()	1113
6.145.5.3 nops()	1113

6.145.5.4 op()	1113
6.145.5.5 map()	1114
6.145.5.6 subs()	1114
6.145.5.7 archive()	1114
6.145.5.8 read_archive()	1114
6.145.5.9 canonical()	1115
6.145.5.10 eval_ncmul()	1115
6.145.5.11 match_same_type()	1115
6.145.5.12 return_type()	1115
6.145.5.13 return_type_tinfo()	1116
6.145.5.14 calchash()	1116
6.145.5.15 do_print()	1116
6.145.5.16 do_print_python_repr()	1116
6.145.5.17 lhs()	1117
6.145.5.18 rhs()	1117
6.145.5.19 make_safe_bool()	1117
6.145.5.20 operator safe_bool()	1117
6.145.5.21 operator"!"()	1117
6.145.6 Member Data Documentation	1118
6.145.6.1 lh	1118
6.145.6.2 rh	1118
6.145.6.3 o	1118
6.146 GiNaC::remember_strategies Class Reference	1118
6.146.1 Detailed Description	1119
6.146.2 Member Enumeration Documentation	1119
6.146.2.1 anonymous enum	1119
6.147 GiNaC::remember_table Class Reference	1120
6.147.1 Detailed Description	1121
6.147.2 Constructor & Destructor Documentation	1121
6.147.2.1 remember_table() [1/2]	1121
6.147.2.2 remember_table() [2/2]	1122
6.147.3 Member Function Documentation	1122
6.147.3.1 lookup_entry()	1122
6.147.3.2 add_entry()	1122
6.147.3.3 clear_all_entries()	1122
6.147.3.4 show_statistics()	1122
6.147.3.5 remember_tables()	1123
6.147.3.6 init_table()	1123
6.147.4 Member Data Documentation	1123
6.147.4.1 table_size	1123
6.147.4.2 max_assoc_size	1123
6.147.4.3 remember_strategy	1123



6.148 GiNaC::remember_table_entry Class Reference . . . . .	1124
6.148.1 Detailed Description . . . . .	1125
6.148.2 Constructor & Destructor Documentation . . . . .	1125
6.148.2.1 remember_table_entry() . . . . .	1125
6.148.3 Member Function Documentation . . . . .	1125
6.148.3.1 is_equal() . . . . .	1125
6.148.3.2 get_result() . . . . .	1125
6.148.3.3 get_last_access() . . . . .	1126
6.148.3.4 get_successful_hits() . . . . .	1126
6.148.4 Member Data Documentation . . . . .	1126
6.148.4.1 hashvalue . . . . .	1126
6.148.4.2 seq . . . . .	1126
6.148.4.3 result . . . . .	1126
6.148.4.4 last_access . . . . .	1126
6.148.4.5 successful_hits . . . . .	1127
6.148.4.6 access_counter . . . . .	1127
6.149 GiNaC::remember_table_list Class Reference . . . . .	1127
6.149.1 Detailed Description . . . . .	1128
6.149.2 Constructor & Destructor Documentation . . . . .	1128
6.149.2.1 remember_table_list() . . . . .	1128
6.149.3 Member Function Documentation . . . . .	1128
6.149.3.1 add_entry() . . . . .	1128
6.149.3.2 lookup_entry() . . . . .	1128
6.149.4 Member Data Documentation . . . . .	1129
6.149.4.1 max_assoc_size . . . . .	1129
6.149.4.2 remember_strategy . . . . .	1129
6.150 GiNaC::return_type_t Struct Reference . . . . .	1129
6.150.1 Detailed Description . . . . .	1129
6.150.2 Member Function Documentation . . . . .	1130
6.150.2.1 operator<() . . . . .	1130
6.150.2.2 operator==( ) . . . . .	1130
6.150.2.3 operator"!=( ) . . . . .	1130
6.150.3 Member Data Documentation . . . . .	1130
6.150.3.1 tinfo . . . . .	1130
6.150.3.2 rl . . . . .	1131
6.151 GiNaC::return_types Class Reference . . . . .	1131
6.151.1 Member Enumeration Documentation . . . . .	1131
6.151.1.1 anonymous enum . . . . .	1131
6.152 GiNaC::relational::safe_bool_helper Struct Reference . . . . .	1131
6.152.1 Member Function Documentation . . . . .	1132
6.152.1.1 nonnull() . . . . .	1132
6.153 GiNaC::scalar_products Class Reference . . . . .	1132

6.153.1 Detailed Description	1132
6.153.2 Member Function Documentation	1133
6.153.2.1 add() [1/2]	1133
6.153.2.2 add() [2/2]	1133
6.153.2.3 add_vectors()	1133
6.153.2.4 clear()	1133
6.153.2.5 is_defined()	1134
6.153.2.6 evaluate()	1134
6.153.2.7 debugprint()	1134
6.153.3 Member Data Documentation	1134
6.153.3.1 spm	1134
6.154 GiNaC::series_options Class Reference	1135
6.154.1 Detailed Description	1135
6.154.2 Member Enumeration Documentation	1135
6.154.2.1 anonymous enum	1135
6.155 GiNaC::solve_algo Class Reference	1135
6.155.1 Detailed Description	1136
6.155.2 Member Enumeration Documentation	1136
6.155.2.1 anonymous enum	1136
6.156 GiNaC::spinidx Class Reference	1137
6.156.1 Detailed Description	1144
6.156.2 Constructor & Destructor Documentation	1144
6.156.2.1 spinidx()	1144
6.156.3 Member Function Documentation	1145
6.156.3.1 is_dummy_pair_same_type()	1145
6.156.3.2 conjugate()	1145
6.156.3.3 archive()	1145
6.156.3.4 read_archive()	1146
6.156.3.5 match_same_type()	1146
6.156.3.6 is_dotted()	1146
6.156.3.7 is_undotted()	1147
6.156.3.8 toggle_dot()	1147
6.156.3.9 toggle_variance_dot()	1147
6.156.3.10 do_print()	1147
6.156.3.11 do_print_latex()	1147
6.156.3.12 do_print_tree()	1148
6.156.4 Member Data Documentation	1148
6.156.4.1 dotted	1148
6.157 GiNaC::spinmetric Class Reference	1148
6.157.1 Detailed Description	1154
6.157.2 Member Function Documentation	1154
6.157.2.1 info()	1154

6.157.2.2 eval_indexed()	1154
6.157.2.3 contract_with()	1155
6.157.2.4 do_print()	1155
6.157.2.5 do_print_latex()	1155
6.158 GiNaC::spmapkey Class Reference	1155
6.158.1 Constructor & Destructor Documentation	1156
6.158.1.1 spmapkey() [1/2]	1157
6.158.1.2 spmapkey() [2/2]	1157
6.158.2 Member Function Documentation	1157
6.158.2.1 operator==( )	1157
6.158.2.2 operator<( )	1157
6.158.2.3 debugprint()	1157
6.158.3 Member Data Documentation	1157
6.158.3.1 v1	1158
6.158.3.2 v2	1158
6.158.3.3 dim	1158
6.159 GiNaC::status_flags Class Reference	1158
6.159.1 Detailed Description	1158
6.159.2 Member Enumeration Documentation	1158
6.159.2.1 anonymous enum	1158
6.160 GiNaC::structure< T, ComparisonPolicy > Class Template Reference	1159
6.160.1 Detailed Description	1165
6.160.2 Constructor & Destructor Documentation	1165
6.160.2.1 structure()	1165
6.160.3 Member Function Documentation	1165
6.160.3.1 get_class_name()	1165
6.160.3.2 eval()	1165
6.160.3.3 evalm()	1166
6.160.3.4 eval_ncmul()	1166
6.160.3.5 eval_indexed()	1166
6.160.3.6 print()	1166
6.160.3.7 precedence()	1167
6.160.3.8 info()	1167
6.160.3.9 nops()	1167
6.160.3.10 op()	1167
6.160.3.11 operator[]() [1/4]	1168
6.160.3.12 operator[]() [2/4]	1168
6.160.3.13 let_op()	1168
6.160.3.14 operator[]() [3/4]	1168
6.160.3.15 operator[]() [4/4]	1168
6.160.3.16 has()	1169
6.160.3.17 match()	1169

6.160.3.18 match_same_type()	1169
6.160.3.19 subs()	1170
6.160.3.20 map()	1170
6.160.3.21 degree()	1170
6.160.3.22 ldegree()	1170
6.160.3.23 coeff()	1171
6.160.3.24 expand()	1171
6.160.3.25 collect()	1171
6.160.3.26 derivative()	1171
6.160.3.27 series()	1172
6.160.3.28 normal()	1172
6.160.3.29 to_rational()	1173
6.160.3.30 to_polynomial()	1173
6.160.3.31 integer_content()	1173
6.160.3.32 smod()	1173
6.160.3.33 max_coefficient()	1174
6.160.3.34 get_free_indices()	1174
6.160.3.35 add_indexed()	1174
6.160.3.36 scalar_mul_indexed()	1175
6.160.3.37 contract_with()	1176
6.160.3.38 return_type()	1176
6.160.3.39 return_type_tinfo()	1176
6.160.3.40 is_equal_same_type()	1177
6.160.3.41 calchash()	1177
6.160.3.42 operator->()	1177
6.160.3.43 get_struct() [1/2]	1177
6.160.3.44 get_struct() [2/2]	1178
6.160.4 Member Data Documentation	1178
6.160.4.1 obj	1178
6.161 GiNaC::su3d Class Reference	1178
6.161.1 Detailed Description	1182
6.161.2 Member Function Documentation	1183
6.161.2.1 eval_indexed()	1183
6.161.2.2 contract_with()	1183
6.161.2.3 return_type()	1183
6.161.2.4 do_print()	1183
6.161.2.5 do_print_latex()	1184
6.162 GiNaC::su3f Class Reference	1184
6.162.1 Detailed Description	1188
6.162.2 Member Function Documentation	1189
6.162.2.1 eval_indexed()	1189
6.162.2.2 contract_with()	1189

6.162.2.3 return_type()	1189
6.162.2.4 do_print()	1189
6.162.2.5 do_print_latex()	1190
6.163 GiNaC::su3one Class Reference	1190
6.163.1 Detailed Description	1194
6.163.2 Member Function Documentation	1194
6.163.2.1 do_print()	1194
6.163.2.2 do_print_latex()	1195
6.164 GiNaC::su3t Class Reference	1195
6.164.1 Detailed Description	1199
6.164.2 Member Function Documentation	1199
6.164.2.1 contract_with()	1200
6.164.2.2 do_print()	1200
6.164.2.3 do_print_latex()	1200
6.165 GiNaC::subs_options Class Reference	1200
6.165.1 Detailed Description	1200
6.165.2 Member Enumeration Documentation	1200
6.165.2.1 anonymous enum	1200
6.166 GiNaC::sy_is_less Class Reference	1201
6.166.1 Constructor & Destructor Documentation	1201
6.166.1.1 sy_is_less()	1201
6.166.2 Member Function Documentation	1201
6.166.2.1 operator()()	1201
6.166.3 Member Data Documentation	1202
6.166.3.1 v	1202
6.167 GiNaC::sy_swap Class Reference	1202
6.167.1 Constructor & Destructor Documentation	1202
6.167.1.1 sy_swap()	1202
6.167.2 Member Function Documentation	1202
6.167.2.1 operator()()	1203
6.167.3 Member Data Documentation	1203
6.167.3.1 v	1203
6.167.3.2 swapped	1203
6.168 GiNaC::sym_desc Struct Reference	1203
6.168.1 Detailed Description	1205
6.168.2 Constructor & Destructor Documentation	1205
6.168.2.1 sym_desc()	1205
6.168.3 Member Function Documentation	1205
6.168.3.1 operator<()	1205
6.168.4 Member Data Documentation	1205
6.168.4.1 sym	1206
6.168.4.2 deg_a	1206

6.168.4.3 deg_b	1206
6.168.4.4 ldeg_a	1206
6.168.4.5 ldeg_b	1206
6.168.4.6 max_deg	1206
6.168.4.7 max_lcnops	1207
6.169 GiNaC::symbol Class Reference	1207
6.169.1 Detailed Description	1212
6.169.2 Constructor & Destructor Documentation	1212
6.169.2.1 symbol() [1/2]	1212
6.169.2.2 symbol() [2/2]	1212
6.169.3 Member Function Documentation	1213
6.169.3.1 info()	1213
6.169.3.2 eval()	1213
6.169.3.3 evalf()	1213
6.169.3.4 series()	1214
6.169.3.5 subs()	1214
6.169.3.6 normal()	1214
6.169.3.7 to_rational()	1215
6.169.3.8 to_polynomial()	1215
6.169.3.9 conjugate()	1215
6.169.3.10 real_part()	1215
6.169.3.11 imag_part()	1215
6.169.3.12 is_polynomial()	1216
6.169.3.13 archive()	1216
6.169.3.14 read_archive()	1216
6.169.3.15 derivative()	1217
6.169.3.16 is_equal_same_type()	1217
6.169.3.17 calchash()	1217
6.169.3.18 get_domain()	1218
6.169.3.19 set_name()	1218
6.169.3.20 set_TeX_name()	1218
6.169.3.21 get_name()	1218
6.169.3.22 get_TeX_name()	1218
6.169.3.23 do_print()	1219
6.169.3.24 do_print_latex()	1219
6.169.3.25 do_print_tree()	1219
6.169.3.26 do_print_python_repr()	1219
6.169.4 Member Data Documentation	1219
6.169.4.1 serial	1219
6.169.4.2 name	1220
6.169.4.3 TeX_name	1220
6.169.4.4 next_serial	1220

6.170 GiNaC::symbolset Class Reference	1220
6.170.1 Constructor & Destructor Documentation	1221
6.170.1.1 symbolset()	1221
6.170.2 Member Function Documentation	1221
6.170.2.1 insert_symbols()	1221
6.170.2.2 has()	1221
6.170.3 Member Data Documentation	1221
6.170.3.1 s	1222
6.171 GiNaC::symmetry Class Reference	1222
6.171.1 Detailed Description	1227
6.171.2 Member Enumeration Documentation	1227
6.171.2.1 symmetry_type	1227
6.171.3 Constructor & Destructor Documentation	1227
6.171.3.1 symmetry() [1/2]	1227
6.171.3.2 symmetry() [2/2]	1227
6.171.4 Member Function Documentation	1228
6.171.4.1 archive()	1228
6.171.4.2 read_archive()	1228
6.171.4.3 calchash()	1228
6.171.4.4 get_type()	1229
6.171.4.5 set_type()	1229
6.171.4.6 add()	1229
6.171.4.7 validate()	1229
6.171.4.8 has_symmetry()	1230
6.171.4.9 has_nonsymmetric()	1230
6.171.4.10 has_cyclic()	1230
6.171.4.11 do_print()	1230
6.171.4.12 do_print_tree()	1230
6.171.5 Friends And Related Function Documentation	1231
6.171.5.1 sy_is_less	1231
6.171.5.2 sy_swap	1231
6.171.5.3 canonicalize	1231
6.171.6 Member Data Documentation	1231
6.171.6.1 type	1231
6.171.6.2 indices	1232
6.171.6.3 children	1232
6.172 GiNaC::symminfo Class Reference	1232
6.172.1 Detailed Description	1234
6.172.2 Constructor & Destructor Documentation	1234
6.172.2.1 symminfo() [1/2]	1234
6.172.2.2 symminfo() [2/2]	1234
6.172.3 Member Data Documentation	1234

6.172.3.1 symmterm	1234
6.172.3.2 coeff	1234
6.172.3.3 orig	1235
6.172.3.4 num	1235
6.173 GiNaC::symminfo_is_less_by_orig Class Reference	1235
6.173.1 Member Function Documentation	1235
6.173.1.1 operator()()	1235
6.174 GiNaC::symminfo_is_less_by_symmterm Class Reference	1235
6.174.1 Member Function Documentation	1236
6.174.1.1 operator()()	1236
6.175 GiNaC::tensdelta Class Reference	1236
6.175.1 Detailed Description	1241
6.175.2 Member Function Documentation	1241
6.175.2.1 info()	1241
6.175.2.2 eval_indexed()	1241
6.175.2.3 contract_with()	1241
6.175.2.4 return_type()	1242
6.175.2.5 do_print()	1242
6.175.2.6 do_print_latex()	1242
6.176 GiNaC::tensepsilon Class Reference	1242
6.176.1 Detailed Description	1247
6.176.2 Constructor & Destructor Documentation	1247
6.176.2.1 tensepsilon()	1247
6.176.3 Member Function Documentation	1247
6.176.3.1 info()	1247
6.176.3.2 eval_indexed()	1248
6.176.3.3 contract_with()	1248
6.176.3.4 archive()	1248
6.176.3.5 read_archive()	1248
6.176.3.6 return_type()	1249
6.176.3.7 do_print()	1249
6.176.3.8 do_print_latex()	1249
6.176.4 Member Data Documentation	1249
6.176.4.1 minkowski	1249
6.176.4.2 pos_sig	1249
6.177 GiNaC::tensmetric Class Reference	1250
6.177.1 Detailed Description	1255
6.177.2 Member Function Documentation	1255
6.177.2.1 info()	1255
6.177.2.2 eval_indexed()	1255
6.177.2.3 contract_with()	1256
6.177.2.4 return_type()	1256



6.177.2.5 do_print()	1256
6.178 GiNaC::tensor Class Reference	1257
6.178.1 Detailed Description	1261
6.178.2 Member Function Documentation	1261
6.178.2.1 return_type()	1261
6.178.2.2 replace_contr_index()	1261
6.179 GiNaC::terminfo Class Reference	1262
6.179.1 Detailed Description	1262
6.179.2 Constructor & Destructor Documentation	1263
6.179.2.1 terminfo()	1263
6.179.3 Member Data Documentation	1263
6.179.3.1 orig	1263
6.179.3.2 symm	1263
6.180 GiNaC::terminfo_is_less Class Reference	1263
6.180.1 Member Function Documentation	1263
6.180.1.1 operator()	1264
6.181 GiNaC::class_info< OPT >::tree_node Struct Reference	1264
6.181.1 Constructor & Destructor Documentation	1264
6.181.1.1 tree_node()	1265
6.181.2 Member Function Documentation	1265
6.181.2.1 add_child()	1265
6.181.3 Member Data Documentation	1265
6.181.3.1 children	1265
6.181.3.2 info	1265
6.182 GiNaC::unarchive_table_t Class Reference	1265
6.182.1 Constructor & Destructor Documentation	1266
6.182.1.1 unarchive_table_t()	1266
6.182.1.2 ~unarchive_table_t()	1266
6.182.2 Member Function Documentation	1266
6.182.2.1 find()	1266
6.182.2.2 insert()	1267
6.182.3 Member Data Documentation	1267
6.182.3.1 usecount	1267
6.182.3.2 unarch_map	1267
6.183 GiNaC::user_defined_kernel Class Reference	1267
6.183.1 Detailed Description	1274
6.183.2 Constructor & Destructor Documentation	1274
6.183.2.1 user_defined_kernel()	1274
6.183.3 Member Function Documentation	1274
6.183.3.1 nops()	1274
6.183.3.2 op()	1274
6.183.3.3 let_op()	1275

6.183.3.4 is_numeric()	1275
6.183.3.5 Laurent_series()	1275
6.183.3.6 uses_Laurent_series()	1275
6.183.3.7 do_print()	1276
6.183.4 Member Data Documentation	1276
6.183.4.1 f	1276
6.183.4.2 x	1276
6.184 GiNaC::varidx Class Reference	1276
6.184.1 Detailed Description	1283
6.184.2 Constructor & Destructor Documentation	1283
6.184.2.1 varidx()	1283
6.184.3 Member Function Documentation	1284
6.184.3.1 is_dummy_pair_same_type()	1284
6.184.3.2 archive()	1284
6.184.3.3 read_archive()	1285
6.184.3.4 match_same_type()	1285
6.184.3.5 is_covariant()	1286
6.184.3.6 is_contravariant()	1286
6.184.3.7 toggle_variance()	1286
6.184.3.8 do_print()	1286
6.184.3.9 do_print_tree()	1286
6.184.4 Member Data Documentation	1287
6.184.4.1 covariant	1287
6.185 GiNaC::visitor Class Reference	1287
6.185.1 Detailed Description	1287
6.185.2 Constructor & Destructor Documentation	1287
6.185.2.1 ~visitor()	1287
6.186 GiNaC::wildcard Class Reference	1288
6.186.1 Detailed Description	1292
6.186.2 Constructor & Destructor Documentation	1292
6.186.2.1 wildcard()	1292
6.186.3 Member Function Documentation	1292
6.186.3.1 match()	1292
6.186.3.2 archive()	1293
6.186.3.3 read_archive()	1293
6.186.3.4 calchash()	1293
6.186.3.5 get_label()	1293
6.186.3.6 do_print()	1294
6.186.3.7 do_print_tree()	1294
6.186.3.8 do_print_python_repr()	1294
6.186.4 Member Data Documentation	1294
6.186.4.1 label	1294

6.187 GiNaC::zeta1_SERIAL Class Reference . . . . .	1295
6.187.1 Detailed Description . . . . .	1295
6.187.2 Member Data Documentation . . . . .	1295
6.187.2.1 serial . . . . .	1295
6.188 GiNaC::zeta2_SERIAL Class Reference . . . . .	1295
6.188.1 Detailed Description . . . . .	1296
6.188.2 Member Data Documentation . . . . .	1296
6.188.2.1 serial . . . . .	1296
<b>7 File Documentation . . . . .</b>	<b>1297</b>
7.1 add.cpp File Reference . . . . .	1297
7.1.1 Detailed Description . . . . .	1297
7.2 add.h File Reference . . . . .	1298
7.2.1 Detailed Description . . . . .	1298
7.3 archive.cpp File Reference . . . . .	1298
7.3.1 Detailed Description . . . . .	1299
7.4 archive.h File Reference . . . . .	1299
7.4.1 Detailed Description . . . . .	1300
7.4.2 Macro Definition Documentation . . . . .	1300
7.4.2.1 GINAC_DECLARE_UNARCHIVER . . . . .	1301
7.4.2.2 GINAC_BIND_UNARCHIVER . . . . .	1301
7.5 assertion.h File Reference . . . . .	1302
7.5.1 Detailed Description . . . . .	1302
7.5.2 Macro Definition Documentation . . . . .	1302
7.5.2.1 GINAC_ASSERT . . . . .	1302
7.6 basic.cpp File Reference . . . . .	1302
7.6.1 Detailed Description . . . . .	1303
7.7 basic.h File Reference . . . . .	1303
7.7.1 Detailed Description . . . . .	1304
7.8 class_info.h File Reference . . . . .	1304
7.8.1 Detailed Description . . . . .	1305
7.9 clifford.cpp File Reference . . . . .	1305
7.9.1 Detailed Description . . . . .	1307
7.10 clifford.h File Reference . . . . .	1307
7.10.1 Detailed Description . . . . .	1309
7.11 color.cpp File Reference . . . . .	1309
7.11.1 Detailed Description . . . . .	1310
7.11.2 Macro Definition Documentation . . . . .	1310
7.11.2.1 TEST_PERMUTATION . . . . .	1311
7.11.2.2 CMPINDICES . . . . .	1311
7.12 color.h File Reference . . . . .	1311
7.12.1 Detailed Description . . . . .	1312

7.13 compiler.h File Reference . . . . .	1312
7.13.1 Detailed Description . . . . .	1312
7.13.2 Macro Definition Documentation . . . . .	1312
7.13.2.1 unlikely . . . . .	1313
7.13.2.2 likely . . . . .	1313
7.13.2.3 attribute_deprecated . . . . .	1313
7.14 constant.cpp File Reference . . . . .	1313
7.14.1 Detailed Description . . . . .	1314
7.15 constant.h File Reference . . . . .	1314
7.15.1 Detailed Description . . . . .	1314
7.16 container.h File Reference . . . . .	1315
7.16.1 Detailed Description . . . . .	1315
7.17 crc32.h File Reference . . . . .	1315
7.17.1 Detailed Description . . . . .	1316
7.18 ex.cpp File Reference . . . . .	1316
7.18.1 Detailed Description . . . . .	1316
7.19 ex.h File Reference . . . . .	1316
7.19.1 Detailed Description . . . . .	1318
7.20 excompiler.cpp File Reference . . . . .	1319
7.20.1 Detailed Description . . . . .	1319
7.21 excompiler.h File Reference . . . . .	1319
7.21.1 Detailed Description . . . . .	1320
7.22 expair.cpp File Reference . . . . .	1320
7.22.1 Detailed Description . . . . .	1321
7.23 expair.h File Reference . . . . .	1321
7.23.1 Detailed Description . . . . .	1321
7.24 expairseq.cpp File Reference . . . . .	1322
7.24.1 Detailed Description . . . . .	1322
7.25 expairseq.h File Reference . . . . .	1322
7.25.1 Detailed Description . . . . .	1323
7.26 exprseq.cpp File Reference . . . . .	1323
7.26.1 Detailed Description . . . . .	1323
7.27 exprseq.h File Reference . . . . .	1324
7.27.1 Detailed Description . . . . .	1324
7.28 factor.cpp File Reference . . . . .	1324
7.28.1 Detailed Description . . . . .	1325
7.28.2 Macro Definition Documentation . . . . .	1325
7.28.2.1 DCOUT . . . . .	1325
7.28.2.2 DCOUTVAR . . . . .	1325
7.28.2.3 DCOUT2 . . . . .	1326
7.28.2.4 USE_SAME_DEGREE_FACTOR . . . . .	1326
7.28.3 Variable Documentation . . . . .	1326

7.28.3.1 value	1326
7.28.3.2 r	1326
7.28.3.3 c	1327
7.28.3.4 m	1327
7.28.3.5 lr	1328
7.28.3.6 cache	1328
7.28.3.7 factors	1328
7.28.3.8 one	1328
7.28.3.9 n	1329
7.28.3.10 len	1329
7.28.3.11 last	1329
7.28.3.12 k	1330
7.28.3.13 poly	1330
7.28.3.14 x	1330
7.28.3.15 evalpoint	1331
7.28.3.16 R	1331
7.28.3.17 syms_wox	1331
7.28.3.18 unit	1332
7.28.3.19 cont	1332
7.28.3.20 pp	1332
7.28.3.21 vn	1332
7.28.3.22 vnlst	1332
7.28.3.23 modulus	1332
7.28.3.24 syms	1332
7.28.3.25 options	1333
7.29 factor.h File Reference	1333
7.29.1 Detailed Description	1333
7.30 fail.cpp File Reference	1333
7.30.1 Detailed Description	1334
7.31 fail.h File Reference	1334
7.31.1 Detailed Description	1334
7.32 fderivative.cpp File Reference	1335
7.32.1 Detailed Description	1335
7.33 fderivative.h File Reference	1335
7.33.1 Detailed Description	1336
7.34 flags.h File Reference	1336
7.34.1 Detailed Description	1336
7.35 function.cpp File Reference	1337
7.35.1 Detailed Description	1337
7.36 function.h File Reference	1337
7.36.1 Detailed Description	1344
7.36.2 Macro Definition Documentation	1344

7.36.2.1 DECLARE_FUNCTION_1P . . . . .	1344
7.36.2.2 DECLARE_FUNCTION_2P . . . . .	1344
7.36.2.3 DECLARE_FUNCTION_3P . . . . .	1345
7.36.2.4 DECLARE_FUNCTION_4P . . . . .	1345
7.36.2.5 DECLARE_FUNCTION_5P . . . . .	1345
7.36.2.6 DECLARE_FUNCTION_6P . . . . .	1345
7.36.2.7 DECLARE_FUNCTION_7P . . . . .	1346
7.36.2.8 DECLARE_FUNCTION_8P . . . . .	1346
7.36.2.9 DECLARE_FUNCTION_9P . . . . .	1346
7.36.2.10 DECLARE_FUNCTION_10P . . . . .	1346
7.36.2.11 DECLARE_FUNCTION_11P . . . . .	1347
7.36.2.12 DECLARE_FUNCTION_12P . . . . .	1347
7.36.2.13 DECLARE_FUNCTION_13P . . . . .	1347
7.36.2.14 DECLARE_FUNCTION_14P . . . . .	1348
7.36.2.15 REGISTER_FUNCTION . . . . .	1348
7.36.2.16 is_ex_the_function . . . . .	1348
7.37 ginac.h File Reference . . . . .	1348
7.37.1 Detailed Description . . . . .	1349
7.38 hash_map.h File Reference . . . . .	1349
7.38.1 Detailed Description . . . . .	1349
7.39 hash_seed.h File Reference . . . . .	1350
7.39.1 Detailed Description . . . . .	1350
7.40 idx.cpp File Reference . . . . .	1350
7.40.1 Detailed Description . . . . .	1351
7.41 idx.h File Reference . . . . .	1351
7.41.1 Detailed Description . . . . .	1352
7.42 indexed.cpp File Reference . . . . .	1352
7.42.1 Detailed Description . . . . .	1354
7.43 indexed.h File Reference . . . . .	1354
7.43.1 Detailed Description . . . . .	1355
7.44 inifcns.cpp File Reference . . . . .	1355
7.44.1 Detailed Description . . . . .	1358
7.45 inifcns.h File Reference . . . . .	1358
7.45.1 Detailed Description . . . . .	1360
7.46 inifcns_elliptic.cpp File Reference . . . . .	1360
7.46.1 Detailed Description . . . . .	1361
7.47 inifcns_gamma.cpp File Reference . . . . .	1361
7.47.1 Detailed Description . . . . .	1362
7.48 inifcns_nstdsums.cpp File Reference . . . . .	1362
7.48.1 Detailed Description . . . . .	1364
7.49 inifcns_trans.cpp File Reference . . . . .	1364
7.49.1 Detailed Description . . . . .	1367

7.50 integral.cpp File Reference . . . . .	1367
7.50.1 Detailed Description . . . . .	1368
7.51 integral.h File Reference . . . . .	1368
7.51.1 Detailed Description . . . . .	1369
7.52 integration_kernel.cpp File Reference . . . . .	1369
7.52.1 Detailed Description . . . . .	1370
7.52.2 Variable Documentation . . . . .	1370
7.52.2.1 qbar . . . . .	1370
7.52.2.2 order . . . . .	1371
7.52.2.3 cache_vec . . . . .	1371
7.52.2.4 x . . . . .	1371
7.53 integration_kernel.h File Reference . . . . .	1371
7.53.1 Detailed Description . . . . .	1373
7.54 lst.cpp File Reference . . . . .	1373
7.54.1 Detailed Description . . . . .	1373
7.55 lst.h File Reference . . . . .	1373
7.55.1 Detailed Description . . . . .	1374
7.56 matrix.cpp File Reference . . . . .	1374
7.56.1 Detailed Description . . . . .	1375
7.57 matrix.h File Reference . . . . .	1375
7.57.1 Detailed Description . . . . .	1376
7.58 mul.cpp File Reference . . . . .	1376
7.58.1 Detailed Description . . . . .	1377
7.59 mul.h File Reference . . . . .	1377
7.59.1 Detailed Description . . . . .	1378
7.60 ncmul.cpp File Reference . . . . .	1378
7.60.1 Detailed Description . . . . .	1378
7.61 ncmul.h File Reference . . . . .	1379
7.61.1 Detailed Description . . . . .	1379
7.62 normal.cpp File Reference . . . . .	1379
7.62.1 Detailed Description . . . . .	1381
7.62.2 Macro Definition Documentation . . . . .	1381
7.62.2.1 FAST_COMPARE . . . . .	1382
7.62.2.2 USE_REMEMBER . . . . .	1382
7.62.2.3 USE_TRIAL_DIVISION . . . . .	1382
7.62.2.4 STATISTICS . . . . .	1382
7.63 normal.h File Reference . . . . .	1382
7.63.1 Detailed Description . . . . .	1383
7.64 numeric.cpp File Reference . . . . .	1383
7.64.1 Detailed Description . . . . .	1386
7.65 numeric.h File Reference . . . . .	1387
7.65.1 Detailed Description . . . . .	1389

7.66 operators.cpp File Reference	1389
7.66.1 Detailed Description	1391
7.67 operators.h File Reference	1391
7.67.1 Detailed Description	1393
7.68 power.cpp File Reference	1393
7.68.1 Detailed Description	1393
7.69 power.h File Reference	1394
7.69.1 Detailed Description	1394
7.70 print.cpp File Reference	1394
7.70.1 Detailed Description	1395
7.71 print.h File Reference	1395
7.71.1 Detailed Description	1396
7.71.2 Macro Definition Documentation	1396
7.71.2.1 GINAC_DECLARE_PRINT_CONTEXT_COMMON	1396
7.71.2.2 GINAC_DECLARE_PRINT_CONTEXT_BASE	1397
7.71.2.3 GINAC_DECLARE_PRINT_CONTEXT	1397
7.71.2.4 GINAC_IMPLEMENT_PRINT_CONTEXT	1397
7.72 pseries.cpp File Reference	1398
7.72.1 Detailed Description	1398
7.73 pseries.h File Reference	1398
7.73.1 Detailed Description	1399
7.74 ptr.h File Reference	1399
7.74.1 Detailed Description	1399
7.75 registrar.cpp File Reference	1400
7.75.1 Detailed Description	1400
7.76 registrar.h File Reference	1400
7.76.1 Detailed Description	1401
7.76.2 Macro Definition Documentation	1401
7.76.2.1 GINAC_DECLARE_REGISTERED_CLASS_COMMON	1402
7.76.2.2 GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS	1402
7.76.2.3 GINAC_DECLARE_REGISTERED_CLASS	1402
7.76.2.4 GINAC_IMPLEMENT_REGISTERED_CLASS	1403
7.76.2.5 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT	1403
7.76.2.6 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T	1403
7.77 relational.cpp File Reference	1404
7.77.1 Detailed Description	1404
7.78 relational.h File Reference	1404
7.78.1 Detailed Description	1405
7.79 remember.cpp File Reference	1405
7.79.1 Detailed Description	1405
7.80 remember.h File Reference	1405
7.80.1 Detailed Description	1406



7.81 structure.h File Reference	1406
7.81.1 Detailed Description	1406
7.82 symbol.cpp File Reference	1407
7.82.1 Detailed Description	1407
7.83 symbol.h File Reference	1407
7.83.1 Detailed Description	1408
7.84 symmetry.cpp File Reference	1408
7.84.1 Detailed Description	1409
7.85 symmetry.h File Reference	1409
7.85.1 Detailed Description	1410
7.86 tensor.cpp File Reference	1411
7.86.1 Detailed Description	1412
7.87 tensor.h File Reference	1412
7.87.1 Detailed Description	1413
7.88 utils.cpp File Reference	1413
7.88.1 Detailed Description	1415
7.89 utils.h File Reference	1415
7.89.1 Detailed Description	1417
7.89.2 Macro Definition Documentation	1417
7.89.2.1 DEFAULT_CTOR	1417
7.89.2.2 DEFAULT_COMPARE	1417
7.89.2.3 DEFAULT_PRINT	1417
7.89.2.4 DEFAULT_PRINT_LATEX	1417
7.90 utils_multi_iterator.h File Reference	1418
7.90.1 Detailed Description	1419
7.91 version.h File Reference	1419
7.91.1 Detailed Description	1420
7.91.2 Macro Definition Documentation	1420
7.91.2.1 GINACLIB_MAJOR_VERSION	1420
7.91.2.2 GINACLIB_MINOR_VERSION	1420
7.91.2.3 GINACLIB_MICRO_VERSION	1420
7.91.2.4 GINAC_LT_CURRENT	1420
7.91.2.5 GINAC_LT_REVISION	1420
7.91.2.6 GINAC_LT_AGE	1420
7.91.2.7 GINACLIB_ARCHIVE_VERSION	1421
7.91.2.8 GINACLIB_ARCHIVE_AGE	1421
7.91.2.9 GINACLIB_STR_HELPER	1421
7.91.2.10 GINACLIB_STR	1421
7.91.2.11 GINACLIB_VERSION	1421
7.92 wildcard.cpp File Reference	1421
7.92.1 Detailed Description	1422
7.93 wildcard.h File Reference	1422

7.93.1 Detailed Description . . . . .	1422
---------------------------------------	------

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">GiNaC</a> . . . . .	<a href="#">19</a>
<a href="#">GiNaC::internal</a> . . . . .	<a href="#">310</a>
<a href="#">std</a> . . . . .	<a href="#">310</a>



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

GiNaC::internal::_iter_rep . . . . .	311
GiNaC::_numeric_digits . . . . .	313
GiNaC::archive . . . . .	334
GiNaC::archive_node . . . . .	342
GiNaC::archive_node::archive_node_cit_range . . . . .	352
GiNaC::archive::archived_ex . . . . .	354
GiNaC::basic_multi_iterator< T > . . . . .	389
GiNaC::multi_iterator_counter< T > . . . . .	889
GiNaC::multi_iterator_counter_indv< T > . . . . .	892
GiNaC::multi_iterator_ordered< T > . . . . .	896
GiNaC::multi_iterator_ordered_eq< T > . . . . .	900
GiNaC::multi_iterator_ordered_eq_indv< T > . . . . .	904
GiNaC::multi_iterator_permutation< T > . . . . .	908
GiNaC::multi_iterator_shuffle< T > . . . . .	912
GiNaC::multi_iterator_shuffle_prime< T > . . . . .	917
GiNaC::basic_partition_generator . . . . .	395
GiNaC::partition_generator . . . . .	977
GiNaC::partition_with_zero_parts_generator . . . . .	980
GiNaC::class_info< OPT > . . . . .	397
GiNaC::compare_all_equal< T > . . . . .	435
GiNaC::compare_bitwise< T > . . . . .	437
GiNaC::compare_std_less< T > . . . . .	438
ComparisonPolicy	
GiNaC::structure< T, ComparisonPolicy > . . . . .	1159
GiNaC::composition_generator . . . . .	439
GiNaC::const_iterator . . . . .	442
GiNaC::const_postorder_iterator . . . . .	449
GiNaC::const_preorder_iterator . . . . .	452
GiNaC::container_storage< C > . . . . .	485
GiNaC::container< C > . . . . .	467
GiNaC::function . . . . .	658
GiNaC::fderivative . . . . .	643
GiNaC::indexed . . . . .	747
GiNaC::clifford . . . . .	401

GiNaC::color . . . . .	422
GiNaC::ncmul . . . . .	929
GiNaC::composition_generator::coolmulti . . . . .	489
GiNaC::determinant_algo . . . . .	493
GiNaC::do_taylor . . . . .	521
GiNaC::domain . . . . .	521
std::domain_error . . . . .	
GiNaC::pole_error . . . . .	1005
GiNaC::dunno . . . . .	522
GiNaC::composition_generator::coolmulti::element . . . . .	552
std::equal_to< GiNaC::ex > . . . . .	562
GiNaC::error_and_integral . . . . .	564
GiNaC::error_and_integral_is_less . . . . .	565
GiNaC::ex . . . . .	570
GiNaC::ex_base_is_less . . . . .	605
GiNaC::ex_is_equal . . . . .	605
GiNaC::ex_is_less . . . . .	606
GiNaC::ex_swap . . . . .	606
GiNaC::expair . . . . .	607
GiNaC::expair_is_less . . . . .	610
GiNaC::expair_rest_is_less . . . . .	611
GiNaC::expair_swap . . . . .	612
GiNaC::expand_options . . . . .	636
GiNaC::factor_options . . . . .	637
GiNaC::function_options . . . . .	680
GiNaC::G2_SERIAL . . . . .	727
GiNaC::G3_SERIAL . . . . .	728
GiNaC::gcd_options . . . . .	728
GiNaC::gcdheu_failed . . . . .	729
GiNaC::has_distance< T > . . . . .	729
GiNaC::has_options . . . . .	731
std::hash< GiNaC::ex > . . . . .	732
GiNaC::idx_is_equal_ignore_dim . . . . .	746
GiNaC::info_flags . . . . .	769
GiNaC::is_not_a_clifford . . . . .	792
GiNaC::is_summation_idx . . . . .	793
GiNaC::iterated_integral2_SERIAL . . . . .	794
GiNaC::iterated_integral3_SERIAL . . . . .	794
GiNaC::lanczos_coeffs . . . . .	813
std::less< GiNaC::ptr< T > > . . . . .	814
GiNaC::library_init . . . . .	815
std::list . . . . .	
GiNaC::remember_table_list . . . . .	1127
GiNaC::make_flat_inserter . . . . .	817
GiNaC::map_function . . . . .	820
GiNaC::derivative_map_function . . . . .	491
GiNaC::eval_integ_map_function . . . . .	566
GiNaC::evalf_map_function . . . . .	567
GiNaC::evalm_map_function . . . . .	569
GiNaC::expand_map_function . . . . .	634
GiNaC::normal_map_function . . . . .	943
GiNaC::pointer_to_map_function . . . . .	983
GiNaC::pointer_to_map_function_1arg< T1 > . . . . .	985
GiNaC::pointer_to_map_function_2args< T1, T2 > . . . . .	988
GiNaC::pointer_to_map_function_3args< T1, T2, T3 > . . . . .	990
GiNaC::pointer_to_member_to_map_function< C > . . . . .	993
GiNaC::pointer_to_member_to_map_function_1arg< C, T1 > . . . . .	996
GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 > . . . . .	998

GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 > . . . . .	1001
GiNaC::basic_partition_generator::mpartition2 . . . . .	864
GiNaC::op0_is_equal . . . . .	977
GiNaC::print_context . . . . .	1032
GiNaC::print_csrc . . . . .	1036
GiNaC::print_csrc_cl_N . . . . .	1038
GiNaC::print_csrc_double . . . . .	1040
GiNaC::print_csrc_float . . . . .	1042
GiNaC::print_dflt . . . . .	1044
GiNaC::print_latex . . . . .	1049
GiNaC::print_python . . . . .	1058
GiNaC::print_python_repr . . . . .	1059
GiNaC::print_tree . . . . .	1061
GiNaC::print_context_options . . . . .	1034
GiNaC::print_functor . . . . .	1045
GiNaC::print_functor_impl . . . . .	1048
GiNaC::print_memfun_handler< T, C > . . . . .	1051
GiNaC::print_ptrfun_handler< T, C > . . . . .	1055
GiNaC::print_options . . . . .	1053
GiNaC::archive_node::property . . . . .	1063
GiNaC::archive_node::property_info . . . . .	1064
GiNaC::psi1_SERIAL . . . . .	1085
GiNaC::psi2_SERIAL . . . . .	1086
GiNaC::ptr< T > . . . . .	1087
GiNaC::ptr< GiNaC::basic > . . . . .	1087
GiNaC::refcounted . . . . .	1100
GiNaC::basic . . . . .	355
GiNaC::constant . . . . .	456
GiNaC::container< C > . . . . .	467
GiNaC::expairseq . . . . .	613
GiNaC::add . . . . .	316
GiNaC::mul . . . . .	866
GiNaC::fail . . . . .	638
GiNaC::idx . . . . .	733
GiNaC::varidx . . . . .	1276
GiNaC::spinidx . . . . .	1137
GiNaC::integral . . . . .	771
GiNaC::integration_kernel . . . . .	783
GiNaC::ELi_kernel . . . . .	554
GiNaC::Ebar_kernel . . . . .	522
GiNaC::Eisenstein_h_kernel . . . . .	531
GiNaC::Eisenstein_kernel . . . . .	542
GiNaC::Kronecker_dtau_kernel . . . . .	795
GiNaC::Kronecker_dz_kernel . . . . .	804
GiNaC::basic_log_kernel . . . . .	383
GiNaC::modular_form_kernel . . . . .	855
GiNaC::multiple_polylog_kernel . . . . .	921
GiNaC::user_defined_kernel . . . . .	1267
GiNaC::matrix . . . . .	822
GiNaC::numeric . . . . .	945
GiNaC::power . . . . .	1014
GiNaC::pseries . . . . .	1066
GiNaC::relational . . . . .	1106
GiNaC::structure< T, ComparisonPolicy > . . . . .	1159
GiNaC::symbol . . . . .	1207
GiNaC::realsymbol . . . . .	1092
GiNaC::possymbol . . . . .	1007

GiNaC::symmetry . . . . .	1222
GiNaC::tensor . . . . .	1257
GiNaC::cliffordunit . . . . .	417
GiNaC::diracgamma . . . . .	494
GiNaC::diracgamma5 . . . . .	501
GiNaC::diracgammaL . . . . .	506
GiNaC::diracgammaR . . . . .	511
GiNaC::diracone . . . . .	516
GiNaC::su3d . . . . .	1178
GiNaC::su3f . . . . .	1184
GiNaC::su3one . . . . .	1190
GiNaC::su3t . . . . .	1195
GiNaC::tensdelta . . . . .	1236
GiNaC::tensepsilon . . . . .	1242
GiNaC::tensmetric . . . . .	1250
GiNaC::minkmetric . . . . .	847
GiNaC::spinmetric . . . . .	1148
GiNaC::wildcard . . . . .	1288
GiNaC::registered_class_options . . . . .	1102
GiNaC::remember_strategies . . . . .	1118
GiNaC::remember_table_entry . . . . .	1124
GiNaC::return_type_t . . . . .	1129
GiNaC::return_types . . . . .	1131
GiNaC::relational::safe_bool_helper . . . . .	1131
GiNaC::scalar_products . . . . .	1132
GiNaC::series_options . . . . .	1135
GiNaC::solve_algo . . . . .	1135
GiNaC::spmapkey . . . . .	1155
GiNaC::status_flags . . . . .	1158
GiNaC::subs_options . . . . .	1200
GiNaC::sy_is_less . . . . .	1201
GiNaC::sy_swap . . . . .	1202
GiNaC::sym_desc . . . . .	1203
GiNaC::symbolset . . . . .	1220
GiNaC::symminfo . . . . .	1232
GiNaC::symminfo_is_less_by_orig . . . . .	1235
GiNaC::symminfo_is_less_by_symmterm . . . . .	1235
GiNaC::terminfo . . . . .	1262
GiNaC::terminfo_is_less . . . . .	1263
GiNaC::class_info< OPT >::tree_node . . . . .	1264
GiNaC::unarchive_table_t . . . . .	1265
std::vector	
GiNaC::remember_table . . . . .	1120
GiNaC::visitor . . . . .	1287
GiNaC::zeta1_SERIAL . . . . .	1295
GiNaC::zeta2_SERIAL . . . . .	1295



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">GiNaC::internal::_iter_rep</a>	311
<a href="#">GiNaC::_numeric_digits</a>	
This class is used to instantiate a global singleton object Digits which behaves just like Maple's	
Digits	313
<a href="#">GiNaC::add</a>	
Sum of expressions	316
<a href="#">GiNaC::archive</a>	
This class holds archived versions of <a href="#">GiNaC</a> expressions (class ex)	334
<a href="#">GiNaC::archive_node</a>	
This class stores all properties needed to record/retrieve the state of one object of class basic	
(or a derived class)	342
<a href="#">GiNaC::archive_node::archive_node_cit_range</a>	352
<a href="#">GiNaC::archive::archived_ex</a>	
Archived expression descriptor	354
<a href="#">GiNaC::basic</a>	
This class is the ABC (abstract base class) of <a href="#">GiNaC</a> 's class hierarchy	355
<a href="#">GiNaC::basic_log_kernel</a>	
The basic integration kernel with a logarithmic singularity at the origin	383
<a href="#">GiNaC::basic_multi_iterator&lt; T &gt;</a>	
Basic_multi_iterator is a base class	389
<a href="#">GiNaC::basic_partition_generator</a>	
Base class for generating all bounded combinatorial partitions of an integer n with exactly m	
parts in non-decreasing order	395
<a href="#">GiNaC::class_info&lt; OPT &gt;</a>	397
<a href="#">GiNaC::clifford</a>	
This class holds an object representing an element of the Clifford algebra (the Dirac gamma	
matrices)	401
<a href="#">GiNaC::cliffordunit</a>	
This class represents the Clifford algebra generators (units)	417
<a href="#">GiNaC::color</a>	
This class holds a generator T_a or the unity element of the Lie algebra of SU(3), as used for	
calculations in quantum chromodynamics	422
<a href="#">GiNaC::compare_all_equal&lt; T &gt;</a>	
Comparison policy: all structures of one type are equal	435
<a href="#">GiNaC::compare_bitwise&lt; T &gt;</a>	
Comparison policy: use bit-wise comparison to compare structures	437

<a href="#">GiNaC::compare_std_less&lt; T &gt;</a>	
Comparison policy: use <code>std::equal_to</code> / <code>std::less</code> (defaults to operators <code>==</code> and <code>&lt;</code> ) to compare structures . . . . .	438
<a href="#">GiNaC::composition_generator</a>	
Generate all compositions of a partition of an integer $n$ , starting with the compositions which has non-decreasing order . . . . .	439
<a href="#">GiNaC::const_iterator</a> . . . . .	442
<a href="#">GiNaC::const_postorder_iterator</a> . . . . .	449
<a href="#">GiNaC::const_preorder_iterator</a> . . . . .	452
<a href="#">GiNaC::constant</a>	
This class holds constants, symbols with specific numerical value . . . . .	456
<a href="#">GiNaC::container&lt; C &gt;</a>	
Wrapper template for making <a href="#">GiNaC</a> classes out of STL containers . . . . .	467
<a href="#">GiNaC::container_storage&lt; C &gt;</a>	
Helper template for encapsulating the <a href="#">reserve()</a> mechanics of STL containers . . . . .	485
<a href="#">GiNaC::composition_generator::coolmulti</a> . . . . .	489
<a href="#">GiNaC::derivative_map_function</a>	
Function object to be applied by <a href="#">basic::derivative()</a> . . . . .	491
<a href="#">GiNaC::determinant_algo</a>	
Switch to control algorithm for determinant computation . . . . .	493
<a href="#">GiNaC::diracgamma</a>	
This class represents the Dirac gamma Lorentz vector . . . . .	494
<a href="#">GiNaC::diracgamma5</a>	
This class represents the Dirac gamma5 object which anticommutes with all other gammas . . . . .	501
<a href="#">GiNaC::diracgammaL</a>	
This class represents the Dirac gammaL object which behaves like $1/2 (1-\text{gamma5})$ . . . . .	506
<a href="#">GiNaC::diracgammaR</a>	
This class represents the Dirac gammaL object which behaves like $1/2 (1+\text{gamma5})$ . . . . .	511
<a href="#">GiNaC::diracone</a>	
This class represents the Clifford algebra unity element . . . . .	516
<a href="#">GiNaC::do_taylor</a>	
Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe . . . . .	521
<a href="#">GiNaC::domain</a>	
Domain of an object . . . . .	521
<a href="#">GiNaC::dunno</a>	
Exception class thrown by functions to signal unimplemented functionality so the expression may just be <code>.hold()</code> . . . . .	522
<a href="#">GiNaC::Ebar_kernel</a>	
The Ebar-kernel . . . . .	522
<a href="#">GiNaC::Eisenstein_h_kernel</a>	
The kernel corresponding to the Eisenstein series $h_{k,N,r,s}(\tau)$ . . . . .	531
<a href="#">GiNaC::Eisenstein_kernel</a>	
The kernel corresponding to the Eisenstein series $E_{k,N,a,b,K}(\tau)$ . . . . .	542
<a href="#">GiNaC::composition_generator::coolmulti::element</a> . . . . .	552
<a href="#">GiNaC::ELi_kernel</a>	
The ELi-kernel . . . . .	554
<a href="#">std::equal_to&lt; GiNaC::ex &gt;</a>	
Specialization of <code>std::equal_to()</code> for <code>ex</code> objects . . . . .	562
<a href="#">GiNaC::error_and_integral</a> . . . . .	564
<a href="#">GiNaC::error_and_integral_is_less</a> . . . . .	565
<a href="#">GiNaC::eval_integ_map_function</a>	
Function object to be applied by <a href="#">basic::eval_integ()</a> . . . . .	566
<a href="#">GiNaC::evalf_map_function</a>	
Function object to be applied by <a href="#">basic::evalf()</a> . . . . .	567
<a href="#">GiNaC::evalm_map_function</a>	
Function object to be applied by <a href="#">basic::evalm()</a> . . . . .	569

<a href="#">GiNaC::ex</a>	
Lightweight wrapper for <a href="#">GiNaC</a> 's symbolic objects	570
<a href="#">GiNaC::ex_base_is_less</a>	605
<a href="#">GiNaC::ex_is_equal</a>	605
<a href="#">GiNaC::ex_is_less</a>	606
<a href="#">GiNaC::ex_swap</a>	606
<a href="#">GiNaC::expair</a>	
A pair of expressions	607
<a href="#">GiNaC::expair_is_less</a>	
Function object for insertion into third argument of STL's sort() etc	610
<a href="#">GiNaC::expair_rest_is_less</a>	
Function object not caring about the numerical coefficients for insertion into third argument of STL's sort()	611
<a href="#">GiNaC::expair_swap</a>	612
<a href="#">GiNaC::expairseq</a>	
A sequence of class expair	613
<a href="#">GiNaC::expand_map_function</a>	
Function object to be applied by <a href="#">basic::expand()</a>	634
<a href="#">GiNaC::expand_options</a>	
Flags to control the behavior of <a href="#">expand()</a>	636
<a href="#">GiNaC::factor_options</a>	
Flags to control the polynomial factorization	637
<a href="#">GiNaC::fail</a>	638
<a href="#">GiNaC::fderivative</a>	
This class represents the (abstract) derivative of a symbolic function	643
<a href="#">GiNaC::function</a>	
The class function is used to implement builtin functions like sin, cos... and user defined functions	658
<a href="#">GiNaC::function_options</a>	680
<a href="#">GiNaC::G2_SERIAL</a>	
Generalized multiple polylogarithm	727
<a href="#">GiNaC::G3_SERIAL</a>	
Generalized multiple polylogarithm with explicit imaginary parts	728
<a href="#">GiNaC::gcd_options</a>	
Flags to control the behavior of <a href="#">gcd()</a> and friends	728
<a href="#">GiNaC::gcdheu_failed</a>	
Exception thrown by <a href="#">heur_gcd()</a> to signal failure	729
<a href="#">GiNaC::has_distance&lt; T &gt;</a>	
SFINAE test for distance	729
<a href="#">GiNaC::has_options</a>	
Flags to control the behavior of <a href="#">has()</a>	731
<a href="#">std::hash&lt; GiNaC::ex &gt;</a>	
Specialization of std::hash() for ex objects	732
<a href="#">GiNaC::idx</a>	
This class holds one index of an indexed object	733
<a href="#">GiNaC::idx_is_equal_ignore_dim</a>	746
<a href="#">GiNaC::indexed</a>	
This class holds an indexed expression	747
<a href="#">GiNaC::info_flags</a>	
Possible attributes an object can have	769
<a href="#">GiNaC::integral</a>	
Symbolic integral	771
<a href="#">GiNaC::integration_kernel</a>	
The base class for integration kernels for iterated integrals	783
<a href="#">GiNaC::is_not_a_clifford</a>	
Predicate for finding non-clifford objects	792
<a href="#">GiNaC::is_summation_idx</a>	793
<a href="#">GiNaC::iterated_integral2_SERIAL</a>	
Complete elliptic integral of the first kind	794

<a href="#">GiNaC::iterated_integral3_SERIAL</a>	
Iterated integral with explicit truncation	794
<a href="#">GiNaC::Kronecker_dtau_kernel</a>	
The kernel corresponding to integrating the Kronecker coefficient function $g^{(n)}(z_j, K\tau)$ in $\tau$ (or equivalently in $\bar{q}$ )	795
<a href="#">GiNaC::Kronecker_dz_kernel</a>	
The kernel corresponding to integrating the Kronecker coefficient function $g^{(n-1)}(z - z_j, K\tau)$ in $z$	804
<a href="#">GiNaC::lanczos_coeffs</a>	813
<a href="#">std::less&lt; GiNaC::ptr&lt; T &gt; &gt;</a>	
Specialization of <code>std::less</code> for <code>ptr&lt;T&gt;</code> to enable ordering of <code>ptr&lt;T&gt;</code> objects (e.g	814
<a href="#">GiNaC::library_init</a>	
Helper class to initialize the library	815
<a href="#">GiNaC::make_flat_inserter</a>	
Class to handle the renaming of dummy indices	817
<a href="#">GiNaC::map_function</a>	
Function object for <code>map()</code>	820
<a href="#">GiNaC::matrix</a>	
Symbolic matrices	822
<a href="#">GiNaC::minkmetric</a>	
This class represents a Minkowski metric tensor	847
<a href="#">GiNaC::modular_form_kernel</a>	
A kernel corresponding to a polynomial in Eisenstein series	855
<a href="#">GiNaC::basic_partition_generator::mpartition2</a>	864
<a href="#">GiNaC::mul</a>	
Product of expressions	866
<a href="#">GiNaC::multi_iterator_counter&lt; T &gt;</a>	
The class <code>multi_iterator_counter</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	889
<a href="#">GiNaC::multi_iterator_counter_indv&lt; T &gt;</a>	
The class <code>multi_iterator_counter_indv</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	892
<a href="#">GiNaC::multi_iterator_ordered&lt; T &gt;</a>	
The class <code>multi_iterator_ordered</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	896
<a href="#">GiNaC::multi_iterator_ordered_eq&lt; T &gt;</a>	
The class <code>multi_iterator_ordered_eq</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	900
<a href="#">GiNaC::multi_iterator_ordered_eq_indv&lt; T &gt;</a>	
The class <code>multi_iterator_ordered_eq_indv</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , such that	904
<a href="#">GiNaC::multi_iterator_permutation&lt; T &gt;</a>	
The class <code>multi_iterator_permutation</code> defines a <code>multi_iterator</code> $(i_1, i_2, \dots, i_k)$ , for which	908
<a href="#">GiNaC::multi_iterator_shuffle&lt; T &gt;</a>	
The class <code>multi_iterator_shuffle</code> defines a <code>multi_iterator</code> , which runs over all shuffles of a and b	912
<a href="#">GiNaC::multi_iterator_shuffle_prime&lt; T &gt;</a>	
The class <code>multi_iterator_shuffle_prime</code> defines a <code>multi_iterator</code> , which runs over all shuffles of a and b, excluding the first one (a,b)	917
<a href="#">GiNaC::multiple_polylog_kernel</a>	
The integration kernel for multiple polylogarithms	921
<a href="#">GiNaC::ncmul</a>	
Non-commutative product of expressions	929
<a href="#">GiNaC::normal_map_function</a>	
Function object to be applied by <code>basic::normal()</code>	943
<a href="#">GiNaC::numeric</a>	
This class is a wrapper around CLN-numbers within the <code>GiNaC</code> class hierarchy	945
<a href="#">GiNaC::op0_is_equal</a>	977
<a href="#">GiNaC::partition_generator</a>	
Generate all bounded combinatorial partitions of an integer n with exactly m parts (not including zero parts) in non-decreasing order	977
<a href="#">GiNaC::partition_with_zero_parts_generator</a>	
Generate all bounded combinatorial partitions of an integer n with exactly m parts (including zero parts) in non-decreasing order	980

<a href="#">GiNaC::pointer_to_map_function</a>	983
<a href="#">GiNaC::pointer_to_map_function_1arg&lt; T1 &gt;</a>	985
<a href="#">GiNaC::pointer_to_map_function_2args&lt; T1, T2 &gt;</a>	988
<a href="#">GiNaC::pointer_to_map_function_3args&lt; T1, T2, T3 &gt;</a>	990
<a href="#">GiNaC::pointer_to_member_to_map_function&lt; C &gt;</a>	993
<a href="#">GiNaC::pointer_to_member_to_map_function_1arg&lt; C, T1 &gt;</a>	996
<a href="#">GiNaC::pointer_to_member_to_map_function_2args&lt; C, T1, T2 &gt;</a>	998
<a href="#">GiNaC::pointer_to_member_to_map_function_3args&lt; C, T1, T2, T3 &gt;</a>	1001
<a href="#">GiNaC::pole_error</a>	
Exception class thrown when a singularity is encountered	1005
<a href="#">GiNaC::possymbol</a>	
Specialization of symbol to real positive domain	1007
<a href="#">GiNaC::power</a>	
This class holds a two-component object, a basis and and exponent representing exponentiation	1014
<a href="#">GiNaC::print_context</a>	
Base class for print_contexts	1032
<a href="#">GiNaC::print_context_options</a>	
This class stores information about a registered <a href="#">print_context</a> class	1034
<a href="#">GiNaC::print_csrc</a>	
Base context for C source output	1036
<a href="#">GiNaC::print_csrc_cl_N</a>	
Context for C source output using CLN numbers	1038
<a href="#">GiNaC::print_csrc_double</a>	
Context for C source output using double precision	1040
<a href="#">GiNaC::print_csrc_float</a>	
Context for C source output using float precision	1042
<a href="#">GiNaC::print_dflt</a>	
Context for default (ginsh-parsable) output	1044
<a href="#">GiNaC::print_functor</a>	
This class represents a print method for a certain algebraic class and <a href="#">print_context</a> type	1045
<a href="#">GiNaC::print_functor_impl</a>	
Base class for <a href="#">print_functor</a> handlers	1048
<a href="#">GiNaC::print_latex</a>	
Context for latex-parsable output	1049
<a href="#">GiNaC::print_memfun_handler&lt; T, C &gt;</a>	
Print_functor handler for member functions of class T, context type C	1051
<a href="#">GiNaC::print_options</a>	
Flags to control the behavior of a <a href="#">print_context</a>	1053
<a href="#">GiNaC::print_ptrfun_handler&lt; T, C &gt;</a>	
Print_functor handler for pointer-to-functions of class T, context type C	1055
<a href="#">GiNaC::print_python</a>	
Context for python pretty-print output	1058
<a href="#">GiNaC::print_python_repr</a>	
Context for python-parsable output	1059
<a href="#">GiNaC::print_tree</a>	
Context for tree-like output for debugging	1061
<a href="#">GiNaC::archive_node::property</a>	
Archived property (data type, name and associated data)	1063
<a href="#">GiNaC::archive_node::property_info</a>	
Information about a stored property	1064
<a href="#">GiNaC::pseries</a>	
This class holds a extended truncated power series (positive and negative integer powers)	1066
<a href="#">GiNaC::psi1_SERIAL</a>	
Polylogarithm and multiple polylogarithm	1085
<a href="#">GiNaC::psi2_SERIAL</a>	
Derivatives of Psi-function (aka polygamma-functions)	1086
<a href="#">GiNaC::ptr&lt; T &gt;</a>	
Class of (intrusively) reference-counted pointers that support copy-on-write semantics	1087

<a href="#">GiNaC::realsymbol</a>	
Specialization of symbol to real domain	1092
<a href="#">GiNaC::refcounted</a>	
Base class for reference-counted objects	1100
<a href="#">GiNaC::registered_class_options</a>	
This class stores information about a registered <a href="#">GiNaC</a> class	1102
<a href="#">GiNaC::relational</a>	
This class holds a relation consisting of two expressions and a logical relation between them	1106
<a href="#">GiNaC::remember_strategies</a>	
Strategies how to clean up the function remember cache	1118
<a href="#">GiNaC::remember_table</a>	
The remember table is organized like an n-fold associative cache in a microprocessor	1120
<a href="#">GiNaC::remember_table_entry</a>	
A single entry in the remember table of a function	1124
<a href="#">GiNaC::remember_table_list</a>	
A list of entries in the remember table having some least significant bits of the hashvalue in common	1127
<a href="#">GiNaC::return_type_t</a>	
To distinguish between different kinds of non-commutative objects	1129
<a href="#">GiNaC::return_types</a>	1131
<a href="#">GiNaC::relational::safe_bool_helper</a>	1131
<a href="#">GiNaC::scalar_products</a>	
Helper class for storing information about known scalar products which are to be automatically replaced by <a href="#">simplify_indexed()</a>	1132
<a href="#">GiNaC::series_options</a>	
Flags to control series expansion	1135
<a href="#">GiNaC::solve_algo</a>	
Switch to control algorithm for linear system solving	1135
<a href="#">GiNaC::spinidx</a>	
This class holds a spinor index that can be dotted or undotted and that also has a variance	1137
<a href="#">GiNaC::spinmetric</a>	
This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors	1148
<a href="#">GiNaC::spmapkey</a>	1155
<a href="#">GiNaC::status_flags</a>	
Flags to store information about the state of an object	1158
<a href="#">GiNaC::structure&lt; T, ComparisonPolicy &gt;</a>	
Wrapper template for making <a href="#">GiNaC</a> classes out of C++ structures	1159
<a href="#">GiNaC::su3d</a>	
This class represents the tensor of symmetric su(3) structure constants	1178
<a href="#">GiNaC::su3f</a>	
This class represents the tensor of antisymmetric su(3) structure constants	1184
<a href="#">GiNaC::su3one</a>	
This class represents the su(3) unity element	1190
<a href="#">GiNaC::su3t</a>	
This class represents an su(3) generator	1195
<a href="#">GiNaC::subs_options</a>	
Flags to control the behavior of <a href="#">subs()</a>	1200
<a href="#">GiNaC::sy_is_less</a>	1201
<a href="#">GiNaC::sy_swap</a>	1202
<a href="#">GiNaC::sym_desc</a>	
This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b"	1203
<a href="#">GiNaC::symbol</a>	
Basic CAS symbol	1207
<a href="#">GiNaC::symbolset</a>	1220
<a href="#">GiNaC::symmetry</a>	
This class describes the symmetry of a group of indices	1222

<a href="#">GiNaC::symminfo</a>	
This structure stores the individual symmetrized terms obtained during the simplification of sums	<a href="#">1232</a>
<a href="#">GiNaC::symminfo_is_less_by_orig</a>	<a href="#">1235</a>
<a href="#">GiNaC::symminfo_is_less_by_symmterm</a>	<a href="#">1235</a>
<a href="#">GiNaC::tensdelta</a>	
This class represents the delta tensor	<a href="#">1236</a>
<a href="#">GiNaC::tensepsilon</a>	
This class represents the totally antisymmetric epsilon tensor	<a href="#">1242</a>
<a href="#">GiNaC::tensmetric</a>	
This class represents a general metric tensor which can be used to raise/lower indices	<a href="#">1250</a>
<a href="#">GiNaC::tensor</a>	
This class holds one of <a href="#">GiNaC</a> 's predefined special tensors such as the delta and the metric tensors	<a href="#">1257</a>
<a href="#">GiNaC::terminfo</a>	
This structure stores the original and symmetrized versions of terms obtained during the simplification of sums	<a href="#">1262</a>
<a href="#">GiNaC::terminfo_is_less</a>	<a href="#">1263</a>
<a href="#">GiNaC::class_info&lt; OPT &gt;::tree_node</a>	<a href="#">1264</a>
<a href="#">GiNaC::unarchive_table_t</a>	<a href="#">1265</a>
<a href="#">GiNaC::user_defined_kernel</a>	
A user-defined integration kernel	<a href="#">1267</a>
<a href="#">GiNaC::varidx</a>	
This class holds an index with a variance (co- or contravariant)	<a href="#">1276</a>
<a href="#">GiNaC::visitor</a>	
Degenerate base class for visitors	<a href="#">1287</a>
<a href="#">GiNaC::wildcard</a>	
This class acts as a wildcard for <a href="#">subs()</a> , <a href="#">match()</a> , <a href="#">has()</a> and <a href="#">find()</a>	<a href="#">1288</a>
<a href="#">GiNaC::zeta1_SERIAL</a>	
Complex conjugate	<a href="#">1295</a>
<a href="#">GiNaC::zeta2_SERIAL</a>	
Alternating Euler sum or colored MZV	<a href="#">1295</a>





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">add.cpp</a>	Implementation of <a href="#">GiNaC</a> 's sums of expressions . . . . .	1297
<a href="#">add.h</a>	Interface to <a href="#">GiNaC</a> 's sums of expressions . . . . .	1298
<a href="#">archive.cpp</a>	Archiving of <a href="#">GiNaC</a> expressions . . . . .	1298
<a href="#">archive.h</a>	Archiving of <a href="#">GiNaC</a> expressions . . . . .	1299
<a href="#">assertion.h</a>	Assertion macro definition . . . . .	1302
<a href="#">basic.cpp</a>	Implementation of <a href="#">GiNaC</a> 's ABC . . . . .	1302
<a href="#">basic.h</a>	Interface to <a href="#">GiNaC</a> 's ABC . . . . .	1303
<a href="#">class_info.h</a>	Helper templates to provide per-class information for class hierarchies . . . . .	1304
<a href="#">clifford.cpp</a>	Implementation of <a href="#">GiNaC</a> 's clifford algebra (Dirac gamma) objects . . . . .	1305
<a href="#">clifford.h</a>	Interface to <a href="#">GiNaC</a> 's clifford algebra (Dirac gamma) objects . . . . .	1307
<a href="#">color.cpp</a>	Implementation of <a href="#">GiNaC</a> 's color (SU(3) Lie algebra) objects . . . . .	1309
<a href="#">color.h</a>	Interface to <a href="#">GiNaC</a> 's color (SU(3) Lie algebra) objects . . . . .	1311
<a href="#">compiler.h</a>	Definition of optimizing macros . . . . .	1312
<a href="#">constant.cpp</a>	Implementation of <a href="#">GiNaC</a> 's constant types and some special constants . . . . .	1313
<a href="#">constant.h</a>	Interface to <a href="#">GiNaC</a> 's constant types and some special constants . . . . .	1314
<a href="#">container.h</a>	Wrapper template for making <a href="#">GiNaC</a> classes out of STL containers . . . . .	1315
<a href="#">crc32.h</a>	CRC32 hash function . . . . .	1315
<a href="#">ex.cpp</a>	Implementation of <a href="#">GiNaC</a> 's light-weight expression handles . . . . .	1316

<a href="#">ex.h</a>	Interface to <a href="#">GiNaC</a> 's light-weight expression handles	1316
<a href="#">excompiler.cpp</a>	Functions to facilitate the conversion of a <a href="#">ex</a> to a function pointer suited for fast numerical integration	1319
<a href="#">excompiler.h</a>	Functions to facilitate the conversion of a <a href="#">ex</a> to a function pointer suited for fast numerical integration	1319
<a href="#">expair.cpp</a>	Implementation of expression pairs (building blocks of <a href="#">expairseq</a> )	1320
<a href="#">expair.h</a>	Definition of expression pairs (building blocks of <a href="#">expairseq</a> )	1321
<a href="#">expairseq.cpp</a>	Implementation of sequences of expression pairs	1322
<a href="#">expairseq.h</a>	Interface to sequences of expression pairs	1322
<a href="#">exprseq.cpp</a>	Implementation of <a href="#">GiNaC</a> 's <a href="#">exprseq</a>	1323
<a href="#">exprseq.h</a>	Definition of <a href="#">GiNaC</a> 's <a href="#">exprseq</a>	1324
<a href="#">factor.cpp</a>	Polynomial factorization (implementation)	1324
<a href="#">factor.h</a>	Polynomial factorization	1333
<a href="#">fail.cpp</a>	Implementation of class signaling failure of operation	1333
<a href="#">fail.h</a>	Interface to class signaling failure of operation	1334
<a href="#">fderivative.cpp</a>	Implementation of abstract derivatives of functions	1335
<a href="#">fderivative.h</a>	Interface to abstract derivatives of functions	1335
<a href="#">flags.h</a>	Collection of all flags used through the <a href="#">GiNaC</a> framework	1336
<a href="#">function.cpp</a>	Implementation of class of symbolic functions	1337
<a href="#">function.h</a>	Interface to class of symbolic functions	1337
<a href="#">ginac.h</a>	This include file includes all other public <a href="#">GiNaC</a> headers	1348
<a href="#">hash_map.h</a>	Replacement for <code>map&lt;&gt;</code> using hash tables	1349
<a href="#">hash_seed.h</a>	Type-specific hash seed	1350
<a href="#">idx.cpp</a>	Implementation of <a href="#">GiNaC</a> 's indices	1350
<a href="#">idx.h</a>	Interface to <a href="#">GiNaC</a> 's indices	1351
<a href="#">indexed.cpp</a>	Implementation of <a href="#">GiNaC</a> 's indexed expressions	1352
<a href="#">indexed.h</a>	Interface to <a href="#">GiNaC</a> 's indexed expressions	1354
<a href="#">inifcns.cpp</a>	Implementation of <a href="#">GiNaC</a> 's initially known functions	1355
<a href="#">inifcns.h</a>	Interface to <a href="#">GiNaC</a> 's initially known functions	1358
<a href="#">inifcns_elliptic.cpp</a>	Implementation of some special functions related to elliptic curves	1360

<a href="#">inifcns_gamma.cpp</a>	Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff	1361
<a href="#">inifcns_nstdsums.cpp</a>	Implementation of some special functions that have a representation as nested sums . . . . .	1362
<a href="#">inifcns_trans.cpp</a>	Implementation of transcendental (and trigonometric and hyperbolic) functions . . . . .	1364
<a href="#">integral.cpp</a>	Implementation of <a href="#">GiNaC</a> 's symbolic integral . . . . .	1367
<a href="#">integral.h</a>	Interface to <a href="#">GiNaC</a> 's symbolic integral . . . . .	1368
<a href="#">integration_kernel.cpp</a>	Implementation of <a href="#">GiNaC</a> 's integration kernels for iterated integrals . . . . .	1369
<a href="#">integration_kernel.h</a>	Interface to <a href="#">GiNaC</a> 's integration kernels for iterated integrals . . . . .	1371
<a href="#">lst.cpp</a>	Implementation of <a href="#">GiNaC</a> 's lst . . . . .	1373
<a href="#">lst.h</a>	Definition of <a href="#">GiNaC</a> 's lst . . . . .	1373
<a href="#">matrix.cpp</a>	Implementation of symbolic matrices . . . . .	1374
<a href="#">matrix.h</a>	Interface to symbolic matrices . . . . .	1375
<a href="#">mul.cpp</a>	Implementation of <a href="#">GiNaC</a> 's products of expressions . . . . .	1376
<a href="#">mul.h</a>	Interface to <a href="#">GiNaC</a> 's products of expressions . . . . .	1377
<a href="#">ncmul.cpp</a>	Implementation of <a href="#">GiNaC</a> 's non-commutative products of expressions . . . . .	1378
<a href="#">ncmul.h</a>	Interface to <a href="#">GiNaC</a> 's non-commutative products of expressions . . . . .	1379
<a href="#">normal.cpp</a>	This file implements several functions that work on univariate and multivariate polynomials and rational functions . . . . .	1379
<a href="#">normal.h</a>	This file defines several functions that work on univariate and multivariate polynomials and rational functions . . . . .	1382
<a href="#">numeric.cpp</a>	This file contains the interface to the underlying bignum package . . . . .	1383
<a href="#">numeric.h</a>	Makes the interface to the underlying bignum package available . . . . .	1387
<a href="#">operators.cpp</a>	Implementation of <a href="#">GiNaC</a> 's overloaded operators . . . . .	1389
<a href="#">operators.h</a>	Interface to <a href="#">GiNaC</a> 's overloaded operators . . . . .	1391
<a href="#">power.cpp</a>	Implementation of <a href="#">GiNaC</a> 's symbolic exponentiation (basis <sup>exponent</sup> ) . . . . .	1393
<a href="#">power.h</a>	Interface to <a href="#">GiNaC</a> 's symbolic exponentiation (basis <sup>exponent</sup> ) . . . . .	1394
<a href="#">print.cpp</a>	Implementation of helper classes for expression output . . . . .	1394
<a href="#">print.h</a>	Definition of helper classes for expression output . . . . .	1395
<a href="#">pseries.cpp</a>	Implementation of class for extended truncated power series and methods for series expansion	1398
<a href="#">pseries.h</a>	Interface to class for extended truncated power series . . . . .	1398
<a href="#">ptr.h</a>	Reference-counted pointer template . . . . .	1399

<a href="#">registrar.cpp</a>	
GiNaC's class registrar (for class basic and all classes derived from it)	1400
<a href="#">registrar.h</a>	
GiNaC's class registrar (for class basic and all classes derived from it)	1400
<a href="#">relational.cpp</a>	
Implementation of relations between expressions	1404
<a href="#">relational.h</a>	
Interface to relations between expressions	1404
<a href="#">remember.cpp</a>	
Implementation of helper classes for using the remember option in GiNaC functions	1405
<a href="#">remember.h</a>	
Interface to helper classes for using the remember option in GiNaC functions	1405
<a href="#">structure.h</a>	
Wrapper template for making GiNaC classes out of C++ structures	1406
<a href="#">symbol.cpp</a>	
Implementation of GiNaC's symbolic objects	1407
<a href="#">symbol.h</a>	
Interface to GiNaC's symbolic objects	1407
<a href="#">symmetry.cpp</a>	
Implementation of GiNaC's symmetry definitions	1408
<a href="#">symmetry.h</a>	
Interface to GiNaC's symmetry definitions	1409
<a href="#">tensor.cpp</a>	
Implementation of GiNaC's special tensors	1411
<a href="#">tensor.h</a>	
Interface to GiNaC's special tensors	1412
<a href="#">utils.cpp</a>	
Implementation of several small and furry utilities needed within GiNaC but not of any interest to the user of the library	1413
<a href="#">utils.h</a>	
Interface to several small and furry utilities needed within GiNaC but not of any interest to the user of the library	1415
<a href="#">utils_multi_iterator.h</a>	
Utilities for summing over multiple indices	1418
<a href="#">version.h</a>	
GiNaC library version information	1419
<a href="#">wildcard.cpp</a>	
Implementation of GiNaC's wildcard objects	1421
<a href="#">wildcard.h</a>	
Interface to GiNaC's wildcard objects	1422

## Chapter 5

# Namespace Documentation

### 5.1 GiNaC Namespace Reference

#### Namespaces

- namespace [internal](#)

#### Classes

- class [\\_numeric\\_digits](#)  
*This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.*
- class [add](#)  
*Sum of expressions.*
- class [archive](#)  
*This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).*
- class [archive\\_node](#)  
*This class stores all properties needed to record/retrieve the state of one object of class [basic](#) (or a derived class).*
- class [basic](#)  
*This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.*
- class [basic\\_log\\_kernel](#)  
*The basic integration kernel with a logarithmic singularity at the origin.*
- class [basic\\_multi\\_iterator](#)  
*[basic\\_multi\\_iterator](#) is a base class.*
- class [basic\\_partition\\_generator](#)  
*Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.*
- class [class\\_info](#)
- class [clifford](#)  
*This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).*
- class [cliffordunit](#)  
*This class represents the Clifford algebra generators (units).*
- class [color](#)  
*This class holds a generator  $T_a$  or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.*
- class [compare\\_all\\_equal](#)

- Comparison policy: all structures of one type are equal.*

  - class [compare\\_bitwise](#)

*Comparison policy: use bit-wise comparison to compare structures.*
  - class [compare\\_std\\_less](#)

*Comparison policy: use `std::equal_to`/`std::less` (defaults to operators `==` and `<`) to compare structures.*
  - class [composition\\_generator](#)

*Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.*
  - class [const\\_iterator](#)
  - class [const\\_postorder\\_iterator](#)
  - class [const\\_preorder\\_iterator](#)
  - class [constant](#)

*This class holds constants, symbols with specific numerical value.*
  - class [container](#)

*Wrapper template for making [GiNaC](#) classes out of STL containers.*
  - class [container\\_storage](#)

*Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.*
  - struct [derivative\\_map\\_function](#)

*Function object to be applied by [basic::derivative\(\)](#).*
  - class [determinant\\_algo](#)

*Switch to control algorithm for determinant computation.*
  - class [diracgamma](#)

*This class represents the Dirac gamma Lorentz vector.*
  - class [diracgamma5](#)

*This class represents the Dirac gamma5 object which anticommutes with all other gammas.*
  - class [diracgammaL](#)

*This class represents the Dirac gammaL object which behaves like  $1/2 (1-\text{gamma5})$ .*
  - class [diracgammaR](#)

*This class represents the Dirac gammaL object which behaves like  $1/2 (1+\text{gamma5})$ .*
  - class [diracone](#)

*This class represents the Clifford algebra unity element.*
  - class [do\\_taylor](#)

*Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.*
  - class [domain](#)

*Domain of an object.*
  - class [dunno](#)

*Exception class thrown by functions to signal unimplemented functionality so the expression may just be `.hold()`*
  - class [Ebar\\_kernel](#)

*The Ebar-kernel.*
  - class [Eisenstein\\_h\\_kernel](#)

*The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .*
  - class [Eisenstein\\_kernel](#)

*The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .*
  - class [ELi\\_kernel](#)

*The ELi-kernel.*
  - struct [error\\_and\\_integral](#)
  - struct [error\\_and\\_integral\\_is\\_less](#)
  - struct [eval\\_integ\\_map\\_function](#)

*Function object to be applied by [basic::eval\\_integ\(\)](#).*
  - struct [evalf\\_map\\_function](#)

- Function object to be applied by `basic::evalf()`.
- struct `evalm_map_function`
  - Function object to be applied by `basic::evalm()`.
- class `ex`
  - Lightweight wrapper for GiNaC's symbolic objects.
- struct `ex_base_is_less`
- struct `ex_is_equal`
- struct `ex_is_less`
- struct `ex_swap`
- class `expair`
  - A pair of expressions.
- struct `expair_is_less`
  - Function object for insertion into third argument of STL's `sort()` etc.
- struct `expair_rest_is_less`
  - Function object not caring about the numerical coefficients for insertion into third argument of STL's `sort()`.
- struct `expair_swap`
- class `expairseq`
  - A sequence of class `expair`.
- struct `expand_map_function`
  - Function object to be applied by `basic::expand()`.
- class `expand_options`
  - Flags to control the behavior of `expand()`.
- class `factor_options`
  - Flags to control the polynomial factorization.
- class `fail`
- class `fderivative`
  - This class represents the (abstract) derivative of a symbolic function.
- class `function`
  - The class `function` is used to implement builtin functions like `sin`, `cos`... and user defined functions.
- class `function_options`
- class `G2_SERIAL`
  - Generalized multiple polylogarithm.
- class `G3_SERIAL`
  - Generalized multiple polylogarithm with explicit imaginary parts.
- struct `gcd_options`
  - Flags to control the behavior of `gcd()` and friends.
- class `gcdheu_failed`
  - Exception thrown by `heur_gcd()` to signal failure.
- class `has_distance`
  - SFINAE test for distance.
- class `has_options`
  - Flags to control the behavior of `has()`.
- class `idx`
  - This class holds one index of an indexed object.
- struct `idx_is_equal_ignore_dim`
- class `indexed`
  - This class holds an indexed expression.
- class `info_flags`
  - Possible attributes an object can have.
- class `integral`
  - Symbolic integral.

- class [integration\\_kernel](#)  
The base class for integration kernels for iterated integrals.
- struct [is\\_not\\_a\\_clifford](#)  
Predicate for finding non-clifford objects.
- struct [is\\_summation\\_idx](#)
- class [iterated\\_integral2\\_SERIAL](#)  
Complete elliptic integral of the first kind.
- class [iterated\\_integral3\\_SERIAL](#)  
Iterated integral with explicit truncation.
- class [Kronecker\\_dtau\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).
- class [Kronecker\\_dz\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .
- class [lanczos\\_coeffs](#)
- class [library\\_init](#)  
Helper class to initialize the library.
- class [make\\_flat\\_inserter](#)  
Class to handle the renaming of dummy indices.
- struct [map\\_function](#)  
Function object for `map()`.
- class [matrix](#)  
Symbolic matrices.
- class [minkmetric](#)  
This class represents a Minkowski metric tensor.
- class [modular\\_form\\_kernel](#)  
A kernel corresponding to a polynomial in Eisenstein series.
- class [mul](#)  
Product of expressions.
- class [multi\\_iterator\\_counter](#)  
The class [multi\\_iterator\\_counter](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.
- class [multi\\_iterator\\_counter\\_indv](#)  
The class [multi\\_iterator\\_counter\\_indv](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.
- class [multi\\_iterator\\_ordered](#)  
The class [multi\\_iterator\\_ordered](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.
- class [multi\\_iterator\\_ordered\\_eq](#)  
The class [multi\\_iterator\\_ordered\\_eq](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.
- class [multi\\_iterator\\_ordered\\_eq\\_indv](#)  
The class [multi\\_iterator\\_ordered\\_eq\\_indv](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.
- class [multi\\_iterator\\_permutation](#)  
The class [multi\\_iterator\\_permutation](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , for which.
- class [multi\\_iterator\\_shuffle](#)  
The class [multi\\_iterator\\_shuffle](#) defines a `multi_iterator`, which runs over all shuffles of  $a$  and  $b$ .
- class [multi\\_iterator\\_shuffle\\_prime](#)  
The class [multi\\_iterator\\_shuffle\\_prime](#) defines a `multi_iterator`, which runs over all shuffles of  $a$  and  $b$ , excluding the first one  $(a,b)$ .
- class [multiple\\_polylog\\_kernel](#)  
The integration kernel for multiple polylogarithms.
- class [ncmul](#)  
Non-commutative product of expressions.
- struct [normal\\_map\\_function](#)  
Function object to be applied by `basic::normal()`.



- class [numeric](#)  
*This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.*
- struct [op0\\_is\\_equal](#)
- class [partition\\_generator](#)  
*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.*
- class [partition\\_with\\_zero\\_parts\\_generator](#)  
*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.*
- class [pointer\\_to\\_map\\_function](#)
- class [pointer\\_to\\_map\\_function\\_1arg](#)
- class [pointer\\_to\\_map\\_function\\_2args](#)
- class [pointer\\_to\\_map\\_function\\_3args](#)
- class [pointer\\_to\\_member\\_to\\_map\\_function](#)
- class [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg](#)
- class [pointer\\_to\\_member\\_to\\_map\\_function\\_2args](#)
- class [pointer\\_to\\_member\\_to\\_map\\_function\\_3args](#)
- class [pole\\_error](#)  
*Exception class thrown when a singularity is encountered.*
- class [possymbol](#)  
*Specialization of symbol to real positive domain.*
- class [power](#)  
*This class holds a two-component object, a basis and an exponent representing exponentiation.*
- class [print\\_context](#)  
*Base class for print\_contexts.*
- class [print\\_context\\_options](#)  
*This class stores information about a registered [print\\_context](#) class.*
- class [print\\_csrc](#)  
*Base context for C source output.*
- class [print\\_csrc\\_cl\\_N](#)  
*Context for C source output using CLN numbers.*
- class [print\\_csrc\\_double](#)  
*Context for C source output using double precision.*
- class [print\\_csrc\\_float](#)  
*Context for C source output using float precision.*
- class [print\\_dflt](#)  
*Context for default (ginsh-parsable) output.*
- class [print\\_functor](#)  
*This class represents a print method for a certain algebraic class and [print\\_context](#) type.*
- class [print\\_functor\\_impl](#)  
*Base class for [print\\_functor](#) handlers.*
- class [print\\_latex](#)  
*Context for latex-parsable output.*
- class [print\\_memfun\\_handler](#)  
*[print\\_functor](#) handler for member functions of class  $T$ , context type  $C$*
- class [print\\_options](#)  
*Flags to control the behavior of a [print\\_context](#).*
- class [print\\_ptrfun\\_handler](#)  
*[print\\_functor](#) handler for pointer-to-functions of class  $T$ , context type  $C$*
- class [print\\_python](#)  
*Context for python pretty-print output.*
- class [print\\_python\\_repr](#)

- Context for python-parsable output.*

  - class [print\\_tree](#)
- Context for tree-like output for debugging.*

  - class [pseries](#)
- This class holds a extended truncated power series (positive and negative integer powers).*

  - class [psi1\\_SERIAL](#)
- Polylogarithm and multiple polylogarithm.*

  - class [psi2\\_SERIAL](#)
- Derivatives of Psi-function (aka polygamma-functions).*

  - class [ptr](#)
- Class of (intrusively) reference-counted pointers that support copy-on-write semantics.*

  - class [realsymbol](#)
- Specialization of symbol to real domain.*

  - class [refcounted](#)
- Base class for reference-counted objects.*

  - class [registered\\_class\\_options](#)
- This class stores information about a registered [GiNaC](#) class.*

  - class [relational](#)
- This class holds a relation consisting of two expressions and a logical relation between them.*

  - class [remember\\_strategies](#)
- Strategies how to clean up the function remember cache.*

  - class [remember\\_table](#)
- The remember table is organized like an n-fold associative cache in a microprocessor.*

  - class [remember\\_table\\_entry](#)
- A single entry in the remember table of a function.*

  - class [remember\\_table\\_list](#)
- A list of entries in the remember table having some least significant bits of the hashvalue in common.*

  - struct [return\\_type\\_t](#)
- To distinguish between different kinds of non-commutative objects.*

  - class [return\\_types](#)
- Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).*

  - class [scalar\\_products](#)
- Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).*

  - class [series\\_options](#)
- Flags to control series expansion.*

  - class [solve\\_algo](#)
- Switch to control algorithm for linear system solving.*

  - class [spinidx](#)
- This class holds a spinor index that can be dotted or undotted and that also has a variance.*

  - class [spinmetric](#)
- This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.*

  - class [smapkey](#)
- Flags to store information about the state of an object.*

  - class [status\\_flags](#)
- Wrapper template for making [GiNaC](#) classes out of C++ structures.*

  - class [structure](#)
- This class represents the tensor of symmetric su(3) structure constants.*

  - class [su3d](#)
- This class represents the tensor of antisymmetric su(3) structure constants.*

  - class [su3f](#)

- class [su3one](#)  
*This class represents the  $su(3)$  unity element.*
- class [su3t](#)  
*This class represents an  $su(3)$  generator.*
- class [subs\\_options](#)  
*Flags to control the behavior of [subs\(\)](#).*
- class [sy\\_is\\_less](#)
- class [sy\\_swap](#)
- struct [sym\\_desc](#)  
*This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".*
- class [symbol](#)  
*Basic CAS symbol.*
- class [symbolset](#)
- class [symmetry](#)  
*This class describes the symmetry of a group of indices.*
- class [symminfo](#)  
*This structure stores the individual symmetrized terms obtained during the simplification of sums.*
- class [symminfo\\_is\\_less\\_by\\_orig](#)
- class [symminfo\\_is\\_less\\_by\\_symmterm](#)
- class [tensdelta](#)  
*This class represents the delta tensor.*
- class [tensepsilon](#)  
*This class represents the totally antisymmetric epsilon tensor.*
- class [tensmetric](#)  
*This class represents a general metric tensor which can be used to raise/lower indices.*
- class [tensor](#)  
*This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.*
- class [terminfo](#)  
*This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.*
- class [terminfo\\_is\\_less](#)
- class [unarchive\\_table\\_t](#)
- class [user\\_defined\\_kernel](#)  
*A user-defined integration kernel.*
- class [varidx](#)  
*This class holds an index with a variance (co- or contravariant).*
- class [visitor](#)  
*Degenerate base class for visitors.*
- class [wildcard](#)  
*This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).*
- class [zeta1\\_SERIAL](#)  
*Complex conjugate.*
- class [zeta2\\_SERIAL](#)  
*Alternating Euler sum or colored MZV.*

## Typedefs

- typedef unsigned [archive\\_node\\_id](#)  
*Numerical ID value to refer to an [archive\\_node](#).*
- typedef unsigned [archive\\_atom](#)  
*Numerical ID value to refer to a string.*
- typedef [basic](#) [\\*\(\\* synthesize\\_func\)](#) ()
- typedef std::map< std::string, [synthesize\\_func](#) > [unarchive\\_map\\_t](#)
- typedef std::vector< [ex](#) > [exvector](#)
- typedef std::set< [ex](#), [ex\\_is\\_less](#) > [exset](#)
- typedef std::map< [ex](#), [ex](#), [ex\\_is\\_less](#) > [exmap](#)
- typedef [ex](#) [\\* evalfunctype](#) ()
- typedef double [\\* FUNCP\\_1P](#) (double)  
*Function pointer with one function parameter.*
- typedef double [\\* FUNCP\\_2P](#) (double, double)  
*Function pointer with two function parameters.*
- typedef void [\\* FUNCP\\_CUBA](#) (const int \*, const double[], const int \*, double[])  
*Function pointer for use with the CUBA library ( <http://www.feynarts.de/cuba>).*
- typedef std::vector< [expair](#) > [epvector](#)  
*expair-vector*
- typedef [epvector::iterator](#) [epp](#)  
*expair-vector pointer*
- typedef [container](#) < std::vector > [exprseq](#)
- typedef std::multiset< unsigned > [paramset](#)
- typedef [ex](#) [\\* eval\\_funcp](#) ()
- typedef [ex](#) [\\* evalf\\_funcp](#) ()
- typedef [ex](#) [\\* conjugate\\_funcp](#) ()
- typedef [ex](#) [\\* real\\_part\\_funcp](#) ()
- typedef [ex](#) [\\* imag\\_part\\_funcp](#) ()
- typedef [ex](#) [\\* expand\\_funcp](#) ()
- typedef [ex](#) [\\* derivative\\_funcp](#) ()
- typedef [ex](#) [\\* expl\\_derivative\\_funcp](#) ()
- typedef [ex](#) [\\* power\\_funcp](#) ()
- typedef [ex](#) [\\* series\\_funcp](#) ()
- typedef void [\\* print\\_funcp](#) ()
- typedef bool [\\* info\\_funcp](#) ()
- typedef [ex](#) [\\* eval\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* evalf\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* conjugate\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* real\\_part\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* imag\\_part\\_funcp\\_1](#) (const [ex](#) &)
- typedef [ex](#) [\\* expand\\_funcp\\_1](#) (const [ex](#) &, unsigned)
- typedef [ex](#) [\\* derivative\\_funcp\\_1](#) (const [ex](#) &, unsigned)
- typedef [ex](#) [\\* expl\\_derivative\\_funcp\\_1](#) (const [ex](#) &, const [symbol](#) &)
- typedef [ex](#) [\\* power\\_funcp\\_1](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* series\\_funcp\\_1](#) (const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void [\\* print\\_funcp\\_1](#) (const [ex](#) &, const [print\\_context](#) &)
- typedef bool [\\* info\\_funcp\\_1](#) (const [ex](#) &, unsigned)
- typedef [ex](#) [\\* eval\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* evalf\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* conjugate\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* real\\_part\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* imag\\_part\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#) [\\* expand\\_funcp\\_2](#) (const [ex](#) &, const [ex](#) &, unsigned)

- [illegible]



- [illegible]







- typedef [ex](#)(\* [power\\_funcp\\_13](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_13](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_13](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_13](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [eval\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [evalf\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [expand\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [derivative\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [expl\\_derivative\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [symbol](#) &)
- typedef [ex](#)(\* [power\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_14](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(\* [eval\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [evalf\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [conjugate\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [real\\_part\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [imag\\_part\\_funcp\\_exvector](#)) (const [exvector](#) &)
- typedef [ex](#)(\* [expand\\_funcp\\_exvector](#)) (const [exvector](#) &, unsigned)
- typedef [ex](#)(\* [derivative\\_funcp\\_exvector](#)) (const [exvector](#) &, unsigned)
- typedef [ex](#)(\* [expl\\_derivative\\_funcp\\_exvector](#)) (const [exvector](#) &, const [symbol](#) &)
- typedef [ex](#)(\* [power\\_funcp\\_exvector](#)) (const [exvector](#) &, const [ex](#) &)
- typedef [ex](#)(\* [series\\_funcp\\_exvector](#)) (const [exvector](#) &, const [relational](#) &, int, unsigned)
- typedef void(\* [print\\_funcp\\_exvector](#)) (const [exvector](#) &, const [print\\_context](#) &)
- typedef bool(\* [info\\_funcp\\_exvector](#)) (const [exvector](#) &, unsigned)
- template<typename T, class Hash = std::hash<[ex](#)>, class KeyEqual = std::equal\_to<[ex](#)>, class Allocator = std::allocator<std::pair<const [ex](#), T>>>
 using [exhashmap](#) = std::unordered\_map< [ex](#), T, Hash, KeyEqual, Allocator >
- typedef std::map< [spmapkey](#), [ex](#) > [spmap](#)
- typedef map< [error\\_and\\_integral](#), [ex](#), [error\\_and\\_integral\\_is\\_less](#) > [lookup\\_map](#)
- typedef [container](#)< std::list > [lst](#)
- typedef std::vector< std::size\_t > [uintvector](#)
- typedef std::vector< unsigned > [unsignedvector](#)
- typedef std::vector< [exvector](#) > [exvectorvector](#)

- typedef std::vector< [sym\\_desc](#) > [sym\\_desc\\_vec](#)
- typedef void(\* [digits\\_changed\\_callback](#)) (long)  
*Function pointer to implement callbacks in the case 'Digits' gets changed.*
- typedef [class\\_info](#)< [print\\_context\\_options](#) > [print\\_context\\_class\\_info](#)
- typedef [class\\_info](#)< [registered\\_class\\_options](#) > [registered\\_class\\_info](#)

## Enumerations

- enum { [callback\\_registered](#) = 1 }

## Functions

- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([add](#), [expairseq](#), [print\\_func](#)< [print\\_context](#) >(&[add::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[add::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_csrc](#) >(&[add::do\\_print\\_csrc](#)), [print\\_func](#)< [print\\_tree](#) >(&[add::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[add::do\\_print\\_python\\_repr](#)))  
[add](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([add](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([add](#))
- static void [write\\_unsigned](#) (std::ostream &os, unsigned val)  
*Write unsigned integer quantity to stream.*
- static unsigned [read\\_unsigned](#) (std::istream &is)  
*Read unsigned integer quantity from stream.*
- std::ostream & [operator<<](#) (std::ostream &os, const [archive\\_node](#) &n)  
*Write [archive\\_node](#) to binary data stream.*
- std::ostream & [operator<<](#) (std::ostream &os, const [archive](#) &ar)  
*Write archive to binary data stream.*
- std::istream & [operator>>](#) (std::istream &is, [archive\\_node](#) &n)  
*Read [archive\\_node](#) from binary data stream.*
- std::istream & [operator>>](#) (std::istream &is, [archive](#) &ar)  
*Read archive from binary data stream.*
- static [synthesize\\_func](#) [find\\_factory\\_fcn](#) (const std::string &name)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([basic](#), void, [print\\_func](#)< [print\\_context](#) >(&[basic::do\\_print](#)), [print\\_func](#)< [print\\_tree](#) >(&[basic::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[basic::do\\_print\\_python\\_repr](#)))  
[basic](#)  
*basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by duplicate()), so it can copy the tinfo\_key and the hash value.*
- template<class T >  
bool [is\\_a](#) (const [basic](#) &obj)  
*Check if obj is a T, including base classes.*
- template<class T >  
bool [is\\_exactly\\_a](#) (const [basic](#) &obj)  
*Check if obj is a T, not including base classes.*
- template<class B , typename... Args>  
B & [dynallocate](#) (Args &&... args)  
*Constructs a new (class basic or derived) B object on the heap.*
- template<class B >  
B & [dynallocate](#) (std::initializer\_list< [ex](#) > il)  
*Constructs a new (class basic or derived) B object on the heap.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([clifford](#), [indexed](#), [print\\_func](#)< [print\\_dflt](#) >(&[clifford::do\\_print\\_dflt](#)), [print\\_func](#)< [print\\_latex](#) >(&[clifford::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_tree](#) >(&[clifford::do\\_print\\_tree](#)))  
[GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)([diracone](#)
- [print\\_func](#)< [print\\_dflt](#) > (&[diracone::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[diracone](#)

- [GINAC\\_BIND\\_UNARCHIVER](#) ([clifford](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([cliffordunit](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracone](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracgamma](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracgamma5](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracgammaL](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([diracgammaR](#))
- static bool [is\\_dirac\\_slash](#) (const [ex](#) &seq0)
- static void [base\\_and\\_index](#) (const [ex](#) &c, [ex](#) &b, [ex](#) &i)  
*This function decomposes  $\gamma \sim \mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$*
- [ex](#) [dirac\\_ONE](#) (unsigned char rl=0)  
*Create a Clifford unity object.*
- static unsigned [get\\_dim\\_uint](#) (const [ex](#) &e)
- [ex](#) [clifford\\_unit](#) (const [ex](#) &mu, const [ex](#) &metr, unsigned char rl=0)  
*Create a Clifford unit object.*
- [ex](#) [dirac\\_gamma](#) (const [ex](#) &mu, unsigned char rl=0)  
*Create a Dirac gamma object.*
- [ex](#) [dirac\\_gamma5](#) (unsigned char rl=0)  
*Create a Dirac gamma5 object.*
- [ex](#) [dirac\\_gammaL](#) (unsigned char rl=0)  
*Create a Dirac gammaL object.*
- [ex](#) [dirac\\_gammaR](#) (unsigned char rl=0)  
*Create a Dirac gammaR object.*
- [ex](#) [dirac\\_slash](#) (const [ex](#) &e, const [ex](#) &dim, unsigned char rl=0)  
*Create a term of the form  $e_\mu * \gamma \sim \mu$  with a unique index  $\mu$ .*
- static unsigned char [get\\_representation\\_label](#) (const [return\\_type\\_t](#) &ti)  
*Extract representation label from tinfo key (as returned by [return\\_type\\_tinfo\(\)](#)).*
- static [ex](#) [trace\\_string](#) (exvector::const\_iterator ix, size\_t num)  
*Take trace of a string of an even number of Dirac gammas given a vector of indices.*
- [ex](#) [dirac\\_trace](#) (const [ex](#) &e, const std::set< unsigned char > &rls, const [ex](#) &trONE=4)  
*Calculate dirac traces over the specified set of representation labels.*
- [ex](#) [dirac\\_trace](#) (const [ex](#) &e, const [lst](#) &rls, const [ex](#) &trONE=4)  
*Calculate dirac traces over the specified list of representation labels.*
- [ex](#) [dirac\\_trace](#) (const [ex](#) &e, unsigned char rl=0, const [ex](#) &trONE=4)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- [ex](#) [canonicalize\\_clifford](#) (const [ex](#) &e)  
*Bring all products of clifford objects in an expression into a canonical order.*
- [ex](#) [clifford\\_star\\_bar](#) (const [ex](#) &e, bool do\_bar, unsigned [options](#))  
*An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).*
- [ex](#) [clifford\\_prime](#) (const [ex](#) &e)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- [ex](#) [remove\\_dirac\\_ONE](#) (const [ex](#) &e, unsigned char rl=0, unsigned [options](#)=0)  
*Replaces [dirac\\_ONE](#)'s (with a [representation\\_label](#) no less than  $rl$ ) in  $e$  with 1.*
- int [clifford\\_max\\_label](#) (const [ex](#) &e, bool ignore\_ONE=false)  
*Returns the maximal representation label of a clifford object if  $e$  contains at least one, otherwise returns -1.*
- [ex](#) [clifford\\_norm](#) (const [ex](#) &e)  
*Calculation of the norm in the Clifford algebra.*
- [ex](#) [clifford\\_inverse](#) (const [ex](#) &e)  
*Calculation of the inverse in the Clifford algebra.*
- [ex](#) [lst\\_to\\_clifford](#) (const [ex](#) &v, const [ex](#) &mu, const [ex](#) &metr, unsigned char rl=0)  
*List or vector conversion into the Clifford vector.*

- `ex lst_to_clifford` (const `ex` &v, const `ex` &e)
- static `ex get_clifford_comp` (const `ex` &e, const `ex` &c, bool root=true)  
*Auxiliary structure to define a function for stripping one Clifford unit from vectors.*
- `lst clifford_to_lst` (const `ex` &e, const `ex` &c, bool algebraic=true)  
*An inverse function to `lst_to_clifford()`.*
- `ex clifford_moebius_map` (const `ex` &a, const `ex` &b, const `ex` &c, const `ex` &d, const `ex` &v, const `ex` &G, unsigned char rl=0)  
*Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b|c d) in linear spaces with arbitrary signature.*
- `ex clifford_moebius_map` (const `ex` &M, const `ex` &v, const `ex` &G, unsigned char rl=0)  
*The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*
- `GINAC_DECLARE_UNARCHIVER` (clifford)
- `GINAC_DECLARE_UNARCHIVER` (diracone)
- `GINAC_DECLARE_UNARCHIVER` (cliffordunit)
- `GINAC_DECLARE_UNARCHIVER` (diracgamma)
- `GINAC_DECLARE_UNARCHIVER` (diracgamma5)
- `GINAC_DECLARE_UNARCHIVER` (diracgammaL)
- `GINAC_DECLARE_UNARCHIVER` (diracgammaR)
- bool `is_clifford_tinfo` (const `return_type_t` &ti)  
*Check whether a given `return_type_t` object (as returned by `return_type_tinfo()`) is that of a clifford object (with an arbitrary representation label).*
- `ex clifford_bar` (const `ex` &e)  
*Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.*
- `ex clifford_star` (const `ex` &e)  
*Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (su3one, tensor, print\_func< `print_dflt` >(&su3one::do\_print). print\_func< `print_latex` >(&su3one::do\_print\_latex)) `GINAC_IMPLEMENT_REGISTERED_CLASS_↵`  
`OPT(su3t`
- `print_func< print_dflt >` (&su3t::do\_print). `print_func< print_latex >` (&su3t
- `GINAC_BIND_UNARCHIVER` (color)
- `GINAC_BIND_UNARCHIVER` (su3one)
- `GINAC_BIND_UNARCHIVER` (su3t)
- `GINAC_BIND_UNARCHIVER` (su3f)
- `GINAC_BIND_UNARCHIVER` (su3d)
- static `ex permute_free_index_to_front` (const `exvector` &iv3, const `exvector` &iv2, int &sig)  
*Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.*
- `ex color_ONE` (unsigned char rl=0)  
*Create the su(3) unity element.*
- `ex color_T` (const `ex` &a, unsigned char rl=0)  
*Create an su(3) generator.*
- `ex color_f` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) antisymmetric structure constant.*
- `ex color_d` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) symmetric structure constant.*
- `ex color_h` (const `ex` &a, const `ex` &b, const `ex` &c)  
*This returns the linear combination d.a.b.c+1\*f.a.b.c.*
- static bool `is_color_tinfo` (const `return_type_t` &ti)  
*Check whether a given tinfo key (as returned by `return_type_tinfo()`) is that of a color object (with an arbitrary representation label).*
- static unsigned char `get_representation_label` (const `return_type_t` &ti)  
*Extract representation label from tinfo key (as returned by `return_type_tinfo()`).*

- `ex color_trace` (const `ex` &`e`, const `std::set< unsigned char >` &`rls`)  
*Calculate color traces over the specified set of representation labels.*
- `ex color_trace` (const `ex` &`e`, const `lst` &`rl`)  
*Calculate color traces over the specified list of representation labels.*
- `ex color_trace` (const `ex` &`e`, unsigned char `rl=0`)  
*Calculate the trace of an expression containing color objects with a specified representation label.*
- `GINAC_DECLARE_UNARCHIVER` (`color`)
- `GINAC_DECLARE_UNARCHIVER` (`su3one`)
- `GINAC_DECLARE_UNARCHIVER` (`su3t`)
- `GINAC_DECLARE_UNARCHIVER` (`su3f`)
- `GINAC_DECLARE_UNARCHIVER` (`su3d`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`constant`, `basic`, `print_func< print_context >` (&`constant::do_print`), `print_func< print_latex >` (&`constant::do_print_latex`), `print_func< print_tree >` (&`constant::do_print_tree`), `print_func< print_python_repr >` (&`constant::do_print_python_repr`)) const ant
- `GINAC_BIND_UNARCHIVER` (`constant`)
- `GINAC_DECLARE_UNARCHIVER` (`constant`)
- static unsigned `crc32` (const char \*`c`, const unsigned `len`, const unsigned `crcinit`)
- bool `are_ex_trivially_equal` (const `ex` &`e1`, const `ex` &`e2`)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- `std::ostream & operator<<` (`std::ostream` &`os`, const `exvector` &`e`)
- `std::ostream & operator<<` (`std::ostream` &`os`, const `exset` &`e`)
- `std::ostream & operator<<` (`std::ostream` &`os`, const `exmap` &`e`)
- `size_t nops` (const `ex` &`thisex`)
- `ex expand` (const `ex` &`thisex`, unsigned `options=0`)
- `ex conjugate` (const `ex` &`thisex`)
- `ex real_part` (const `ex` &`thisex`)
- `ex imag_part` (const `ex` &`thisex`)
- bool `has` (const `ex` &`thisex`, const `ex` &`pattern`, unsigned `options=0`)
- bool `find` (const `ex` &`thisex`, const `ex` &`pattern`, `exset` &`found`)
- bool `is_polynomial` (const `ex` &`thisex`, const `ex` &`vars`)
- int `degree` (const `ex` &`thisex`, const `ex` &`s`)
- int `ldegree` (const `ex` &`thisex`, const `ex` &`s`)
- `ex coeff` (const `ex` &`thisex`, const `ex` &`s`, int `n=1`)
- `ex numer` (const `ex` &`thisex`)
- `ex denom` (const `ex` &`thisex`)
- `ex numer_denom` (const `ex` &`thisex`)
- `ex normal` (const `ex` &`thisex`)
- `ex to_rational` (const `ex` &`thisex`, `exmap` &`repl`)
- `ex to_polynomial` (const `ex` &`thisex`, `exmap` &`repl`)
- `ex collect` (const `ex` &`thisex`, const `ex` &`s`, bool `distributed=false`)
- `ex eval` (const `ex` &`thisex`)
- `ex evalf` (const `ex` &`thisex`)
- `ex evalm` (const `ex` &`thisex`)
- `ex eval_integ` (const `ex` &`thisex`)
- `ex diff` (const `ex` &`thisex`, const `symbol` &`s`, unsigned `nth=1`)
- `ex series` (const `ex` &`thisex`, const `ex` &`r`, int `order`, unsigned `options=0`)
- bool `match` (const `ex` &`thisex`, const `ex` &`pattern`, `exmap` &`repl_lst`)
- `ex simplify_indexed` (const `ex` &`thisex`, unsigned `options=0`)
- `ex simplify_indexed` (const `ex` &`thisex`, const `scalar_products` &`sp`, unsigned `options=0`)
- `ex symmetrize` (const `ex` &`thisex`)
- `ex symmetrize` (const `ex` &`thisex`, const `lst` &`l`)
- `ex antisymmetrize` (const `ex` &`thisex`)
- `ex antisymmetrize` (const `ex` &`thisex`, const `lst` &`l`)
- `ex symmetrize_cyclic` (const `ex` &`thisex`)

- `ex symmetrize_cyclic` (const `ex` &thisex, const `lst` &l)
- `ex op` (const `ex` &thisex, size\_t i)
- `ex lhs` (const `ex` &thisex)
- `ex rhs` (const `ex` &thisex)
- `bool is_zero` (const `ex` &thisex)
- `void swap` (`ex` &e1, `ex` &e2)
- `ex subs` (const `ex` &thisex, const `exmap` &m, unsigned `options`=0)
- `ex subs` (const `ex` &thisex, const `lst` &ls, const `lst` &lr, unsigned `options`=0)
- `ex subs` (const `ex` &thisex, const `ex` &e, unsigned `options`=0)
- `template<class T >`  
`bool is_a` (const `ex` &obj)  
*Check if ex is a handle to a T, including base classes.*
- `template<class T >`  
`bool is_exactly_a` (const `ex` &obj)  
*Check if ex is a handle to a T, not including base classes.*
- `template<class T >`  
`const T & ex_to` (const `ex` &e)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- `void compile_ex` (const `ex` &expr, const `symbol` &sym, `FUNC_P_1P` &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- `void compile_ex` (const `ex` &expr, const `symbol` &sym1, const `symbol` &sym2, `FUNC_P_2P` &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- `void compile_ex` (const `lst` &exprs, const `lst` &syms, `FUNC_P_CUBA` &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- `void link_ex` (const std::string filename, `FUNC_P_1P` &fp)  
*Opens an existing so-file and returns a function pointer of type FUNC\_P\_1P to the contained function.*
- `void link_ex` (const std::string filename, `FUNC_P_2P` &fp)  
*Opens an existing so-file and returns a function pointer of type FUNC\_P\_2P to the contained function.*
- `void link_ex` (const std::string filename, `FUNC_P_CUBA` &fp)  
*Opens an existing so-file and returns a function pointer of type FUNC\_P\_CUBA to the contained function.*
- `void unlink_ex` (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*
- `void swap` (`expair` &e1, `expair` &e2)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`expairseq`, `basic`, print\_func< `print_context` >(&`expairseq`::do\_print). print\_func< `print_tree` >(&`expairseq`::do\_print\_tree)) class `epp_is_less`
- `epvector * conjugateepvector` (const `epvector` &)  
*Complex conjugate every element of an epvector.*
- `ex factor` (const `ex` &poly, unsigned `options`)  
*Interface function to the outside world.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`fail`, `basic`, print\_func< `print_context` >(&`fail`::do\_print). print\_func< `print_tree` >(&`fail`::do\_print\_tree)) `GINAC_BIND_UNARCHIVER`(`fail`)
- `GINAC_DECLARE_UNARCHIVER` (`fail`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`fderivative`, `function`, print\_func< `print_context` >(&`fderivative`::do\_print). print\_func< `print_latex` >(&`fderivative`::do\_print\_latex). print\_func< `print_csrc` >(&`fderivative`::do\_print\_csrc). print\_func< `print_tree` >(&`fderivative`::do\_print\_tree)) `fderivative`
- `GINAC_BIND_UNARCHIVER` (`fderivative`)
- `GINAC_DECLARE_UNARCHIVER` (`fderivative`)
- `GINAC_BIND_UNARCHIVER` (`function`)
- `GINAC_DECLARE_UNARCHIVER` (`function`)
- `template<typename T >`  
`bool is_the_function` (const `ex` &x)
- `static unsigned make_hash_seed` (const std::type\_info &tinfo)

*We need a hash function which gives different values for objects of different types.*

- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`idx`, `basic`, `print_func< print_context >(&idx::do_print)`, `print_func< print_latex >(&idx::do_print_latex)`, `print_func< print_csrc >(&idx::do_print_csrc)`, `print_func< print_tree >(&idx::do_print_tree)`) `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT`(`varidx`
- `print_func< print_context >(&varidx::do_print)`, `print_func< print_latex >(&varidx`
- `GINAC_BIND_UNARCHIVER` (`idx`)
- `GINAC_BIND_UNARCHIVER` (`varidx`)
- `GINAC_BIND_UNARCHIVER` (`spinidx`)
- `bool is_dummy_pair` (`const idx &i1`, `const idx &i2`)

*Check whether two indices form a dummy pair.*

- `bool is_dummy_pair` (`const ex &e1`, `const ex &e2`)

*Check whether two expressions form a dummy index pair.*

- `void find_free_and_dummy` (`exvector::const_iterator it`, `exvector::const_iterator itend`, `exvector &out_free`, `exvector &out_dummy`)

*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*

- `ex minimal_dim` (`const ex &dim1`, `const ex &dim2`)

*Return the minimum of two index dimensions.*

- `GINAC_DECLARE_UNARCHIVER` (`idx`)
- `GINAC_DECLARE_UNARCHIVER` (`varidx`)
- `GINAC_DECLARE_UNARCHIVER` (`spinidx`)
- `void find_free_and_dummy` (`const exvector &v`, `exvector &out_free`, `exvector &out_dummy`)

*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*

- `void find_dummy_indices` (`const exvector &v`, `exvector &out_dummy`)

*Given a vector of indices, find the dummy indices.*

- `size_t count_dummy_indices` (`const exvector &v`)

*Count the number of dummy index pairs in an index vector.*

- `size_t count_free_indices` (`const exvector &v`)

*Count the number of dummy index pairs in an index vector.*

- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`indexed`, `exprseq`, `print_func< print_context >(&indexed::do_print)`, `print_func< print_latex >(&indexed::do_print_latex)`, `print_func< print_tree >(&indexed::do_print_tree)`) `indexed`
- `GINAC_BIND_UNARCHIVER` (`indexed`)
- `static bool indices_consistent` (`const exvector &v1`, `const exvector &v2`)

*Check whether two sorted index vectors are consistent (i.e.*

- `template<class T >`  
`size_t number_of_type` (`const exvector &v`)
- `template<class T >`  
`static ex rename_dummy_indices` (`const ex &e`, `exvector &global_dummy_indices`, `exvector &local_dummy_↵`  
`_indices`)

*Rename dummy indices in an expression.*

- `static void find_variant_indices` (`const exvector &v`, `exvector &variant_indices`)

*Given a set of indices, extract those of class varidx.*

- `bool reposition_dummy_indices` (`ex &e`, `exvector &variant_dummy_indices`, `exvector &moved_indices`)

*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*

- `static void product_to_exvector` (`const ex &e`, `exvector &v`, `bool &non_commutative`)

- `template<class T >`

`ex idx_symmetrization` (`const ex &r`, `const exvector &local_dummy_indices`)

- `ex simplify_indexed` (`const ex &e`, `exvector &free_indices`, `exvector &dummy_indices`, `const scalar_products &sp`)

*Simplify indexed expression, return list of free indices.*

- `ex simplify_indexed_product` (`const ex &e`, `exvector &free_indices`, `exvector &dummy_indices`, `const scalar_products &sp`)



*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.*

- `bool hasindex (const ex &x, const ex &sym)`
- `exvector get_all_dummy_indices_safely (const ex &e)`

*More reliable version of the form.*

- `exvector get_all_dummy_indices (const ex &e)`

*Returns all dummy indices from the exvector.*

- `lst rename_dummy_indices_uniquely (const exvector &va, const exvector &vb)`

*Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.*

- `ex rename_dummy_indices_uniquely (const exvector &va, const exvector &vb, const ex &b)`

*Same as above, where va and vb contain the indices of a and b and are sorted.*

- `ex rename_dummy_indices_uniquely (const ex &a, const ex &b)`

*Returns b with all dummy indices, which are common with a, renamed.*

- `ex rename_dummy_indices_uniquely (exvector &va, const ex &b, bool modify_va=false)`

*Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b will be appended to va.*

- `ex expand_dummy_sum (const ex &e, bool subs_idx=false)`

*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*

- `GINAC\_DECLARE\_UNARCHIVER (indexed)`

- `static ex conjugate_evalf (const ex &arg)`

- `static ex conjugate_eval (const ex &arg)`

- `static void conjugate_print_latex (const ex &arg, const print\_context &c)`

- `static ex conjugate_conjugate (const ex &arg)`

- `static ex conjugate_expl_derivative (const ex &arg, const symbol &s)`

- `static ex conjugate_real_part (const ex &arg)`

- `static ex conjugate_imag_part (const ex &arg)`

- `static bool func_arg_info (const ex &arg, unsigned inf)`

- `static bool conjugate_info (const ex &arg, unsigned inf)`

- `REGISTER\_FUNCTION (conjugate_function, eval_func(conjugate\_eval). evalf_func(conjugate\_evalf).  
expl_derivative_func(conjugate\_expl\_derivative). info_func(conjugate\_info). print_func< print\_latex  
>(conjugate\_print\_latex). conjugate_func(conjugate\_conjugate). real_part_func(conjugate\_real\_part).  
imag_part_func(conjugate\_imag\_part). set_name("conjugate","conjugate"))`

- `static ex real_part_evalf (const ex &arg)`

- `static ex real_part_eval (const ex &arg)`

- `static void real_part_print_latex (const ex &arg, const print\_context &c)`

- `static ex real_part_conjugate (const ex &arg)`

- `static ex real_part_real_part (const ex &arg)`

- `static ex real_part_imag_part (const ex &arg)`

- `static ex real_part_expl_derivative (const ex &arg, const symbol &s)`

- `REGISTER\_FUNCTION (real_part_function, eval_func(real\_part\_eval). evalf_func(real\_part\_evalf). expl↵  
_derivative_func(real\_part\_expl\_derivative). print_func< print\_latex >(real\_part\_print\_latex). conjugate↵  
_func(real\_part\_conjugate). real_part_func(real\_part\_real\_part). imag_part_func(real\_part\_imag\_part).  
set_name("real_part","real_part"))`

- `static ex imag_part_evalf (const ex &arg)`

- `static ex imag_part_eval (const ex &arg)`

- `static void imag_part_print_latex (const ex &arg, const print\_context &c)`

- `static ex imag_part_conjugate (const ex &arg)`

- `static ex imag_part_real_part (const ex &arg)`

- `static ex imag_part_imag_part (const ex &arg)`

- `static ex imag_part_expl_derivative (const ex &arg, const symbol &s)`

- `REGISTER\_FUNCTION (imag_part_function, eval_func(imag\_part\_eval). evalf_func(imag\_part\_evalf).  
expl_derivative_func(imag\_part\_expl\_derivative). print_func< print\_latex >(imag\_part\_print\_latex).  
conjugate_func(imag\_part\_conjugate). real_part_func(imag\_part\_real\_part). imag_part_func(imag\_part\_imag\_part).  
set_name("imag_part","imag_part"))`



- static `ex abs_evalf` (const `ex` &arg)
- static `ex abs_eval` (const `ex` &arg)
- static `ex abs_expand` (const `ex` &arg, unsigned `options`)
- static `ex abs_expl_derivative` (const `ex` &arg, const `symbol` &s)
- static void `abs_print_latex` (const `ex` &arg, const `print_context` &c)
- static void `abs_print_csrc_float` (const `ex` &arg, const `print_context` &c)
- static `ex abs_conjugate` (const `ex` &arg)
- static `ex abs_real_part` (const `ex` &arg)
- static `ex abs_imag_part` (const `ex` &arg)
- static `ex abs_power` (const `ex` &arg, const `ex` &exp)
- bool `abs_info` (const `ex` &arg, unsigned inf)
- `REGISTER_FUNCTION` (`abs`, `eval_func(abs_eval)`, `evalf_func(abs_evalf)`, `expand_func(abs_expand)`, `expl_derivative_func(abs_expl_derivative)`, `info_func(abs_info)`, `print_func< print_latex >(abs_print_latex)`, `print_func< print_csrc_float >(abs_print_csrc_float)`, `print_func< print_csrc_double >(abs_print_csrc_double)`, `conjugate_func(abs_conjugate)`, `real_part_func(abs_real_part)`, `imag_part_func(abs_imag_part)`, `power_←_func(abs_power)`)
- static `ex step_evalf` (const `ex` &arg)
- static `ex step_eval` (const `ex` &arg)
- static `ex step_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex step_conjugate` (const `ex` &arg)
- static `ex step_real_part` (const `ex` &arg)
- static `ex step_imag_part` (const `ex` &arg)
- `REGISTER_FUNCTION` (`step`, `eval_func(step_eval)`, `evalf_func(step_evalf)`, `series_func(step_series)`, `conjugate_func(step_conjugate)`, `real_part_func(step_real_part)`, `imag_part_func(step_imag_part)`)
- static `ex csgn_evalf` (const `ex` &arg)
- static `ex csgn_eval` (const `ex` &arg)
- static `ex csgn_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex csgn_conjugate` (const `ex` &arg)
- static `ex csgn_real_part` (const `ex` &arg)
- static `ex csgn_imag_part` (const `ex` &arg)
- static `ex csgn_power` (const `ex` &arg, const `ex` &exp)
- `REGISTER_FUNCTION` (`csgn`, `eval_func(csgn_eval)`, `evalf_func(csgn_evalf)`, `series_func(csgn_series)`, `conjugate_func(csgn_conjugate)`, `real_part_func(csgn_real_part)`, `imag_part_func(csgn_imag_part)`, `power_func(csgn_power)`)
- static `ex eta_evalf` (const `ex` &x, const `ex` &y)
- static `ex eta_eval` (const `ex` &x, const `ex` &y)
- static `ex eta_series` (const `ex` &x, const `ex` &y, const `relational` &rel, int `order`, unsigned `options`)
- static `ex eta_conjugate` (const `ex` &x, const `ex` &y)
- static `ex eta_real_part` (const `ex` &x, const `ex` &y)
- static `ex eta_imag_part` (const `ex` &x, const `ex` &y)
- `REGISTER_FUNCTION` (`eta`, `eval_func(eta_eval)`, `evalf_func(eta_evalf)`, `series_func(eta_series)`, `latex_name("\\eta")`, `set_symmetry(sy_symm(0, 1))`, `conjugate_func(eta_conjugate)`, `real_part_←_func(eta_real_part)`, `imag_part_func(eta_imag_part)`)
- static `ex Li2_evalf` (const `ex` &x)
- static `ex Li2_eval` (const `ex` &x)
- static `ex Li2_deriv` (const `ex` &x, unsigned `deriv_param`)
- static `ex Li2_series` (const `ex` &x, const `relational` &rel, int `order`, unsigned `options`)
- static `ex Li2_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`Li2`, `eval_func(Li2_eval)`, `evalf_func(Li2_evalf)`, `derivative_func(Li2_deriv)`, `series_←_func(Li2_series)`, `conjugate_func(Li2_conjugate)`, `latex_name("\\mathrm{Li}_2")`)
- static `ex Li3_eval` (const `ex` &x)
- `REGISTER_FUNCTION` (`Li3`, `eval_func(Li3_eval)`, `latex_name("\\mathrm{Li}_3")`)
- static `ex zetaderiv_eval` (const `ex` &n, const `ex` &x)
- static `ex zetaderiv_deriv` (const `ex` &n, const `ex` &x, unsigned `deriv_param`)

- `REGISTER_FUNCTION` (zetaderiv, eval\_func(zetaderiv\_eval). derivative\_func(zetaderiv\_deriv). latex\_name("\\zeta^\\prime"))
  - static `ex factorial_evalf` (const `ex` &`x`)
  - static `ex factorial_eval` (const `ex` &`x`)
  - static void `factorial_print_dflt_latex` (const `ex` &`x`, const `print_context` &`c`)
  - static `ex factorial_conjugate` (const `ex` &`x`)
  - static `ex factorial_real_part` (const `ex` &`x`)
  - static `ex factorial_imag_part` (const `ex` &`x`)
  - `REGISTER_FUNCTION` (factorial, eval\_func(factorial\_eval). evalf\_func(factorial\_evalf). print\_func<print\_dflt>(factorial\_print\_dflt\_latex). print\_func<print\_latex>(factorial\_print\_dflt\_latex). conjugate\_func(factorial\_conjugate). real\_part\_func(factorial\_real\_part). imag\_part\_func(factorial\_imag\_part))
  - static `ex binomial_evalf` (const `ex` &`x`, const `ex` &`y`)
  - static `ex binomial_sym` (const `ex` &`x`, const `numeric` &`y`)
  - static `ex binomial_eval` (const `ex` &`x`, const `ex` &`y`)
  - static `ex binomial_conjugate` (const `ex` &`x`, const `ex` &`y`)
  - static `ex binomial_real_part` (const `ex` &`x`, const `ex` &`y`)
  - static `ex binomial_imag_part` (const `ex` &`x`, const `ex` &`y`)
  - `REGISTER_FUNCTION` (binomial, eval\_func(binomial\_eval). evalf\_func(binomial\_evalf). conjugate\_func(binomial\_conjugate). real\_part\_func(binomial\_real\_part). imag\_part\_func(binomial\_imag\_part))
  - static `ex Order_eval` (const `ex` &`x`)
  - static `ex Order_series` (const `ex` &`x`, const `relational` &`r`, int `order`, unsigned `options`)
  - static `ex Order_conjugate` (const `ex` &`x`)
  - static `ex Order_real_part` (const `ex` &`x`)
  - static `ex Order_imag_part` (const `ex` &`x`)
  - static `ex Order_expl_derivative` (const `ex` &`arg`, const `symbol` &`s`)
  - `REGISTER_FUNCTION` (Order, eval\_func(Order\_eval). series\_func(Order\_series). latex\_name("\\mathcal{O}"). expl\_derivative\_func(Order\_expl\_derivative). conjugate\_func(Order\_conjugate). real\_part\_func(Order\_real\_part). imag\_part\_func(Order\_imag\_part))
  - `ex lsolve` (const `ex` &`eqns`, const `ex` &`symbols`, unsigned `options=solve_algo::automatic`)
- Factorial function.*
- const `numeric fsolve` (const `ex` &`f`, const `symbol` &`x`, const `numeric` &`x1`, const `numeric` &`x2`)
- Find a real root of real-valued function f(x) numerically within a given interval.*
- template<typename T1 >  
function `zeta` (const T1 &`p1`)
  - template<typename T1 , typename T2 >  
function `zeta` (const T1 &`p1`, const T2 &`p2`)
  - template<> bool `is_the_function<zeta_SERIAL>` (const `ex` &`x`)
  - template<typename T1 , typename T2 >  
function `G` (const T1 &`x`, const T2 &`y`)
  - template<typename T1 , typename T2 , typename T3 >  
function `G` (const T1 &`x`, const T2 &`s`, const T3 &`y`)
  - template<> bool `is_the_function<G_SERIAL>` (const `ex` &`x`)
  - template<typename T1 >  
function `psi` (const T1 &`p1`)
  - template<typename T1 , typename T2 >  
function `psi` (const T1 &`p1`, const T2 &`p2`)
  - template<> bool `is_the_function<psi_SERIAL>` (const `ex` &`x`)
  - template<typename T1 , typename T2 >  
function `iterated_integral` (const T1 &`kernel_lst`, const T2 &`lambda`)
  - template<typename T1 , typename T2 , typename T3 >  
function `iterated_integral` (const T1 &`kernel_lst`, const T2 &`lambda`, const T3 &`N_trunc`)
  - template<> bool `is_the_function<iterated_integral_SERIAL>` (const `ex` &`x`)
  - bool `is_order_function` (const `ex` &`e`)
- Check whether a function is the Order (O(n)) function.*
- `ex convert_H_to_Li` (const `ex` &`parameterlst`, const `ex` &`arg`)

*Converts a given list containing parameters for  $H$  in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*

- static [ex EllipticK\\_evalf](#) (const [ex](#) &k)
- static [ex EllipticK\\_eval](#) (const [ex](#) &k)
- static [ex EllipticK\\_deriv](#) (const [ex](#) &k, unsigned deriv\_param)
- static [ex EllipticK\\_series](#) (const [ex](#) &k, const [relational](#) &rel, int [order](#), unsigned [options](#))
- static void [EllipticK\\_print\\_latex](#) (const [ex](#) &k, const [print\\_context](#) &c)
- [REGISTER\\_FUNCTION](#) (EllipticK, evalf\_func([EllipticK\\_evalf](#)). eval\_func([EllipticK\\_eval](#)). derivative\_↵  
func([EllipticK\\_deriv](#)). series\_func([EllipticK\\_series](#)). print\_func< [print\\_latex](#) >([EllipticK\\_print\\_latex](#)). do\_↵  
not\_evalf\_params())
- static [ex EllipticE\\_evalf](#) (const [ex](#) &k)
- static [ex EllipticE\\_eval](#) (const [ex](#) &k)
- static [ex EllipticE\\_deriv](#) (const [ex](#) &k, unsigned deriv\_param)
- static [ex EllipticE\\_series](#) (const [ex](#) &k, const [relational](#) &rel, int [order](#), unsigned [options](#))
- static void [EllipticE\\_print\\_latex](#) (const [ex](#) &k, const [print\\_context](#) &c)
- [REGISTER\\_FUNCTION](#) (EllipticE, evalf\_func([EllipticE\\_evalf](#)). eval\_func([EllipticE\\_eval](#)). derivative\_↵  
func([EllipticE\\_deriv](#)). series\_func([EllipticE\\_series](#)). print\_func< [print\\_latex](#) >([EllipticE\\_print\\_latex](#)). do\_↵  
not\_evalf\_params())
- static [ex iterated\\_integral\\_evalf\\_impl](#) (const [ex](#) &kernel\_lst, const [ex](#) &lambda, const [ex](#) &N\_trunc)
- static [ex iterated\\_integral2\\_evalf](#) (const [ex](#) &kernel\_lst, const [ex](#) &lambda)
- static [ex iterated\\_integral3\\_evalf](#) (const [ex](#) &kernel\_lst, const [ex](#) &lambda, const [ex](#) &N\_trunc)
- static [ex iterated\\_integral2\\_eval](#) (const [ex](#) &kernel\_lst, const [ex](#) &lambda)
- static [ex iterated\\_integral3\\_eval](#) (const [ex](#) &kernel\_lst, const [ex](#) &lambda, const [ex](#) &N\_trunc)
- static [ex lgamma\\_evalf](#) (const [ex](#) &x)
- static [ex lgamma\\_eval](#) (const [ex](#) &x)

*Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.*

- static [ex lgamma\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex lgamma\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int [order](#), unsigned [options](#))
- static [ex lgamma\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) (lgamma, eval\_func([lgamma\\_eval](#)). evalf\_func([lgamma\\_evalf](#)). derivative\_↵  
func([lgamma\\_deriv](#)). series\_func([lgamma\\_series](#)). conjugate\_func([lgamma\\_conjugate](#)). latex\_name("\log  
\Gamma"))
- static [ex tgamma\\_evalf](#) (const [ex](#) &x)
- static [ex tgamma\\_eval](#) (const [ex](#) &x)

*Evaluation of  $tgamma(x)$ , the true Gamma function.*

- static [ex tgamma\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex tgamma\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int [order](#), unsigned [options](#))
- static [ex tgamma\\_conjugate](#) (const [ex](#) &x)
- [REGISTER\\_FUNCTION](#) (tgamma, eval\_func([tgamma\\_eval](#)). evalf\_func([tgamma\\_evalf](#)). derivative\_↵  
\_func([tgamma\\_deriv](#)). series\_func([tgamma\\_series](#)). conjugate\_func([tgamma\\_conjugate](#)). latex\_↵  
name("\Gamma"))
- static [ex beta\\_evalf](#) (const [ex](#) &x, const [ex](#) &y)
- static [ex beta\\_eval](#) (const [ex](#) &x, const [ex](#) &y)
- static [ex beta\\_deriv](#) (const [ex](#) &x, const [ex](#) &y, unsigned deriv\_param)
- static [ex beta\\_series](#) (const [ex](#) &arg1, const [ex](#) &arg2, const [relational](#) &rel, int [order](#), unsigned [options](#))
- [REGISTER\\_FUNCTION](#) (beta, eval\_func([beta\\_eval](#)). evalf\_func([beta\\_evalf](#)). derivative\_func([beta\\_deriv](#)).  
series\_func([beta\\_series](#)). latex\_name("\mathrm{B}"). set\_symmetry([sy\\_symm](#)(0, 1)))
- static [ex psi1\\_evalf](#) (const [ex](#) &x)
- static [ex psi1\\_eval](#) (const [ex](#) &x)

*Evaluation of digamma-function  $\psi(x)$ .*

- static [ex psi1\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex psi1\\_series](#) (const [ex](#) &arg, const [relational](#) &rel, int [order](#), unsigned [options](#))
- static [ex psi2\\_evalf](#) (const [ex](#) &n, const [ex](#) &x)
- static [ex psi2\\_eval](#) (const [ex](#) &n, const [ex](#) &x)

*Evaluation of polygamma-function  $\psi(n, x)$ .*

- static `ex psi2_deriv` (const `ex` &n, const `ex` &x, unsigned deriv\_param)
- static `ex psi2_series` (const `ex` &n, const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex G2_evalf` (const `ex` &x\_, const `ex` &y)
- static `ex G2_eval` (const `ex` &x\_, const `ex` &y)
- static `ex G3_evalf` (const `ex` &x\_, const `ex` &s\_, const `ex` &y)
- static `ex G3_eval` (const `ex` &x\_, const `ex` &s\_, const `ex` &y)
- static `ex Li_evalf` (const `ex` &m\_, const `ex` &x\_)
- static `ex Li_eval` (const `ex` &m\_, const `ex` &x\_)
- static `ex Li_series` (const `ex` &m, const `ex` &x, const `relational` &rel, int `order`, unsigned `options`)
- static `ex Li_deriv` (const `ex` &m\_, const `ex` &x\_, unsigned deriv\_param)
- static void `Li_print_latex` (const `ex` &m\_, const `ex` &x\_, const `print_context` &c)
- `REGISTER_FUNCTION` (Li, evalf\_func(Li\_evalf). eval\_func(Li\_eval). series\_func(Li\_series). derivative\_↔  
func(Li\_deriv). print\_func< `print_latex` >(Li\_print\_latex). do\_not\_evalf\_params())
- static `ex S_evalf` (const `ex` &n, const `ex` &p, const `ex` &x)
- static `ex S_eval` (const `ex` &n, const `ex` &p, const `ex` &x)
- static `ex S_series` (const `ex` &n, const `ex` &p, const `ex` &x, const `relational` &rel, int `order`, unsigned `options`)
- static `ex S_deriv` (const `ex` &n, const `ex` &p, const `ex` &x, unsigned deriv\_param)
- static void `S_print_latex` (const `ex` &n, const `ex` &p, const `ex` &x, const `print_context` &c)
- `REGISTER_FUNCTION` (S, evalf\_func(S\_evalf). eval\_func(S\_eval). series\_func(S\_series). derivative\_↔  
func(S\_deriv). print\_func< `print_latex` >(S\_print\_latex). do\_not\_evalf\_params())
- static `ex H_evalf` (const `ex` &x1, const `ex` &x2)
- static `ex H_eval` (const `ex` &m\_, const `ex` &x)
- static `ex H_series` (const `ex` &m, const `ex` &x, const `relational` &rel, int `order`, unsigned `options`)
- static `ex H_deriv` (const `ex` &m\_, const `ex` &x, unsigned deriv\_param)
- static void `H_print_latex` (const `ex` &m\_, const `ex` &x, const `print_context` &c)
- `REGISTER_FUNCTION` (H, evalf\_func(H\_evalf). eval\_func(H\_eval). series\_func(H\_series). derivative\_↔  
func(H\_deriv). print\_func< `print_latex` >(H\_print\_latex). do\_not\_evalf\_params())
- static `ex zeta1_evalf` (const `ex` &x)
- static `ex zeta1_eval` (const `ex` &m)
- static `ex zeta1_deriv` (const `ex` &m, unsigned deriv\_param)
- static void `zeta1_print_latex` (const `ex` &m\_, const `print_context` &c)
- static `ex zeta2_evalf` (const `ex` &x, const `ex` &s)
- static `ex zeta2_eval` (const `ex` &m, const `ex` &s\_)
- static `ex zeta2_deriv` (const `ex` &m, const `ex` &s, unsigned deriv\_param)
- static void `zeta2_print_latex` (const `ex` &m\_, const `ex` &s\_, const `print_context` &c)
- static `ex exp_evalf` (const `ex` &x)
- static `ex exp_eval` (const `ex` &x)
- static `ex exp_expand` (const `ex` &arg, unsigned `options`)
- static `ex exp_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex exp_real_part` (const `ex` &x)
- static `ex exp_imag_part` (const `ex` &x)
- static `ex exp_conjugate` (const `ex` &x)
- static `ex exp_power` (const `ex` &x, const `ex` &a)
- static bool `exp_info` (const `ex` &x, unsigned inf)
- `REGISTER_FUNCTION` (exp, eval\_func(exp\_eval). evalf\_func(exp\_evalf). info\_func(exp\_info). expand\_↔  
func(exp\_expand). derivative\_func(exp\_deriv). real\_part\_func(exp\_real\_part). imag\_part\_func(exp\_imag\_part).  
conjugate\_func(exp\_conjugate). power\_func(exp\_power). latex\_name("\\exp"))
- static `ex log_evalf` (const `ex` &x)
- static `ex log_eval` (const `ex` &x)
- static `ex log_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex log_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex log_real_part` (const `ex` &x)
- static `ex log_imag_part` (const `ex` &x)
- static `ex log_expand` (const `ex` &arg, unsigned `options`)
- static `ex log_conjugate` (const `ex` &x)

- static bool `log_info` (const `ex` &`x`, unsigned `inf`)
- `REGISTER_FUNCTION` (`log`, `eval_func(log_eval)`. `evalf_func(log_evalf)`. `info_func(log_info)`. `expand_`  
`_func(log_expand)`. `derivative_func(log_deriv)`. `series_func(log_series)`. `real_part_func(log_real_part)`.  
`imag_part_func(log_imag_part)`. `conjugate_func(log_conjugate)`. `latex_name("\\ln")`)
- static `ex` `sin_evalf` (const `ex` &`x`)
- static `ex` `sin_eval` (const `ex` &`x`)
- static `ex` `sin_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `sin_real_part` (const `ex` &`x`)
- static `ex` `sin_imag_part` (const `ex` &`x`)
- static `ex` `sin_conjugate` (const `ex` &`x`)
- static bool `trig_info` (const `ex` &`x`, unsigned `inf`)
- `REGISTER_FUNCTION` (`sin`, `eval_func(sin_eval)`. `evalf_func(sin_evalf)`. `info_func(trig_info)`. `derivative_`  
`_func(sin_deriv)`. `real_part_func(sin_real_part)`. `imag_part_func(sin_imag_part)`. `conjugate_`  
`func(sin_conjugate)`. `latex_name("\\sin")`)
- static `ex` `cos_evalf` (const `ex` &`x`)
- static `ex` `cos_eval` (const `ex` &`x`)
- static `ex` `cos_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `cos_real_part` (const `ex` &`x`)
- static `ex` `cos_imag_part` (const `ex` &`x`)
- static `ex` `cos_conjugate` (const `ex` &`x`)
- `REGISTER_FUNCTION` (`cos`, `eval_func(cos_eval)`. `info_func(trig_info)`. `evalf_func(cos_evalf)`. `derivative_`  
`_func(cos_deriv)`. `real_part_func(cos_real_part)`. `imag_part_func(cos_imag_part)`. `conjugate_`  
`func(cos_conjugate)`. `latex_name("\\cos")`)
- static `ex` `tan_evalf` (const `ex` &`x`)
- static `ex` `tan_eval` (const `ex` &`x`)
- static `ex` `tan_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `tan_real_part` (const `ex` &`x`)
- static `ex` `tan_imag_part` (const `ex` &`x`)
- static `ex` `tan_series` (const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `tan_conjugate` (const `ex` &`x`)
- `REGISTER_FUNCTION` (`tan`, `eval_func(tan_eval)`. `evalf_func(tan_evalf)`. `info_func(trig_info)`. `derivative_`  
`_func(tan_deriv)`. `series_func(tan_series)`. `real_part_func(tan_real_part)`. `imag_part_func(tan_imag_part)`.  
`conjugate_func(tan_conjugate)`. `latex_name("\\tan")`)
- static `ex` `asin_evalf` (const `ex` &`x`)
- static `ex` `asin_eval` (const `ex` &`x`)
- static `ex` `asin_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `asin_conjugate` (const `ex` &`x`)
- static bool `asin_info` (const `ex` &`x`, unsigned `inf`)
- `REGISTER_FUNCTION` (`asin`, `eval_func(asin_eval)`. `evalf_func(asin_evalf)`. `info_func(asin_info)`.  
`derivative_func(asin_deriv)`. `conjugate_func(asin_conjugate)`. `latex_name("\\arcsin")`)
- static `ex` `acos_evalf` (const `ex` &`x`)
- static `ex` `acos_eval` (const `ex` &`x`)
- static `ex` `acos_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `acos_conjugate` (const `ex` &`x`)
- `REGISTER_FUNCTION` (`acos`, `eval_func(acos_eval)`. `evalf_func(acos_evalf)`. `info_func(asin_info)`.  
`derivative_func(acos_deriv)`. `conjugate_func(acos_conjugate)`. `latex_name("\\arccos")`)
- static `ex` `atan_evalf` (const `ex` &`x`)
- static `ex` `atan_eval` (const `ex` &`x`)
- static `ex` `atan_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `atan_series` (const `ex` &`arg`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `atan_conjugate` (const `ex` &`x`)
- static bool `atan_info` (const `ex` &`x`, unsigned `inf`)
- `REGISTER_FUNCTION` (`atan`, `eval_func(atan_eval)`. `evalf_func(atan_evalf)`. `info_func(atan_info)`.  
`derivative_func(atan_deriv)`. `series_func(atan_series)`. `conjugate_func(atan_conjugate)`. `latex_`  
`name("\\arctan")`)

- static `ex atan2_evalf` (const `ex` &y, const `ex` &x)
- static `ex atan2_eval` (const `ex` &y, const `ex` &x)
- static `ex atan2_deriv` (const `ex` &y, const `ex` &x, unsigned deriv\_param)
- static bool `atan2_info` (const `ex` &y, const `ex` &x, unsigned inf)
- `REGISTER_FUNCTION` (`atan2`, `eval_func(atan2_eval)`, `evalf_func(atan2_evalf)`, `info_func(atan2_info)`, `evalf_func(atan2_evalf)`, `derivative_func(atan2_deriv)`)
- static `ex sinh_evalf` (const `ex` &x)
- static `ex sinh_eval` (const `ex` &x)
- static `ex sinh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex sinh_real_part` (const `ex` &x)
- static `ex sinh_imag_part` (const `ex` &x)
- static `ex sinh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`sinh`, `eval_func(sinh_eval)`, `evalf_func(sinh_evalf)`, `info_func(atan_info)`, `derivative_func(sinh_deriv)`, `real_part_func(sinh_real_part)`, `imag_part_func(sinh_imag_part)`, `conjugate_func(sinh_conjugate)`, `latex_name("\\sinh")`)
- static `ex cosh_evalf` (const `ex` &x)
- static `ex cosh_eval` (const `ex` &x)
- static `ex cosh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex cosh_real_part` (const `ex` &x)
- static `ex cosh_imag_part` (const `ex` &x)
- static `ex cosh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`cosh`, `eval_func(cosh_eval)`, `evalf_func(cosh_evalf)`, `info_func(exp_info)`, `derivative_func(cosh_deriv)`, `real_part_func(cosh_real_part)`, `imag_part_func(cosh_imag_part)`, `conjugate_func(cosh_conjugate)`, `latex_name("\\cosh")`)
- static `ex tanh_evalf` (const `ex` &x)
- static `ex tanh_eval` (const `ex` &x)
- static `ex tanh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex tanh_series` (const `ex` &x, const `relational` &rel, int `order`, unsigned `options`)
- static `ex tanh_real_part` (const `ex` &x)
- static `ex tanh_imag_part` (const `ex` &x)
- static `ex tanh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`tanh`, `eval_func(tanh_eval)`, `evalf_func(tanh_evalf)`, `info_func(atan_info)`, `derivative_func(tanh_deriv)`, `series_func(tanh_series)`, `real_part_func(tanh_real_part)`, `imag_part_func(tanh_imag_part)`, `conjugate_func(tanh_conjugate)`, `latex_name("\\tanh")`)
- static `ex asinh_evalf` (const `ex` &x)
- static `ex asinh_eval` (const `ex` &x)
- static `ex asinh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex asinh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`asinh`, `eval_func(asinh_eval)`, `evalf_func(asinh_evalf)`, `info_func(atan_info)`, `derivative_func(asinh_deriv)`, `conjugate_func(asinh_conjugate)`)
- static `ex acosh_evalf` (const `ex` &x)
- static `ex acosh_eval` (const `ex` &x)
- static `ex acosh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex acosh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`acosh`, `eval_func(acosh_eval)`, `evalf_func(acosh_evalf)`, `info_func(asin_info)`, `derivative_func(acosh_deriv)`, `conjugate_func(acosh_conjugate)`)
- static `ex atanh_evalf` (const `ex` &x)
- static `ex atanh_eval` (const `ex` &x)
- static `ex atanh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex atanh_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex atanh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`atanh`, `eval_func(atanh_eval)`, `evalf_func(atanh_evalf)`, `info_func(asin_info)`, `derivative_func(atanh_deriv)`, `series_func(atanh_series)`, `conjugate_func(atanh_conjugate)`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`integral`, `basic`, `print_func< print_dflt >(&integral::do_print)`, `print_func< print_python >(&integral::do_print)`, `print_func< print_latex >(&integral::do_print_latex)`) `integral`



- [ex subsvalue](#) (const [ex](#) &var, const [ex](#) &value, const [ex](#) &fun)
- [ex adaptivesimpson](#) (const [ex](#) &x, const [ex](#) &a\_in, const [ex](#) &b\_in, const [ex](#) &f, const [ex](#) &error)
 

*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*
- [GINAC\\_BIND\\_UNARCHIVER](#) (integral)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (integral)
- [ex ifactor](#) (const [numeric](#) &n)
 

*Returns the decomposition of the positive integer n into prime numbers in the form  $lst(lst(p1,...,pr), lst(a1,...,ar))$  such that  $n = p1^{a1} \dots pr^{ar}$ .*
- [bool is\\_discriminant\\_of\\_quadratic\\_number\\_field](#) (const [numeric](#) &n)
 

*Returns true if the integer n is either one or the discriminant of a quadratic number field.*
- [numeric kronecker\\_symbol](#) (const [numeric](#) &a, const [numeric](#) &n)
 

*Returns the Kronecker symbol a: integer n: integer.*
- [numeric primitive\\_dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a)
 

*Defines a primitive Dirichlet character through the Kronecker symbol.*
- [numeric dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a, const [numeric](#) &N)
 

*Defines a Dirichlet character through the Kronecker symbol.*
- [numeric generalised\\_Bernoulli\\_number](#) (const [numeric](#) &k, const [numeric](#) &b)
 

*The generalised Bernoulli number.*
- [ex Bernoulli\\_polynomial](#) (const [numeric](#) &k, const [ex](#) &x)
 

*The Bernoulli polynomials.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (integration\_kernel, basic, print\_func< [print\\_context](#) >(&integration\_kernel::do\_print)) integration\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (integration\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (basic\_log\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&basic\_log\_kernel::do\_print)) basic\_log\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (basic\_log\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (multiple\_polylog\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&multiple\_polylog\_kernel::do\_print)) multiple\_polylog\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (multiple\_polylog\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (ELi\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&ELi\_kernel::do\_print)) ELi\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (ELi\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Ebar\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Ebar\_kernel::do\_print)) Ebar\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Ebar\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dtau\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Kronecker\_dtau\_kernel::do\_print)) Kronecker\_dtau\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dtau\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dz\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Kronecker\_dz\_kernel::do\_print)) Kronecker\_dz\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dz\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Eisenstein\_kernel::do\_print)) Eisenstein\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_h\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Eisenstein\_h\_kernel::do\_print)) Eisenstein\_h\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_h\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (modular\_form\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&modular\_form\_kernel::do\_print)) modular\_form\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (modular\_form\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (user\_defined\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&user\_defined\_kernel::do\_print)) user\_defined\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (user\_defined\_kernel)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (integration\_kernel)

- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([basic\\_log\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([multiple\\_polylog\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([ELi\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Ebar\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dtau\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dz\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_h\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([modular\\_form\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([user\\_defined\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([lst](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([matrix](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[matrix::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[matrix::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_tree](#) >(&[matrix::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[matrix::do\\_print\\_python\\_repr](#))) [matrix](#)
- *Default ctor.*
- [GINAC\\_BIND\\_UNARCHIVER](#) ([matrix](#))
- [ex lst\\_to\\_matrix](#) (const [lst](#) &l)
- *Convert list of lists to matrix.*
- [ex diag\\_matrix](#) (const [lst](#) &l)
- *Convert list of diagonal elements to matrix.*
- [ex diag\\_matrix](#) (std::initializer\_list< [ex](#) > l)
- [ex unit\\_matrix](#) (unsigned [r](#), unsigned [c](#))
- *Create an r times c unit matrix.*
- [ex symbolic\\_matrix](#) (unsigned [r](#), unsigned [c](#), const std::string &base\_name, const std::string &tex\_base\_name)
- *Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- [ex reduced\\_matrix](#) (const [matrix](#) &m, unsigned [r](#), unsigned [c](#))
- *Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- [ex sub\\_matrix](#) (const [matrix](#) &m, unsigned [r](#), unsigned [nr](#), unsigned [c](#), unsigned [nc](#))
- *Return the nr times nc submatrix starting at position r, c of matrix m.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([matrix](#))
- [size\\_t nops](#) (const [matrix](#) &m)
- [ex expand](#) (const [matrix](#) &m, unsigned [options](#)=0)
- [ex evalf](#) (const [matrix](#) &m)
- unsigned [rows](#) (const [matrix](#) &m)
- unsigned [cols](#) (const [matrix](#) &m)
- [matrix transpose](#) (const [matrix](#) &m)
- [ex determinant](#) (const [matrix](#) &m, unsigned [options](#)=[determinant\\_algo::automatic](#))
- [ex trace](#) (const [matrix](#) &m)
- [ex charpoly](#) (const [matrix](#) &m, const [ex](#) &lambda)
- [matrix inverse](#) (const [matrix](#) &m)
- [matrix inverse](#) (const [matrix](#) &m, unsigned [algo](#))
- unsigned [rank](#) (const [matrix](#) &m)
- unsigned [rank](#) (const [matrix](#) &m, unsigned [solve\\_algo](#))
- [ex unit\\_matrix](#) (unsigned [x](#))
- *Create a x times x unit matrix.*
- [ex symbolic\\_matrix](#) (unsigned [r](#), unsigned [c](#), const std::string &base\_name)
- *Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([mul](#), [expairseq](#), [print\\_func](#)< [print\\_context](#) >(&[mul::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[mul::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_csrc](#) >(&[mul::do\\_print\\_csrc](#)), [print\\_func](#)< [print\\_tree](#) >(&[mul::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[mul::do\\_print\\_python\\_repr](#))) [mul](#)



- bool `tryfactsubs` (const `ex` &origfactor, const `ex` &patternfactor, int &nummatches, `exmap` &repls)
- bool `algebraic_match_mul_with_mul` (const `mul` &e, const `ex` &pat, `exmap` &repls, int `factor`, int &nummatches, const std::vector< bool > &subsed, std::vector< bool > &matched)
- Checks whether e matches to the pattern pat and the (possibly to be updated) list of replacements repls.*
- `GINAC_BIND_UNARCHIVER` (`mul`)
- `GINAC_DECLARE_UNARCHIVER` (`mul`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`ncmul`, `exprseq`, print\_func< `print_context` >(&`ncmul::do_print`), print\_func< `print_tree` >(&`ncmul::do_print_tree`), print\_func< `print_csrc` >(&`ncmul::do_print_csrc`), print\_func< `print_python_repr` >(&`ncmul::do_print_csrc`)) `ncmul`
- `ex reeval_ncmul` (const `exvector` &v)
- `ex hold_ncmul` (const `exvector` &v)
- `GINAC_BIND_UNARCHIVER` (`ncmul`)
- `GINAC_DECLARE_UNARCHIVER` (`ncmul`)
- static bool `get_first_symbol` (const `ex` &e, `ex` &x)
- Return pointer to first symbol found in expression.*
- static void `add_symbol` (const `ex` &s, `sym_desc_vec` &v)
- static void `collect_symbols` (const `ex` &e, `sym_desc_vec` &v)
- static void `get_symbol_stats` (const `ex` &a, const `ex` &b, `sym_desc_vec` &v)
- Collect statistical information about symbols in polynomials.*
- static `numeric lcmcoeff` (const `ex` &e, const `numeric` &l)
- static `numeric lcm_of_coefficients_denominators` (const `ex` &e)
- Compute LCM of denominators of coefficients of a polynomial.*
- static `ex multiply_lcm` (const `ex` &e, const `numeric` &lcm)
- Bring polynomial from Q[X] to Z[X] by multiplying in the previously determined LCM of the coefficient's denominators.*
- `ex quo` (const `ex` &a, const `ex` &b, const `ex` &x, bool check\_args)
- Quotient q(x) of polynomials a(x) and b(x) in Q[x].*
- `ex rem` (const `ex` &a, const `ex` &b, const `ex` &x, bool check\_args)
- Remainder r(x) of polynomials a(x) and b(x) in Q[x].*
- `ex decomp_rational` (const `ex` &a, const `ex` &x)
- Decompose rational function a(x)=N(x)/D(x) into P(x)+n(x)/D(x) with degree(n, x) < degree(D, x).*
- `ex prem` (const `ex` &a, const `ex` &b, const `ex` &x, bool check\_args)
- Pseudo-remainder of polynomials a(x) and b(x) in Q[x].*
- `ex sprem` (const `ex` &a, const `ex` &b, const `ex` &x, bool check\_args)
- Sparse pseudo-remainder of polynomials a(x) and b(x) in Q[x].*
- bool `divide` (const `ex` &a, const `ex` &b, `ex` &q, bool check\_args)
- Exact polynomial division of a(X) by b(X) in Q[X].*
- static bool `divide_in_z` (const `ex` &a, const `ex` &b, `ex` &q, `sym_desc_vec::const_iterator` var)
- Exact polynomial division of a(X) by b(X) in Z[X].*
- static `ex sr_gcd` (const `ex` &a, const `ex` &b, `sym_desc_vec::const_iterator` var)
- Compute GCD of multivariate polynomials using the subresultant PRS algorithm.*
- static `ex interpolate` (const `ex` &gamma, const `numeric` &xi, const `ex` &x, int degree\_hint=1)
- xi-adic polynomial interpolation*
- static bool `heur_gcd_z` (`ex` &res, const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, `sym_desc_vec::const_iterator` var)
- Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
- static bool `heur_gcd` (`ex` &res, const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, `sym_desc_vec::const_iterator` var)
- Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
- static `ex gcd_pf_pow` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- static `ex gcd_pf_mul` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- `ex gcd` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, bool check\_args, unsigned `options`)
- Compute GCD (Greatest Common Divisor) of multivariate polynomials a(X) and b(X) in Z[X].*
- static `ex gcd_pf_pow_pow` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
- `ex lcm` (const `ex` &a, const `ex` &b, bool check\_args)

- Compute LCM (Least Common Multiple) of multivariate polynomials in  $\mathbb{Z}[X]$ .*

  - static `epvector sqrfree_yun` (const `ex` &a, const `symbol` &x)

*Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.*

  - `ex sqrfree` (const `ex` &a, const `lst` &l)

*Compute a square-free factorization of a multivariate polynomial in  $\mathbb{Q}[X]$ .*

  - `ex sqrfree_parfrac` (const `ex` &a, const `symbol` &x)

*Compute square-free partial fraction decomposition of rational function  $a(x)$ .*

  - static `ex replace_with_symbol` (const `ex` &e, `exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier)

*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*

  - static `ex replace_with_symbol` (const `ex` &e, `exmap` &repl)

*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*

  - static `ex frac_cancel` (const `ex` &n, const `ex` &d)

*Fraction cancellation.*

  - static `ex find_common_factor` (const `ex` &e, `ex` &factor, `exmap` &repl)

*Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).*

  - `ex collect_common_factors` (const `ex` &e)

*Collect common factors in sums.*

  - `ex resultant` (const `ex` &e1, const `ex` &e2, const `ex` &s)

*Resultant of two expressions  $e_1, e_2$  with respect to symbol  $s$ .*

  - `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`numeric`, `basic`, `print_func` < `print_context` > (&`numeric::do_print`).  
`print_func` < `print_latex` > (&`numeric::do_print_latex`). `print_func` < `print_csrc` > (&`numeric::do_print_csrc`).  
`print_func` < `print_csrc_cl_N` > (&`numeric::do_print_csrc_cl_N`). `print_func` < `print_tree` > (&`numeric::do_print_tree`).  
`print_func` < `print_python_repr` > (&`numeric::do_print_python_repr`)) `numeric`

*default ctor.*

  - static const `cln::cl_F make_real_float` (const `cln::cl_idcoded_float` &dec)

*Construct a floating point number from sign, mantissa, and exponent.*

  - static const `cln::cl_F read_real_float` (`std::istream` &s)

*Read serialized floating point number.*

  - `GINAC_BIND_UNARCHIVER` (`numeric`)
  - static void `write_real_float` (`std::ostream` &s, const `cln::cl_R` &n)
  - static void `print_real_number` (const `print_context` &c, const `cln::cl_R` &x)

*Helper function to print a real number in a nicer way than is CLN's default.*

  - static void `print_integer_csrc` (const `print_context` &c, const `cln::cl_I` &x)

*Helper function to print integer number in C++ source format.*

  - static void `print_real_csrc` (const `print_context` &c, const `cln::cl_R` &x)

*Helper function to print real number in C++ source format.*

  - template<typename T1, typename T2>  
static bool `coerce` (T1 &dst, const T2 &arg)
  - template<> bool `coerce` < `int`, `cln::cl_I` > (`int` &dst, const `cln::cl_I` &arg)

*Check if CLN integer can be converted into int.*

  - template<> bool `coerce` < `unsigned int`, `cln::cl_I` > (`unsigned int` &dst, const `cln::cl_I` &arg)
  - static void `print_real_cl_N` (const `print_context` &c, const `cln::cl_R` &x)

*Helper function to print real number in C++ source format using `cl_N` types.*

  - const `numeric exp` (const `numeric` &x)

*Exponential function.*

  - const `numeric log` (const `numeric` &x)

*Natural logarithm.*

  - const `numeric sin` (const `numeric` &x)

*Numeric sine (trigonometric function).*

  - const `numeric cos` (const `numeric` &x)

*Numeric cosine (trigonometric function).*

- const [numeric tan](#) (const [numeric &x](#))  
*Numeric tangent (trigonometric function).*
- const [numeric asin](#) (const [numeric &x](#))  
*Numeric inverse sine (trigonometric function).*
- const [numeric acos](#) (const [numeric &x](#))  
*Numeric inverse cosine (trigonometric function).*
- const [numeric atan](#) (const [numeric &x](#))  
*Numeric arcustangent.*
- const [numeric atan](#) (const [numeric &y](#), const [numeric &x](#))  
*Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- const [numeric sinh](#) (const [numeric &x](#))  
*Numeric hyperbolic sine (trigonometric function).*
- const [numeric cosh](#) (const [numeric &x](#))  
*Numeric hyperbolic cosine (trigonometric function).*
- const [numeric tanh](#) (const [numeric &x](#))  
*Numeric hyperbolic tangent (trigonometric function).*
- const [numeric asinh](#) (const [numeric &x](#))  
*Numeric inverse hyperbolic sine (trigonometric function).*
- const [numeric acosh](#) (const [numeric &x](#))  
*Numeric inverse hyperbolic cosine (trigonometric function).*
- const [numeric atanh](#) (const [numeric &x](#))  
*Numeric inverse hyperbolic tangent (trigonometric function).*
- static [cln::cl\\_N Li2\\_series](#) (const [cln::cl\\_N &x](#), const [cln::float\\_format\\_t &prec](#))  
*Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.*
- static [cln::cl\\_N Li2\\_projection](#) (const [cln::cl\\_N &x](#), const [cln::float\\_format\\_t &prec](#))  
*Folds Li2's argument inside a small rectangle to enhance convergence.*
- const [cln::cl\\_N Li2\\_](#) (const [cln::cl\\_N &value](#))  
*Numeric evaluation of Dilogarithm.*
- const [numeric Li2](#) (const [numeric &x](#))
- const [numeric zeta](#) (const [numeric &x](#))  
*Numeric evaluation of Riemann's Zeta function.*
- static [cln::float\\_format\\_t guess\\_precision](#) (const [cln::cl\\_N &x](#))
- const [cln::cl\\_N lgamma](#) (const [cln::cl\\_N &x](#))  
*The Gamma function.*
- const [numeric lgamma](#) (const [numeric &x](#))
- const [cln::cl\\_N tgamma](#) (const [cln::cl\\_N &x](#))
- const [numeric tgamma](#) (const [numeric &x](#))
- const [numeric psi](#) (const [numeric &x](#))  
*The psi function (aka polygamma function).*
- const [numeric psi](#) (const [numeric &n](#), const [numeric &x](#))  
*The psi functions (aka polygamma functions).*
- const [numeric factorial](#) (const [numeric &n](#))  
*Factorial combinatorial function.*
- const [numeric doublefactorial](#) (const [numeric &n](#))  
*The double factorial combinatorial function.*
- const [numeric binomial](#) (const [numeric &n](#), const [numeric &k](#))  
*The Binomial coefficients.*
- const [numeric bernoulli](#) (const [numeric &nn](#))  
*Bernoulli number.*
- const [numeric fibonacci](#) (const [numeric &n](#))  
*Fibonacci number.*

- `const numeric abs` (`const numeric &x`)  
*Absolute value.*
- `const numeric mod` (`const numeric &a`, `const numeric &b`)  
*Modulus (in positive representation).*
- `const numeric smod` (`const numeric &a_`, `const numeric &b_`)  
*Modulus (in symmetric representation).*
- `const numeric irem` (`const numeric &a`, `const numeric &b`)  
*Numeric integer remainder.*
- `const numeric irem` (`const numeric &a`, `const numeric &b`, `numeric &q`)  
*Numeric integer remainder.*
- `const numeric iquo` (`const numeric &a`, `const numeric &b`)  
*Numeric integer quotient.*
- `const numeric iquo` (`const numeric &a`, `const numeric &b`, `numeric &r`)  
*Numeric integer quotient.*
- `const numeric gcd` (`const numeric &a`, `const numeric &b`)  
*Greatest Common Divisor.*
- `const numeric lcm` (`const numeric &a`, `const numeric &b`)  
*Least Common Multiple.*
- `const numeric sqrt` (`const numeric &x`)  
*Numeric square root.*
- `const numeric isqrt` (`const numeric &x`)  
*Integer numeric square root.*
- `ex PiEvalf` ()  
*Floating point evaluation of Archimedes' constant Pi.*
- `ex EulerEvalf` ()  
*Floating point evaluation of Euler's constant gamma.*
- `ex CatalanEvalf` ()  
*Floating point evaluation of Catalan's constant.*
- `std::ostream & operator<<` (`std::ostream &os`, `const _numeric_digits &e`)
- `GINAC_DECLARE_UNARCHIVER` (`numeric`)
- `const numeric pow` (`const numeric &x`, `const numeric &y`)
- `const numeric inverse` (`const numeric &x`)
- `numeric step` (`const numeric &x`)
- `int csgn` (`const numeric &x`)
- `bool is_zero` (`const numeric &x`)
- `bool is_positive` (`const numeric &x`)
- `bool is_negative` (`const numeric &x`)
- `bool is_integer` (`const numeric &x`)
- `bool is_pos_integer` (`const numeric &x`)
- `bool is_nonneg_integer` (`const numeric &x`)
- `bool is_even` (`const numeric &x`)
- `bool is_odd` (`const numeric &x`)
- `bool is_prime` (`const numeric &x`)
- `bool is_rational` (`const numeric &x`)
- `bool is_real` (`const numeric &x`)
- `bool is_cinteger` (`const numeric &x`)
- `bool is_crational` (`const numeric &x`)
- `int to_int` (`const numeric &x`)
- `long to_long` (`const numeric &x`)
- `double to_double` (`const numeric &x`)
- `const numeric real` (`const numeric &x`)
- `const numeric imag` (`const numeric &x`)

- const [numeric numer](#) (const [numeric &x](#))
- const [numeric denom](#) (const [numeric &x](#))
- static const [ex exadd](#) (const [ex &lh](#), const [ex &rh](#))
  - Used internally by [operator+\(\)](#) to add two ex objects.*
- static const [ex exmul](#) (const [ex &lh](#), const [ex &rh](#))
  - Used internally by [operator\\*\(\)](#) to multiply two ex objects.*
- static const [ex exminus](#) (const [ex &lh](#))
  - Used internally by [operator-\(\)](#) and friends to change the sign of an argument.*
- const [ex operator+](#) (const [ex &lh](#), const [ex &rh](#))
- const [ex operator-](#) (const [ex &lh](#), const [ex &rh](#))
- const [ex operator\\*](#) (const [ex &lh](#), const [ex &rh](#))
- const [ex operator/](#) (const [ex &lh](#), const [ex &rh](#))
- const [numeric operator+](#) (const [numeric &lh](#), const [numeric &rh](#))
- const [numeric operator-](#) (const [numeric &lh](#), const [numeric &rh](#))
- const [numeric operator\\*](#) (const [numeric &lh](#), const [numeric &rh](#))
- const [numeric operator/](#) (const [numeric &lh](#), const [numeric &rh](#))
- [ex & operator+=](#) ([ex &lh](#), const [ex &rh](#))
- [ex & operator-=](#) ([ex &lh](#), const [ex &rh](#))
- [ex & operator\\*=](#) ([ex &lh](#), const [ex &rh](#))
- [ex & operator/=](#) ([ex &lh](#), const [ex &rh](#))
- [numeric & operator+=](#) ([numeric &lh](#), const [numeric &rh](#))
- [numeric & operator-=](#) ([numeric &lh](#), const [numeric &rh](#))
- [numeric & operator\\*=](#) ([numeric &lh](#), const [numeric &rh](#))
- [numeric & operator/=](#) ([numeric &lh](#), const [numeric &rh](#))
- const [ex operator+](#) (const [ex &lh](#))
- const [ex operator-](#) (const [ex &lh](#))
- const [numeric operator+](#) (const [numeric &lh](#))
- const [numeric operator-](#) (const [numeric &lh](#))
- [ex & operator++](#) ([ex &rh](#))
  - Expression prefix increment.*
- [ex & operator--](#) ([ex &rh](#))
  - Expression prefix decrement.*
- const [ex operator++](#) ([ex &lh](#), int)
  - Expression postfix increment.*
- const [ex operator--](#) ([ex &lh](#), int)
  - Expression postfix decrement.*
- [numeric & operator++](#) ([numeric &rh](#))
  - Numeric prefix increment.*
- [numeric & operator--](#) ([numeric &rh](#))
  - Numeric prefix decrement.*
- const [numeric operator++](#) ([numeric &lh](#), int)
  - Numeric postfix increment.*
- const [numeric operator--](#) ([numeric &lh](#), int)
  - Numeric postfix decrement.*
- const [relational operator==](#) (const [ex &lh](#), const [ex &rh](#))
- const [relational operator!=](#) (const [ex &lh](#), const [ex &rh](#))
- const [relational operator<](#) (const [ex &lh](#), const [ex &rh](#))
- const [relational operator<=](#) (const [ex &lh](#), const [ex &rh](#))
- const [relational operator>](#) (const [ex &lh](#), const [ex &rh](#))
- const [relational operator>=](#) (const [ex &lh](#), const [ex &rh](#))
- static int [my\\_ios\\_index](#) ()
- static void [my\\_ios\\_callback](#) (std::ios\_base::event ev, std::ios\_base &s, int i)
- static [print\\_context](#) \* [get\\_print\\_context](#) (std::ios\_base &s)

- static void [set\\_print\\_context](#) (std::ios\_base &s, const [print\\_context](#) &c)
- static unsigned [get\\_print\\_options](#) (std::ios\_base &s)
- static void [set\\_print\\_options](#) (std::ostream &s, unsigned [options](#))
- std::ostream & [operator<<](#) (std::ostream &os, const [ex](#) &e)
- std::istream & [operator>>](#) (std::istream &is, [ex](#) &e)
- std::ostream & [dflt](#) (std::ostream &os)
- std::ostream & [latex](#) (std::ostream &os)
- std::ostream & [python](#) (std::ostream &os)
- std::ostream & [python\\_repr](#) (std::ostream &os)
- std::ostream & [tree](#) (std::ostream &os)
- std::ostream & [csrc](#) (std::ostream &os)
- std::ostream & [csrc\\_float](#) (std::ostream &os)
- std::ostream & [csrc\\_double](#) (std::ostream &os)
- std::ostream & [csrc\\_cl\\_N](#) (std::ostream &os)
- std::ostream & [index\\_dimensions](#) (std::ostream &os)
- std::ostream & [no\\_index\\_dimensions](#) (std::ostream &os)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([power](#), [basic](#), print\_func< [print\\_dflt](#) >(&[power::do\\_print\\_dflt](#)), print\_func< [print\\_latex](#) >(&[power::do\\_print\\_latex](#)), print\_func< [print\\_csrc](#) >(&[power::do\\_print\\_csrc](#)), print\_func< [print\\_python](#) >(&[power::do\\_print\\_python](#)), print\_func< [print\\_python\\_repr](#) >(&[power::do\\_print\\_python\\_repr](#)), print\_func< [print\\_csrc\\_cl\\_N](#) >(&[power::do\\_print\\_csrc\\_cl\\_N](#))) [power](#)
- static void [print\\_sym\\_pow](#) (const [print\\_context](#) &c, const [symbol](#) &x, int [exp](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([power](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([power](#))
- [ex pow](#) (const [ex](#) &b, const [ex](#) &e)  
*Symbolic exponentiation.*
- template<typename T1 , typename T2 >  
[ex pow](#) (const T1 &b, const T2 &e)
- [ex sqrt](#) (const [ex](#) &a)  
*Square root expression.*
- template<class T >  
bool [is\\_a](#) (const [print\\_context](#) &obj)  
*Check if obj is a T, including base classes.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([pseries](#), [basic](#), print\_func< [print\\_context](#) >(&[pseries::do\\_print](#)), print\_func< [print\\_latex](#) >(&[pseries::do\\_print\\_latex](#)), print\_func< [print\\_tree](#) >(&[pseries::do\\_print\\_tree](#)), print\_func< [print\\_python](#) >(&[pseries::do\\_print\\_python](#)), print\_func< [print\\_python\\_repr](#) >(&[pseries::do\\_print\\_python\\_repr](#))) [pseries](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([pseries](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([pseries](#))
- [ex series\\_to\\_poly](#) (const [ex](#) &e)  
*Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.*
- bool [is\\_terminating](#) (const [pseries](#) &s)
- template<typename T >  
[return\\_type\\_t](#) [make\\_return\\_type\\_t](#) (const unsigned [rl](#)=0)
- template<class Alg , class Ctx , class T , class C >  
void [set\\_print\\_func](#) (void f(const T &, const C &c, unsigned))  
*Add or replace a print method.*
- template<class Alg , class Ctx , class T , class C >  
void [set\\_print\\_func](#) (void (T::\*f)(const C &, unsigned))  
*Add or replace a print method.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([relational](#), [basic](#), print\_func< [print\\_context](#) >(&[relational::do\\_print](#)), print\_func< [print\\_tree](#) >(&[relational::do\\_print\\_tree](#)), print\_func< [print\\_python\\_repr](#) >(&[relational::do\\_print\\_python\\_repr](#))) [relational](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([relational](#))
- static void [print\\_operator](#) (const [print\\_context](#) &c, [relational::operators](#) o)

- [GINAC\\_DECLARE\\_UNARCHIVER](#) (relational)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (symbol, basic, print\_func< [print\\_context](#) >(&symbol::do\_print). print\_func< [print\\_latex](#) >(&symbol::do\_print\_latex). print\_func< [print\\_tree](#) >(&symbol::do\_print\_tree). print\_func< [print\\_python\\_repr](#) >(&symbol::do\_print\_python\_repr)) symbol
- static const std::string & [get\\_default\\_TeX\\_name](#) (const std::string &name)  
*Return default TeX name for symbol.*
- [GINAC\\_BIND\\_UNARCHIVER](#) (symbol)
- [GINAC\\_BIND\\_UNARCHIVER](#) (realsymbol)
- [GINAC\\_BIND\\_UNARCHIVER](#) (possymbol)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (symbol)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (realsymbol)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (possymbol)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (symmetry, basic, print\_func< [print\\_context](#) >(&symmetry::do\_print). print\_func< [print\\_tree](#) >(&symmetry::do\_print\_tree)) symmetry
- [GINAC\\_BIND\\_UNARCHIVER](#) (symmetry)
- static const [symmetry](#) & [index0](#) ()
- static const [symmetry](#) & [index1](#) ()
- static const [symmetry](#) & [index2](#) ()
- static const [symmetry](#) & [index3](#) ()
- const [symmetry](#) & [not\\_symmetric](#) ()
- const [symmetry](#) & [symmetric2](#) ()
- const [symmetry](#) & [symmetric3](#) ()
- const [symmetry](#) & [symmetric4](#) ()
- const [symmetry](#) & [antisymmetric2](#) ()
- const [symmetry](#) & [antisymmetric3](#) ()
- const [symmetry](#) & [antisymmetric4](#) ()
- int [canonicalize](#) (exvector::iterator v, const [symmetry](#) &symm)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- static [ex symm](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator [last](#), bool asymmetric)
- [ex symmetrize](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Symmetrize expression over a set of objects (symbols, indices).*
- [ex antisymmetrize](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex symmetrize\\_cyclic](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (symmetry)
- [symmetry sy\\_none](#) ()
- [symmetry sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry sy\\_symm](#) ()
- [symmetry sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry sy\\_anti](#) ()
- [symmetry sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry sy\\_cycl](#) ()
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [ex symmetrize](#) (const [ex](#) &e, const [exvector](#) &v)



- Symmetrize expression over a set of objects (symbols, indices).*

  - `ex antisymmetrize` (const `ex` &e, const `exvector` &v)
- Antisymmetrize expression over a set of objects (symbols, indices).*

  - `ex symmetrize_cyclic` (const `ex` &e, const `exvector` &v)
- Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*

  - `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`tensdelta`, `tensor`, `print_func`< `print_dflt` >(&`tensdelta::do_print`), `print_func`< `print_latex` >(&`tensdelta::do_print_latex`)) `GINAC_IMPLEMENT_REGISTERED_CLASS_↵`  
`OPT`(`tensmetric`
  - `print_func`< `print_dflt` > (&`tensmetric::do_print`). `print_func`< `print_latex` >(&`tensmetric`
  - `GINAC_BIND_UNARCHIVER` (`minkmetric`)
  - `GINAC_BIND_UNARCHIVER` (`tensepsilon`)
  - `GINAC_BIND_UNARCHIVER` (`tensdelta`)
  - `GINAC_BIND_UNARCHIVER` (`tensmetric`)
  - `GINAC_BIND_UNARCHIVER` (`spinmetric`)
  - `ex delta_tensor` (const `ex` &i1, const `ex` &i2)
- Create a delta tensor with specified indices.*

  - `ex metric_tensor` (const `ex` &i1, const `ex` &i2)
- Create a symmetric metric tensor with specified indices.*

  - `ex lorentz_g` (const `ex` &i1, const `ex` &i2, bool `pos_sig`=false)
- Create a Minkowski metric tensor with specified indices.*

  - `ex spinor_metric` (const `ex` &i1, const `ex` &i2)
- Create a spinor metric tensor with specified indices.*

  - `ex epsilon_tensor` (const `ex` &i1, const `ex` &i2)
- Create an epsilon tensor in a Euclidean space with two indices.*

  - `ex epsilon_tensor` (const `ex` &i1, const `ex` &i2, const `ex` &i3)
- Create an epsilon tensor in a Euclidean space with three indices.*

  - `ex lorentz_eps` (const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4, bool `pos_sig`=false)
- Create an epsilon tensor in a Minkowski space with four indices.*

  - `GINAC_DECLARE_UNARCHIVER` (`tensdelta`)
  - `GINAC_DECLARE_UNARCHIVER` (`tensmetric`)
  - `GINAC_DECLARE_UNARCHIVER` (`minkmetric`)
  - `GINAC_DECLARE_UNARCHIVER` (`spinmetric`)
  - `GINAC_DECLARE_UNARCHIVER` (`tensepsilon`)
  - unsigned `log2` (unsigned `n`)
- Integer binary logarithm.*

  - const `numeric multinomial_coefficient` (const `std::vector`< unsigned > &p)
- Compute the multinomial coefficient  $n!/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.*

  - unsigned `rotate_left` (unsigned `n`)
- Rotate bits of unsigned value by one bit to the left.*

  - `template`<class `T` >  
int `compare_pointers` (const `T` \*a, const `T` \*b)
- Compare two pointers (just to establish some sort of canonical order).*

  - unsigned `golden_ratio_hash` (uintptr\_t `n`)
- Truncated multiplication with golden ratio, for computing hash values.*

  - `template`<class `It` >  
int `permutation_sign` (`It` first, `It` last)
  - `template`<class `It` , class `Cmp` , class `Swap` >  
int `permutation_sign` (`It` first, `It` last, `Cmp` comp, `Swap` swapit)
  - `template`<class `It` , class `Cmp` , class `Swap` >  
void `shaker_sort` (`It` first, `It` last, `Cmp` comp, `Swap` swapit)
  - `template`<class `It` , class `Swap` >  
void `cyclic_permutation` (`It` first, `It` last, `It` new\_first, `Swap` swapit)



- `template<typename T >`  
`std::enable_if< has\_distance< T >::value, typename std::iterator_traits< T >::difference_type >::type`  
`format\_index\_value (const T &a, const T &b)`  
*For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.*
- `template<typename T >`  
`std::enable_if<!has\_distance< T >::value, T >::type format\_index\_value (const T &a, const T &b)`  
*For all other cases we simply print the value.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const basic\_multi\_iterator< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_ordered< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq\_indv< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_counter< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_counter\_indv< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_permutation< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_shuffle< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_shuffle\_prime< T > &v)`  
*Output operator.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (wildcard, basic, print_func< print\_context >(&wildcard::do_print).`  
`print_func< print\_tree >(&wildcard::do_print_tree). print_func< print\_python\_repr >(&wildcard::do_print_python_repr))`  
`wildcard`
- `GINAC_BIND_UNARCHIVER (wildcard)`
- `bool haswild (const ex &x)`  
*Check whether x has a wildcard anywhere as a subexpression.*
- `GINAC_DECLARE_UNARCHIVER (wildcard)`
- `ex wild (unsigned label=0)`  
*Create a wildcard object with the specified label.*

## Variables

- static `unarchive_table_t unarch_table_instance`
- `GiNaC::evalm_map_function map_evalm`
- `GiNaC::evalinteg_map_function map_eval_integ`
- `tensor`
- `const constant Pi ("Pi", PiEvalf, "\\pi", domain::positive)`

- Pi.*
- const [constant Euler](#) ("Euler", [EulerEvalf](#), "\\gamma\_E", [domain::positive](#))
- Euler's constant.*
- const [constant Catalan](#) ("Catalan", [CatalanEvalf](#), "G", [domain::positive](#))
- Catalan's constant.*
- static unsigned const [crctab](#) [256]
- static [library\\_init](#) [library\\_initializer](#)
- For construction of flyweights, etc.*
- const [basic](#) \* [\\_num0\\_bp](#)
- [idx](#)
- unsigned [force\\_include\\_tgamma](#) = [tgamma\\_SERIAL::serial](#)
- unsigned [force\\_include\\_zeta1](#) = [zeta1\\_SERIAL::serial](#)
- template<> [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\\_T](#)([lst](#), [basic](#), [print\\_func](#)< [print\\_context](#)>(&[lst::do\\_print](#))). [print\\_func](#)< [print\\_tree](#)>(&[lst::do\\_print\\_tree](#))) template<> bool [lst](#) [GINAC\\_BIND\\_UNARCHIVER](#) ([lst](#))
- Specialization of [container::info\(\)](#) for [lst](#).*
- const [numeric](#) [l](#) = [numeric](#)([cln::complex](#)([cln::cl\\_l](#)(0),[cln::cl\\_l](#)(1)))
- Imaginary unit.*
- [\\_numeric\\_digits](#) [Digits](#)
- Accuracy in decimal digits.*
- unsigned [next\\_print\\_context\\_id](#) = 0
- Next free ID for [print\\_context](#) types.*
- const int [version\\_major](#) = [GINACLIB\\_MAJOR\\_VERSION](#)
- const int [version\\_minor](#) = [GINACLIB\\_MINOR\\_VERSION](#)
- const int [version\\_micro](#) = [GINACLIB\\_MICRO\\_VERSION](#)
- const [numeric](#) \* [\\_num\\_120\\_p](#)
- const [ex](#) [\\_ex\\_120](#) = [ex](#)(\* [\\_num\\_120\\_p](#))
- const [numeric](#) \* [\\_num\\_60\\_p](#)
- const [ex](#) [\\_ex\\_60](#) = [ex](#)(\* [\\_num\\_60\\_p](#))
- const [numeric](#) \* [\\_num\\_48\\_p](#)
- const [ex](#) [\\_ex\\_48](#) = [ex](#)(\* [\\_num\\_48\\_p](#))
- const [numeric](#) \* [\\_num\\_30\\_p](#)
- const [ex](#) [\\_ex\\_30](#) = [ex](#)(\* [\\_num\\_30\\_p](#))
- const [numeric](#) \* [\\_num\\_25\\_p](#)
- const [ex](#) [\\_ex\\_25](#) = [ex](#)(\* [\\_num\\_25\\_p](#))
- const [numeric](#) \* [\\_num\\_24\\_p](#)
- const [ex](#) [\\_ex\\_24](#) = [ex](#)(\* [\\_num\\_24\\_p](#))
- const [numeric](#) \* [\\_num\\_20\\_p](#)
- const [ex](#) [\\_ex\\_20](#) = [ex](#)(\* [\\_num\\_20\\_p](#))
- const [numeric](#) \* [\\_num\\_18\\_p](#)
- const [ex](#) [\\_ex\\_18](#) = [ex](#)(\* [\\_num\\_18\\_p](#))
- const [numeric](#) \* [\\_num\\_15\\_p](#)
- const [ex](#) [\\_ex\\_15](#) = [ex](#)(\* [\\_num\\_15\\_p](#))
- const [numeric](#) \* [\\_num\\_12\\_p](#)
- const [ex](#) [\\_ex\\_12](#) = [ex](#)(\* [\\_num\\_12\\_p](#))
- const [numeric](#) \* [\\_num\\_11\\_p](#)
- const [ex](#) [\\_ex\\_11](#) = [ex](#)(\* [\\_num\\_11\\_p](#))
- const [numeric](#) \* [\\_num\\_10\\_p](#)
- const [ex](#) [\\_ex\\_10](#) = [ex](#)(\* [\\_num\\_10\\_p](#))
- const [numeric](#) \* [\\_num\\_9\\_p](#)
- const [ex](#) [\\_ex\\_9](#) = [ex](#)(\* [\\_num\\_9\\_p](#))
- const [numeric](#) \* [\\_num\\_8\\_p](#)
- const [ex](#) [\\_ex\\_8](#) = [ex](#)(\* [\\_num\\_8\\_p](#))
- const [numeric](#) \* [\\_num\\_7\\_p](#)

- const [ex\\_ex\\_7](#) = [ex](#)(\*[\\_num\\_7\\_p](#))
- const [numeric](#) \* [\\_num\\_6\\_p](#)
- const [ex\\_ex\\_6](#) = [ex](#)(\*[\\_num\\_6\\_p](#))
- const [numeric](#) \* [\\_num\\_5\\_p](#)
- const [ex\\_ex\\_5](#) = [ex](#)(\*[\\_num\\_5\\_p](#))
- const [numeric](#) \* [\\_num\\_4\\_p](#)
- const [ex\\_ex\\_4](#) = [ex](#)(\*[\\_num\\_4\\_p](#))
- const [numeric](#) \* [\\_num\\_3\\_p](#)
- const [ex\\_ex\\_3](#) = [ex](#)(\*[\\_num\\_3\\_p](#))
- const [numeric](#) \* [\\_num\\_2\\_p](#)
- const [ex\\_ex\\_2](#) = [ex](#)(\*[\\_num\\_2\\_p](#))
- const [numeric](#) \* [\\_num\\_1\\_p](#)
- const [ex\\_ex\\_1](#) = [ex](#)(\*[\\_num\\_1\\_p](#))
- const [numeric](#) \* [\\_num\\_1\\_2\\_p](#)
- const [ex\\_ex\\_1\\_2](#) = [ex](#)(\*[\\_num\\_1\\_2\\_p](#))
- const [numeric](#) \* [\\_num\\_1\\_3\\_p](#)
- const [ex\\_ex\\_1\\_3](#) = [ex](#)(\*[\\_num\\_1\\_3\\_p](#))
- const [numeric](#) \* [\\_num\\_1\\_4\\_p](#)
- const [ex\\_ex\\_1\\_4](#) = [ex](#)(\*[\\_num\\_1\\_4\\_p](#))
- const [numeric](#) \* [\\_num0\\_p](#)
- const [ex\\_ex0](#) = [ex](#)(\*[\\_num0\\_p](#))
- const [numeric](#) \* [\\_num1\\_4\\_p](#)
- const [ex\\_ex1\\_4](#) = [ex](#)(\*[\\_num1\\_4\\_p](#))
- const [numeric](#) \* [\\_num1\\_3\\_p](#)
- const [ex\\_ex1\\_3](#) = [ex](#)(\*[\\_num1\\_3\\_p](#))
- const [numeric](#) \* [\\_num1\\_2\\_p](#)
- const [ex\\_ex1\\_2](#) = [ex](#)(\*[\\_num1\\_2\\_p](#))
- const [numeric](#) \* [\\_num1\\_p](#)
- const [ex\\_ex1](#) = [ex](#)(\*[\\_num1\\_p](#))
- const [numeric](#) \* [\\_num2\\_p](#)
- const [ex\\_ex2](#) = [ex](#)(\*[\\_num2\\_p](#))
- const [numeric](#) \* [\\_num3\\_p](#)
- const [ex\\_ex3](#) = [ex](#)(\*[\\_num3\\_p](#))
- const [numeric](#) \* [\\_num4\\_p](#)
- const [ex\\_ex4](#) = [ex](#)(\*[\\_num4\\_p](#))
- const [numeric](#) \* [\\_num5\\_p](#)
- const [ex\\_ex5](#) = [ex](#)(\*[\\_num5\\_p](#))
- const [numeric](#) \* [\\_num6\\_p](#)
- const [ex\\_ex6](#) = [ex](#)(\*[\\_num6\\_p](#))
- const [numeric](#) \* [\\_num7\\_p](#)
- const [ex\\_ex7](#) = [ex](#)(\*[\\_num7\\_p](#))
- const [numeric](#) \* [\\_num8\\_p](#)
- const [ex\\_ex8](#) = [ex](#)(\*[\\_num8\\_p](#))
- const [numeric](#) \* [\\_num9\\_p](#)
- const [ex\\_ex9](#) = [ex](#)(\*[\\_num9\\_p](#))
- const [numeric](#) \* [\\_num10\\_p](#)
- const [ex\\_ex10](#) = [ex](#)(\*[\\_num10\\_p](#))
- const [numeric](#) \* [\\_num11\\_p](#)
- const [ex\\_ex11](#) = [ex](#)(\*[\\_num11\\_p](#))
- const [numeric](#) \* [\\_num12\\_p](#)
- const [ex\\_ex12](#) = [ex](#)(\*[\\_num12\\_p](#))
- const [numeric](#) \* [\\_num15\\_p](#)
- const [ex\\_ex15](#) = [ex](#)(\*[\\_num15\\_p](#))
- const [numeric](#) \* [\\_num18\\_p](#)
- const [ex\\_ex18](#) = [ex](#)(\*[\\_num18\\_p](#))

- `const numeric * _num20_p`
- `const ex _ex20 = ex(*_num20_p)`
- `const numeric * _num24_p`
- `const ex _ex24 = ex(*_num24_p)`
- `const numeric * _num25_p`
- `const ex _ex25 = ex(*_num25_p)`
- `const numeric * _num30_p`
- `const ex _ex30 = ex(*_num30_p)`
- `const numeric * _num48_p`
- `const ex _ex48 = ex(*_num48_p)`
- `const numeric * _num60_p`
- `const ex _ex60 = ex(*_num60_p)`
- `const numeric * _num120_p`
- `const ex _ex120 = ex(*_num120_p)`

### 5.1.1 Typedef Documentation

#### 5.1.1.1 `archive_node_id`

```
typedef unsigned GiNaC::archive_node_id
```

Numerical ID value to refer to an [archive\\_node](#).

#### 5.1.1.2 `archive_atom`

```
typedef unsigned GiNaC::archive_atom
```

Numerical ID value to refer to a string.

#### 5.1.1.3 `synthesize_func`

```
typedef basic *(* GiNaC::synthesize_func) ()
```

#### 5.1.1.4 `unarchive_map_t`

```
typedef std::map<std::string, synthesize_func> GiNaC::unarchive_map_t
```

#### 5.1.1.5 exvector

```
typedef std::vector<ex> GiNaC::exvector
```

#### 5.1.1.6 exset

```
typedef std::set<ex, ex_is_less> GiNaC::exset
```

#### 5.1.1.7 exmap

```
typedef std::map<ex, ex, ex_is_less> GiNaC::exmap
```

#### 5.1.1.8 evalffunctype

```
typedef ex(* GiNaC::evalffunctype) ()
```

#### 5.1.1.9 FUNCP\_1P

```
typedef double(* GiNaC::FUNCP_1P) (double)
```

Function pointer with one function parameter.

#### 5.1.1.10 FUNCP\_2P

```
typedef double(* GiNaC::FUNCP_2P) (double, double)
```

Function pointer with two function parameters.

#### 5.1.1.11 FUNCP\_CUBA

```
typedef void(* GiNaC::FUNCP_CUBA) (const int *, const double[], const int *, double[])
```

Function pointer for use with the CUBA library ( <http://www.feynarts.de/cuba>).

#### 5.1.1.12 `epvector`

```
typedef std::vector<expair> GiNaC::epvector
```

expair-vector

#### 5.1.1.13 `epp`

```
typedef epvector::iterator GiNaC::epp
```

expair-vector pointer

#### 5.1.1.14 `exprseq`

```
typedef container<std::vector> GiNaC::exprseq
```

#### 5.1.1.15 `paramset`

```
typedef std::multiset<unsigned> GiNaC::paramset
```

#### 5.1.1.16 `eval_funcp`

```
typedef ex(* GiNaC::eval_funcp) ()
```

#### 5.1.1.17 `evalf_funcp`

```
typedef ex(* GiNaC::evalf_funcp) ()
```

#### 5.1.1.18 `conjugate_funcp`

```
typedef ex(* GiNaC::conjugate_funcp) ()
```

**5.1.1.19 real\_part\_funcp**

```
typedef ex(* GiNaC::real_part_funcp) ()
```

**5.1.1.20 imag\_part\_funcp**

```
typedef ex(* GiNaC::imag_part_funcp) ()
```

**5.1.1.21 expand\_funcp**

```
typedef ex(* GiNaC::expand_funcp) ()
```

**5.1.1.22 derivative\_funcp**

```
typedef ex(* GiNaC::derivative_funcp) ()
```

**5.1.1.23 expl\_derivative\_funcp**

```
typedef ex(* GiNaC::expl_derivative_funcp) ()
```

**5.1.1.24 power\_funcp**

```
typedef ex(* GiNaC::power_funcp) ()
```

**5.1.1.25 series\_funcp**

```
typedef ex(* GiNaC::series_funcp) ()
```

**5.1.1.26 print\_funcp**

```
typedef void(* GiNaC::print_funcp) ()
```

#### 5.1.1.27 info\_funcp

```
typedef bool(* GiNaC::info_funcp) ()
```

#### 5.1.1.28 eval\_funcp\_1

```
typedef ex(* GiNaC::eval_funcp_1) (const ex &)
```

#### 5.1.1.29 evalf\_funcp\_1

```
typedef ex(* GiNaC::evalf_funcp_1) (const ex &)
```

#### 5.1.1.30 conjugate\_funcp\_1

```
typedef ex(* GiNaC::conjugate_funcp_1) (const ex &)
```

#### 5.1.1.31 real\_part\_funcp\_1

```
typedef ex(* GiNaC::real_part_funcp_1) (const ex &)
```

#### 5.1.1.32 imag\_part\_funcp\_1

```
typedef ex(* GiNaC::imag_part_funcp_1) (const ex &)
```

#### 5.1.1.33 expand\_funcp\_1

```
typedef ex(* GiNaC::expand_funcp_1) (const ex &, unsigned)
```

#### 5.1.1.34 derivative\_funcp\_1

```
typedef ex(* GiNaC::derivative_funcp_1) (const ex &, unsigned)
```



#### 5.1.1.35 expl\_derivative\_funcp\_1

```
typedef ex(* GiNaC::expl_derivative_funcp_1) (const ex &, const symbol &)
```

#### 5.1.1.36 power\_funcp\_1

```
typedef ex(* GiNaC::power_funcp_1) (const ex &, const ex &)
```

#### 5.1.1.37 series\_funcp\_1

```
typedef ex(* GiNaC::series_funcp_1) (const ex &, const relational &, int, unsigned)
```

#### 5.1.1.38 print\_funcp\_1

```
typedef void(* GiNaC::print_funcp_1) (const ex &, const print_context &)
```

#### 5.1.1.39 info\_funcp\_1

```
typedef bool(* GiNaC::info_funcp_1) (const ex &, unsigned)
```

#### 5.1.1.40 eval\_funcp\_2

```
typedef ex(* GiNaC::eval_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.41 evalf\_funcp\_2

```
typedef ex(* GiNaC::evalf_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.42 conjugate\_funcp\_2

```
typedef ex(* GiNaC::conjugate_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.43 real\_part\_funcp\_2

```
typedef ex(* GiNaC::real_part_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.44 imag\_part\_funcp\_2

```
typedef ex(* GiNaC::imag_part_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.45 expand\_funcp\_2

```
typedef ex(* GiNaC::expand_funcp_2) (const ex &, const ex &, unsigned)
```

#### 5.1.1.46 derivative\_funcp\_2

```
typedef ex(* GiNaC::derivative_funcp_2) (const ex &, const ex &, unsigned)
```

#### 5.1.1.47 expl\_derivative\_funcp\_2

```
typedef ex(* GiNaC::expl_derivative_funcp_2) (const ex &, const ex &, const symbol &)
```

#### 5.1.1.48 power\_funcp\_2

```
typedef ex(* GiNaC::power_funcp_2) (const ex &, const ex &, const ex &)
```

#### 5.1.1.49 series\_funcp\_2

```
typedef ex(* GiNaC::series_funcp_2) (const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.50 print\_funcp\_2

```
typedef void(* GiNaC::print_funcp_2) (const ex &, const ex &, const print_context &)
```

#### 5.1.1.51 info\_funcp\_2

```
typedef bool(* GiNaC::info_funcp_2) (const ex &, const ex &, unsigned)
```

#### 5.1.1.52 eval\_funcp\_3

```
typedef ex(* GiNaC::eval_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.53 evalf\_funcp\_3

```
typedef ex(* GiNaC::evalf_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.54 conjugate\_funcp\_3

```
typedef ex(* GiNaC::conjugate_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.55 real\_part\_funcp\_3

```
typedef ex(* GiNaC::real_part_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.56 imag\_part\_funcp\_3

```
typedef ex(* GiNaC::imag_part_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.57 expand\_funcp\_3

```
typedef ex(* GiNaC::expand_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.58 derivative\_funcp\_3

```
typedef ex(* GiNaC::derivative_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.59 `expl_derivative_funcp_3`

```
typedef ex(* GiNaC::expl_derivative_funcp_3) (const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.60 `power_funcp_3`

```
typedef ex(* GiNaC::power_funcp_3) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.61 `series_funcp_3`

```
typedef ex(* GiNaC::series_funcp_3) (const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.62 `print_funcp_3`

```
typedef void(* GiNaC::print_funcp_3) (const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.63 `info_funcp_3`

```
typedef bool(* GiNaC::info_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.64 `eval_funcp_4`

```
typedef ex(* GiNaC::eval_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.65 `evalf_funcp_4`

```
typedef ex(* GiNaC::evalf_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.66 conjugate\_funcp\_4

```
typedef ex(* GiNaC::conjugate_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.67 real\_part\_funcp\_4

```
typedef ex(* GiNaC::real_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.68 imag\_part\_funcp\_4

```
typedef ex(* GiNaC::imag_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.69 expand\_funcp\_4

```
typedef ex(* GiNaC::expand_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.70 derivative\_funcp\_4

```
typedef ex(* GiNaC::derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.71 expl\_derivative\_funcp\_4

```
typedef ex(* GiNaC::expl_derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &,  
const symbol &)
```

#### 5.1.1.72 power\_funcp\_4

```
typedef ex(* GiNaC::power_funcp_4) (const ex &, const ex &, const ex &, const ex &, const ex  
&)
```

#### 5.1.1.73 series\_funcp\_4

```
typedef ex(* GiNaC::series_funcp_4) (const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.74 print\_funcp\_4

```
typedef void(* GiNaC::print_funcp_4) (const ex &, const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.75 info\_funcp\_4

```
typedef bool(* GiNaC::info_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.76 eval\_funcp\_5

```
typedef ex(* GiNaC::eval_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.77 evalf\_funcp\_5

```
typedef ex(* GiNaC::evalf_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.78 conjugate\_funcp\_5

```
typedef ex(* GiNaC::conjugate_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.79 real\_part\_funcp\_5

```
typedef ex(* GiNaC::real_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.80 imag\_part\_funcp\_5

```
typedef ex(* GiNaC::imag_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.81 expand\_funcp\_5

```
typedef ex(* GiNaC::expand_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.82 derivative\_funcp\_5

```
typedef ex(* GiNaC::derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.83 expl\_derivative\_funcp\_5

```
typedef ex(* GiNaC::expl_derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.84 power\_funcp\_5

```
typedef ex(* GiNaC::power_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.85 series\_funcp\_5

```
typedef ex(* GiNaC::series_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.86 print\_funcp\_5

```
typedef void(* GiNaC::print_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.87 info\_funcp\_5

```
typedef bool(* GiNaC::info_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.88 eval\_funcp\_6

```
typedef ex(* GiNaC::eval_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.89 evalf\_funcp\_6

```
typedef ex(* GiNaC::evalf_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.90 conjugate\_funcp\_6

```
typedef ex(* GiNaC::conjugate_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.91 real\_part\_funcp\_6

```
typedef ex(* GiNaC::real_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.92 imag\_part\_funcp\_6

```
typedef ex(* GiNaC::imag_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.93 expand\_funcp\_6

```
typedef ex(* GiNaC::expand_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```



#### 5.1.1.94 derivative\_funcp\_6

```
typedef ex(* GiNaC::derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.95 expl\_derivative\_funcp\_6

```
typedef ex(* GiNaC::expl_derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.96 power\_funcp\_6

```
typedef ex(* GiNaC::power_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.97 series\_funcp\_6

```
typedef ex(* GiNaC::series_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.98 print\_funcp\_6

```
typedef void(* GiNaC::print_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.99 info\_funcp\_6

```
typedef bool(* GiNaC::info_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.100 eval\_funcp\_7

```
typedef ex(* GiNaC::eval_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.101 evalf\_funcp\_7

```
typedef ex(* GiNaC::evalf_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &)
```

#### 5.1.1.102 conjugate\_funcp\_7

```
typedef ex(* GiNaC::conjugate_funcp_7) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &)
```

#### 5.1.1.103 real\_part\_funcp\_7

```
typedef ex(* GiNaC::real_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &)
```

#### 5.1.1.104 imag\_part\_funcp\_7

```
typedef ex(* GiNaC::imag_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &)
```

#### 5.1.1.105 expand\_funcp\_7

```
typedef ex(* GiNaC::expand_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, unsigned)
```

#### 5.1.1.106 derivative\_funcp\_7

```
typedef ex(* GiNaC::derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.107 expl\_derivative\_funcp\_7

```
typedef ex(* GiNaC::expl_derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.108 power\_funcp\_7

```
typedef ex(* GiNaC::power_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &)
```

#### 5.1.1.109 series\_funcp\_7

```
typedef ex(* GiNaC::series_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.110 print\_funcp\_7

```
typedef void(* GiNaC::print_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const print_context &)
```

#### 5.1.1.111 info\_funcp\_7

```
typedef bool(* GiNaC::info_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, unsigned)
```

#### 5.1.1.112 eval\_funcp\_8

```
typedef ex(* GiNaC::eval_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &)
```

#### 5.1.1.113 evalf\_funcp\_8

```
typedef ex(* GiNaC::evalf_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &)
```

#### 5.1.1.114 conjugate\_funcp\_8

```
typedef ex(* GiNaC::conjugate_funcp_8) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.115 real\_part\_funcp\_8

```
typedef ex(* GiNaC::real_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.116 imag\_part\_funcp\_8

```
typedef ex(* GiNaC::imag_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.117 expand\_funcp\_8

```
typedef ex(* GiNaC::expand_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.118 derivative\_funcp\_8

```
typedef ex(* GiNaC::derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.119 expl\_derivative\_funcp\_8

```
typedef ex(* GiNaC::expl_derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.120 power\_funcp\_8

```
typedef ex(* GiNaC::power_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.121 series\_funcp\_8

```
typedef ex(* GiNaC::series_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.122 print\_funcp\_8

```
typedef void(* GiNaC::print_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.123 info\_funcp\_8

```
typedef bool(* GiNaC::info_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.124 eval\_funcp\_9

```
typedef ex(* GiNaC::eval_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.125 evalf\_funcp\_9

```
typedef ex(* GiNaC::evalf_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.126 conjugate\_funcp\_9

```
typedef ex(* GiNaC::conjugate_funcp_9) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.127 real\_part\_funcp\_9

```
typedef ex(* GiNaC::real_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.128 imag\_part\_funcp\_9

```
typedef ex(* GiNaC::imag_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.129 expand\_funcp\_9**

```
typedef ex(* GiNaC::expand_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.130 derivative\_funcp\_9**

```
typedef ex(* GiNaC::derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.131 expl\_derivative\_funcp\_9**

```
typedef ex(* GiNaC::expl_derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

**5.1.1.132 power\_funcp\_9**

```
typedef ex(* GiNaC::power_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.133 series\_funcp\_9**

```
typedef ex(* GiNaC::series_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

**5.1.1.134 print\_funcp\_9**

```
typedef void(* GiNaC::print_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

**5.1.1.135 info\_funcp\_9**

```
typedef bool(* GiNaC::info_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.136 eval\_funcp\_10

```
typedef ex(* GiNaC::eval_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.137 evalf\_funcp\_10

```
typedef ex(* GiNaC::evalf_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.138 conjugate\_funcp\_10

```
typedef ex(* GiNaC::conjugate_funcp_10) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.139 real\_part\_funcp\_10

```
typedef ex(* GiNaC::real_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.140 imag\_part\_funcp\_10

```
typedef ex(* GiNaC::imag_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.141 expand\_funcp\_10

```
typedef ex(* GiNaC::expand_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.142 derivative\_funcp\_10

```
typedef ex(* GiNaC::derivative_funcp_10) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.143 expl\_derivative\_funcp\_10

```
typedef ex(* GiNaC::expl_derivative_funcp_10) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.144 power\_funcp\_10

```
typedef ex(* GiNaC::power_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.145 series\_funcp\_10

```
typedef ex(* GiNaC::series_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int,
unsigned)
```

#### 5.1.1.146 print\_funcp\_10

```
typedef void(* GiNaC::print_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.147 info\_funcp\_10

```
typedef bool(* GiNaC::info_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.148 eval\_funcp\_11

```
typedef ex(* GiNaC::eval_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.149 evalf\_funcp\_11

```
typedef ex(* GiNaC::evalf_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```



#### 5.1.1.150 conjugate\_funcp\_11

```
typedef ex(* GiNaC::conjugate_funcp_11) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.151 real\_part\_funcp\_11

```
typedef ex(* GiNaC::real_part_funcp_11) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.152 imag\_part\_funcp\_11

```
typedef ex(* GiNaC::imag_part_funcp_11) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.153 expand\_funcp\_11

```
typedef ex(* GiNaC::expand_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.154 derivative\_funcp\_11

```
typedef ex(* GiNaC::derivative_funcp_11) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.155 expl\_derivative\_funcp\_11

```
typedef ex(* GiNaC::expl_derivative_funcp_11) (const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
symbol &)
```

#### 5.1.1.156 power\_funcp\_11

```
typedef ex(* GiNaC::power_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.157 series\_funcp\_11

```
typedef ex(* GiNaC::series_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &  
int, unsigned)
```

#### 5.1.1.158 print\_funcp\_11

```
typedef void(* GiNaC::print_funcp_11) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context  
&)
```

#### 5.1.1.159 info\_funcp\_11

```
typedef bool(* GiNaC::info_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.160 eval\_funcp\_12

```
typedef ex(* GiNaC::eval_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.161 evalf\_funcp\_12

```
typedef ex(* GiNaC::evalf_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.162 conjugate\_funcp\_12

```
typedef ex(* GiNaC::conjugate_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.163 real\_part\_funcp\_12

```
typedef ex(* GiNaC::real_part_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.164 imag\_part\_funcp\_12

```
typedef ex(* GiNaC::imag_part_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.165 expand\_funcp\_12

```
typedef ex(* GiNaC::expand_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,  
unsigned)
```

#### 5.1.1.166 derivative\_funcp\_12

```
typedef ex(* GiNaC::derivative_funcp_12) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex  
&, unsigned)
```

#### 5.1.1.167 expl\_derivative\_funcp\_12

```
typedef ex(* GiNaC::expl_derivative_funcp_12) (const ex &, const ex &, const ex &, const ex &,  
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex  
&, const symbol &)
```

#### 5.1.1.168 power\_funcp\_12

```
typedef ex(* GiNaC::power_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &)
```

#### 5.1.1.169 series\_funcp\_12

```
typedef ex(* GiNaC::series_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
relational &, int, unsigned)
```

#### 5.1.1.170 print\_funcp\_12

```
typedef void(* GiNaC::print_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
const print_context &)
```

#### 5.1.1.171 info\_funcp\_12

```
typedef bool(* GiNaC::info_funcp_12) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
unsigned)
```

#### 5.1.1.172 eval\_funcp\_13

```
typedef ex(* GiNaC::eval_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &)
```

#### 5.1.1.173 evalf\_funcp\_13

```
typedef ex(* GiNaC::evalf_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &)
```

#### 5.1.1.174 conjugate\_funcp\_13

```
typedef ex(* GiNaC::conjugate_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
const ex &)
```

#### 5.1.1.175 real\_part\_funcp\_13

```
typedef ex(* GiNaC::real_part_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
const ex &)
```

#### 5.1.1.176 imag\_part\_funcp\_13

```
typedef ex(* GiNaC::imag_part_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &  
const ex &)
```

#### 5.1.1.177 expand\_funcp\_13

```
typedef ex(* GiNaC::expand_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, unsigned)
```

#### 5.1.1.178 derivative\_funcp\_13

```
typedef ex(* GiNaC::derivative_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, unsigned)
```

#### 5.1.1.179 expl\_derivative\_funcp\_13

```
typedef ex(* GiNaC::expl_derivative_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const symbol &)
```

#### 5.1.1.180 power\_funcp\_13

```
typedef ex(* GiNaC::power_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.181 series\_funcp\_13

```
typedef ex(* GiNaC::series_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const relational &, int, unsigned)
```

#### 5.1.1.182 print\_funcp\_13

```
typedef void(* GiNaC::print_funcp_13) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const print_context &)
```

#### 5.1.1.183 info\_funcp\_13

```
typedef bool(* GiNaC::info_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, unsigned)
```

#### 5.1.1.184 eval\_funcp\_14

```
typedef ex(* GiNaC::eval_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.185 evalf\_funcp\_14

```
typedef ex(* GiNaC::evalf_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.186 conjugate\_funcp\_14

```
typedef ex(* GiNaC::conjugate_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.187 real\_part\_funcp\_14

```
typedef ex(* GiNaC::real_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.188 imag\_part\_funcp\_14

```
typedef ex(* GiNaC::imag_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &)
```

#### 5.1.1.189 expand\_funcp\_14

```
typedef ex(* GiNaC::expand_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, unsigned)
```

#### 5.1.1.190 derivative\_funcp\_14

```
typedef ex(* GiNaC::derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, unsigned)
```

#### 5.1.1.191 expl\_derivative\_funcp\_14

```
typedef ex(* GiNaC::expl_derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const symbol &)
```

#### 5.1.1.192 power\_funcp\_14

```
typedef ex(* GiNaC::power_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &)
```

#### 5.1.1.193 series\_funcp\_14

```
typedef ex(* GiNaC::series_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.194 print\_funcp\_14

```
typedef void(* GiNaC::print_funcp_14) (const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, const print_context &)
```

#### 5.1.1.195 info\_funcp\_14

```
typedef bool(* GiNaC::info_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex  
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const  
ex &, const ex &, unsigned)
```

#### 5.1.1.196 eval\_funcp\_exvector

```
typedef ex(* GiNaC::eval_funcp_exvector) (const exvector &)
```

#### 5.1.1.197 evalf\_funcp\_exvector

```
typedef ex(* GiNaC::evalf_funcp_exvector) (const exvector &)
```

#### 5.1.1.198 conjugate\_funcp\_exvector

```
typedef ex(* GiNaC::conjugate_funcp_exvector) (const exvector &)
```

#### 5.1.1.199 real\_part\_funcp\_exvector

```
typedef ex(* GiNaC::real_part_funcp_exvector) (const exvector &)
```

#### 5.1.1.200 imag\_part\_funcp\_exvector

```
typedef ex(* GiNaC::imag_part_funcp_exvector) (const exvector &)
```



**5.1.1.201 expand\_funcp\_exvector**

```
typedef ex(* GiNaC::expand_funcp_exvector) (const exvector &, unsigned)
```

**5.1.1.202 derivative\_funcp\_exvector**

```
typedef ex(* GiNaC::derivative_funcp_exvector) (const exvector &, unsigned)
```

**5.1.1.203 expl\_derivative\_funcp\_exvector**

```
typedef ex(* GiNaC::expl_derivative_funcp_exvector) (const exvector &, const symbol &)
```

**5.1.1.204 power\_funcp\_exvector**

```
typedef ex(* GiNaC::power_funcp_exvector) (const exvector &, const ex &)
```

**5.1.1.205 series\_funcp\_exvector**

```
typedef ex(* GiNaC::series_funcp_exvector) (const exvector &, const relational &, int, unsigned)
```

**5.1.1.206 print\_funcp\_exvector**

```
typedef void(* GiNaC::print_funcp_exvector) (const exvector &, const print\_context &)
```

**5.1.1.207 info\_funcp\_exvector**

```
typedef bool(* GiNaC::info_funcp_exvector) (const exvector &, unsigned)
```

#### 5.1.1.208 exhashmap

```
template<typename T , class Hash = std::hash<ex>, class KeyEqual = std::equal_to<ex>, class
Allocator = std::allocator<std::pair<const ex, T>>>
using GiNaC::exhashmap = typedef std::unordered_map<ex, T, Hash, KeyEqual, Allocator>
```

#### 5.1.1.209 spmap

```
typedef std::map<spmapkey, ex> GiNaC::spmap
```

#### 5.1.1.210 lookup\_map

```
typedef map<error_and_integral, ex, error_and_integral_is_less> GiNaC::lookup_map
```

#### 5.1.1.211 lst

```
typedef container< std::list > GiNaC::lst
```

#### 5.1.1.212 uintvector

```
typedef std::vector<std::size_t> GiNaC::uintvector
```

#### 5.1.1.213 unsignedvector

```
typedef std::vector<unsigned> GiNaC::unsignedvector
```

#### 5.1.1.214 exvectorvector

```
typedef std::vector<exvector> GiNaC::exvectorvector
```

**5.1.1.215 sym\_desc\_vec**

```
typedef std::vector<sym_desc> GiNaC::sym_desc_vec
```

**5.1.1.216 digits\_changed\_callback**

```
typedef void(* GiNaC::digits_changed_callback) (long)
```

Function pointer to implement callbacks in the case 'Digits' gets changed.

Main purpose of such callbacks is to adjust look-up tables of certain functions to the new precision. Parameter contains the signed difference between new Digits and old Digits.

**5.1.1.217 print\_context\_class\_info**

```
typedef class_info<print_context_options> GiNaC::print_context_class_info
```

**5.1.1.218 registered\_class\_info**

```
typedef class_info<registered_class_options> GiNaC::registered_class_info
```

**5.1.2 Enumeration Type Documentation****5.1.2.1 anonymous enum**

anonymous enum

Enumerator

callback_registered
---------------------

**5.1.3 Function Documentation****5.1.3.1 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [1/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    add ,
```

```

    expairseq ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
print_tree. print_func< print_python_repr > &::do_print_python_repr )

```

#### 5.1.3.2 GINAC\_BIND\_UNARCHIVER() [1/49]

```

GiNaC::GINAC_BIND_UNARCHIVER (
    add )

```

#### 5.1.3.3 GINAC\_DECLARE\_UNARCHIVER() [1/51]

```

GiNaC::GINAC_DECLARE_UNARCHIVER (
    add )

```

#### 5.1.3.4 write\_unsigned()

```

static void GiNaC::write_unsigned (
    std::ostream & os,
    unsigned val ) [static]

```

Write unsigned integer quantity to stream.

#### 5.1.3.5 read\_unsigned()

```

static unsigned GiNaC::read_unsigned (
    std::istream & is ) [static]

```

Read unsigned integer quantity from stream.

#### 5.1.3.6 operator<<() [1/16]

```

std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const archive_node & n )

```

Write [archive\\_node](#) to binary data stream.

**5.1.3.7 operator<<() [2/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const archive & ar )
```

Write archive to binary data stream.

**5.1.3.8 operator>>() [1/3]**

```
std::istream & GiNaC::operator>> (
    std::istream & is,
    archive_node & n )
```

Read [archive\\_node](#) from binary data stream.

**5.1.3.9 operator>>() [2/3]**

```
std::istream & GiNaC::operator>> (
    std::istream & is,
    archive & ar )
```

Read archive from binary data stream.

**5.1.3.10 find\_factory\_fcn()**

```
static synthesize_func GiNaC::find_factory_fcn (
    const std::string & name ) [static]
```

References [GiNaC::unarchive\\_table\\_t::find\(\)](#).

Referenced by [GiNaC::archive\\_node::unarchive\(\)](#).

**5.1.3.11 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [2/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    basic ,
    void ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print←
_tree. print_func< print_python_repr > &::do_print_python_repr )
```

basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by `duplicate()`), so it can copy the `tinfo_key` and the hash value.

**5.1.3.12 is\_a()** [1/3]

```
template<class T >
bool GiNaC::is_a (
    const basic & obj ) [inline]
```

Check if *obj* is a T, including base classes.

Referenced by [GiNaC::container< C >::subs\(\)](#).

**5.1.3.13 is\_exactly\_a()** [1/2]

```
template<class T >
bool GiNaC::is_exactly_a (
    const basic & obj ) [inline]
```

Check if *obj* is a T, not including base classes.

**5.1.3.14 dynallocate()** [1/2]

```
template<class B , typename... Args>
B & GiNaC::dynallocate (
    Args &&... args ) [inline]
```

Constructs a new (class basic or derived) B object on the heap.

This function picks the object's ctor based on the given argument types.

This helps the constructor of *ex* from basic (or a derived class B) because then the constructor doesn't have to duplicate the object onto the heap. See [ex::construct\\_from\\_basic\(const basic &\)](#) for more information.

References [GiNaC::status\\_flags::dynallocated](#).

**5.1.3.15 dynallocate()** [2/2]

```
template<class B >
B & GiNaC::dynallocate (
    std::initializer_list< ex > il ) [inline]
```

Constructs a new (class basic or derived) B object on the heap.

This function is needed for [GiNaC](#) classes which have public ctors from initializer lists of expressions (which are not a type and not captured by the variadic template version).

References [GiNaC::status\\_flags::dynallocated](#).

**5.1.3.16 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [3/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    clifford ,
    indexed ,
    print_func< print_dflt > &::do_print_dflt. print_func< print_latex > &::do_←
print_latex. print_func< print_tree > &::do_print_tree )
```

**5.1.3.17 print\_func< print\_dflt >()** [1/3]

```
GiNaC::print_func< print_dflt > (
    &diracone::do_print ) &
```

**5.1.3.18 GINAC\_BIND\_UNARCHIVER()** [2/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    clifford )
```

**5.1.3.19 GINAC\_BIND\_UNARCHIVER()** [3/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    cliffordunit )
```

**5.1.3.20 GINAC\_BIND\_UNARCHIVER()** [4/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracone )
```

**5.1.3.21 GINAC\_BIND\_UNARCHIVER()** [5/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgamma )
```

**5.1.3.22 GINAC\_BIND\_UNARCHIVER()** [6/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgamma5 )
```

**5.1.3.23 GINAC\_BIND\_UNARCHIVER()** [7/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgammaL )
```

**5.1.3.24 GINAC\_BIND\_UNARCHIVER()** [8/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    diracgammaR )
```

**5.1.3.25 is\_dirac\_slash()**

```
static bool GiNaC::is_dirac_slash (
    const ex & seq0 ) [static]
```

Referenced by [GiNaC::clifford::do\\_print\\_dflt\(\)](#), and [GiNaC::clifford::do\\_print\\_latex\(\)](#).

**5.1.3.26 base\_and\_index()**

```
static void GiNaC::base_and_index (
    const ex & c,
    ex & b,
    ex & i ) [static]
```

This function decomposes  $\gamma_\mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$

**5.1.3.27 dirac\_ONE()**

```
ex GiNaC::dirac_ONE (
    unsigned char r1 = 0 )
```

Create a Clifford unity object.



## Parameters

<i>rl</i>	Representation label
-----------	----------------------

## Returns

newly constructed object

Referenced by [GiNaC::add::coeff\(\)](#).

### 5.1.3.28 get\_dim\_uint()

```
static unsigned GiNaC::get_dim_uint (
    const ex & e ) [static]
```

### 5.1.3.29 clifford\_unit()

```
ex GiNaC::clifford_unit (
    const ex & mu,
    const ex & metr,
    unsigned char rl = 0 )
```

Create a Clifford unit object.

## Parameters

<i>mu</i>	Index (must be of class varidx or a derived class)
<i>metr</i>	Metric (should be indexed, tensmetric or a derived class, or a matrix)
<i>rl</i>	Representation label

## Returns

newly constructed Clifford unit object

### 5.1.3.30 dirac\_gamma()

```
ex GiNaC::dirac_gamma (
    const ex & mu,
    unsigned char rl = 0 )
```

Create a Dirac gamma object.

**Parameters**

<i>mu</i>	Index (must be of class <code>varidx</code> or a derived class)
<i>rl</i>	Representation label

**Returns**

newly constructed gamma object

**5.1.3.31 `dirac_gamma5()`**

```
ex GiNaC::dirac_gamma5 (
    unsigned char rl = 0 )
```

Create a Dirac gamma5 object.

**Parameters**

<i>rl</i>	Representation label
-----------	----------------------

**Returns**

newly constructed object

**5.1.3.32 `dirac_gammaL()`**

```
ex GiNaC::dirac_gammaL (
    unsigned char rl = 0 )
```

Create a Dirac gammaL object.

**Parameters**

<i>rl</i>	Representation label
-----------	----------------------

**Returns**

newly constructed object

**5.1.3.33 `dirac_gammaR()`**

```
ex GiNaC::dirac_gammaR (
    unsigned char rl = 0 )
```

Create a Dirac gammaR object.

**Parameters**

<i>rl</i>	Representation label
-----------	----------------------

**Returns**

newly constructed object

**5.1.3.34 `dirac_slash()`**

```
ex GiNaC::dirac_slash (
    const ex & e,
    const ex & dim,
    unsigned char rl = 0 )
```

Create a term of the form  $e_\mu * \gamma_{\sim\mu}$  with a unique index  $\mu$ .

**Parameters**

<i>e</i>	Original expression
<i>dim</i>	Dimension of index
<i>rl</i>	Representation label

**5.1.3.35 `get_representation_label()` [1/2]**

```
static unsigned char GiNaC::get_representation_label (
    const return_type_t & ti ) [static]
```

Extract representation label from tinfo key (as returned by `return_type_tinfo()`).

Referenced by [color\\_trace\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), and [GiNaC::su3d::contract\\_with\(\)](#).

**5.1.3.36 `trace_string()`**

```
static ex GiNaC::trace_string (
    exvector::const_iterator ix,
    size_t num ) [static]
```

Take trace of a string of an even number of Dirac gammas given a vector of indices.

**5.1.3.37** `dirac_trace()` [1/3]

```
ex GiNaC::dirac_trace (
    const ex & e,
    const std::set< unsigned char > & rls,
    const ex & trONE = 4 )
```

Calculate dirac traces over the specified set of representation labels.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when  $\gamma_5$  is involved.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rls</i>	Set of representation labels
<i>trONE</i>	Expression to be returned as the trace of the unit matrix

**5.1.3.38** `dirac_trace()` [2/3]

```
ex GiNaC::dirac_trace (
    const ex & e,
    const lst & rll,
    const ex & trONE = 4 )
```

Calculate dirac traces over the specified list of representation labels.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when  $\gamma_5$  is involved.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rll</i>	List of representation labels
<i>trONE</i>	Expression to be returned as the trace of the unit matrix

**5.1.3.39** `dirac_trace()` [3/3]

```
ex GiNaC::dirac_trace (
    const ex & e,
    unsigned char rl = 0,
    const ex & trONE = 4 )
```

Calculate the trace of an expression containing gamma objects with a specified representation label.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when  $\gamma_5$  is involved.

## Parameters

<i>e</i>	Expression to take the trace of
<i>rl</i>	Representation label
<i>trONE</i>	Expression to be returned as the trace of the unit matrix

**5.1.3.40 canonicalize\_clifford()**

```
ex GiNaC::canonicalize_clifford (
    const ex & e )
```

Bring all products of clifford objects in an expression into a canonical order.

This is not necessarily the most simple form but it will allow to check two expressions for equality.

**5.1.3.41 clifford\_star\_bar()**

```
ex GiNaC::clifford_star_bar (
    const ex & e,
    bool do_bar,
    unsigned options )
```

An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).

Referenced by [clifford\\_bar\(\)](#), and [clifford\\_star\(\)](#).

**5.1.3.42 clifford\_prime()**

```
ex GiNaC::clifford_prime (
    const ex & e )
```

Automorphism of the Clifford algebra, simply changes signs of all clifford units.

**5.1.3.43 remove\_dirac\_ONE()**

```
ex GiNaC::remove_dirac_ONE (
    const ex & e,
    unsigned char rl = 0,
    unsigned options = 0 )
```

Replaces `dirac_ONE`'s (with a `representation_label` no less than `rl`) in `e` with 1.

For the default value `rl = 0` remove all of them. Aborts if `e` contains any `clifford_unit` with `representation_label` to be removed.

## Parameters

<i>e</i>	Expression to be processed
<i>rl</i>	Value of representation label
<i>options</i>	Defines some internal use

**5.1.3.44 clifford\_max\_label()**

```
int GiNaC::clifford_max_label (
    const ex & e,
    bool ignore_ONE = false )
```

Returns the maximal representation label of a clifford object if e contains at least one, otherwise returns -1.

## Parameters

<i>e</i>	Expression to be processed @ignore_ONE defines if clifford_ONE should be ignored in the search
----------	--

Referenced by [GiNaC::add::coeff\(\)](#).

**5.1.3.45 clifford\_norm()**

```
ex GiNaC::clifford_norm (
    const ex & e )
```

Calculation of the norm in the Clifford algebra.

**5.1.3.46 clifford\_inverse()**

```
ex GiNaC::clifford_inverse (
    const ex & e )
```

Calculation of the inverse in the Clifford algebra.

**5.1.3.47 lst\_to\_clifford()** [1/2]

```
ex GiNaC::lst_to_clifford (
    const ex & v,
    const ex & mu,
    const ex & metr,
    unsigned char rl = 0 )
```

List or vector conversion into the Clifford vector.

## Parameters

<i>v</i>	List or vector of coordinates
<i>mu</i>	Index (must be of class varidx or a derived class)
<i>metr</i>	Metric (should be indexed, tensmetric or a derived class, or a matrix)
<i>rl</i>	Representation label
<i>e</i>	Clifford unit object

## Returns

Clifford vector with given components

5.1.3.48 `lst_to_clifford()` [2/2]

```
ex GiNaC::lst_to_clifford (
    const ex & v,
    const ex & e )
```

5.1.3.49 `get_clifford_comp()`

```
static ex GiNaC::get_clifford_comp (
    const ex & e,
    const ex & c,
    bool root = true ) [static]
```

Auxiliary structure to define a function for stripping one Clifford unit from vectors.

Used in [clifford\\_to\\_lst\(\)](#).

5.1.3.50 `clifford_to_lst()`

```
lst GiNaC::clifford_to_lst (
    const ex & e,
    const ex & c,
    bool algebraic = true )
```

An inverse function to [lst\\_to\\_clifford\(\)](#).

For given Clifford vector extracts its components with respect to given Clifford unit. Obtained components may contain Clifford units with a different metric. Extraction is based on the algebraic formula  $(e * c.i + c.i * e) / \text{pow}(e.i, 2)$  for non-degenerate cases (i.e. neither  $\text{pow}(e.i, 2) = 0$ ).

## Parameters

<i>e</i>	Clifford expression to be decomposed into components
<i>c</i>	Clifford unit defining the metric for splitting (should have numeric dimension of indices)
<i>algebraic</i>	Use algebraic or symbolic algorithm for extractions

**Returns**

List of components of a Clifford vector

**5.1.3.51 clifford\_moebius\_map()** [1/2]

```
ex GiNaC::clifford_moebius_map (
    const ex & a,
    const ex & b,
    const ex & c,
    const ex & d,
    const ex & v,
    const ex & G,
    unsigned char rl = 0 )
```

Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b\c d) in linear spaces with arbitrary signature.

The expression is  $(a * x + b)/(c * x + d)$ , where x is a vector build from list v with metric G. (see Jan Cnops. An introduction to {D}irac operators on manifolds, v.24 of Progress in Mathematical Physics. Birkhauser Boston Inc., Boston, MA, 2002.)

**Parameters**

<i>a</i>	(1,1) entry of the defining matrix
<i>b</i>	(1,2) entry of the defining matrix
<i>c</i>	(2,1) entry of the defining matrix
<i>d</i>	(2,2) entry of the defining matrix
<i>v</i>	Vector to be transformed
<i>G</i>	Metric of the surrounding space, may be a Clifford unit then the next parameter is ignored
<i>rl</i>	Representation label

**Returns**

List of components of the transformed vector

**5.1.3.52 clifford\_moebius\_map()** [2/2]

```
ex GiNaC::clifford_moebius_map (
    const ex & M,
    const ex & v,
    const ex & G,
    unsigned char rl = 0 )
```

The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.



## Parameters

$M$	the defining matrix
$v$	Vector to be transformed
$G$	Metric of the surrounding space, may be a Clifford unit then the next parameter is ignored
$rl$	Representation label

## Returns

List of components of the transformed vector

**5.1.3.53 GINAC\_DECLARE\_UNARCHIVER()** [2/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    clifford )
```

**5.1.3.54 GINAC\_DECLARE\_UNARCHIVER()** [3/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracone )
```

**5.1.3.55 GINAC\_DECLARE\_UNARCHIVER()** [4/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    cliffordunit )
```

**5.1.3.56 GINAC\_DECLARE\_UNARCHIVER()** [5/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgamma )
```

**5.1.3.57 GINAC\_DECLARE\_UNARCHIVER()** [6/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgamma5 )
```

**5.1.3.58 GINAC\_DECLARE\_UNARCHIVER()** [7/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgammaL )
```

**5.1.3.59 GINAC\_DECLARE\_UNARCHIVER()** [8/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    diracgammaR )
```

**5.1.3.60 is\_clifford\_tinfo()**

```
bool GiNaC::is_clifford_tinfo (
    const return_type_t & ti ) [inline]
```

Check whether a given [return\\_type\\_t](#) object (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a clifford object (with an arbitrary representation label).

**Parameters**

<i>ti</i>	tinfo key
-----------	-----------

References [GiNaC::return\\_type\\_t::tinfo](#).

Referenced by [GiNaC::ncmul::conjugate\(\)](#).

**5.1.3.61 clifford\_bar()**

```
ex GiNaC::clifford_bar (
    const ex & e ) [inline]
```

Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.

References [clifford\\_star\\_bar\(\)](#).

**5.1.3.62 clifford\_star()**

```
ex GiNaC::clifford_star (
    const ex & e ) [inline]
```

Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.

References [clifford\\_star\\_bar\(\)](#).

**5.1.3.63 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [4/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    su3one ,
    tensor ,
    print_func< print_dflt > &::do_print. print_func< print_latex > &::do_print_↵
    latex )
```

**5.1.3.64 print\_func< print\_dflt >()** [2/3]

```
GiNaC::print_func< print_dflt > (
    &su3t::do_print ) &
```

**5.1.3.65 GINAC\_BIND\_UNARCHIVER()** [9/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    color )
```

**5.1.3.66 GINAC\_BIND\_UNARCHIVER()** [10/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3one )
```

**5.1.3.67 GINAC\_BIND\_UNARCHIVER()** [11/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3t )
```

**5.1.3.68 GINAC\_BIND\_UNARCHIVER()** [12/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3f )
```

### 5.1.3.69 GINAC\_BIND\_UNARCHIVER() [13/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    su3d )
```

### 5.1.3.70 permute\_free\_index\_to\_front()

```
static ex GiNaC::permute_free_index_to_front (
    const exvector & iv3,
    const exvector & iv2,
    int & sig ) [static]
```

Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.

#### Parameters

<i>iv3</i>	Vector of 3 indices
<i>iv2</i>	Vector of 2 indices, must be a subset of iv3
<i>sig</i>	Returns sign introduced by index permutation

#### Returns

the free index (the one that is in iv3 but not in iv2)

References [GINAC\\_ASSERT](#), and [TEST\\_PERMUTATION](#).

Referenced by [GiNaC::su3f::contract\\_with\(\)](#), and [GiNaC::su3d::contract\\_with\(\)](#).

### 5.1.3.71 color\_ONE()

```
ex GiNaC::color_ONE (
    unsigned char rl = 0 )
```

Create the su(3) unity element.

This is an indexed object, although it has no indices.

#### Parameters

<i>rl</i>	Representation label
-----------	----------------------

#### Returns

newly constructed unity element

Referenced by [GiNaC::su3t::contract\\_with\(\)](#).

### 5.1.3.72 color\_T()

```
ex GiNaC::color_T (
    const ex & a,
    unsigned char rl = 0 )
```

Create an  $su(3)$  generator.

#### Parameters

<i>a</i>	Index
<i>rl</i>	Representation label

#### Returns

newly constructed unity generator

Referenced by [color\\_trace\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), and [GiNaC::su3d::contract\\_with\(\)](#).

### 5.1.3.73 color\_f()

```
ex GiNaC::color_f (
    const ex & a,
    const ex & b,
    const ex & c )
```

Create an  $su(3)$  antisymmetric structure constant.

#### Parameters

<i>a</i>	First index
<i>b</i>	Second index
<i>c</i>	Third index

#### Returns

newly constructed structure constant

References [antisymmetric3\(\)](#), and [c](#).

Referenced by [color\\_h\(\)](#).

#### 5.1.3.74 `color_d()`

```
ex GiNaC::color_d (
    const ex & a,
    const ex & b,
    const ex & c )
```

Create an  $su(3)$  symmetric structure constant.

##### Parameters

<i>a</i>	First index
<i>b</i>	Second index
<i>c</i>	Third index

##### Returns

newly constructed structure constant

References [c](#), and [symmetric3\(\)](#).

Referenced by [color\\_h\(\)](#).

#### 5.1.3.75 `color_h()`

```
ex GiNaC::color_h (
    const ex & a,
    const ex & b,
    const ex & c )
```

This returns the linear combination  $d.a.b.c + l.f.a.b.c$ .

References [c](#), [color\\_d\(\)](#), [color\\_f\(\)](#), and [l](#).

Referenced by [color\\_trace\(\)](#).

#### 5.1.3.76 `is_color_tinfo()`

```
static bool GiNaC::is_color_tinfo (
    const return_type_t & ti ) [static]
```

Check whether a given tinfo key (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a color object (with an arbitrary representation label).

References [GiNaC::return\\_type\\_t::tinfo](#).

Referenced by [color\\_trace\(\)](#).

**5.1.3.77 `get_representation_label()` [2/2]**

```
static unsigned char GiNaC::get_representation_label (
    const return\_type\_t & ti ) [static]
```

Extract representation label from tinfo key (as returned by `return_type_tinfo()`).

References [GiNaC::return\\_type\\_t::rl](#).

**5.1.3.78 `color_trace()` [1/3]**

```
ex GiNaC::color_trace (
    const ex & e,
    const std::set< unsigned char > & rls )
```

Calculate color traces over the specified set of representation labels.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rls</i>	Set of representation labels

References [\\_ex0](#), [\\_ex1](#), [\\_ex3](#), [color\\_h\(\)](#), [color\\_T\(\)](#), [color\\_trace\(\)](#), [delta\\_tensor\(\)](#), [GiNaC::ex::expand\(\)](#), [get\\_representation\\_label\(\)](#), [is\\_color\\_tinfo\(\)](#), [GiNaC::ex::map\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#)

Referenced by [color\\_trace\(\)](#), and [GiNaC::su3t::contract\\_with\(\)](#).

**5.1.3.79 `color_trace()` [2/3]**

```
ex GiNaC::color_trace (
    const ex & e,
    const lst & rl1 )
```

Calculate color traces over the specified list of representation labels.

**Parameters**

<i>e</i>	Expression to take the trace of
<i>rl1</i>	List of representation labels

References [color\\_trace\(\)](#), [GiNaC::info\\_flags::nonnegint](#), and [to\\_int\(\)](#).

**5.1.3.80 `color_trace()` [3/3]**

```
ex GiNaC::color_trace (
```

```
const ex & e,
unsigned char rl = 0 )
```

Calculate the trace of an expression containing color objects with a specified representation label.

#### Parameters

<i>e</i>	Expression to take the trace of
<i>rl</i>	Representation label

References [color\\_trace\(\)](#).

#### 5.1.3.81 GINAC\_DECLARE\_UNARCHIVER() [9/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    color )
```

#### 5.1.3.82 GINAC\_DECLARE\_UNARCHIVER() [10/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3one )
```

#### 5.1.3.83 GINAC\_DECLARE\_UNARCHIVER() [11/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3t )
```

#### 5.1.3.84 GINAC\_DECLARE\_UNARCHIVER() [12/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3f )
```

#### 5.1.3.85 GINAC\_DECLARE\_UNARCHIVER() [13/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    su3d )
```



**5.1.3.86 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [5/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    constant ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_←
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr > &←
::do_print_python_repr ) const
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**5.1.3.87 GINAC\_BIND\_UNARCHIVER()** [14/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    constant )
```

**5.1.3.88 GINAC\_DECLARE\_UNARCHIVER()** [14/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    constant )
```

**5.1.3.89 crc32()**

```
static unsigned GiNaC::crc32 (
    const char * c,
    const unsigned len,
    const unsigned crcinit ) [static]
```

References [c](#), [crctab](#), and [len](#).

**5.1.3.90 are\_ex\_trivially\_equal()**

```
bool GiNaC::are_ex_trivially_equal (
    const ex & e1,
    const ex & e2 ) [inline]
```

Compare two objects of class quickly without doing a deep tree traversal.

**Returns**

"true" if they are equal "false" if equality cannot be established quickly (e1 and e2 may still be equal, in this case).

Referenced by [GiNaC::expair::conjugate\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::expairseq::conjugate\(\)](#), [GiNaC::integral::conjugate\(\)](#), [GiNaC::matrix::conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::power::conjugate\(\)](#), [GiNaC::pseries::conjugate\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::expairseq::expand\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::make\\_flat\\_inserter::make\\_flat\(\)](#), [GiNaC::basic::map\(\)](#), [GiNaC::idx::map\(\)](#), [GiNaC::power::map\(\)](#), [GiNaC::relational::map\(\)](#), [GiNaC::internal::\\_iter\\_rep::operator==\(\)](#), [GiNaC::const\\_iterator::operator==\(\)](#), [GiNaC::basic::subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::container< C >::subschildren\(\)](#), and [GiNaC::expairseq::subschildren\(\)](#).

**5.1.3.91 operator<<() [3/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const exvector & e )
```

References [get\\_print\\_context\(\)](#).

**5.1.3.92 operator<<() [4/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const exset & e )
```

References [get\\_print\\_context\(\)](#).

**5.1.3.93 operator<<() [5/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const exmap & e )
```

References [get\\_print\\_context\(\)](#).

**5.1.3.94 nops() [1/2]**

```
size_t GiNaC::nops (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::nops\(\)](#).

Referenced by [GiNaC::basic::calchash\(\)](#), [GiNaC::basic::derivative\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::do\\_print](#), [GiNaC::basic::eval\\_integ\(\)](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::basic::evalm\(\)](#), [GiNaC::basic::expand\(\)](#), [GiNaC::basic::has\(\)](#), [GiNaC::container< C >::let\\_op\(\)](#), [GiNaC::basic::map\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::basic::subs\(\)](#).

**5.1.3.95 expand() [1/2]**

```
ex GiNaC::expand (
    const ex & thisex,
    unsigned options = 0 ) [inline]
```

References [GiNaC::ex::expand\(\)](#), and [options](#).

Referenced by [GiNaC::basic::collect\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [log\\_expand\(\)](#), [prem\(\)](#), [quo\(\)](#), [rem\(\)](#), and [sprem\(\)](#).

### 5.1.3.96 conjugate()

```
ex GiNaC::conjugate (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::conjugate\(\)](#).

Referenced by [conjugate\\_expl\\_derivative\(\)](#).

### 5.1.3.97 real\_part()

```
ex GiNaC::real_part (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::real\\_part\(\)](#).

Referenced by [cos\\_imag\\_part\(\)](#), [cos\\_real\\_part\(\)](#), [cosh\\_imag\\_part\(\)](#), [cosh\\_real\\_part\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_real\\_part\(\)](#), [log\\_imag\\_part\(\)](#), [real\\_part\\_expl\\_derivative\(\)](#), [sin\\_imag\\_part\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_imag\\_part\(\)](#), [sinh\\_real\\_part\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

### 5.1.3.98 imag\_part()

```
ex GiNaC::imag_part (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::imag\\_part\(\)](#).

Referenced by [cos\\_imag\\_part\(\)](#), [cos\\_real\\_part\(\)](#), [cosh\\_imag\\_part\(\)](#), [cosh\\_real\\_part\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_real\\_part\(\)](#), [imag\\_part\\_expl\\_derivative\(\)](#), [log\\_imag\\_part\(\)](#), [sin\\_imag\\_part\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_imag\\_part\(\)](#), [sinh\\_real\\_part\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

### 5.1.3.99 has()

```
bool GiNaC::has (
    const ex & thisex,
    const ex & pattern,
    unsigned options = 0 ) [inline]
```

References [GiNaC::ex::has\(\)](#), and [options](#).

Referenced by [GiNaC::basic::is\\_polynomial\(\)](#).

#### 5.1.3.100 find()

```
bool GiNaC::find (
    const ex & thisex,
    const ex & pattern,
    exset & found ) [inline]
```

References [GiNaC::ex::find\(\)](#).

#### 5.1.3.101 is\_polynomial()

```
bool GiNaC::is_polynomial (
    const ex & thisex,
    const ex & vars ) [inline]
```

References [GiNaC::ex::is\\_polynomial\(\)](#).

#### 5.1.3.102 degree()

```
int GiNaC::degree (
    const ex & thisex,
    const ex & s ) [inline]
```

References [GiNaC::ex::degree\(\)](#).

Referenced by [GiNaC::basic::collect\(\)](#), [replace\\_with\\_symbol\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.103 ldegree()

```
int GiNaC::ldegree (
    const ex & thisex,
    const ex & s ) [inline]
```

References [GiNaC::ex::ldegree\(\)](#).

Referenced by [GiNaC::basic::collect\(\)](#).

#### 5.1.3.104 `coeff()`

```
ex GiNaC::coeff (
    const ex & thisex,
    const ex & s,
    int n = 1 ) [inline]
```

References [GiNaC::ex::coeff\(\)](#), and [n](#).

Referenced by [GiNaC::basic::collect\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [iterated\\_integral\\_evalf\\_impl\(\)](#), [log\\_series\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.105 `numer()` [1/2]

```
ex GiNaC::numer (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::numer\(\)](#).

Referenced by [decomp\\_rational\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [print\\_real\\_csrc\(\)](#), [GiNaC::power::real\\_part\(\)](#), [replace\\_with\\_symbol\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.106 `denom()` [1/2]

```
ex GiNaC::denom (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::denom\(\)](#).

Referenced by [decomp\\_rational\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [lcmcoeff\(\)](#), [print\\_real\\_csrc\(\)](#), [replace\\_with\\_symbol\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.107 `numer_denom()`

```
ex GiNaC::numer_denom (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::numer\\_denom\(\)](#).

Referenced by [decomp\\_rational\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.108 normal()

```
ex GiNaC::normal (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::normal\(\)](#).

Referenced by [fsolve\(\)](#), [GiNaC::normal\\_map\\_function::operator\(\)](#), and [replace\\_with\\_symbol\(\)](#).

#### 5.1.3.109 to\_rational()

```
ex GiNaC::to_rational (
    const ex & thisex,
    exmap & repl ) [inline]
```

References [GiNaC::ex::to\\_rational\(\)](#).

#### 5.1.3.110 to\_polynomial()

```
ex GiNaC::to_polynomial (
    const ex & thisex,
    exmap & repl ) [inline]
```

References [GiNaC::ex::to\\_polynomial\(\)](#).

#### 5.1.3.111 collect()

```
ex GiNaC::collect (
    const ex & thisex,
    const ex & s,
    bool distributed = false ) [inline]
```

References [GiNaC::ex::collect\(\)](#).

Referenced by [GiNaC::basic::collect\(\)](#).

#### 5.1.3.112 eval()

```
ex GiNaC::eval (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::eval\(\)](#).

#### 5.1.3.113 evalf() [1/2]

```
ex GiNaC::evalf (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::evalf\(\)](#).

Referenced by [beta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [GiNaC::evalf\\_map\\_function::operator\(\)](#), and [zeta2\\_evalf\(\)](#).

#### 5.1.3.114 evalm()

```
ex GiNaC::evalm (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::evalm\(\)](#).

Referenced by [GiNaC::evalm\\_map\\_function::operator\(\)](#).

#### 5.1.3.115 eval\_integ()

```
ex GiNaC::eval_integ (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::eval\\_integ\(\)](#).

Referenced by [GiNaC::eval\\_integ\\_map\\_function::operator\(\)](#).

#### 5.1.3.116 diff()

```
ex GiNaC::diff (
    const ex & thisex,
    const symbol & s,
    unsigned nth = 1 ) [inline]
```

References [GiNaC::ex::diff\(\)](#).

Referenced by [GiNaC::derivative\\_map\\_function::operator\(\)](#).

**5.1.3.117 series()**

```

ex GiNaC::series (
    const ex & thisex,
    const ex & r,
    int order,
    unsigned options = 0 ) [inline]

```

References [options](#), [order](#), [r](#), and [GiNaC::ex::series\(\)](#).

Referenced by [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [Li2\\_series\(\)](#), and [log\\_series\(\)](#).

**5.1.3.118 match()**

```

bool GiNaC::match (
    const ex & thisex,
    const ex & pattern,
    exmap & repl_lst ) [inline]

```

References [GiNaC::ex::match\(\)](#).

Referenced by [GiNaC::basic::has\(\)](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

**5.1.3.119 simplify\_indexed() [1/3]**

```

ex GiNaC::simplify_indexed (
    const ex & thisex,
    unsigned options = 0 ) [inline]

```

References [options](#), and [GiNaC::ex::simplify\\_indexed\(\)](#).

Referenced by [GiNaC::ex::simplify\\_indexed\(\)](#).

**5.1.3.120 simplify\_indexed() [2/3]**

```

ex GiNaC::simplify_indexed (
    const ex & thisex,
    const scalar_products & sp,
    unsigned options = 0 ) [inline]

```

References [options](#), and [GiNaC::ex::simplify\\_indexed\(\)](#).



**5.1.3.121 symmetrize()** [1/4]

```
ex GiNaC::symmetrize (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::symmetrize\(\)](#).

Referenced by [idx\\_symmetrization\(\)](#), [GiNaC::ex::symmetrize\(\)](#), [symmetrize\(\)](#), and [symmetrize\\_cyclic\(\)](#).

**5.1.3.122 symmetrize()** [2/4]

```
ex GiNaC::symmetrize (
    const ex & thisex,
    const lst & l ) [inline]
```

References [GiNaC::ex::symmetrize\(\)](#).

**5.1.3.123 antisymmetrize()** [1/4]

```
ex GiNaC::antisymmetrize (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::antisymmetrize\(\)](#).

Referenced by [GiNaC::ex::antisymmetrize\(\)](#), and [antisymmetrize\(\)](#).

**5.1.3.124 antisymmetrize()** [2/4]

```
ex GiNaC::antisymmetrize (
    const ex & thisex,
    const lst & l ) [inline]
```

References [GiNaC::ex::antisymmetrize\(\)](#).

**5.1.3.125 symmetrize\_cyclic()** [1/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

Referenced by [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

**5.1.3.126 symmetrize\_cyclic()** [2/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & thisex,
    const lst & l ) [inline]
```

References [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

**5.1.3.127 op()**

```
ex GiNaC::op (
    const ex & thisex,
    size_t i ) [inline]
```

References [GiNaC::ex::op\(\)](#).

Referenced by [GiNaC::basic::calchash\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::basic::has\(\)](#), [Li2\\_series\(\)](#), [GiNaC::basic::map\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [rename\\_dummy\\_indices\(\)](#), and [GiNaC::basic::subs\(\)](#).

**5.1.3.128 lhs()**

```
ex GiNaC::lhs (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::lhs\(\)](#).

**5.1.3.129 rhs()**

```
ex GiNaC::rhs (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::rhs\(\)](#).

Referenced by [lsolve\(\)](#), [GiNaC::ptr< T >::operator!=\(\)](#), [GiNaC::ptr< T >::operator==\(\)](#), [GiNaC::matrix::solve\(\)](#), and [sqrfree\\_parfrac\(\)](#).

**5.1.3.130 is\_zero()** [1/2]

```
bool GiNaC::is_zero (
    const ex & thisex ) [inline]
```

References [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [cosh\\_eval\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::mul\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::matrix::pivot\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

#### 5.1.3.131 `swap()` [1/2]

```
void GiNaC::swap (
    ex & e1,
    ex & e2 ) [inline]
```

References [GiNaC::ex::swap\(\)](#).

Referenced by [permutation\\_sign\(\)](#), and [GiNaC::matrix::pivot\(\)](#).

#### 5.1.3.132 `subs()` [1/3]

```
ex GiNaC::subs (
    const ex & thisex,
    const exmap & m,
    unsigned options = 0 ) [inline]
```

References [m](#), [options](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [Li2\\_series\(\)](#).

#### 5.1.3.133 `subs()` [2/3]

```
ex GiNaC::subs (
    const ex & thisex,
    const lst & ls,
    const lst & lr,
    unsigned options = 0 ) [inline]
```

References [lr](#), [options](#), and [GiNaC::ex::subs\(\)](#).

#### 5.1.3.134 `subs()` [3/3]

```
ex GiNaC::subs (
    const ex & thisex,
    const ex & e,
    unsigned options = 0 ) [inline]
```

References [options](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.135 is\_a()** [2/3]

```
template<class T >
bool GiNaC::is_a (
    const ex & obj ) [inline]
```

Check if *ex* is a handle to a T, including base classes.

**5.1.3.136 is\_exactly\_a()** [2/2]

```
template<class T >
bool GiNaC::is_exactly_a (
    const ex & obj ) [inline]
```

Check if *ex* is a handle to a T, not including base classes.

**5.1.3.137 ex\_to()**

```
template<class T >
const T & GiNaC::ex_to (
    const ex & e ) [inline]
```

Return a reference to the basic-derived class T object embedded in an expression.

This is fast but unsafe: the result is undefined if the expression does not contain a T object at its top level. Hence, you should generally check the type of *e* first. Also, you shouldn't cache the returned reference because GiNaC's garbage collector may destroy the referenced object any time it's used in another expression.

**Parameters**

<i>e</i>	expression
----------	------------

**Returns**

reference to object of class T

**See also**

[is\\_exactly\\_a<class T>\(\)](#)

**5.1.3.138 compile\_ex()** [1/3]

```
void GiNaC::compile_ex (
    const ex & expr,
```

```
const symbol & sym,  
FUNCP_1P & fp,  
const std::string filename = "" )
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type FUNCP\_1P.

#### Parameters

<i>expr</i>	Expression to be compiled
<i>sym</i>	Symbol from the expression to become the function parameter
<i>fp</i>	Returned function pointer
<i>filename</i>	Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted

#### 5.1.3.139 compile\_ex() [2/3]

```
void GiNaC::compile_ex (  
    const ex & expr,  
    const symbol & sym1,  
    const symbol & sym2,  
    FUNCP_2P & fp,  
    const std::string filename = "" )
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type FUNCP\_2P.

#### Parameters

<i>expr</i>	Expression to be compiled
<i>sym</i>	Symbol from the expression to become the function parameter
<i>fp</i>	Returned function pointer
<i>filename</i>	Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted

#### 5.1.3.140 compile\_ex() [3/3]

```
void GiNaC::compile_ex (  
    const lst & exprs,  
    const lst & syms,  
    FUNCP_CUBA & fp,  
    const std::string filename = "" )
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type FUNCP\_CUBA.

#### Parameters

<i>expr</i>	Expression to be compiled
<i>sym</i>	Symbol from the expression to become the function parameter
<i>fp</i>	Returned function pointer
<i>filename</i>	Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted

#### 5.1.3.141 link\_ex() [1/3]

```
void GiNaC::link_ex (
    const std::string filename,
    FUNCP_1P & fp )
```

Opens an existing so-file and returns a function pointer of type FUNCP\_1P to the contained function.

The so-file has to be generated by compile\_ex in advance.

#### Parameters

<i>filename</i>	Name of the so-file to open and link
<i>fp</i>	Returned function pointer

#### 5.1.3.142 link\_ex() [2/3]

```
void GiNaC::link_ex (
    const std::string filename,
    FUNCP_2P & fp )
```

Opens an existing so-file and returns a function pointer of type FUNCP\_2P to the contained function.

The so-file has to be generated by compile\_ex in advance.

#### Parameters

<i>filename</i>	Name of the so-file to open and link
<i>fp</i>	Returned function pointer

**5.1.3.143 link\_ex()** [3/3]

```
void GiNaC::link_ex (
    const std::string filename,
    FUNCP_CUBA & fp )
```

Opens an existing so-file and returns a function pointer of type FUNCP\_CUBA to the contained function.

The so-file has to be generated by compile\_ex in advance.

**Parameters**

<i>filename</i>	Name of the so-file to open and link
<i>fp</i>	Returned function pointer

**5.1.3.144 unlink\_ex()**

```
void GiNaC::unlink_ex (
    const std::string filename )
```

Closes all linked .so files that have the supplied filename.

**Parameters**

<i>filename</i>	Name of the so-file to close
-----------------	------------------------------

**5.1.3.145 swap()** [2/2]

```
void GiNaC::swap (
    expair & e1,
    expair & e2 ) [inline]
```

References [GiNaC::expair::swap\(\)](#).

**5.1.3.146 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [6/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    expairseq ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print↵
    _tree )
```

### 5.1.3.147 conjugateepvector()

```
epvector * GiNaC::conjugateepvector (
    const epvector & )
```

Complex conjugate every element of an epvector.

Returns zero if this does not change anything.

References [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [x](#).

Referenced by [GiNaC::expairseq::conjugate\(\)](#), and [GiNaC::pseries::conjugate\(\)](#).

### 5.1.3.148 factor()

```
ex GiNaC::factor (
    const ex & poly,
    unsigned options )
```

Interface function to the outside world.

Factorizes univariate and multivariate polynomials.

It uses [factor1\(\)](#) on each of the explicitly present factors of [poly](#).

The default option is [factor\\_options::polynomial](#), which means that [factor\(\)](#) will only factorize an expression if it is a proper polynomial (i.e. the flag [info\\_flags::polynomial](#) is set). Given the option [factor\\_options::all](#), [factor\(\)](#) will factorize all subexpressions, e.g. polynomials containing functions or polynomials inside function arguments.

#### Parameters

in	<i>poly</i>	expression to factorize
in	<i>option</i>	options to influence the factorization

#### Returns

factorized expression

References [options](#), [poly](#), and [pow\(\)](#).

Referenced by [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [collect\\_common\\_factors\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::mul::series\(\)](#), and [sqrfree\\_parfrac\(\)](#).

### 5.1.3.149 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [7/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    fail ,
```



```
        basic ,  
        print_func< print_context > &::do_print. print_func< print_tree > &::do_print↵  
_tree )
```

#### 5.1.3.150 GINAC\_DECLARE\_UNARCHIVER() [15/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (  
    fail )
```

#### 5.1.3.151 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [8/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (  
    fderivative ,  
    function ,  
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵  
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵  
print_tree )
```

#### 5.1.3.152 GINAC\_BIND\_UNARCHIVER() [15/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (  
    fderivative )
```

#### 5.1.3.153 GINAC\_DECLARE\_UNARCHIVER() [16/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (  
    fderivative )
```

#### 5.1.3.154 GINAC\_BIND\_UNARCHIVER() [16/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (  
    function )
```

**5.1.3.155 GINAC\_DECLARE\_UNARCHIVER()** [17/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    function )
```

**5.1.3.156 is\_the\_function()**

```
template<typename T >
bool GiNaC::is_the_function (
    const ex & x ) [inline]
```

References [x](#).

**5.1.3.157 make\_hash\_seed()**

```
static unsigned GiNaC::make_hash_seed (
    const std::type_info & tinfo ) [inline], [static]
```

We need a hash function which gives different values for objects of different types.

Hence we need some unique integer for each type. Fortunately, standard C++ RTTI class 'type\_info' stores a pointer to mangled type name. Normally this pointer is the same for all objects of the same type (although it changes from run to run), so it can be used for computing hashes. However, on some platforms (such as woe32) the pointer returned by type\_info::name() might be different even for objects of the same type! Hence we need to resort to comparing string representation of the (mangled) type names. This is quite expensive, so we compare crc32 hashes of those strings. We might get more hash collisions (and slower evaluation as a result), but being a bit slower is much better than being wrong.

References [golden\\_ratio\\_hash\(\)](#).

Referenced by [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symbol::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), and [GiNaC::wildcard::calchash\(\)](#).

**5.1.3.158 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [9/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    idx ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
print_tree )
```

**5.1.3.159 print\_func< print\_context >()**

```
GiNaC::print_func< print_context > (
    &varidx::do_print ) &
```

**5.1.3.160 GINAC\_BIND\_UNARCHIVER() [17/49]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    idx )
```

**5.1.3.161 GINAC\_BIND\_UNARCHIVER() [18/49]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    varidx )
```

**5.1.3.162 GINAC\_BIND\_UNARCHIVER() [19/49]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    spinidx )
```

**5.1.3.163 is\_dummy\_pair() [1/2]**

```
bool GiNaC::is_dummy_pair (
    const idx & i1,
    const idx & i2 )
```

Check whether two indices form a dummy pair.

References [GiNaC::idx::is\\_dummy\\_pair\\_same\\_type\(\)](#).

Referenced by [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [find\\_free\\_and\\_dummy\(\)](#), [GiNaC::indexed::has\\_dummy\\_index\\_for\(\)](#), [is\\_dummy\\_pair\(\)](#), [GiNaC::is\\_summation\\_idx::operator\(\)\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

**5.1.3.164 is\_dummy\_pair()** [2/2]

```
bool GiNaC::is_dummy_pair (
    const ex & e1,
    const ex & e2 )
```

Check whether two expressions form a dummy index pair.

References [is\\_dummy\\_pair\(\)](#).

**5.1.3.165 find\_free\_and\_dummy()** [1/2]

```
void GiNaC::find_free_and_dummy (
    exvector::const_iterator it,
    exvector::const_iterator itend,
    exvector & out_free,
    exvector & out_dummy )
```

Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).

**Parameters**

<i>it</i>	Pointer to start of index vector
<i>itend</i>	Pointer to end of index vector
<i>out_free</i>	Vector of free indices (returned, sorted)
<i>out_dummy</i>	Vector of dummy indices (returned, sorted)

References [is\\_dummy\\_pair\(\)](#), [last](#), and [shaker\\_sort\(\)](#).

Referenced by [GiNaC::su3d::contract\\_with\(\)](#), [count\\_dummy\\_indices\(\)](#), [count\\_free\\_indices\(\)](#), [find\\_dummy\\_indices\(\)](#), [find\\_free\\_and\\_dummy\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::indexed::get\\_dummy\\_indices\(\)](#), [GiNaC::indexed::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), and [GiNaC::ncmul::get\\_free\\_indices\(\)](#).

**5.1.3.166 minimal\_dim()**

```
ex GiNaC::minimal_dim (
    const ex & dim1,
    const ex & dim2 )
```

Return the minimum of two index dimensions.

If this is undecidable, throw an exception. Numeric dimensions are always considered "smaller" than symbolic dimensions.

References [GiNaC::ex::is\\_equal\(\)](#).

Referenced by [GiNaC::idx::minimal\\_dim\(\)](#).

**5.1.3.167 GINAC\_DECLARE\_UNARCHIVER()** [18/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    idx )
```

**5.1.3.168 GINAC\_DECLARE\_UNARCHIVER()** [19/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    varidx )
```

**5.1.3.169 GINAC\_DECLARE\_UNARCHIVER()** [20/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    spinidx )
```

**5.1.3.170 find\_free\_and\_dummy()** [2/2]

```
void GiNaC::find_free_and_dummy (
    const exvector & v,
    exvector & out_free,
    exvector & out_dummy ) [inline]
```

Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).

**Parameters**

<i>v</i>	Index vector
<i>out_free</i>	Vector of free indices (returned, sorted)
<i>out_dummy</i>	Vector of dummy indices (returned, sorted)

References [find\\_free\\_and\\_dummy\(\)](#).

**5.1.3.171 find\_dummy\_indices()**

```
void GiNaC::find_dummy_indices (
    const exvector & v,
    exvector & out_dummy ) [inline]
```

Given a vector of indices, find the dummy indices.

## Parameters

<i>v</i>	Index vector
<i>out_dummy</i>	Vector of dummy indices (returned, sorted)

References [find\\_free\\_and\\_dummy\(\)](#).

Referenced by [GiNaC::indexed::get\\_dummy\\_indices\(\)](#).

### 5.1.3.172 count\_dummy\_indices()

```
size_t GiNaC::count_dummy_indices (
    const exvector & v ) [inline]
```

Count the number of dummy index pairs in an index vector.

References [find\\_free\\_and\\_dummy\(\)](#).

### 5.1.3.173 count\_free\_indices()

```
size_t GiNaC::count_free_indices (
    const exvector & v ) [inline]
```

Count the number of dummy index pairs in an index vector.

References [find\\_free\\_and\\_dummy\(\)](#).

### 5.1.3.174 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [10/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    indexed ,
    exprseq ,
    print_func< print\_context > &::do_print. print_func< print\_latex > &::do_↵
    print_latex. print_func< print\_tree > &::do_print_tree )
```

### 5.1.3.175 GINAC\_BIND\_UNARCHIVER() [20/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    indexed )
```

#### 5.1.3.176 indices\_consistent()

```
static bool GiNaC::indices_consistent (
    const exvector & v1,
    const exvector & v2 ) [static]
```

Check whether two sorted index vectors are consistent (i.e. equal).

Referenced by [GiNaC::add::get\\_free\\_indices\(\)](#).

#### 5.1.3.177 number\_of\_type()

```
template<class T >
size_t GiNaC::number_of_type (
    const exvector & v )
```

#### 5.1.3.178 rename\_dummy\_indices()

```
template<class T >
static ex GiNaC::rename_dummy_indices (
    const ex & e,
    exvector & global_dummy_indices,
    exvector & local_dummy_indices ) [static]
```

Rename dummy indices in an expression.

##### Parameters

<i>e</i>	Expression to work on
<i>local_dummy_indices</i>	The set of dummy indices that appear in the expression "e"
<i>global_dummy_indices</i>	The set of dummy indices that have appeared before and which we would like to use in "e", too. This gets updated by the function

References [GiNaC::ex::begin\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [op\(\)](#), [shaker\\_sort\(\)](#), and [GiNaC::ex::subs\(\)](#).

#### 5.1.3.179 find\_variant\_indices()

```
static void GiNaC::find_variant_indices (
    const exvector & v,
    exvector & variant_indices ) [static]
```

Given a set of indices, extract those of class `varidx`.

**5.1.3.180 reposition\_dummy\_indices()**

```
bool GiNaC::reposition_dummy_indices (
    ex & e,
    exvector & variant_dummy_indices,
    exvector & moved_indices )
```

Raise/lower dummy indices in a single indexed objects to canonicalize their variance.

**Parameters**

<i>e</i>	Object to work on
<i>variant_dummy_indices</i>	The set of indices that might need repositioning (will be changed by this function)
<i>moved_indices</i>	The set of indices that have been repositioned (will be changed by this function)

**Returns**

true if 'e' was changed

**5.1.3.181 product\_to\_exvector()**

```
static void GiNaC::product_to_exvector (
    const ex & e,
    exvector & v,
    bool & non_commutative ) [static]
```

References [\\_ex2](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [get\\_all\\_dummy\\_indices\(\)](#).

**5.1.3.182 idx\_symmetrization()**

```
template<class T >
ex GiNaC::idx_symmetrization (
    const ex & r,
    const exvector & local_dummy_indices )
```

References [r](#), and [symmetrize\(\)](#).

**5.1.3.183 simplify\_indexed() [3/3]**

```
ex GiNaC::simplify_indexed (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp )
```

Simplify indexed expression, return list of free indices.



#### 5.1.3.184 `simplify_indexed_product()`

```
ex GiNaC::simplify_indexed_product (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp )
```

Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.

#### 5.1.3.185 `hasindex()`

```
bool GiNaC::hasindex (
    const ex & x,
    const ex & sym )
```

References [hasindex\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [x](#).

Referenced by [hasindex\(\)](#).

#### 5.1.3.186 `get_all_dummy_indices_safely()`

```
exvector GiNaC::get_all_dummy_indices_safely (
    const ex & e )
```

More reliable version of the form.

The former assumes that e is an expanded expression.

References [find\\_free\\_and\\_dummy\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [GiNaC::mul::expand\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), and [rename\\_dummy\\_indices\\_uniquely\(\)](#).

#### 5.1.3.187 `get_all_dummy_indices()`

```
exvector GiNaC::get_all_dummy_indices (
    const ex & e )
```

Returns all dummy indices from the exvector.

Returns all dummy indices from the expression.

References [product\\_to\\_exvector\(\)](#).

Referenced by [expand\\_dummy\\_sum\(\)](#), and [GiNaC::power::expand\\_mul\(\)](#).

**5.1.3.188 rename\_dummy\_indices\_uniquely()** [1/4]

```
lst GiNaC::rename_dummy_indices_uniquely (
    const exvector & va,
    const exvector & vb )
```

Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.

References [GiNaC::container\\_storage< C >::reserve\(\)](#).

Referenced by [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), and [rename\\_dummy\\_indices\\_uniquely\(\)](#).

**5.1.3.189 rename\_dummy\_indices\_uniquely()** [2/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
    const exvector & va,
    const exvector & vb,
    const ex & b )
```

Same as above, where va and vb contain the indices of a and b and are sorted.

References [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.190 rename\_dummy\_indices\_uniquely()** [3/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
    const ex & a,
    const ex & b )
```

Returns b with all dummy indices, which are common with a, renamed.

References [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.191 rename\_dummy\_indices\_uniquely()** [4/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
    exvector & va,
    const ex & b,
    bool modify_va )
```

Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b will be appended to va.

References [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.192 expand\_dummy\_sum()**

```
ex GiNaC::expand_dummy_sum (
    const ex & e,
    bool subs_idx = false )
```

This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.

Optionally all indices with a variance will be substituted by indices with the corresponding numeric values without variance.

**Parameters**

<i>e</i>	the given expression
<i>subs_idx</i>	indicates if variance of dummy indices should be neglected

References [GiNaC::ex::expand\(\)](#), [expand\\_dummy\\_sum\(\)](#), [get\\_all\\_dummy\\_indices\(\)](#), [GiNaC::ex::map\(\)](#), [GiNaC::info\\_flags::nonnegint](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [expand\\_dummy\\_sum\(\)](#).

**5.1.3.193 GINAC\_DECLARE\_UNARCHIVER() [21/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    indexed )
```

**5.1.3.194 conjugate\_evalf()**

```
static ex GiNaC::conjugate_evalf (
    const ex & arg ) [static]
```

References [GiNaC::ex::conjugate\(\)](#).

**5.1.3.195 conjugate\_eval()**

```
static ex GiNaC::conjugate_eval (
    const ex & arg ) [static]
```

References [GiNaC::ex::conjugate\(\)](#).

#### 5.1.3.196 conjugate\_print\_latex()

```
static void GiNaC::conjugate_print_latex (
    const ex & arg,
    const print\_context & c ) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

#### 5.1.3.197 conjugate\_conjugate()

```
static ex GiNaC::conjugate_conjugate (
    const ex & arg ) [static]
```

#### 5.1.3.198 conjugate\_expl\_derivative()

```
static ex GiNaC::conjugate_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References [conjugate\(\)](#), [GiNaC::ex::diff\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::symbol::info\(\)](#), and [GiNaC::info\\_flags::real](#).

#### 5.1.3.199 conjugate\_real\_part()

```
static ex GiNaC::conjugate_real_part (
    const ex & arg ) [static]
```

References [GiNaC::ex::real\\_part\(\)](#).

#### 5.1.3.200 conjugate\_imag\_part()

```
static ex GiNaC::conjugate_imag_part (
    const ex & arg ) [static]
```

References [GiNaC::ex::imag\\_part\(\)](#).

**5.1.3.201 func\_arg\_info()**

```
static bool GiNaC::func_arg_info (
    const ex & arg,
    unsigned inf ) [static]
```

References [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [GiNaC::info\\_flags::real](#).

Referenced by [conjugate\\_info\(\)](#).

**5.1.3.202 conjugate\_info()**

```
static bool GiNaC::conjugate_info (
    const ex & arg,
    unsigned inf ) [static]
```

References [func\\_arg\\_info\(\)](#).

**5.1.3.203 REGISTER\_FUNCTION() [1/36]**

```
GiNaC::REGISTER_FUNCTION (
    conjugate_function ,
    eval_func(conjugate\_eval). evalf_func(conjugate\_evalf). expl_derivative_↵
func(conjugate\_expl\_derivative). info_func(conjugate\_info). print_func< print\_latex >(conjugate\_print\_latex
conjugate_func(conjugate\_conjugate). real_part_func(conjugate\_real\_part). imag_part_func(conjugate\_imag\_part
set_name("conjugate","conjugate") )
```

**5.1.3.204 real\_part\_evalf()**

```
static ex GiNaC::real_part_evalf (
    const ex & arg ) [static]
```

**5.1.3.205 real\_part\_eval()**

```
static ex GiNaC::real_part_eval (
    const ex & arg ) [static]
```

References [GiNaC::ex::real\\_part\(\)](#).

**5.1.3.206 real\_part\_print\_latex()**

```
static void GiNaC::real_part_print_latex (
    const ex & arg,
    const print\_context & c ) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

**5.1.3.207 real\_part\_conjugate()**

```
static ex GiNaC::real_part_conjugate (
    const ex & arg ) [static]
```

**5.1.3.208 real\_part\_real\_part()**

```
static ex GiNaC::real_part_real_part (
    const ex & arg ) [static]
```

**5.1.3.209 real\_part\_imag\_part()**

```
static ex GiNaC::real_part_imag_part (
    const ex & arg ) [static]
```

**5.1.3.210 real\_part\_expl\_derivative()**

```
static ex GiNaC::real_part_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References [GiNaC::ex::diff\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::symbol::info\(\)](#), [GiNaC::info\\_flags::real](#), and [real\\_part\(\)](#).

**5.1.3.211 REGISTER\_FUNCTION() [2/36]**

```
GiNaC::REGISTER_FUNCTION (
    real_part_function ,
    eval_func(real\_part\_eval). evalf_func(real\_part\_evalf). expl_derivative_↵
func(real\_part\_expl\_derivative). print_func< print\_latex >(real\_part\_print\_latex). conjugate_↵
_func(real\_part\_conjugate). real_part_func(real\_part\_real\_part). imag_part_func(real\_part\_imag\_part).
set_name("real_part","real_part") )
```

#### 5.1.3.212 `imag_part_evalf()`

```
static ex GiNaC::imag_part_evalf (  
    const ex & arg ) [static]
```

#### 5.1.3.213 `imag_part_eval()`

```
static ex GiNaC::imag_part_eval (  
    const ex & arg ) [static]
```

References [GiNaC::ex::imag\\_part\(\)](#).

#### 5.1.3.214 `imag_part_print_latex()`

```
static void GiNaC::imag_part_print_latex (  
    const ex & arg,  
    const print\_context & c ) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

#### 5.1.3.215 `imag_part_conjugate()`

```
static ex GiNaC::imag_part_conjugate (  
    const ex & arg ) [static]
```

#### 5.1.3.216 `imag_part_real_part()`

```
static ex GiNaC::imag_part_real_part (  
    const ex & arg ) [static]
```

#### 5.1.3.217 `imag_part_imag_part()`

```
static ex GiNaC::imag_part_imag_part (  
    const ex & arg ) [static]
```

### 5.1.3.218 `imag_part_expl_derivative()`

```
static ex GiNaC::imag_part_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References [GiNaC::ex::diff\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\\_part\(\)](#), [GiNaC::symbol::info\(\)](#), and [GiNaC::info\\_flags::real](#).

### 5.1.3.219 `REGISTER_FUNCTION()` [3/36]

```
GiNaC::REGISTER_FUNCTION (
    imag_part_function ,
    eval_func(imag_part_eval). evalf_func(imag_part_evalf). expl_derivative_↔
func(imag_part_expl_derivative). print_func< print_latex >(imag_part_print_latex). conjugate_↔
_func(imag_part_conjugate). real_part_func(imag_part_real_part). imag_part_func(imag_part_imag_part).
set_name("imag_part","imag_part") )
```

### 5.1.3.220 `abs_evalf()`

```
static ex GiNaC::abs_evalf (
    const ex & arg ) [static]
```

References [abs\(\)](#), and [GiNaC::basic::hold\(\)](#).

### 5.1.3.221 `abs_eval()`

```
static ex GiNaC::abs_eval (
    const ex & arg ) [static]
```

References [abs\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), and [step\(\)](#).

### 5.1.3.222 `abs_expand()`

```
static ex GiNaC::abs_expand (
    const ex & arg,
    unsigned options ) [static]
```

References [abs\(\)](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_function\\_args](#), [GiNaC::expand\\_options::expand\\_transcendental](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::nops\(\)](#), and [options](#).



#### 5.1.3.223 `abs_expl_derivative()`

```
static ex GiNaC::abs_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References [abs\(\)](#), [GiNaC::ex::conjugate\(\)](#), and [GiNaC::ex::diff\(\)](#).

#### 5.1.3.224 `abs_print_latex()`

```
static void GiNaC::abs_print_latex (
    const ex & arg,
    const print\_context & c ) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

#### 5.1.3.225 `abs_print_csrc_float()`

```
static void GiNaC::abs_print_csrc_float (
    const ex & arg,
    const print\_context & c ) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

#### 5.1.3.226 `abs_conjugate()`

```
static ex GiNaC::abs_conjugate (
    const ex & arg ) [static]
```

References [abs\(\)](#), and [GiNaC::basic::hold\(\)](#).

#### 5.1.3.227 `abs_real_part()`

```
static ex GiNaC::abs_real_part (
    const ex & arg ) [static]
```

References [abs\(\)](#), and [GiNaC::basic::hold\(\)](#).

**5.1.3.228 abs\_imag\_part()**

```
static ex GiNaC::abs_imag_part (
    const ex & arg ) [static]
```

**5.1.3.229 abs\_power()**

```
static ex GiNaC::abs_power (
    const ex & arg,
    const ex & exp ) [static]
```

References [abs\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::info\\_flags::even](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_even\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::real](#).

**5.1.3.230 abs\_info()**

```
bool GiNaC::abs_info (
    const ex & arg,
    unsigned inf )
```

References [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), and [GiNaC::info\\_flags::real](#).

**5.1.3.231 REGISTER\_FUNCTION() [4/36]**

```
GiNaC::REGISTER_FUNCTION (
    abs ,
    eval_func(abs\_eval) . evalf_func(abs\_evalf) . expand_func(abs\_expand) . expl_↵
    derivative_func(abs\_expl\_derivative) . info_func(abs\_info) . print_func< print\_latex >(abs\_print\_latex) .
    print_func< print\_csrf\_float >(abs\_print\_csrf\_float) . print_func< print\_csrf\_double >(abs\_print\_csrf\_float)
    conjugate_func(abs\_conjugate) . real_part_func(abs\_real\_part) . imag_part_func(abs\_imag\_part) .
    power_func(abs\_power) )
```

**5.1.3.232 step\_evalf()**

```
static ex GiNaC::step_evalf (
    const ex & arg ) [static]
```

References [GiNaC::basic::hold\(\)](#), and [step\(\)](#).

#### 5.1.3.233 `step_eval()`

```
static ex GiNaC::step_eval (
    const ex & arg ) [static]
```

References [GiNaC::basic::hold\(\)](#), [l](#), [GiNaC::numeric::imag\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::numeric::real\(\)](#), and [step\(\)](#).

#### 5.1.3.234 `step_series()`

```
static ex GiNaC::step_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex0](#), [GiNaC::ex::info\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [options](#), [step\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

#### 5.1.3.235 `step_conjugate()`

```
static ex GiNaC::step_conjugate (
    const ex & arg ) [static]
```

References [GiNaC::basic::hold\(\)](#), and [step\(\)](#).

#### 5.1.3.236 `step_real_part()`

```
static ex GiNaC::step_real_part (
    const ex & arg ) [static]
```

References [GiNaC::basic::hold\(\)](#), and [step\(\)](#).

#### 5.1.3.237 `step_imag_part()`

```
static ex GiNaC::step_imag_part (
    const ex & arg ) [static]
```

**5.1.3.238 REGISTER\_FUNCTION() [5/36]**

```
GiNaC::REGISTER_FUNCTION (
    step ,
    eval_func(step_eval). evalf_func(step_evalf). series_func(step_series). conjugate←
_func(step_conjugate). real_part_func(step_real_part). imag_part_func(step_imag_part) )
```

**5.1.3.239 csgn\_evalf()**

```
static ex GiNaC::csgn_evalf (
    const ex & arg ) [static]
```

References [csgn\(\)](#).

**5.1.3.240 csgn\_eval()**

```
static ex GiNaC::csgn_eval (
    const ex & arg ) [static]
```

References [csgn\(\)](#), [I](#), [GiNaC::numeric::imag\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::numeric::real\(\)](#).

**5.1.3.241 csgn\_series()**

```
static ex GiNaC::csgn_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex0](#), [csgn\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [options](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

**5.1.3.242 csgn\_conjugate()**

```
static ex GiNaC::csgn_conjugate (
    const ex & arg ) [static]
```

References [csgn\(\)](#).

**5.1.3.243 csgn\_real\_part()**

```
static ex GiNaC::csgn_real_part (
    const ex & arg ) [static]
```

References [csgn\(\)](#).

**5.1.3.244 csgn\_imag\_part()**

```
static ex GiNaC::csgn_imag_part (
    const ex & arg ) [static]
```

**5.1.3.245 csgn\_power()**

```
static ex GiNaC::csgn_power (
    const ex & arg,
    const ex & exp ) [static]
```

References [\\_ex2](#), [csgn\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::info\(\)](#), [is\\_odd\(\)](#), and [GiNaC::info\\_flags::positive](#).

**5.1.3.246 REGISTER\_FUNCTION() [6/36]**

```
GiNaC::REGISTER_FUNCTION (
    csgn ,
    eval_func(csgn\_eval). evalf_func(csgn\_evalf). series_func(csgn\_series). conjugate↔
_func(csgn\_conjugate). real_part_func(csgn\_real\_part). imag_part_func(csgn\_imag\_part). power↔
_func(csgn\_power) )
```

**5.1.3.247 eta\_evalf()**

```
static ex GiNaC::eta_evalf (
    const ex & x,
    const ex & y ) [static]
```

References [\\_ex0](#), [csgn\(\)](#), [evalf\(\)](#), [I](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), [GiNaC::info\\_flags::positive](#), and [x](#).

**5.1.3.248 eta\_eval()**

```
static ex GiNaC::eta_eval (
    const ex & x,
    const ex & y ) [static]
```

References [\\_ex0](#), [csgn\(\)](#), [I](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), [GiNaC::info\\_flags::positive](#), and [x](#).

**5.1.3.249 eta\_series()**

```
static ex GiNaC::eta_series (
    const ex & x,
    const ex & y,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex0](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.250 eta\_conjugate()**

```
static ex GiNaC::eta_conjugate (
    const ex & x,
    const ex & y ) [static]
```

References [x](#).

**5.1.3.251 eta\_real\_part()**

```
static ex GiNaC::eta_real_part (
    const ex & x,
    const ex & y ) [static]
```

**5.1.3.252 eta\_imag\_part()**

```
static ex GiNaC::eta_imag_part (
    const ex & x,
    const ex & y ) [static]
```

References [GiNaC::basic::hold\(\)](#), [I](#), and [x](#).

**5.1.3.253 REGISTER\_FUNCTION()** [7/36]

```
GiNaC::REGISTER_FUNCTION (
    eta ,
    eval_func(eta_eval). evalf_func(eta_evalf). series_func(eta_series). latex↵
_name("\\eta"). set_symmetry(sy_symm(0, 1)). conjugate_func(eta_conjugate). real_part_↵
func(eta_real_part). imag_part_func(eta_imag_part) )
```

**5.1.3.254 Li2\_evalf()**

```
static ex GiNaC::Li2_evalf (
    const ex & x ) [static]
```

References [GiNaC::basic::hold\(\)](#), [Li2\(\)](#), and [x](#).

**5.1.3.255 Li2\_eval()**

```
static ex GiNaC::Li2_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex12](#), [\\_ex1\\_2](#), [\\_ex2](#), [\\_ex6](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_ex\\_48](#), [Catalan](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [Li2\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.256 Li2\_deriv()**

```
static ex GiNaC::Li2_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), [log\(\)](#), and [x](#).

**5.1.3.257 Li2\_series()** [1/2]

```
static ex GiNaC::Li2_series (
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_num2\\_p](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_real\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::relational::lhs\(\)](#), [Li2\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [op\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [order](#), [Pi](#), [pow\(\)](#), [GiNaC::relational::rhs\(\)](#), [GiNaC::ex::series\(\)](#), [series\(\)](#), [subs\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::series\\_options::suppress\\_branchcut](#), [x](#), and [zeta\(\)](#).

Referenced by [Li2\\_projection\(\)](#).

**5.1.3.258 Li2\_conjugate()**

```
static ex GiNaC::Li2_conjugate (
    const ex & x ) [static]
```

References [\\_num1\\_p](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [Li2\(\)](#), [GiNaC::info\\_flags::negative](#), and [x](#).

**5.1.3.259 REGISTER\_FUNCTION() [8/36]**

```
GiNaC::REGISTER_FUNCTION (
    Li2 ,
    eval_func(Li2\_eval). evalf_func(Li2\_evalf). derivative_func(Li2\_deriv). series↔
_func(Li2\_series). conjugate_func(Li2\_conjugate). latex_name("\\mathrm{Li}_2") )
```

**5.1.3.260 Li3\_eval()**

```
static ex GiNaC::Li3_eval (
    const ex & x ) [static]
```

References [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.261 REGISTER\_FUNCTION() [9/36]**

```
GiNaC::REGISTER_FUNCTION (
    Li3 ,
    eval_func(Li3\_eval). latex_name("\\mathrm{Li}_3") )
```

**5.1.3.262 zetaderiv\_eval()**

```
static ex GiNaC::zetaderiv_eval (
    const ex & n,
    const ex & x ) [static]
```

References [GiNaC::basic::hold\(\)](#), [n](#), [GiNaC::info\\_flags::numeric](#), [x](#), and [zeta\(\)](#).



#### 5.1.3.263 zetaderiv\_deriv()

```
static ex GiNaC::zetaderiv_deriv (  
    const ex & n,  
    const ex & x,  
    unsigned deriv_param ) [static]
```

References [GINAC\\_ASSERT](#), [n](#), and [x](#).

#### 5.1.3.264 REGISTER\_FUNCTION() [10/36]

```
GiNaC::REGISTER_FUNCTION (  
    zetaderiv ,  
    eval_func(zetaderiv_eval). derivative_func(zetaderiv_deriv). latex_name("\\zeta^\\prime")  
)
```

#### 5.1.3.265 factorial\_evalf()

```
static ex GiNaC::factorial_evalf (  
    const ex & x ) [static]
```

References [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

#### 5.1.3.266 factorial\_eval()

```
static ex GiNaC::factorial_eval (  
    const ex & x ) [static]
```

References [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

#### 5.1.3.267 factorial\_print\_dflt\_latex()

```
static void GiNaC::factorial_print_dflt_latex (  
    const ex & x,  
    const print_context & c ) [static]
```

References [c](#), [GiNaC::ex::print\(\)](#), and [x](#).

**5.1.3.268 factorial\_conjugate()**

```
static ex GiNaC::factorial_conjugate (
    const ex & x ) [static]
```

References [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.269 factorial\_real\_part()**

```
static ex GiNaC::factorial_real_part (
    const ex & x ) [static]
```

References [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.270 factorial\_imag\_part()**

```
static ex GiNaC::factorial_imag_part (
    const ex & x ) [static]
```

**5.1.3.271 REGISTER\_FUNCTION() [11/36]**

```
GiNaC::REGISTER_FUNCTION (
    factorial ,
    eval_func(factorial\_eval). evalf_func(factorial\_evalf). print_func< print\_dflt
>(factorial\_print\_dflt\_latex). print_func< print\_latex >(factorial\_print\_dflt\_latex). conjugate←
_func(factorial\_conjugate). real_part_func(factorial\_real\_part). imag_part_func(factorial\_imag\_part)
)
```

**5.1.3.272 binomial\_evalf()**

```
static ex GiNaC::binomial_evalf (
    const ex & x,
    const ex & y ) [static]
```

References [binomial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

#### 5.1.3.273 `binomial_sym()`

```
static ex GiNaC::binomial_sym (  
    const ex & x,  
    const numeric & y ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [binomial\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_nonneg\\_integer\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and [x](#).

Referenced by [binomial\\_eval\(\)](#).

#### 5.1.3.274 `binomial_eval()`

```
static ex GiNaC::binomial_eval (  
    const ex & x,  
    const ex & y ) [static]
```

References [binomial\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_integer\(\)](#), and [x](#).

#### 5.1.3.275 `binomial_conjugate()`

```
static ex GiNaC::binomial_conjugate (  
    const ex & x,  
    const ex & y ) [static]
```

References [binomial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

#### 5.1.3.276 `binomial_real_part()`

```
static ex GiNaC::binomial_real_part (  
    const ex & x,  
    const ex & y ) [static]
```

References [binomial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

#### 5.1.3.277 `binomial_imag_part()`

```
static ex GiNaC::binomial_imag_part (  
    const ex & x,  
    const ex & y ) [static]
```

**5.1.3.278 REGISTER\_FUNCTION() [12/36]**

```
GiNaC::REGISTER_FUNCTION (
    binomial ,
    eval_func(binomial_eval). evalf_func(binomial_evalf). conjugate_func(binomial_conjugate).
    real_part_func(binomial_real_part). imag_part_func(binomial_imag_part) )
```

**5.1.3.279 Order\_eval()**

```
static ex GiNaC::Order_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::is\\_zero\(\)](#), [m](#), and [x](#).

**5.1.3.280 Order\_series()**

```
static ex GiNaC::Order_series (
    const ex & x,
    const relational & r,
    int order,
    unsigned options ) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), [GiNaC::ex::ldegree\(\)](#), [order](#), [r](#), and [x](#).

**5.1.3.281 Order\_conjugate()**

```
static ex GiNaC::Order_conjugate (
    const ex & x ) [static]
```

References [x](#).

**5.1.3.282 Order\_real\_part()**

```
static ex GiNaC::Order_real_part (
    const ex & x ) [static]
```

References [x](#).

**5.1.3.283 Order\_imag\_part()**

```
static ex GiNaC::Order_imag_part (
    const ex & x ) [static]
```

References [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.284 Order\_expl\_derivative()**

```
static ex GiNaC::Order_expl_derivative (
    const ex & arg,
    const symbol & s ) [static]
```

References [GiNaC::ex::diff\(\)](#).

**5.1.3.285 REGISTER\_FUNCTION() [13/36]**

```
GiNaC::REGISTER_FUNCTION (
    Order ,
    eval_func(Order\_eval). series_func(Order\_series). latex_name("\\mathcal{O}").
    expl_derivative_func(Order\_expl\_derivative). conjugate_func(Order\_conjugate). real_part_↔
    func(Order\_real\_part). imag_part_func(Order\_imag\_part) )
```

**5.1.3.286 Isolve()**

```
ex GiNaC::lsolve (
    const ex & eqns,
    const ex & symbols,
    unsigned options = solve\_algo::automatic )
```

Factorial function.

Binomial function. Order term function (for truncated power series).

References [GiNaC::container< C >::append\(\)](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::matrix::cols\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::info\\_flags::exprseq](#), [GINAC\\_ASSERT](#), [GiNaC::symbolset::has\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::list](#), [Isolve\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [r](#), [GiNaC::info\\_flags::relation\\_equal](#), [rhs\(\)](#), [GiNaC::matrix::rows\(\)](#), [GiNaC::matrix::solve\(\)](#), [GiNaC::info\\_flags::symbol](#), and [syms](#).

Referenced by [Isolve\(\)](#).

**5.1.3.287 fsolve()**

```
const numeric GiNaC::fsolve (
    const ex & f,
    const symbol & x,
    const numeric & x1,
    const numeric & x2 )
```

Find a real root of real-valued function f(x) numerically within a given interval.

The function must change sign across interval. Uses Newton- Raphson method combined with bisection in order to guarantee convergence.

## Parameters

<i>f</i>	Function $f(x)$
<i>x</i>	Symbol $f(x)$
<i>x1</i>	lower interval limit
<i>x2</i>	upper interval limit

## Exceptions

<i>runtime_error</i>	(if interval is invalid).
----------------------	---------------------------

References [GiNaC::ex::diff\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::lhs\(\)](#), [normal\(\)](#), [GiNaC::ex::rhs\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.288 zeta()** [1/3]

```
template<typename T1 >
function GiNaC::zeta (
    const T1 & p1 ) [inline]
```

References [GiNaC::zeta1\\_SERIAL::serial](#).

Referenced by [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [psi2\\_eval\(\)](#), [S\\_eval\(\)](#), [zeta1\\_eval\(\)](#), [zeta1\\_evalf\(\)](#), [zeta2\\_eval\(\)](#), [zeta2\\_evalf\(\)](#), and [zetaderiv\\_eval\(\)](#).

**5.1.3.289 zeta()** [2/3]

```
template<typename T1 , typename T2 >
function GiNaC::zeta (
    const T1 & p1,
    const T2 & p2 ) [inline]
```

References [GiNaC::zeta2\\_SERIAL::serial](#).

**5.1.3.290 is\_the\_function< zeta\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< zeta_SERIAL > (
    const ex & x ) [inline]
```

References [x](#).

**5.1.3.291 G()** [1/2]

```
template<typename T1 , typename T2 >
function GiNaC::G (
    const T1 & x,
    const T2 & y ) [inline]
```

References [GiNaC::G2\\_SERIAL::serial](#), and [x](#).

Referenced by [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), and [G3\\_evalf\(\)](#).

**5.1.3.292 G()** [2/2]

```
template<typename T1 , typename T2 , typename T3 >
function GiNaC::G (
    const T1 & x,
    const T2 & s,
    const T3 & y ) [inline]
```

References [GiNaC::G3\\_SERIAL::serial](#), and [x](#).

**5.1.3.293 is\_the\_function< G\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< G_SERIAL > (
    const ex & x ) [inline]
```

References [x](#).

**5.1.3.294 psi()** [1/4]

```
template<typename T1 >
function GiNaC::psi (
    const T1 & p1 ) [inline]
```

References [GiNaC::psi1\\_SERIAL::serial](#).

Referenced by [beta\\_deriv\(\)](#), [lgamma\\_deriv\(\)](#), [psi1\\_deriv\(\)](#), [psi1\\_eval\(\)](#), [psi1\\_evalf\(\)](#), [psi1\\_series\(\)](#), [psi2\\_deriv\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_evalf\(\)](#), [psi2\\_series\(\)](#), [sr\\_gcd\(\)](#), and [tgamma\\_deriv\(\)](#).

**5.1.3.295 psi()** [2/4]

```
template<typename T1 , typename T2 >
function GiNaC::psi (
    const T1 & p1,
    const T2 & p2 ) [inline]
```

References [GiNaC::psi2\\_SERIAL::serial](#).

**5.1.3.296 is\_the\_function< psi\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< psi_SERIAL > (
    const ex & x ) [inline]
```

References [x](#).

**5.1.3.297 iterated\_integral()** [1/2]

```
template<typename T1 , typename T2 >
function GiNaC::iterated_integral (
    const T1 & kernel_lst,
    const T2 & lambda ) [inline]
```

References [GiNaC::iterated\\_integral2\\_SERIAL::serial](#).

Referenced by [iterated\\_integral2\\_eval\(\)](#), [iterated\\_integral3\\_eval\(\)](#), and [iterated\\_integral\\_evalf\\_impl\(\)](#).

**5.1.3.298 iterated\_integral()** [2/2]

```
template<typename T1 , typename T2 , typename T3 >
function GiNaC::iterated_integral (
    const T1 & kernel_lst,
    const T2 & lambda,
    const T3 & N_trunc ) [inline]
```

References [GiNaC::iterated\\_integral3\\_SERIAL::serial](#).

**5.1.3.299 is\_the\_function< iterated\_integral\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< iterated_integral_SERIAL > (
    const ex & x ) [inline]
```

References [x](#).



**5.1.3.300 is\_order\_function()**

```
bool GiNaC::is_order_function (
    const ex & e ) [inline]
```

Check whether a function is the Order ( $O(n)$ ) function.

References [is\\_ex\\_the\\_function](#).

Referenced by [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::pseries::is\\_terminating\(\)](#), [GiNaC::pseries::mul\\_const\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::pseries::op\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::pseries::pseries\(\)](#), and [GiNaC::integral::series\(\)](#).

**5.1.3.301 convert\_H\_to\_Li()**

```
ex GiNaC::convert_H_to_Li (
    const ex & parameterlst,
    const ex & arg )
```

Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding [GiNaC](#) functions.

References [m](#), and [x](#).

**5.1.3.302 EllipticK\_evalf()**

```
static ex GiNaC::EllipticK_evalf (
    const ex & k ) [static]
```

References [GiNaC::ex::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [sqrt\(\)](#).

**5.1.3.303 EllipticK\_eval()**

```
static ex GiNaC::EllipticK_eval (
    const ex & k ) [static]
```

References [\\_ex0](#), [GiNaC::info\\_flags::crational](#), [GiNaC::constant::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

**5.1.3.304 EllipticK\_deriv()**

```
static ex GiNaC::EllipticK_deriv (
    const ex & k,
    unsigned deriv_param ) [static]
```

References [k](#).

**5.1.3.305 EllipticK\_series()**

```
static ex GiNaC::EllipticK_series (
    const ex & k,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [k](#), [GiNaC::subs\\_options::no\\_pattern](#), [order](#), [Pi](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.306 EllipticK\_print\_latex()**

```
static void GiNaC::EllipticK_print_latex (
    const ex & k,
    const print_context & c ) [static]
```

References [c](#), and [k](#).

**5.1.3.307 REGISTER\_FUNCTION() [14/36]**

```
GiNaC::REGISTER_FUNCTION (
    EllipticK ,
    evalf_func(EllipticK\_evalf). eval_func(EllipticK\_eval). derivative_func(EllipticK\_deriv).
    series_func(EllipticK\_series). print_func< print_latex >(EllipticK\_print\_latex). do_not_↵
    evalf_params() )
```

**5.1.3.308 EllipticE\_evalf()**

```
static ex GiNaC::EllipticE_evalf (
    const ex & k ) [static]
```

References [GiNaC::ex::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [sqrt\(\)](#).

**5.1.3.309 EllipticE\_eval()**

```
static ex GiNaC::EllipticE_eval (
    const ex & k ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::info\\_flags::crational](#), [GiNaC::constant::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

**5.1.3.310 EllipticE\_deriv()**

```
static ex GiNaC::EllipticE_deriv (
    const ex & k,
    unsigned deriv_param ) [static]
```

References [k](#).

**5.1.3.311 EllipticE\_series()**

```
static ex GiNaC::EllipticE_series (
    const ex & k,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [k](#), [GiNaC::subs\\_options::no\\_pattern](#), [order](#), [Pi](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.312 EllipticE\_print\_latex()**

```
static void GiNaC::EllipticE_print_latex (
    const ex & k,
    const print\_context & c ) [static]
```

References [c](#), and [k](#).

**5.1.3.313 REGISTER\_FUNCTION() [15/36]**

```
GiNaC::REGISTER_FUNCTION (
    EllipticE ,
    evalf_func(EllipticE\_evalf). eval_func(EllipticE\_eval). derivative_func(EllipticE\_deriv).
    series_func(EllipticE\_series). print_func< print\_latex >(EllipticE\_print\_latex). do_not_↔
    evalf_params() )
```

**5.1.3.314 iterated\_integral\_evalf\_impl()**

```
static ex GiNaC::iterated_integral_evalf_impl (
    const ex & kernel_lst,
    const ex & lambda,
    const ex & N_trunc ) [static]
```

References [coeff\(\)](#), [Digits](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [iterated\\_integral\(\)](#), [GiNaC::info\\_flags::list](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), and [one](#).

Referenced by [iterated\\_integral2\\_evalf\(\)](#), and [iterated\\_integral3\\_evalf\(\)](#).

**5.1.3.315 iterated\_integral2\_evalf()**

```
static ex GiNaC::iterated_integral2_evalf (
    const ex & kernel_lst,
    const ex & lambda ) [static]
```

References [iterated\\_integral\\_evalf\\_impl\(\)](#).

**5.1.3.316 iterated\_integral3\_evalf()**

```
static ex GiNaC::iterated_integral3_evalf (
    const ex & kernel_lst,
    const ex & lambda,
    const ex & N_trunc ) [static]
```

References [iterated\\_integral\\_evalf\\_impl\(\)](#).

**5.1.3.317 iterated\_integral2\_eval()**

```
static ex GiNaC::iterated_integral2_eval (
    const ex & kernel_lst,
    const ex & lambda ) [static]
```

References [GiNaC::info\\_flags::crational](#), [GiNaC::function::evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [iterated\\_integral\(\)](#), and [GiNaC::info\\_flags::numeric](#).

**5.1.3.318 iterated\_integral3\_eval()**

```
static ex GiNaC::iterated_integral3_eval (
    const ex & kernel_lst,
    const ex & lambda,
    const ex & N_trunc ) [static]
```

References [GiNaC::info\\_flags::crational](#), [GiNaC::function::evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [iterated\\_integral\(\)](#), and [GiNaC::info\\_flags::numeric](#).

**5.1.3.319 lgamma\_evalf()**

```
static ex GiNaC::lgamma_evalf (
    const ex & x ) [static]
```

References [lgamma\(\)](#), and [x](#).

**5.1.3.320 lgamma\_eval()**

```
static ex GiNaC::lgamma_eval (
    const ex & x ) [static]
```

Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.

Handles integer arguments as a special case.

## Exceptions

<a href="#"><i>GiNaC::pole_error("lgamma_eval()")</i></a>	logarithmic pole",0)
---	----------------------

References [\\_ex\\_1](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [is\\_rational\(\)](#), [lgamma\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), and [x](#).

**5.1.3.321 lgamma\_deriv()**

```
static ex GiNaC::lgamma_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [GINAC\\_ASSERT](#), [psi\(\)](#), and [x](#).

**5.1.3.322 lgamma\_series()**

```
static ex GiNaC::lgamma_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [lgamma\(\)](#), [log\(\)](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [GiNaC::basic::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.323 lgamma\_conjugate()**

```
static ex GiNaC::lgamma_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lgamma\(\)](#), [GiNaC::info\\_flags::positive](#), and [x](#).

**5.1.3.324 REGISTER\_FUNCTION() [16/36]**

```
GiNaC::REGISTER_FUNCTION (
    lgamma ,
    eval_func(lgamma\_eval) . evalf_func(lgamma\_evalf) . derivative_func(lgamma\_deriv) .
    series_func(lgamma\_series) . conjugate_func(lgamma\_conjugate) . latex_name("\\log \\Gamma") )
```

### 5.1.3.325 `tgamma_evalf()`

```
static ex GiNaC::tgamma_evalf (
    const ex & x ) [static]
```

References [tgamma\(\)](#), and [x](#).

### 5.1.3.326 `tgamma_eval()`

```
static ex GiNaC::tgamma_eval (
    const ex & x ) [static]
```

Evaluation of `tgamma(x)`, the true Gamma function.

Knows about integer arguments, half-integer arguments and that's it. Somebody ought to provide some good numerical evaluation some day...

#### Exceptions

<code>pole_error("tgamma_eval() simple pole",0)</code>
--

References [\\_num1\\_2\\_p](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [\\_num\\_2\\_p](#), [abs\(\)](#), [GiNaC::numeric::div\(\)](#), [doublefactorial\(\)](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_even\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [is\\_rational\(\)](#), [n](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), [pow\(\)](#), [sqrt\(\)](#), [tgamma\(\)](#), and [x](#).

### 5.1.3.327 `tgamma_deriv()`

```
static ex GiNaC::tgamma_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [GINAC\\_ASSERT](#), [psi\(\)](#), [tgamma\(\)](#), and [x](#).

### 5.1.3.328 `tgamma_series()`

```
static ex GiNaC::tgamma_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [GiNaC::ex::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [tgamma\(\)](#).

**5.1.3.329 `tgamma_conjugate()`**

```
static ex GiNaC::tgamma_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [tgamma\(\)](#), and [x](#).

**5.1.3.330 `REGISTER_FUNCTION()` [17/36]**

```
GiNaC::REGISTER_FUNCTION (
    tgamma ,
    eval_func(tgamma\_eval) . evalf_func(tgamma\_evalf) . derivative_func(tgamma\_deriv) .
    series_func(tgamma\_series) . conjugate_func(tgamma\_conjugate) . latex_name("\\Gamma") )
```

**5.1.3.331 `beta_evalf()`**

```
static ex GiNaC::beta_evalf (
    const ex & x,
    const ex & y ) [static]
```

References [exp\(\)](#), [lgamma\(\)](#), and [x](#).

**5.1.3.332 `beta_eval()`**

```
static ex GiNaC::beta_eval (
    const ex & x,
    const ex & y ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_num\\_1\\_p](#), [evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [is\\_positive\(\)](#), [is\\_rational\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [is\\_real\(\)](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), [tgamma\(\)](#), and [x](#).

**5.1.3.333 `beta_deriv()`**

```
static ex GiNaC::beta_deriv (
    const ex & x,
    const ex & y,
    unsigned deriv_param ) [static]
```

References [GINAC\\_ASSERT](#), [psi\(\)](#), and [x](#).

#### 5.1.3.334 `beta_series()`

```
static ex GiNaC::beta_series (
    const ex & arg1,
    const ex & arg2,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [GiNaC::ex::subs\(\)](#), and [tgamma\(\)](#).

#### 5.1.3.335 `REGISTER_FUNCTION()` [18/36]

```
GiNaC::REGISTER_FUNCTION (
    beta ,
    eval_func(beta_eval). evalf_func(beta_evalf). derivative_func(beta_deriv).
series_func(beta_series). latex_name("\\mathrm{B}"). set_symmetry(sy_symm(0, 1)) )
```

#### 5.1.3.336 `psi1_evalf()`

```
static ex GiNaC::psi1_evalf (
    const ex & x ) [static]
```

References [GiNaC::basic::hold\(\)](#), [psi\(\)](#), and [x](#).

#### 5.1.3.337 `psi1_eval()`

```
static ex GiNaC::psi1_eval (
    const ex & x ) [static]
```

Evaluation of digamma-function  $\psi(x)$ .

Somebody ought to provide some good numerical evaluation some day...

References [\\_ex1\\_2](#), [\\_ex2](#), [\\_num2\\_p](#), [\\_num\\_1\\_p](#), [Euler](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [is\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), [psi\(\)](#), and [x](#).



**5.1.3.338 psi1\_deriv()**

```
static ex GiNaC::psi1_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), [psi\(\)](#), and [x](#).

**5.1.3.339 psi1\_series()**

```
static ex GiNaC::psi1_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [psi\(\)](#), [GiNaC::power::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.340 psi2\_evalf()**

```
static ex GiNaC::psi2_evalf (
    const ex & n,
    const ex & x ) [static]
```

References [GiNaC::basic::hold\(\)](#), [n](#), [psi\(\)](#), and [x](#).

**5.1.3.341 psi2\_eval()**

```
static ex GiNaC::psi2_eval (
    const ex & n,
    const ex & x ) [static]
```

Evaluation of polygamma-function  $\psi(n,x)$ .

Somebody ought to provide some good numerical evaluation some day...

References [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_num1\\_2\\_p](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [\\_num\\_1\\_p](#), [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [is\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [log\(\)](#), [m](#), [n](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [psi\(\)](#), [tgamma\(\)](#), [x](#), and [zeta\(\)](#).

#### 5.1.3.342 `psi2_deriv()`

```
static ex GiNaC::psi2_deriv (
    const ex & n,
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), [n](#), [psi\(\)](#), and [x](#).

#### 5.1.3.343 `psi2_series()`

```
static ex GiNaC::psi2_series (
    const ex & n,
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [m](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [psi\(\)](#), and [GiNaC::ex::subs\(\)](#).

#### 5.1.3.344 `G2_evalf()`

```
static ex GiNaC::G2_evalf (
    const ex & x_,
    const ex & y ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_real\(\)](#), [log\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), and [x](#).

#### 5.1.3.345 `G2_eval()`

```
static ex GiNaC::G2_eval (
    const ex & x_,
    const ex & y ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::info\\_flags::crational](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_real\(\)](#), [log\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), and [x](#).

**5.1.3.346 G3\_evalf()**

```
static ex GiNaC::G3_evalf (
    const ex & x_,
    const ex & s_,
    const ex & y ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.347 G3\_eval()**

```
static ex GiNaC::G3_eval (
    const ex & x_,
    const ex & s_,
    const ex & y ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::ex::begin\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::ex::end\(\)](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.348 Li\_evalf()**

```
static ex GiNaC::Li_evalf (
    const ex & m_,
    const ex & x_ ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::posint](#), and [x](#).

**5.1.3.349 Li\_eval()**

```
static ex GiNaC::Li_eval (
    const ex & m_,
    const ex & x_ ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [\\_ex\\_48](#), [GiNaC::container< C >::append\(\)](#), [GiNaC::ex::begin\(\)](#), [Catalan](#), [GiNaC::info\\_flags::crational](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [log\(\)](#), [m](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), [x](#), and [zeta\(\)](#).

**5.1.3.350 Li\_series()**

```
static ex GiNaC::Li_series (
    const ex & m,
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [order](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.351 Li\_deriv()**

```
static ex GiNaC::Li_deriv (
    const ex & m_,
    const ex & x_,
    unsigned deriv_param ) [static]
```

References [\\_ex0](#), [GINAC\\_ASSERT](#), [m](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [x](#).

**5.1.3.352 Li\_print\_latex()**

```
static void GiNaC::Li_print_latex (
    const ex & m_,
    const ex & x_,
    const print\_context & c ) [static]
```

References [GiNaC::ex::begin\(\)](#), [c](#), [GiNaC::ex::end\(\)](#), [m](#), and [x](#).

**5.1.3.353 REGISTER\_FUNCTION() [19/36]**

```
GiNaC::REGISTER_FUNCTION (
    Li ,
    evalf_func(Li_evalf). eval_func(Li_eval). series_func(Li_series). derivative_↔
func(Li_deriv). print_func< print\_latex >(Li_print_latex). do_not_evalf_params() )
```

**5.1.3.354 S\_evalf()**

```
static ex GiNaC::S_evalf (
    const ex & n,
    const ex & p,
    const ex & x ) [static]
```

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [n](#), [GiNaC::info\\_flags::posint](#), and [x](#).

**5.1.3.355 S\_eval()**

```
static ex GiNaC::S_eval (
    const ex & n,
    const ex & p,
    const ex & x ) [static]
```

References [\\_ex0](#), [GiNaC::info\\_flags::crational](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [m](#), [n](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [to\\_int\(\)](#), [x](#), and [zeta\(\)](#).

**5.1.3.356 S\_series()**

```
static ex GiNaC::S_series (
    const ex & n,
    const ex & p,
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [options](#), [order](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.357 S\_deriv()**

```
static ex GiNaC::S_deriv (
    const ex & n,
    const ex & p,
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex0](#), [GINAC\\_ASSERT](#), [n](#), and [x](#).

**5.1.3.358 S\_print\_latex()**

```
static void GiNaC::S_print_latex (
    const ex & n,
    const ex & p,
    const ex & x,
    const print\_context & c ) [static]
```

References [c](#), [n](#), [GiNaC::ex::print\(\)](#), and [x](#).

**5.1.3.359 REGISTER\_FUNCTION() [20/36]**

```
GiNaC::REGISTER_FUNCTION (
    S ,
    evalf_func(S_evalf). eval_func(S_eval). series_func(S_series). derivative_↔
func(S_deriv). print_func< print_latex >(S_print_latex). do_not_evalf_params() )
```

**5.1.3.360 H\_evalf()**

```
static ex GiNaC::H_evalf (
    const ex & x1,
    const ex & x2 ) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::ex::evalf\(\)](#), [l](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [m](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [Pi](#), [GiNaC::ex::subs\(\)](#), [to\\_int\(\)](#), and [x](#).

**5.1.3.361 H\_eval()**

```
static ex GiNaC::H_eval (
    const ex & m_,
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::info\\_flags::crational](#), [GiNaC::ex::evalf\(\)](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [log\(\)](#), [m](#), [n](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), [step\(\)](#), and [x](#).

**5.1.3.362 H\_series()**

```
static ex GiNaC::H_series (
    const ex & m,
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [m](#), and [x](#).

**5.1.3.363 H\_deriv()**

```
static ex GiNaC::H_deriv (
    const ex & m_,
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GINAC\\_ASSERT](#), [m](#), and [x](#).

**5.1.3.364 H\_print\_latex()**

```
static void GiNaC::H_print_latex (
    const ex & m_,
    const ex & x,
    const print\_context & c ) [static]
```

References [c](#), [m](#), [GiNaC::ex::print\(\)](#), and [x](#).

**5.1.3.365 REGISTER\_FUNCTION() [21/36]**

```
GiNaC::REGISTER_FUNCTION (
    H ,
    evalf_func(H\_evalf) . eval_func(H\_eval) . series_func(H\_series) . derivative_↔
func(H\_deriv) . print_func< print\_latex >(H\_print\_latex) . do_not_evalf_params() )
```

**5.1.3.366 zeta1\_evalf()**

```
static ex GiNaC::zeta1_evalf (
    const ex & x ) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [Digits](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::posint](#), [r](#), [x](#), and [zeta\(\)](#).

Referenced by [zeta1\\_eval\(\)](#).

**5.1.3.367 zeta1\_eval()**

```
static ex GiNaC::zeta1_eval (
    const ex & m ) [static]
```

References [\\_ex0](#), [\\_ex\\_1\\_2](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [abs\(\)](#), [bernoulli\(\)](#), [GiNaC::info\\_flags::crational](#), [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::info\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [m](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::odd](#), [Pi](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [zeta\(\)](#), and [zeta1\\_evalf\(\)](#).

**5.1.3.368 zeta1\_deriv()**

```
static ex GiNaC::zeta1_deriv (
    const ex & m,
    unsigned deriv\_param ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GINAC\\_ASSERT](#), and [m](#).

**5.1.3.369 zeta1\_print\_latex()**

```
static void GiNaC::zeta1_print_latex (
    const ex & m_,
    const print\_context & c ) [static]
```

References [c](#), [m](#), and [GiNaC::ex::print\(\)](#).

**5.1.3.370 zeta2\_evalf()**

```
static ex GiNaC::zeta2_evalf (
    const ex & x,
    const ex & s ) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::posint](#), [x](#), and [zeta\(\)](#).

**5.1.3.371 zeta2\_eval()**

```
static ex GiNaC::zeta2_eval (
    const ex & m,
    const ex & s_ ) [static]
```

References [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [GiNaC::info\\_flags::positive](#), and [zeta\(\)](#).

**5.1.3.372 zeta2\_deriv()**

```
static ex GiNaC::zeta2_deriv (
    const ex & m,
    const ex & s,
    unsigned deriv_param ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [m](#), [GiNaC::ex::op\(\)](#), and [GiNaC::info\\_flags::positive](#).

**5.1.3.373 zeta2\_print\_latex()**

```
static void GiNaC::zeta2_print_latex (
    const ex & m_,
    const ex & s_,
    const print\_context & c ) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [c](#), and [m](#).



**5.1.3.374 exp\_evalf()**

```
static ex GiNaC::exp_evalf (
    const ex & x ) [static]
```

References [exp\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.375 exp\_eval()**

```
static ex GiNaC::exp_eval (
    const ex & x ) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [\\_num0\\_p](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [\\_num3\\_p](#), [\\_num4\\_p](#), [GiNaC::info\\_flags::crational](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [mod\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), and [x](#).

**5.1.3.376 exp\_expand()**

```
static ex GiNaC::exp_expand (
    const ex & arg,
    unsigned options ) [static]
```

References [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [exp\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_function\\_args](#), [GiNaC::expand\\_options::expand\\_transcendental](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::nops\(\)](#), and [options](#).

**5.1.3.377 exp\_deriv()**

```
static ex GiNaC::exp_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [exp\(\)](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.378 exp\_real\_part()**

```
static ex GiNaC::exp_real_part (
    const ex & x ) [static]
```

References [cos\(\)](#), [exp\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [x](#).

**5.1.3.379 exp\_imag\_part()**

```
static ex GiNaC::exp_imag_part (
    const ex & x ) [static]
```

References [exp\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), and [x](#).

**5.1.3.380 exp\_conjugate()**

```
static ex GiNaC::exp_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [exp\(\)](#), and [x](#).

**5.1.3.381 exp\_power()**

```
static ex GiNaC::exp_power (
    const ex & x,
    const ex & a ) [static]
```

References [\\_ex\\_1](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.382 exp\_info()**

```
static bool GiNaC::exp_info (
    const ex & x,
    unsigned inf ) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.383 REGISTER\_FUNCTION() [22/36]**

```
GiNaC::REGISTER_FUNCTION (
    exp ,
    eval_func(exp\_eval). evalf_func(exp\_evalf). info_func(exp\_info). expand_↔
func(exp\_expand). derivative_func(exp\_deriv). real_part_func(exp\_real\_part). imag_part_↔
func(exp\_imag\_part). conjugate_func(exp\_conjugate). power_func(exp\_power). latex_name("\\exp")
)
```

**5.1.3.384 log\_evalf()**

```
static ex GiNaC::log_evalf (
    const ex & x ) [static]
```

References [GiNaC::basic::hold\(\)](#), [log\(\)](#), and [x](#).

**5.1.3.385 log\_eval()**

```
static ex GiNaC::log_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1\\_2](#), [GiNaC::info\\_flags::crational](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.386 log\_deriv()**

```
static ex GiNaC::log_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex\\_1](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.387 log\_series()**

```
static ex GiNaC::log_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::pseries::coeff\(\)](#), [coeff\(\)](#), [csgn\(\)](#), [GiNaC::ex::diff\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::pseries::is\\_terminating\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::pseries::ldegree\(\)](#), [GiNaC::relational::lhs\(\)](#), [log\(\)](#), [n](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::pseries::nops\(\)](#), [options](#), [order](#), [Pi](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::relational::rhs\(\)](#), [GiNaC::ex::series\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

**5.1.3.388 log\_real\_part()**

```
static ex GiNaC::log_real_part (
    const ex & x ) [static]
```

References [abs\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::nonnegative](#), and [x](#).

**5.1.3.389 log\_imag\_part()**

```
static ex GiNaC::log_imag_part (
    const ex & x ) [static]
```

References [imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::nonnegative](#), [real\\_part\(\)](#), and [x](#).

**5.1.3.390 log\_expand()**

```
static ex GiNaC::log_expand (
    const ex & arg,
    unsigned options ) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_func](#), [GiNaC::expand\\_options::expand\\_transcendental](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::ex::nops\(\)](#), [options](#), [GiNaC::info\\_flags::positive](#), [GiNaC::status\\_flags::purely\\_indefinite](#), and [GiNaC::basic::setflag\(\)](#).

**5.1.3.391 log\_conjugate()**

```
static ex GiNaC::log_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::positive](#), and [x](#).

**5.1.3.392 log\_info()**

```
static bool GiNaC::log_info (
    const ex & x,
    unsigned inf ) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.393 REGISTER\_FUNCTION() [23/36]**

```
GiNaC::REGISTER_FUNCTION (
    log ,
    eval_func(log\_eval). evalf_func(log\_evalf). info_func(log\_info). expand_↵
func(log\_expand). derivative_func(log\_deriv). series_func(log\_series). real_part_func(log\_real\_part).
imag_part_func(log\_imag\_part). conjugate_func(log\_conjugate). latex_name("\\ln" ) )
```

#### 5.1.3.394 `sin_evalf()`

```
static ex GiNaC::sin_evalf (
    const ex & x ) [static]
```

References [GiNaC::basic::hold\(\)](#), [sin\(\)](#), and [x](#).

#### 5.1.3.395 `sin_eval()`

```
static ex GiNaC::sin_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex1\\_3](#), [\\_ex1\\_4](#), [\\_ex2](#), [\\_ex3](#), [\\_ex5](#), [\\_ex6](#), [\\_ex60](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_ex\\_1\\_3](#), [\\_ex\\_1\\_4](#), [\\_num0\\_p](#), [\\_num10\\_p](#), [\\_num120\\_p](#), [\\_num15\\_p](#), [\\_num18\\_p](#), [\\_num20\\_p](#), [\\_num25\\_p](#), [\\_num30\\_p](#), [\\_num5\\_p](#), [\\_num60\\_p](#), [\\_num6\\_p](#), [acos\(\)](#), [asin\(\)](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [mod\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [sin\(\)](#), [sqrt\(\)](#), and [x](#).

#### 5.1.3.396 `sin_deriv()`

```
static ex GiNaC::sin_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [cos\(\)](#), [GINAC\\_ASSERT](#), and [x](#).

#### 5.1.3.397 `sin_real_part()`

```
static ex GiNaC::sin_real_part (
    const ex & x ) [static]
```

References [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), and [x](#).

#### 5.1.3.398 `sin_imag_part()`

```
static ex GiNaC::sin_imag_part (
    const ex & x ) [static]
```

References [cos\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.399 sin\_conjugate()**

```
static ex GiNaC::sin_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [sin\(\)](#), and [x](#).

**5.1.3.400 trig\_info()**

```
static bool GiNaC::trig_info (
    const ex & x,
    unsigned inf ) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.401 REGISTER\_FUNCTION() [24/36]**

```
GiNaC::REGISTER_FUNCTION (
    sin ,
    eval_func(sin\_eval). evalf_func(sin\_evalf). info_func(trig\_info). derivative_↵
func(sin\_deriv). real_part_func(sin\_real\_part). imag_part_func(sin\_imag\_part). conjugate_↵
func(sin\_conjugate). latex_name("\\sin" ) )
```

**5.1.3.402 cos\_evalf()**

```
static ex GiNaC::cos_evalf (
    const ex & x ) [static]
```

References [cos\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.403 cos\_eval()**

```
static ex GiNaC::cos_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex1\\_3](#), [\\_ex1\\_4](#), [\\_ex2](#), [\\_ex3](#), [\\_ex5](#), [\\_ex6](#), [\\_ex60](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_ex\\_1\\_3](#), [\\_ex\\_1\\_4](#), [\\_num0\\_p](#), [\\_num10\\_p](#), [\\_num120\\_p](#), [\\_num12\\_p](#), [\\_num15\\_p](#), [\\_num20\\_p](#), [\\_num24\\_p](#), [\\_num25\\_p](#), [\\_num30\\_p](#), [\\_num5\\_p](#), [\\_num60\\_p](#), [acos\(\)](#), [asin\(\)](#), [atan\(\)](#), [cos\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [mod\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [sqrt\(\)](#), and [x](#).

**5.1.3.404 cos\_deriv()**

```
static ex GiNaC::cos_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [GINAC\\_ASSERT](#), [sin\(\)](#), and [x](#).

**5.1.3.405 cos\_real\_part()**

```
static ex GiNaC::cos_real_part (
    const ex & x ) [static]
```

References [cos\(\)](#), [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [x](#).

**5.1.3.406 cos\_imag\_part()**

```
static ex GiNaC::cos_imag_part (
    const ex & x ) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.407 cos\_conjugate()**

```
static ex GiNaC::cos_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [cos\(\)](#), and [x](#).

**5.1.3.408 REGISTER\_FUNCTION()** [25/36]

```
GiNaC::REGISTER_FUNCTION (
    cos ,
    eval_func(cos\_eval). info_func(trig\_info). evalf_func(cos\_evalf). derivative_↵
func(cos\_deriv). real_part_func(cos\_real\_part). imag_part_func(cos\_imag\_part). conjugate_↵
func(cos\_conjugate). latex_name("\\cos" ) )
```

**5.1.3.409 tan\_evalf()**

```
static ex GiNaC::tan_evalf (
    const ex & x ) [static]
```

References [GiNaC::basic::hold\(\)](#), [tan\(\)](#), and [x](#).

**5.1.3.410 tan\_eval()**

```
static ex GiNaC::tan_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_3](#), [\\_ex2](#), [\\_ex3](#), [\\_ex60](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_num0\\_p](#), [\\_num10\\_p](#), [\\_num15\\_p](#), [\\_num20\\_p](#), [\\_num25\\_p](#), [\\_num30\\_p](#), [\\_num5\\_p](#), [\\_num60\\_p](#), [acos\(\)](#), [asin\(\)](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [mod\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [sqrt\(\)](#), [tan\(\)](#), and [x](#).

**5.1.3.411 tan\_deriv()**

```
static ex GiNaC::tan_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex1](#), [\\_ex2](#), [GINAC\\_ASSERT](#), [tan\(\)](#), and [x](#).

**5.1.3.412 tan\_real\_part()**

```
static ex GiNaC::tan_real_part (
    const ex & x ) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), and [x](#).

**5.1.3.413 tan\_imag\_part()**

```
static ex GiNaC::tan_imag_part (
    const ex & x ) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), [tanh\(\)](#), and [x](#).



**5.1.3.414 tan\_series()**

```
static ex GiNaC::tan_series (
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [cos\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::odd](#), [options](#), [order](#), [Pi](#), [sin\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.415 tan\_conjugate()**

```
static ex GiNaC::tan_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [tan\(\)](#), and [x](#).

**5.1.3.416 REGISTER\_FUNCTION() [26/36]**

```
GiNaC::REGISTER_FUNCTION (
    tan ,
    eval_func(tan_eval). evalf_func(tan_evalf). info_func(trig_info). derivative↵
    _func(tan_deriv). series_func(tan_series). real_part_func(tan_real_part). imag_part_↵
    func(tan_imag_part). conjugate_func(tan_conjugate). latex_name("\\tan") )
```

**5.1.3.417 asin\_evalf()**

```
static ex GiNaC::asin_evalf (
    const ex & x ) [static]
```

References [asin\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.418 asin\_eval()**

```
static ex GiNaC::asin_eval (
    const ex & x ) [static]
```

References [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [asin\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.419 asin\_deriv()**

```
static ex GiNaC::asin_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex2](#), [\\_ex\\_1\\_2](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.420 asin\_conjugate()**

```
static ex GiNaC::asin_conjugate (
    const ex & x ) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [asin\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.421 asin\_info()**

```
static bool GiNaC::asin_info (
    const ex & x,
    unsigned inf ) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), and [x](#).

**5.1.3.422 REGISTER\_FUNCTION() [27/36]**

```
GiNaC::REGISTER_FUNCTION (
    asin ,
    eval_func(asin\_eval). evalf_func(asin\_evalf). info_func(asin\_info). derivative↔
_func(asin\_deriv). conjugate_func(asin\_conjugate). latex_name("\\arcsin") )
```

**5.1.3.423 acos\_evalf()**

```
static ex GiNaC::acos_evalf (
    const ex & x ) [static]
```

References [acos\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.424 `acos_eval()`**

```
static ex GiNaC::acos_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex1\\_3](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [acos\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.425 `acos_deriv()`**

```
static ex GiNaC::acos_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex2](#), [\\_ex\\_1\\_2](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.426 `acos_conjugate()`**

```
static ex GiNaC::acos_conjugate (
    const ex & x ) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [acos\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.427 `REGISTER_FUNCTION()` [28/36]**

```
GiNaC::REGISTER_FUNCTION (
    acos ,
    eval_func(acos\_eval). evalf_func(acos\_evalf). info_func(asin\_info). derivative↔
_func(acos\_deriv). conjugate_func(acos\_conjugate). latex_name("\\arccos") )
```

**5.1.3.428 `atan_evalf()`**

```
static ex GiNaC::atan_evalf (
    const ex & x ) [static]
```

References [atan\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.429 atan\_eval()**

```
static ex GiNaC::atan_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_4](#), [\\_ex\\_1](#), [\\_ex\\_1\\_4](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.430 atan\_deriv()**

```
static ex GiNaC::atan_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.431 atan\_series()**

```
static ex GiNaC::atan_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [abs\(\)](#), [atan\(\)](#), [csgn\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::relational::lhs\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [options](#), [order](#), [Pi](#), [GiNaC::info\\_flags::real](#), [GiNaC::relational::rhs\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

**5.1.3.432 atan\_conjugate()**

```
static ex GiNaC::atan_conjugate (
    const ex & x ) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [atan\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), and [x](#).

**5.1.3.433 atan\_info()**

```
static bool GiNaC::atan_info (
    const ex & x,
    unsigned inf ) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.434 REGISTER\_FUNCTION() [29/36]**

```
GiNaC::REGISTER_FUNCTION (
    atan ,
    eval_func(atan_eval). evalf_func(atan_evalf). info_func(atan_info). derivative↔
_func(atan_deriv). series_func(atan_series). conjugate_func(atan_conjugate). latex_name("\\arctan")
)
```

**5.1.3.435 atan2\_evalf()**

```
static ex GiNaC::atan2_evalf (
    const ex & y,
    const ex & x ) [static]
```

References [atan\(\)](#), and [x](#).

**5.1.3.436 atan2\_eval()**

```
static ex GiNaC::atan2_eval (
    const ex & y,
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1\\_2](#), [\\_ex1\\_4](#), [\\_ex\\_1\\_2](#), [\\_ex\\_1\\_4](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [Pi](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.437 atan2\_deriv()**

```
static ex GiNaC::atan2_deriv (
    const ex & y,
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex2](#), [\\_ex\\_1](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.438 atan2\_info()**

```
static bool GiNaC::atan2_info (
    const ex & y,
    const ex & x,
    unsigned inf ) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.439 REGISTER\_FUNCTION() [30/36]**

```
GiNaC::REGISTER_FUNCTION (
    atan2 ,
    eval_func(atan2_eval). evalf_func(atan2_evalf). info_func(atan2_info). evalf_←
func(atan2_evalf). derivative_func(atan2_deriv) )
```

**5.1.3.440 sinh\_evalf()**

```
static ex GiNaC::sinh_evalf (
    const ex & x ) [static]
```

References [GiNaC::basic::hold\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.441 sinh\_eval()**

```
static ex GiNaC::sinh_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1\\_2](#), [acosh\(\)](#), [asinh\(\)](#), [atanh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [l](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [real\(\)](#), [sin\(\)](#), [sinh\(\)](#), [sqrt\(\)](#), and [x](#).

**5.1.3.442 sinh\_deriv()**

```
static ex GiNaC::sinh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [cosh\(\)](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.443 sinh\_real\_part()**

```
static ex GiNaC::sinh_real_part (
    const ex & x ) [static]
```

References [cos\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.444 sinh\_imag\_part()**

```
static ex GiNaC::sinh_imag_part (
    const ex & x ) [static]
```

References [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), and [x](#).

**5.1.3.445 sinh\_conjugate()**

```
static ex GiNaC::sinh_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.446 REGISTER\_FUNCTION()** [31/36]

```
GiNaC::REGISTER_FUNCTION (
    sinh ,
    eval_func(sinh_eval). evalf_func(sinh_evalf). info_func(atan_info). derivative↔
    _func(sinh_deriv). real_part_func(sinh_real_part). imag_part_func(sinh_imag_part). conjugate↔
    _func(sinh_conjugate). latex_name("\\sinh") )
```

**5.1.3.447 cosh\_evalf()**

```
static ex GiNaC::cosh_evalf (
    const ex & x ) [static]
```

References [cosh\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.448 cosh\_eval()**

```
static ex GiNaC::cosh_eval (
    const ex & x ) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1\\_2](#), [acosh\(\)](#), [asinh\(\)](#), [atanh\(\)](#), [cos\(\)](#), [cosh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [real\(\)](#), [sqrt\(\)](#), and [x](#).

**5.1.3.449 cosh\_deriv()**

```
static ex GiNaC::cosh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [GINAC\\_ASSERT](#), [sinh\(\)](#), and [x](#).

**5.1.3.450 cosh\_real\_part()**

```
static ex GiNaC::cosh_real_part (
    const ex & x ) [static]
```

References [cos\(\)](#), [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [x](#).

**5.1.3.451 cosh\_imag\_part()**

```
static ex GiNaC::cosh_imag_part (
    const ex & x ) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.452 cosh\_conjugate()**

```
static ex GiNaC::cosh_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [cosh\(\)](#), and [x](#).

**5.1.3.453 REGISTER\_FUNCTION()** [32/36]

```
GiNaC::REGISTER_FUNCTION (
    cosh ,
    eval_func(cosh\_eval). evalf_func(cosh\_evalf). info_func(exp\_info). derivative←
_func(cosh\_deriv). real_part_func(cosh\_real\_part). imag_part_func(cosh\_imag\_part). conjugate←
_func(cosh\_conjugate). latex_name("\\cosh") )
```



**5.1.3.454 tanh\_evalf()**

```
static ex GiNaC::tanh_evalf (
    const ex & x ) [static]
```

References [GiNaC::basic::hold\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.455 tanh\_eval()**

```
static ex GiNaC::tanh_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [acosh\(\)](#), [asinh\(\)](#), [atanh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [real\(\)](#), [sqrt\(\)](#), [tan\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.456 tanh\_deriv()**

```
static ex GiNaC::tanh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex1](#), [\\_ex2](#), [GINAC\\_ASSERT](#), [tanh\(\)](#), and [x](#).

**5.1.3.457 tanh\_series()**

```
static ex GiNaC::tanh_series (
    const ex & x,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [cosh\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::odd](#), [options](#), [order](#), [Pi](#), [sinh\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.458 tanh\_real\_part()**

```
static ex GiNaC::tanh_real_part (
    const ex & x ) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.459 tanh\_imag\_part()**

```
static ex GiNaC::tanh_imag_part (
    const ex & x ) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.460 tanh\_conjugate()**

```
static ex GiNaC::tanh_conjugate (
    const ex & x ) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.461 REGISTER\_FUNCTION() [33/36]**

```
GiNaC::REGISTER_FUNCTION (
    tanh ,
    eval_func(tanh\_eval). evalf_func(tanh\_evalf). info_func(atan\_info). derivative↔
    _func(tanh\_deriv). series_func(tanh\_series). real_part_func(tanh\_real\_part). imag_part_↔
    func(tanh\_imag\_part). conjugate_func(tanh\_conjugate). latex_name("\\tanh") )
```

**5.1.3.462 asinh\_evalf()**

```
static ex GiNaC::asinh_evalf (
    const ex & x ) [static]
```

References [asinh\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.463 asinh\_eval()**

```
static ex GiNaC::asinh_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [asinh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), and [x](#).

**5.1.3.464 asinh\_deriv()**

```
static ex GiNaC::asinh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1\\_2](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.465 asinh\_conjugate()**

```
static ex GiNaC::asinh_conjugate (
    const ex & x ) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [asinh\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), and [x](#).

**5.1.3.466 REGISTER\_FUNCTION() [34/36]**

```
GiNaC::REGISTER_FUNCTION (
    asinh ,
    eval_func(asinh\_eval) . evalf_func(asinh\_evalf) . info_func(atan\_info) . derivative←
_func(asinh\_deriv) . conjugate_func(asinh\_conjugate) )
```

**5.1.3.467 acosh\_evalf()**

```
static ex GiNaC::acosh_evalf (
    const ex & x ) [static]
```

References [acosh\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.468 acosh\_eval()**

```
static ex GiNaC::acosh_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [acosh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.469 acosh\_deriv()**

```
static ex GiNaC::acosh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.470 acosh\_conjugate()**

```
static ex GiNaC::acosh_conjugate (
    const ex & x ) [static]
```

References [\\_num1\\_p](#), [acosh\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.471 REGISTER\_FUNCTION() [35/36]**

```
GiNaC::REGISTER_FUNCTION (
    acosh ,
    eval_func(acosh\_eval) . evalf_func(acosh\_evalf) . info_func(asin\_info) . derivative←
    _func(acosh\_deriv) . conjugate_func(acosh\_conjugate) )
```

**5.1.3.472 atanh\_evalf()**

```
static ex GiNaC::atanh_evalf (
    const ex & x ) [static]
```

References [atanh\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.473 atanh\_eval()**

```
static ex GiNaC::atanh_eval (
    const ex & x ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [atanh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), and [x](#).

**5.1.3.474 atanh\_deriv()**

```
static ex GiNaC::atanh_deriv (
    const ex & x,
    unsigned deriv_param ) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.475 atanh\_series()**

```
static ex GiNaC::atanh_series (
    const ex & arg,
    const relational & rel,
    int order,
    unsigned options ) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [abs\(\)](#), [atanh\(\)](#), [csgn\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::relational::lhs\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [options](#), [order](#), [Pi](#), [GiNaC::info\\_flags::real](#), [GiNaC::relational::rhs\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

**5.1.3.476 atanh\_conjugate()**

```
static ex GiNaC::atanh_conjugate (
    const ex & x ) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [atanh\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.477 REGISTER\_FUNCTION() [36/36]**

```
GiNaC::REGISTER_FUNCTION (
    atanh ,
    eval_func(atanh\_eval) . evalf_func(atanh\_evalf) . info_func(asin\_info) . derivative↔
_func(atanh\_deriv) . series_func(atanh\_series) . conjugate_func(atanh\_conjugate) )
```

**5.1.3.478 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [11/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    integral ,
    basic ,
    print_func< print\_dflt > &::do_print . print_func< print\_python > &::do_print .
    print_func< print\_latex > &::do_print_latex )
```

**5.1.3.479 subsvalue()**

```
ex GiNaC::subsvalue (
    const ex & var,
    const ex & value,
    const ex & fun )
```

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::subs\(\)](#), and [value](#).

Referenced by [adaptivesimpson\(\)](#).

**5.1.3.480 adaptivesimpson()**

```
GiNaC::ex GiNaC::adaptivesimpson (
    const ex & x,
    const ex & a_in,
    const ex & b_in,
    const ex & f,
    const ex & error )
```

Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.

Parameters are integration variable, left boundary, right boundary, function to be integrated and the relative integration error. The function should evalf into a number after substituting the integration variable by a number. Another thing to note is that this implementation is no good at integrating functions with discontinuities.

References [abs\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::integral::max\\_integration\\_level](#), [GiNaC::ex::subs\(\)](#), [subsvalue\(\)](#), and [x](#).

Referenced by [GiNaC::integral::evalf\(\)](#).

**5.1.3.481 GINAC\_BIND\_UNARCHIVER() [21/49]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    integral )
```

**5.1.3.482 GINAC\_DECLARE\_UNARCHIVER() [22/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    integral )
```

**5.1.3.483 ifactor()**

```
ex GiNaC::ifactor (
    const numeric & n )
```

Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $\text{lst}(p_1, \dots, p_r), \text{lst}(a_1, \dots, a_r)$  such that  $n = p_1^{a_1} \dots p_r^{a_r}$ .

References [GiNaC::container< C >::append\(\)](#), [irem\(\)](#),  $n$ , and [GiNaC::info\\_flags::prime](#).

Referenced by [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), and [kronecker\\_symbol\(\)](#).

**5.1.3.484 is\_discriminant\_of\_quadratic\_number\_field()**

```
bool GiNaC::is_discriminant_of_quadratic_number_field (
    const numeric & n )
```

Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.

Returns false otherwise.

Ref.: Toshitsune Miyake, Modular Forms, Chapter 3.1

References [abs\(\)](#), [ifactor\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::numeric::is\\_odd\(\)](#), [mod\(\)](#),  $n$ , [GiNaC::container< C >::nops\(\)](#), and [GiNaC::container< C >::op\(\)](#).

Referenced by [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#).

**5.1.3.485 kronecker\_symbol()**

```
numeric GiNaC::kronecker_symbol (
    const numeric & a,
    const numeric & n )
```

Returns the Kronecker symbol  $a$ : integer  $n$ : integer.

This routine defines  $\text{kronecker\_symbol}(1,0) = 1$   $\text{kronecker\_symbol}(-1,0) = 1$   $\text{kronecker\_symbol}(a,0) = 0$ ,  $a \neq 1, -1$

In particular  $\text{kronecker\_symbol}(-1,0) = 1$  (in agreement with Sage)

Ref.: Toshitsune Miyake, Modular Forms, Chapter 3.1

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ifactor\(\)](#), [GiNaC::numeric::is\\_even\(\)](#),  $n$ , [GiNaC::container< C >::op\(\)](#), [pow\(\)](#), and [unit](#).

Referenced by [primitive\\_dirichlet\\_character\(\)](#).

**5.1.3.486 primitive\_dirichlet\_character()**

```
numeric GiNaC::primitive_dirichlet_character (
    const numeric & n,
    const numeric & a )
```

Defines a primitive Dirichlet character through the Kronecker symbol.

n: integer a: discriminant of a quadratic field  $|a|$ : conductor

The character takes the values -1,0,1.

References [kronecker\\_symbol\(\)](#), and [n](#).

Referenced by [dirichlet\\_character\(\)](#), and [generalised\\_Bernoulli\\_number\(\)](#).

**5.1.3.487 dirichlet\_character()**

```
numeric GiNaC::dirichlet_character (
    const numeric & n,
    const numeric & a,
    const numeric & N )
```

Defines a Dirichlet character through the Kronecker symbol.

n: integer a: discriminant of a quadratic field  $|a|$ : conductor N: modulus, needs to be multiple of  $|a|$

The character takes the values -1,0,1.

References [gcd\(\)](#), [n](#), and [primitive\\_dirichlet\\_character\(\)](#).

**5.1.3.488 generalised\_Bernoulli\_number()**

```
numeric GiNaC::generalised_Bernoulli_number (
    const numeric & k,
    const numeric & b )
```

The generalised Bernoulli number.

k: index / modular weight

b: discriminant of a quadratic field, defines primitive character  $\psi$   $M=|b|$ : conductor of primitive character  $\psi$

The generalised Bernoulli number is computed from the series expansion of the generating function. The generating function is given in eq.(34), arXiv:1704.08895

References [abs\(\)](#), [GiNaC::ex::coeff\(\)](#), [exp\(\)](#), [factorial\(\)](#), [k](#), [primitive\\_dirichlet\\_character\(\)](#), [GiNaC::ex::series\(\)](#), [series\\_to\\_poly\(\)](#), and [x](#).



#### 5.1.3.489 Bernoulli\_polynomial()

```
ex GiNaC::Bernoulli_polynomial (
    const numeric & k,
    const ex & x )
```

The Bernoulli polynomials.

References [GiNaC::ex::coeff\(\)](#), [exp\(\)](#), [factorial\(\)](#), [k](#), [GiNaC::ex::series\(\)](#), [series\\_to\\_poly\(\)](#), and [x](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#).

#### 5.1.3.490 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [12/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    integration_kernel ,
    basic ,
    print_func< print_context > &::do_print )
```

#### 5.1.3.491 GINAC\_BIND\_UNARCHIVER() [22/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    integration_kernel )
```

#### 5.1.3.492 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [13/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    basic_log_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

#### 5.1.3.493 GINAC\_BIND\_UNARCHIVER() [23/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    basic_log_kernel )
```

**5.1.3.494 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [14/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    multiple_polylog_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

References [\\_ex1](#).

**5.1.3.495 GINAC\_BIND\_UNARCHIVER()** [24/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    multiple_polylog_kernel )
```

**5.1.3.496 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [15/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    ELi_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.497 GINAC\_BIND\_UNARCHIVER()** [25/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    ELi_kernel )
```

**5.1.3.498 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [16/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Ebar_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.499 GINAC\_BIND\_UNARCHIVER()** [26/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Ebar_kernel )
```

**5.1.3.500 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [17/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Kronecker_dtau_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.501 GINAC\_BIND\_UNARCHIVER()** [27/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Kronecker_dtau_kernel )
```

**5.1.3.502 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [18/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Kronecker_dz_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.503 GINAC\_BIND\_UNARCHIVER()** [28/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Kronecker_dz_kernel )
```

**5.1.3.504 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [19/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Eisenstein_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.505 GINAC\_BIND\_UNARCHIVER()** [29/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Eisenstein_kernel )
```

**5.1.3.506 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [20/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    Eisenstein_h_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.507 GINAC\_BIND\_UNARCHIVER()** [30/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    Eisenstein_h_kernel )
```

**5.1.3.508 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [21/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    modular_form_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.509 GINAC\_BIND\_UNARCHIVER()** [31/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    modular_form_kernel )
```

**5.1.3.510 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [22/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    user_defined_kernel ,
    integration_kernel ,
    print_func< print_context > &::do_print )
```

**5.1.3.511 GINAC\_BIND\_UNARCHIVER()** [32/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    user_defined_kernel )
```

**5.1.3.512 GINAC\_DECLARE\_UNARCHIVER()** [23/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    integration_kernel )
```

**5.1.3.513 GINAC\_DECLARE\_UNARCHIVER()** [24/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    basic_log_kernel )
```

**5.1.3.514 GINAC\_DECLARE\_UNARCHIVER()** [25/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    multiple_polylog_kernel )
```

**5.1.3.515 GINAC\_DECLARE\_UNARCHIVER()** [26/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    ELi_kernel )
```

**5.1.3.516 GINAC\_DECLARE\_UNARCHIVER()** [27/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Ebar_kernel )
```

**5.1.3.517 GINAC\_DECLARE\_UNARCHIVER()** [28/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Kronecker_dtau_kernel )
```

**5.1.3.518 GINAC\_DECLARE\_UNARCHIVER()** [29/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Kronecker_dz_kernel )
```

**5.1.3.519 GINAC\_DECLARE\_UNARCHIVER()** [30/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Eisenstein_kernel )
```

**5.1.3.520 GINAC\_DECLARE\_UNARCHIVER()** [31/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    Eisenstein_h_kernel )
```

**5.1.3.521 GINAC\_DECLARE\_UNARCHIVER()** [32/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    modular_form_kernel )
```

**5.1.3.522 GINAC\_DECLARE\_UNARCHIVER()** [33/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    user_defined_kernel )
```

**5.1.3.523 GINAC\_DECLARE\_UNARCHIVER()** [34/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    lst )
```

**5.1.3.524 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [23/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    matrix ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↔
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr > &↔
::do_print_python_repr )
```

Default ctor.

Initializes to 1 x 1-dimensional zero-matrix.

References [GiNaC::status\\_flags::not\\_shareable](#).

**5.1.3.525 GINAC\_BIND\_UNARCHIVER()** [33/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    matrix )
```

**5.1.3.526 lst\_to\_matrix()**

```
ex GiNaC::lst_to_matrix (
    const lst & l )
```

Convert list of lists to matrix.

References [cols\(\)](#), [GiNaC::container< C >::nops\(\)](#), and [rows\(\)](#).

**5.1.3.527 diag\_matrix()** [1/2]

```
ex GiNaC::diag_matrix (
    const lst & l )
```

Convert list of diagonal elements to matrix.

References [GiNaC::container< C >::nops\(\)](#).

**5.1.3.528 diag\_matrix()** [2/2]

```
ex GiNaC::diag_matrix (
    std::initializer_list< ex > l )
```

**5.1.3.529 unit\_matrix()** [1/2]

```
ex GiNaC::unit_matrix (
    unsigned r,
    unsigned c )
```

Create an r times c unit matrix.

References [\\_ex1](#), [c](#), [GiNaC::status\\_flags::evaluated](#), [r](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [unit\\_matrix\(\)](#).

**5.1.3.530 symbolic\_matrix()** [1/2]

```
ex GiNaC::symbolic_matrix (
    unsigned r,
    unsigned c,
    const std::string & base_name,
    const std::string & tex_base_name )
```

Create an  $r$  times  $c$  matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.

The base name for LaTeX output is specified separately.

References [c](#), [GiNaC::status\\_flags::evaluated](#), [r](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [symbolic\\_matrix\(\)](#).

**5.1.3.531 reduced\_matrix()**

```
ex GiNaC::reduced_matrix (
    const matrix & m,
    unsigned r,
    unsigned c )
```

Return the reduced matrix that is formed by deleting the  $r$ th row and  $c$ th column of matrix  $m$ .

The determinant of the result is the Minor  $r, c$ .

References [c](#), [cols\(\)](#), [GiNaC::status\\_flags::evaluated](#), [m](#), [r](#), [rows\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**5.1.3.532 sub\_matrix()**

```
ex GiNaC::sub_matrix (
    const matrix & m,
    unsigned r,
    unsigned nr,
    unsigned c,
    unsigned nc )
```

Return the  $nr$  times  $nc$  submatrix starting at position  $r, c$  of matrix  $m$ .

References [c](#), [GiNaC::status\\_flags::evaluated](#), [m](#), [r](#), and [GiNaC::basic::setflag\(\)](#).

**5.1.3.533 GINAC\_DECLARE\_UNARCHIVER()** [35/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    matrix )
```



**5.1.3.534 nops()** [2/2]

```
size_t GiNaC::nops (
    const matrix & m ) [inline]
```

References [m](#).

**5.1.3.535 expand()** [2/2]

```
ex GiNaC::expand (
    const matrix & m,
    unsigned options = 0 ) [inline]
```

References [m](#), and [options](#).

**5.1.3.536 evalf()** [2/2]

```
ex GiNaC::evalf (
    const matrix & m ) [inline]
```

References [m](#).

**5.1.3.537 rows()**

```
unsigned GiNaC::rows (
    const matrix & m ) [inline]
```

References [m](#).

Referenced by [lst\\_to\\_matrix\(\)](#), and [reduced\\_matrix\(\)](#).

**5.1.3.538 cols()**

```
unsigned GiNaC::cols (
    const matrix & m ) [inline]
```

References [m](#).

Referenced by [lst\\_to\\_matrix\(\)](#), and [reduced\\_matrix\(\)](#).

#### 5.1.3.539 transpose()

```
matrix GiNaC::transpose (
    const matrix & m ) [inline]
```

References [m](#).

#### 5.1.3.540 determinant()

```
ex GiNaC::determinant (
    const matrix & m,
    unsigned options = determinant_algo::automatic ) [inline]
```

References [m](#), and [options](#).

#### 5.1.3.541 trace()

```
ex GiNaC::trace (
    const matrix & m ) [inline]
```

References [m](#).

#### 5.1.3.542 charpoly()

```
ex GiNaC::charpoly (
    const matrix & m,
    const ex & lambda ) [inline]
```

References [m](#).

#### 5.1.3.543 inverse() [1/3]

```
matrix GiNaC::inverse (
    const matrix & m ) [inline]
```

References [GiNaC::solve\\_algo::automatic](#), and [m](#).

**5.1.3.544 inverse()** [2/3]

```
matrix GiNaC::inverse (
    const matrix & m,
    unsigned algo ) [inline]
```

References [m](#).

**5.1.3.545 rank()** [1/2]

```
unsigned GiNaC::rank (
    const matrix & m ) [inline]
```

References [m](#).

**5.1.3.546 rank()** [2/2]

```
unsigned GiNaC::rank (
    const matrix & m,
    unsigned solve_algo ) [inline]
```

References [m](#).

**5.1.3.547 unit\_matrix()** [2/2]

```
ex GiNaC::unit_matrix (
    unsigned x ) [inline]
```

Create a x times x unit matrix.

References [unit\\_matrix\(\)](#), and [x](#).

**5.1.3.548 symbolic\_matrix()** [2/2]

```
ex GiNaC::symbolic_matrix (
    unsigned r,
    unsigned c,
    const std::string & base_name ) [inline]
```

Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.

References [c](#), [r](#), and [symbolic\\_matrix\(\)](#).

**5.1.3.549 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [24/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    mul ,
    expairseq ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
print_tree. print_func< print_python_repr > &::do_print_python_repr )
```

**5.1.3.550 tryfactsubs()**

```
bool GiNaC::tryfactsubs (
    const ex & origfactor,
    const ex & patternfactor,
    int & nummatches,
    exmap & repls )
```

References [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::match\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), and [GiNaC::power::subs\(\)](#).

**5.1.3.551 algebraic\_match\_mul\_with\_mul()**

```
bool GiNaC::algebraic_match_mul_with_mul (
    const mul & e,
    const ex & pat,
    exmap & repls,
    int factor,
    int & nummatches,
    const std::vector< bool > & substed,
    std::vector< bool > & matched )
```

Checks whether e matches to the pattern pat and the (possibly to be updated) list of replacements repls.

This matching is in the sense of algebraic substitutions. Matching starts with pat.op(factor) of the pattern because the factors before this one have already been matched. The (possibly updated) number of matches is in nummatches. substed[i] is true for factors that already have been replaced by previous substitutions and matched[i] is true for factors that have been matched by the current match.

References [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [factor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::nops\(\)](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::expairseq::op\(\)](#), and [tryfactsubs\(\)](#).

Referenced by [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), and [GiNaC::mul::has\(\)](#).

**5.1.3.552 GINAC\_BIND\_UNARCHIVER()** [34/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    mul )
```

**5.1.3.553 GINAC\_DECLARE\_UNARCHIVER()** [36/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    mul )
```

**5.1.3.554 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [25/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    ncmul ,
    exprseq ,
    print_func< print_context > &::do_print, print_func< print_tree > &::do_print↵
    _tree, print_func< print_csrc > &::do_print_csrc, print_func< print_python_repr > &::do_↵
    print_csrc )
```

**5.1.3.555 reeval\_ncmul()**

```
ex GiNaC::reeval_ncmul (
    const exvector & v )
```

**5.1.3.556 hold\_ncmul()**

```
ex GiNaC::hold_ncmul (
    const exvector & v )
```

Referenced by [GiNaC::basic::eval\\_ncmul\(\)](#), [GiNaC::color::eval\\_ncmul\(\)](#), and [GiNaC::structure< T, ComparisonPolicy >::eval\\_ncmul\(\)](#)

**5.1.3.557 GINAC\_BIND\_UNARCHIVER()** [35/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    ncmul )
```

**5.1.3.558 GINAC\_DECLARE\_UNARCHIVER()** [37/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    ncmul )
```

**5.1.3.559 get\_first\_symbol()**

```
static bool GiNaC::get_first_symbol (
    const ex & e,
    ex & x ) [static]
```

Return pointer to first symbol found in expression.

Due to [GiNaC](#)'s internal ordering of terms, it may not be obvious which symbol this function returns for a given expression.

**Parameters**

<i>e</i>	expression to search
<i>x</i>	first symbol found (returned)

**Returns**

"false" if no symbol was found, "true" otherwise

References [get\\_first\\_symbol\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [x](#).

Referenced by [divide\(\)](#), [frac\\_cancel\(\)](#), [get\\_first\\_symbol\(\)](#), and [GiNaC::ex::unit\(\)](#).

**5.1.3.560 add\_symbol()**

```
static void GiNaC::add_symbol (
    const ex & s,
    sym\_desc\_vec & v ) [static]
```

Referenced by [collect\\_symbols\(\)](#).

**5.1.3.561 collect\_symbols()**

```
static void GiNaC::collect_symbols (
    const ex & e,
    sym\_desc\_vec & v ) [static]
```

References [add\\_symbol\(\)](#), [collect\\_symbols\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [collect\\_symbols\(\)](#), and [get\\_symbol\\_stats\(\)](#).

**5.1.3.562 get\_symbol\_stats()**

```
static void GiNaC::get_symbol_stats (
    const ex & a,
    const ex & b,
    sym\_desc\_vec & v ) [static]
```

Collect statistical information about symbols in polynomials.

This function fills in a vector of "sym\_desc" structs which contain information about the highest and lowest degrees of all symbols that appear in two polynomials. The vector is then sorted by minimum degree (lowest to highest). The information gathered by this function is used by the GCD routines to identify trivial factors and to determine which variable to choose as the main variable for GCD computation.

## Parameters

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>v</i>	vector of <a href="#">sym_desc</a> structs (filled in)

References [collect\\_symbols\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::ex::lcoeff\(\)](#), [GiNaC::ex::ldegree\(\)](#), and [GiNaC::ex::nops\(\)](#).

Referenced by [divide\\_in\\_z\(\)](#), [gcd\(\)](#), and [sqrfree\(\)](#).

**5.1.3.563 lcmcoeff()**

```
static numeric GiNaC::lcmcoeff (
    const ex & e,
    const numeric & l ) [static]
```

References [\\_num1\\_p](#), [c](#), [denom\(\)](#), [GiNaC::ex::info\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::rational](#).

Referenced by [heur\\_gcd\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [lcmcoeff\(\)](#), and [multiply\\_lcm\(\)](#).

**5.1.3.564 lcm\_of\_coefficients\_denominators()**

```
static numeric GiNaC::lcm_of_coefficients_denominators (
    const ex & e ) [static]
```

Compute LCM of denominators of coefficients of a polynomial.

Given a polynomial with rational coefficients, this function computes the LCM of the denominators of all coefficients. This can be used to bring a polynomial from  $\mathbb{Q}[X]$  to  $\mathbb{Z}[X]$ .

## Parameters

<i>e</i>	multivariate polynomial (need not be expanded)
----------	--

## Returns

LCM of denominators of coefficients

References [\\_num1\\_p](#), and [lcmcoeff\(\)](#).

Referenced by [frac\\_cancel\(\)](#), [heur\\_gcd\(\)](#), and [sqrfree\(\)](#).

### 5.1.3.565 multiply\_lcm()

```
static ex GiNaC::multiply_lcm (
    const ex & e,
    const numeric & lcm ) [static]
```

Bring polynomial from  $\mathbb{Q}[X]$  to  $\mathbb{Z}[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.

#### Parameters

<i>e</i>	multivariate polynomial (need not be expanded)
<i>lcm</i>	LCM to multiply in

References [\\_num1\\_p](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_rational\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [frac\\_cancel\(\)](#), [multiply\\_lcm\(\)](#), and [sqrfree\(\)](#).

### 5.1.3.566 quo()

```
ex GiNaC::quo (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args )
```

Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

It satisfies  $a(x)=b(x)*q(x)+r(x)$ .

#### Parameters

<i>a</i>	first polynomial in x (dividend)
<i>b</i>	second polynomial in x (divisor)
<i>x</i>	a and b are polynomials in x
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

#### Returns

quotient of a and b in  $\mathbb{Q}[x]$

References [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), *r*, [GiNaC::info\\_flags::rational\\_polynomial](#), and *x*.

Referenced by [decomp\\_rational\(\)](#), [GiNaC::ex::primpart\(\)](#), [sqrfree\\_parfrac\(\)](#), [sqrfree\\_yun\(\)](#), and [GiNaC::ex::unitcontprim\(\)](#).



**5.1.3.567 rem()**

```
ex GiNaC::rem (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args )
```

Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

It satisfies  $a(x)=b(x)*q(x)+r(x)$ .

**Parameters**

$a$	first polynomial in $x$ (dividend)
$b$	second polynomial in $x$ (divisor)
$x$	$a$ and $b$ are polynomials in $x$
$check\_args$	check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true")

**Returns**

remainder of  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$

References [\\_ex0](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [x](#).

Referenced by [decomp\\_rational\(\)](#), and [sqrfree\\_parfrac\(\)](#).

**5.1.3.568 decomp\_rational()**

```
ex GiNaC::decomp_rational (
    const ex & a,
    const ex & x )
```

Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .

**Parameters**

$a$	rational function in $x$
$x$	$a$ is a function of $x$

**Returns**

decomposed function.

References [denom\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [GiNaC::ex::op\(\)](#), [quo\(\)](#), [rem\(\)](#), and [x](#).

### 5.1.3.569 prem()

```
ex GiNaC::prem (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args )
```

Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

#### Parameters

<i>a</i>	first polynomial in $x$ (dividend)
<i>b</i>	second polynomial in $x$ (divisor)
<i>x</i>	$a$ and $b$ are polynomials in $x$
<i>check_args</i>	check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true")

#### Returns

pseudo-remainder of  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [x](#).

Referenced by [sr\\_gcd\(\)](#).

### 5.1.3.570 sprem()

```
ex GiNaC::sprem (
    const ex & a,
    const ex & b,
    const ex & x,
    bool check_args )
```

Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

#### Parameters

<i>a</i>	first polynomial in $x$ (dividend)
<i>b</i>	second polynomial in $x$ (divisor)
<i>x</i>	$a$ and $b$ are polynomials in $x$
<i>check_args</i>	check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true")

#### Returns

sparse pseudo-remainder of  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [x](#).

**5.1.3.571 divide()**

```
bool GiNaC::divide (
    const ex & a,
    const ex & b,
    ex & q,
    bool check_args )
```

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $\mathbb{Q}[X]$ .

**Parameters**

<i>a</i>	first multivariate polynomial (dividend)
<i>b</i>	second multivariate polynomial (divisor)
<i>q</i>	quotient (returned)
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

**Returns**

"true" when exact division succeeds (quotient returned in q), "false" otherwise (q left untouched)

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [get\\_first\\_symbol\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [x](#).

Referenced by [divide\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [gcd\(\)](#), [quo\(\)](#), and [rem\(\)](#).

**5.1.3.572 divide\_in\_z()**

```
static bool GiNaC::divide_in_z (
    const ex & a,
    const ex & b,
    ex & q,
    sym_desc_vec::const_iterator var ) [static]
```

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $\mathbb{Z}[X]$ .

This functions works like [divide\(\)](#) but the input and output polynomials are in  $\mathbb{Z}[X]$  instead of  $\mathbb{Q}[X]$  (i.e. they have integer coefficients). Unlike [divide\(\)](#), it doesn't check whether the input polynomials really are integer polynomials, so be careful of what you pass in. Also, you have to run [get\\_symbol\\_stats\(\)](#) over the input polynomials before calling this function and pass an iterator to the first element of the [sym\\_desc](#) vector. This function is used internally by the [heur\\_gcd\(\)](#).

**Parameters**

<i>a</i>	first multivariate polynomial (dividend)
<i>b</i>	second multivariate polynomial (divisor)
<i>q</i>	quotient (returned)
<i>var</i>	iterator to first element of vector of <a href="#">sym_desc</a> structs

**Returns**

"true" when exact division succeeds (the quotient is returned in q), "false" otherwise.

**See also**

[get\\_symbol\\_stats](#), [heur\\_gcd](#)

References [\\_ex0](#), [\\_ex1](#), [\\_num0\\_p](#), [\\_num1\\_p](#), [GiNaC::ex::begin\(\)](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::find\(\)](#), [get\\_symbol\\_stats\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [k](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [qbar](#), [r](#), [GiNaC::ex::subs\(\)](#), and [x](#).

Referenced by [divide\\_in\\_z\(\)](#), [heur\\_gcd\\_z\(\)](#), and [sr\\_gcd\(\)](#).

**5.1.3.573 sr\_gcd()**

```
static ex GiNaC::sr_gcd (
    const ex & a,
    const ex & b,
    sym_desc_vec::const_iterator var ) [static]
```

Compute GCD of multivariate polynomials using the subresultant PRS algorithm.

This function is used internally by [gcd\(\)](#).

**Parameters**

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>var</i>	iterator to first element of vector of <a href="#">sym_desc</a> structs

**Returns**

the GCD as a new expression

**See also**

[gcd](#)

References [\\_ex0](#), [\\_ex1](#), [c](#), [GiNaC::ex::content\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [gcd\(\)](#), [pow\(\)](#), [prem\(\)](#), [GiNaC::ex::primpart\(\)](#), [psi\(\)](#), [r](#), and [x](#).

Referenced by [gcd\(\)](#).

**5.1.3.574 interpolate()**

```
static ex GiNaC::interpolate (
    const ex & gamma,
    const numeric & xi,
    const ex & x,
    int degree_hint = 1 ) [static]
```

xi-adic polynomial interpolation

References [GiNaC::numeric::inverse\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), [GiNaC::ex::smod\(\)](#), and [x](#).

Referenced by [heur\\_gcd\\_z\(\)](#).

**5.1.3.575 heur\_gcd\_z()**

```
static bool GiNaC::heur_gcd_z (
    ex & res,
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb,
    sym_desc_vec::const_iterator var ) [static]
```

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

[get\\_symbol\\_stats\(\)](#) must have been called previously with the input polynomials and an iterator to the first element of the [sym\\_desc](#) vector passed in. This function is used internally by [gcd\(\)](#).

**Parameters**

<i>a</i>	first integer multivariate polynomial (expanded)
<i>b</i>	second integer multivariate polynomial (expanded)
<i>ca</i>	cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor
<i>cb</i>	cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor
<i>var</i>	iterator to first element of vector of <a href="#">sym_desc</a> structs
<i>res</i>	the GCD (returned)

**Returns**

true if GCD was computed, false otherwise.

**See also**

[gcd](#)

**Exceptions**

<a href="#">gcdheu_failed()</a>
---------------------------------

References [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [heur\\_gcd\\_z\(\)](#), [GiNaC::numeric::int\\_length\(\)](#), [GiNaC::ex::integer\\_content\(\)](#), [interpolate\(\)](#), [GiNaC::numeric::inverse\(\)](#), [iquo\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [isqrt\(\)](#), [GiNaC::ex::max\\_coefficient\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::subs\(\)](#), and [x](#).

Referenced by [heur\\_gcd\(\)](#), and [heur\\_gcd\\_z\(\)](#).

### 5.1.3.576 [heur\\_gcd\(\)](#)

```
static bool GiNaC::heur_gcd (
    ex & res,
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb,
    sym_desc_vec::const_iterator var ) [static]
```

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

[get\\_symbol\\_stats\(\)](#) must have been called previously with the input polynomials and an iterator to the first element of the [sym\\_desc](#) vector passed in. This function is used internally by [gcd\(\)](#).

#### Parameters

<i>a</i>	first rational multivariate polynomial (expanded)
<i>b</i>	second rational multivariate polynomial (expanded)
<i>ca</i>	cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor
<i>cb</i>	cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor
<i>var</i>	iterator to first element of vector of <a href="#">sym_desc</a> structs
<i>res</i>	the GCD (returned)

#### Returns

true if GCD was computed, false otherwise.

#### See also

[heur\\_gcd\\_z](#)

[gcd](#)

References [heur\\_gcd\\_z\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), and [lcmcoeff\(\)](#).

Referenced by [gcd\(\)](#).

**5.1.3.577 gcd\_pf\_pow()**

```
static ex GiNaC::gcd_pf_pow (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb ) [static]
```

References [\\_ex1](#), [expand\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::lddegree\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [gcd\(\)](#), and [gcd\\_pf\\_pow\(\)](#).

**5.1.3.578 gcd\_pf\_mul()**

```
static ex GiNaC::gcd_pf_mul (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb ) [static]
```

References [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [gcd\(\)](#), and [gcd\\_pf\\_mul\(\)](#).

**5.1.3.579 gcd() [1/2]**

```
ex GiNaC::gcd (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb,
    bool check_args,
    unsigned options )
```

Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $\mathbb{Z}[X]$ .

Optionally also compute the cofactors of  $a$  and  $b$ , defined by  $a = ca * \text{gcd}(a, b)$  and  $b = cb * \text{gcd}(a, b)$ .

**Parameters**

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>ca</i>	pointer to expression that will receive the cofactor of $a$ , or nullptr
<i>cb</i>	pointer to expression that will receive the cofactor of $b$ , or nullptr
<i>check_args</i>	check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true")

**Returns**

the GCD as a new expression

References [\\_ex0](#), [\\_ex1](#), [divide\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [gcd\\_pf\\_pow\(\)](#), [get\\_symbol\\_stats\(\)](#), [heur\\_gcd\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::integer\\_content\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [n](#), [GiNaC::gcd\\_options::no\\_heur\\_gcd](#), [GiNaC::gcd\\_options::no\\_part\\_factored](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [pow\(\)](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [sr\\_gcd\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [GiNaC::gcd\\_options::use\\_sr\\_gcd](#), and [x](#).

Referenced by [GiNaC::ex::content\(\)](#), [dirichlet\\_character\(\)](#), [find\\_common\\_factor\(\)](#), [frac\\_cancel\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [heur\\_gcd\\_z\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [lcm\(\)](#), [GiNaC::add::normal\(\)](#), [sqrfree\\_yun\(\)](#), and [sr\\_gcd\(\)](#).

**5.1.3.580 gcd\_pf\_pow\_pow()**

```
static ex GiNaC::gcd_pf_pow_pow (
    const ex & a,
    const ex & b,
    ex * ca,
    ex * cb ) [static]
```

References [\\_ex1](#), [gcd\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [gcd\\_pf\\_pow\(\)](#).

**5.1.3.581 lcm() [1/2]**

```
ex GiNaC::lcm (
    const ex & a,
    const ex & b,
    bool check_args )
```

Compute LCM (Least Common Multiple) of multivariate polynomials in  $\mathbb{Z}[X]$ .

**Parameters**

<i>a</i>	first multivariate polynomial
<i>b</i>	second multivariate polynomial
<i>check_args</i>	check whether a and b are polynomials with rational coefficients (defaults to "true")

**Returns**

the LCM as a new expression

References [gcd\(\)](#), [GiNaC::ex::info\(\)](#), [lcm\(\)](#), and [GiNaC::info\\_flags::rational\\_polynomial](#).

Referenced by [GiNaC::add::integer\\_content\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [multiply\\_lcm\(\)](#), and [sqrfree\(\)](#).



**5.1.3.582 sqrfree\_yun()**

```
static epvector GiNaC::sqrfree_yun (
    const ex & a,
    const symbol & x ) [static]
```

Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.

Used internally by [sqrfree\(\)](#).

**Parameters**

$a$	multivariate polynomial over $\mathbb{Z}[X]$ , treated here as univariate polynomial in $x$ (needs not be expanded).
$x$	variable to factor in

**Returns**

vector of expairs (factor, exponent), sorted by exponents

References [\\_ex1](#), [GiNaC::ex::diff\(\)](#), [factors](#), [gcd\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [quo\(\)](#), and [x](#).

Referenced by [sqrfree\(\)](#), and [sqrfree\\_parfrac\(\)](#).

**5.1.3.583 sqrfree()**

```
ex GiNaC::sqrfree (
    const ex & a,
    const lst & l )
```

Compute a square-free factorization of a multivariate polynomial in  $\mathbb{Q}[X]$ .

**Parameters**

$a$	multivariate polynomial over $\mathbb{Q}[X]$ (needs not be expanded)
$l$	lst of variables to factor in, may be left empty for autodetection

**Returns**

a square-free factorization of  $a$ .

**Note**

A polynomial  $p(X) \in C[X]$  is said *square-free* if, whenever any two polynomials  $q(X)$  and  $r(X)$  are such that

$$p(X) = q(X)^2 r(X),$$

we have  $q(X) \in C$ . This means that  $p(X)$  has no repeated factors, apart eventually from constants. Given a polynomial  $p(X) \in C[X]$ , we say that the decomposition

$$p(X) = b \cdot p_1(X)^{a_1} \cdot p_2(X)^{a_2} \cdots p_r(X)^{a_r}$$

is a *square-free factorization* of  $p(X)$  if the following conditions hold:

1.  $b \in C$  and  $b \neq 0$ ;
2.  $a_i$  is a positive integer for  $i = 1, \dots, r$ ;
3. the degree of the polynomial  $p_i$  is strictly positive for  $i = 1, \dots, r$ ;
4. the polynomial  $\prod_{i=1}^r p_i(X)$  is square-free.

Square-free factorizations need not be unique. For example, if  $a_i$  is even, we could change the polynomial  $p_i(X)$  into  $-p_i(X)$ . Observe also that the factors  $p_i(X)$  need not be irreducible polynomials.

References [\\_ex0](#), [GiNaC::container< C >::append\(\)](#), [factors](#), [get\\_symbol\\_stats\(\)](#), [lcm\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::container< C >::remove\\_first\(\)](#), [sqrfree\(\)](#), [sqrfree\\_yun\(\)](#), and [x](#).

Referenced by [sqrfree\(\)](#).

### 5.1.3.584 sqrfree\_parfrac()

```
ex GiNaC::sqrfree_parfrac (
    const ex & a,
    const symbol & x )
```

Compute square-free partial fraction decomposition of rational function a(x).

#### Parameters

a	rational function over $\mathbb{Z}[x]$ , treated as univariate polynomial in x
x	variable to factor in

#### Returns

decomposed rational function

References [\\_ex1](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::ex::coeff\(\)](#), [coeff\(\)](#), [degree\(\)](#), [denom\(\)](#), [GiNaC::ex::expand\(\)](#), [factor\(\)](#), [GINAC\\_ASSERT](#), [k](#), [n](#), [numer\(\)](#), [numer\\_denom\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [quo\(\)](#), [r](#), [rem\(\)](#), [rhs\(\)](#), [GiNaC::matrix::solve\(\)](#), [sqrfree\\_yun\(\)](#), [to\\_int\(\)](#), and [x](#).

### 5.1.3.585 replace\_with\_symbol() [1/2]

```
static ex GiNaC::replace_with_symbol (
    const ex & e,
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) [static]
```

Create a symbol for replacing the expression "e" (or return a previously assigned symbol).

The symbol and expression are appended to repl, for a later application of [subs\(\)](#). An entry in the replacement table repl can be changed in some cases. If it was altered, we need to provide the modifier for the previously build expressions. The modifier is an (ordered) list, because those substitutions need to be done in the incremental order. As

an example let us consider a rationalisation of the expression  $e = \exp(2x) \cdot \cos(\exp(2x) + 1) \cdot \exp(x)$ . The first factor [GiNaC](#) denotes by something like `symbol1` and will record: `e = symbol1 * cos(symbol1 + 1) * exp(x)` `repl = {symbol1 : exp(2*x)}`. Similarly, the second factor would be denoted as `symbol2` and we will have `e = symbol1 * symbol2 * exp(x)` `repl = {symbol1 : exp(2*x), symbol2 : cos(symbol1 + 1)}`. Denoting the third term as `symbol3` [GiNaC](#) is willing to re-think `exp(2*x)` as `symbol3^2` rather than just `symbol1`. Here are two issues: 1) The replacement "`symbol1 -> symbol3^2`" in the previous part of the expression needs to be done outside of the present routine; 2) The pair "`symbol1 : exp(2*x)`" shall be deleted from the replacement table `repl`. However, this will create illegal substitution "`symbol2 : cos(symbol1 + 1)`" with undefined `symbol1`. These both problems are mitigated through the additions of the record "`symbol1 == symbol3^2`" to the list modifier. Changed length of the modifier signals to the calling code that the previous portion of the expression needs to be altered (it solves 1). Thus [GiNaC](#) can record now `e = symbol3^2 * symbol2 * symbol3` `repl = {symbol2 : cos(symbol1 + 1), symbol3 : exp(x)}` `modifier = {symbol1 == symbol3^2}`. Then, doing the backward substitutions the list modifier will be used to restore such iterative substitutions in the right way (this solves 2).

See also

[ex::normal](#)

References [\\_ex\\_1](#), [GiNaC::container< C >::append\(\)](#), [degree\(\)](#), [denom\(\)](#), [exp\(\)](#), [GiNaC::ex::find\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_integer\(\)](#), [is\\_rational\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [normal\(\)](#), [numer\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::basic::normal\(\)](#), [GiNaC::numeric::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::basic::to\\_polynomial\(\)](#), [GiNaC::numeric::to\\_polynomial\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::basic::to\\_rational\(\)](#), [GiNaC::numeric::to\\_rational\(\)](#), and [GiNaC::power::to\\_rational\(\)](#).

### 5.1.3.586 replace\_with\_symbol() [2/2]

```
static ex GiNaC::replace_with_symbol (
    const ex & e,
    exmap & repl ) [static]
```

Create a symbol for replacing the expression "e" (or return a previously assigned symbol).

The symbol and expression are appended to `repl`, and the symbol is returned.

See also

[basic::to\\_rational](#)

[basic::to\\_polynomial](#)

References [GiNaC::subs\\_options::no\\_pattern](#), and [GiNaC::ex::subs\(\)](#).

### 5.1.3.587 frac\_cancel()

```
static ex GiNaC::frac_cancel (
    const ex & n,
    const ex & d ) [static]
```

Fraction cancellation.

## Parameters

<i>n</i>	numerator
<i>d</i>	denominator

## Returns

cancelled fraction {n, d} as a list

References [\\_ex1](#), [\\_ex\\_1](#), [\\_num1\\_p](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [get\\_first\\_symbol\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_negative\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [multiply\\_lcm\(\)](#), [n](#), [GiNaC::numeric::numer\(\)](#), [GiNaC::ex::unit\(\)](#), and [x](#).

Referenced by [GiNaC::add::normal\(\)](#), and [GiNaC::mul::normal\(\)](#).

**5.1.3.588 find\_common\_factor()**

```
static ex GiNaC::find_common_factor (
    const ex & e,
    ex & factor,
    exmap & repl ) [static]
```

Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).

References [\\_ex0](#), [\\_ex1](#), [divide\(\)](#), [factor\(\)](#), [find\\_common\\_factor\(\)](#), [gcd\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [k](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [GiNaC::ex::to\\_polynomial\(\)](#), and [x](#).

Referenced by [collect\\_common\\_factors\(\)](#), and [find\\_common\\_factor\(\)](#).

**5.1.3.589 collect\_common\_factors()**

```
ex GiNaC::collect_common_factors (
    const ex & e )
```

Collect common factors in sums.

This converts expressions like 'a\*(b\*x+b\*y)' to 'a\*b\*(x+y)'.

References [factor\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [r](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::power::to\\_polynomial\(\)](#).

**5.1.3.590 resultant()**

```
ex GiNaC::resultant (
    const ex & e1,
    const ex & e2,
    const ex & s )
```

Resultant of two expressions e1,e2 with respect to symbol s.

Method: Compute determinant of Sylvester matrix of e1,e2,s.

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [k](#), [GiNaC::ex::lddegree\(\)](#), [m](#), and [GiNaC::info\\_flags::polynomial](#).

**5.1.3.591 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [26/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    numeric ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_csrc_cl_N > &↵
::do_print_csrc_cl_N. print_func< print_tree > &::do_print_tree. print_func< print_python_repr
> &::do_print_python_repr )
```

default ctor.

Numerically it initializes to an integer zero.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [GiNaC::numeric::value](#).

**5.1.3.592 make\_real\_float()**

```
static const cln::cl_F GiNaC::make_real_float (
    const cln::cl_idcoded_float & dec ) [static]
```

Construct a floating point number from sign, mantissa, and exponent.

References [x](#).

Referenced by [read\\_real\\_float\(\)](#).

#### 5.1.3.593 read\_real\_float()

```
static const cln::cl_F GiNaC::read_real_float (
    std::istream & s ) [static]
```

Read serialized floating point number.

References [make\\_real\\_float\(\)](#), and [x](#).

Referenced by [GiNaC::numeric::read\\_archive\(\)](#).

#### 5.1.3.594 GINAC\_BIND\_UNARCHIVER() [36/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    numeric )
```

#### 5.1.3.595 write\_real\_float()

```
static void GiNaC::write_real_float (
    std::ostream & s,
    const cln::cl_R & n ) [static]
```

References [n](#).

Referenced by [GiNaC::numeric::archive\(\)](#).

#### 5.1.3.596 print\_real\_number()

```
static void GiNaC::print_real_number (
    const print_context & c,
    const cln::cl_R & x ) [static]
```

Helper function to print a real number in a nicer way than is CLN's default.

Instead of printing 42.0L0 this just prints 42.0 to ostream os and instead of 3.99168L7 it prints 3.99168E7. This is fine in [GiNaC](#) as long as it only uses `cl_LF` and no other floating point types that we might want to visibly distinguish from `cl_LF`.

See also

[numeric::print\(\)](#)

References [c](#), and [x](#).

Referenced by [GiNaC::numeric::print\\_numeric\(\)](#), and [print\\_real\\_cl\\_N\(\)](#).

### 5.1.3.597 `print_integer_csrc()`

```
static void GiNaC::print_integer_csrc (
    const print\_context & c,
    const cln::cl\_I & x ) [static]
```

Helper function to print integer number in C++ source format.

See also

[numeric::print\(\)](#)

References [c](#), and [x](#).

Referenced by [print\\_real\\_csrc\(\)](#).

### 5.1.3.598 `print_real_csrc()`

```
static void GiNaC::print_real_csrc (
    const print\_context & c,
    const cln::cl\_R & x ) [static]
```

Helper function to print real number in C++ source format.

See also

[numeric::print\(\)](#)

References [c](#), [denom\(\)](#), [numer\(\)](#), [print\\_integer\\_csrc\(\)](#), and [x](#).

Referenced by [GiNaC::numeric::do\\_print\\_csrc\(\)](#).

### 5.1.3.599 `coerce()`

```
template<typename T1 , typename T2 >
static bool GiNaC::coerce (
    T1 & dst,
    const T2 & arg ) [inline], [static]
```

Referenced by [print\\_real\\_cl\\_N\(\)](#).

**5.1.3.600 coerce< int, cln::cl\_I >()**

```
template<>
bool GiNaC::coerce< int, cln::cl_I > (
    int & dst,
    const cln::cl_I & arg ) [inline]
```

Check if CLN integer can be converted into int.

See also

<https://www.ginac.de/pipermail/cln-list/2006-October/000248.html>

**5.1.3.601 coerce< unsigned int, cln::cl\_I >()**

```
template<>
bool GiNaC::coerce< unsigned int, cln::cl_I > (
    unsigned int & dst,
    const cln::cl_I & arg ) [inline]
```

**5.1.3.602 print\_real\_cl\_N()**

```
static void GiNaC::print_real_cl_N (
    const print_context & c,
    const cln::cl_R & x ) [static]
```

Helper function to print real number in C++ source format using cl\_N types.

See also

[numeric::print\(\)](#)

References [c](#), [coerce\(\)](#), [Digits](#), [print\\_real\\_number\(\)](#), and [x](#).

Referenced by [GiNaC::numeric::do\\_print\\_csrc\\_cl\\_N\(\)](#).

**5.1.3.603 exp()**

```
const numeric GiNaC::exp (
    const numeric & x )
```

Exponential function.

Returns

arbitrary precision numerical exp(x).

References [x](#).

Referenced by [abs\\_eval\(\)](#), [abs\\_power\(\)](#), [Bernoulli\\_polynomial\(\)](#), [beta\\_evalf\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [csgn\\_power\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [exp\\_conjugate\(\)](#), [exp\\_deriv\(\)](#), [exp\\_eval\(\)](#), [exp\\_evalf\(\)](#), [exp\\_expand\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_power\(\)](#), [exp\\_real\\_part\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_val](#), [GiNaC::power::imag\\_part\(\)](#), [log\\_eval\(\)](#), [print\\_sym\\_pow\(\)](#), [GiNaC::power::real\\_part\(\)](#), [replace\\_with\\_symbol\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), and [tgamma\(\)](#).



**5.1.3.604 log()**

```
const numeric GiNaC::log (
    const numeric & x )
```

Natural logarithm.

**Parameters**

x	complex number
---	----------------

**Returns**

arbitrary precision numerical  $\log(x)$ .

**Exceptions**

<i>pole_error("log()"</i>	<i>logarithmic pole",0)</i>
---------------------------	-----------------------------

References [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

Referenced by [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [exp\\_eval\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [H\\_eval\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [lgamma\(\)](#), [lgamma\\_eval\(\)](#), [lgamma\\_series\(\)](#), [Li2\\_deriv\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [log\\_conjugate\(\)](#), [log\\_eval\(\)](#), [log\\_evalf\(\)](#), [log\\_expand\(\)](#), [log\\_real\\_part\(\)](#), [log\\_series\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::power::real\\_part\(\)](#), [S\\_eval\(\)](#), and [GiNaC::power::series\(\)](#).

**5.1.3.605 sin()**

```
const numeric GiNaC::sin (
    const numeric & x )
```

Numeric sine (trigonometric function).

**Returns**

arbitrary precision numerical  $\sin(x)$ .

References [x](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [cos\\_deriv\(\)](#), [cos\\_imag\\_part\(\)](#), [cosh\\_imag\\_part\(\)](#), [exp\\_imag\\_part\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [lgamma\(\)](#), [sin\\_conjugate\(\)](#), [sin\\_eval\(\)](#), [sin\\_evalf\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_eval\(\)](#), [sinh\\_imag\\_part\(\)](#), and [tan\\_series\(\)](#).

### 5.1.3.606 `cos()`

```
const numeric GiNaC::cos (
    const numeric & x )
```

Numeric cosine (trigonometric function).

#### Returns

arbitrary precision numerical  $\cos(x)$ .

References [x](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [cos\\_conjugate\(\)](#), [cos\\_eval\(\)](#), [cos\\_evalf\(\)](#), [cos\\_real\\_part\(\)](#), [cosh\\_eval\(\)](#), [cosh\\_real\\_part\(\)](#), [exp\\_real\\_part\(\)](#), [GiNaC::power::real\\_part\(\)](#), [sin\\_deriv\(\)](#), [sin\\_imag\\_part\(\)](#), [sinh\\_real\\_part\(\)](#), and [tan\\_series\(\)](#).

### 5.1.3.607 `tan()`

```
const numeric GiNaC::tan (
    const numeric & x )
```

Numeric tangent (trigonometric function).

#### Returns

arbitrary precision numerical  $\tan(x)$ .

References [x](#).

Referenced by [tan\\_conjugate\(\)](#), [tan\\_deriv\(\)](#), [tan\\_eval\(\)](#), [tan\\_evalf\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

### 5.1.3.608 `asin()`

```
const numeric GiNaC::asin (
    const numeric & x )
```

Numeric inverse sine (trigonometric function).

#### Returns

arbitrary precision numerical  $\arcsin(x)$ .

References [x](#).

Referenced by [asin\\_conjugate\(\)](#), [asin\\_eval\(\)](#), [asin\\_evalf\(\)](#), [cos\\_eval\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**5.1.3.609 acos()**

```
const numeric GiNaC::acos (
    const numeric & x )
```

Numeric inverse cosine (trigonometric function).

**Returns**

arbitrary precision numerical acos(x).

References [x](#).

Referenced by [acos\\_conjugate\(\)](#), [acos\\_eval\(\)](#), [acos\\_evalf\(\)](#), [cos\\_eval\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**5.1.3.610 atan() [1/2]**

```
const numeric GiNaC::atan (
    const numeric & x )
```

Numeric arcustangent.

**Parameters**

<i>x</i>	complex number
----------	----------------

**Returns**

atan(x)

**Exceptions**

<i>pole_error("atan()"</i>	<i>logarithmic pole",0)</i> if $x=1$ or $x=-1$ .
----------------------------	--

References [\\_num1\\_p](#), [abs\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

Referenced by [atan2\\_eval\(\)](#), [atan2\\_evalf\(\)](#), [atan\\_conjugate\(\)](#), [atan\\_eval\(\)](#), [atan\\_evalf\(\)](#), [atan\\_series\(\)](#), [cos\\_eval\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**5.1.3.611 atan() [2/2]**

```
const numeric GiNaC::atan (
    const numeric & y,
    const numeric & x )
```

Numeric arcustangent of two arguments, analytically continued in a suitable way.

**Parameters**

<i>y</i>	complex number
<i>x</i>	complex number

**Returns**

$-i \cdot \log((x + i \cdot y) / \sqrt{x^2 + y^2})$ , which is equal to  $\operatorname{atan}(y/x)$  if  $y$  and  $x$  are both real.

**Exceptions**

<i>pole_error("atan()", logarithmic pole", 0)</i>	if $y/x == +i$ or $y/x == -i$ .
---	---------------------------------

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::numeric::to\\_cl\\_N\(\)](#), and [x](#).

**5.1.3.612 sinh()**

```
const numeric GiNaC::sinh (
    const numeric & x )
```

Numeric hyperbolic sine (trigonometric function).

**Returns**

arbitrary precision numerical  $\sinh(x)$ .

References [x](#).

Referenced by [cos\\_imag\\_part\(\)](#), [cosh\\_deriv\(\)](#), [cosh\\_imag\\_part\(\)](#), [sin\\_imag\\_part\(\)](#), [sinh\\_conjugate\(\)](#), [sinh\\_eval\(\)](#), [sinh\\_evalf\(\)](#), [sinh\\_real\\_part\(\)](#), and [tanh\\_series\(\)](#).

**5.1.3.613 cosh()**

```
const numeric GiNaC::cosh (
    const numeric & x )
```

Numeric hyperbolic cosine (trigonometric function).

**Returns**

arbitrary precision numerical  $\cosh(x)$ .

References [x](#).

Referenced by [cos\\_real\\_part\(\)](#), [cosh\\_conjugate\(\)](#), [cosh\\_eval\(\)](#), [cosh\\_evalf\(\)](#), [cosh\\_real\\_part\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_deriv\(\)](#), [sinh\\_imag\\_part\(\)](#), and [tanh\\_series\(\)](#).

#### 5.1.3.614 `tanh()`

```
const numeric GiNaC::tanh (  
    const numeric & x )
```

Numeric hyperbolic tangent (trigonometric function).

##### Returns

arbitrary precision numerical  $\tanh(x)$ .

References [x](#).

Referenced by [tan\\_imag\\_part\(\)](#), [tanh\\_conjugate\(\)](#), [tanh\\_deriv\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_evalf\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

#### 5.1.3.615 `asinh()`

```
const numeric GiNaC::asinh (  
    const numeric & x )
```

Numeric inverse hyperbolic sine (trigonometric function).

##### Returns

arbitrary precision numerical  $\operatorname{asinh}(x)$ .

References [x](#).

Referenced by [asinh\\_conjugate\(\)](#), [asinh\\_eval\(\)](#), [asinh\\_evalf\(\)](#), [cosh\\_eval\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

#### 5.1.3.616 `acosh()`

```
const numeric GiNaC::acosh (  
    const numeric & x )
```

Numeric inverse hyperbolic cosine (trigonometric function).

##### Returns

arbitrary precision numerical  $\operatorname{acosh}(x)$ .

References [x](#).

Referenced by [acosh\\_conjugate\(\)](#), [acosh\\_eval\(\)](#), [acosh\\_evalf\(\)](#), [cosh\\_eval\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

**5.1.3.617 atanh()**

```
const numeric GiNaC::atanh (
    const numeric & x )
```

Numeric inverse hyperbolic tangent (trigonometric function).

**Returns**

arbitrary precision numerical atanh(x).

References [x](#).

Referenced by [atanh\\_conjugate\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_evalf\(\)](#), [atanh\\_series\(\)](#), [cosh\\_eval\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

**5.1.3.618 Li2\_series()** [2/2]

```
static cln::cl_N GiNaC::Li2_series (
    const cln::cl_N & x,
    const cln::float_format_t & prec ) [static]
```

Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.

References [x](#).

**5.1.3.619 Li2\_projection()**

```
static cln::cl_N GiNaC::Li2_projection (
    const cln::cl_N & x,
    const cln::float_format_t & prec ) [static]
```

Folds Li2's argument inside a small rectangle to enhance convergence.

References [Li2\\_projection\(\)](#), [Li2\\_series\(\)](#), and [x](#).

Referenced by [Li2\\_\(\)](#), and [Li2\\_projection\(\)](#).

### 5.1.3.620 `Li2_()`

```
const cln::cl_N GiNaC::Li2_ (
    const cln::cl_N & value )
```

Numeric evaluation of Dilogarithm.

The domain is the entire complex plane, the branch cut lies along the positive real axis, starting at 1 and continuous with quadrant IV.

#### Returns

arbitrary precision numerical  $\text{Li}_2(x)$ .

References [Li2\\_projection\(\)](#), and [value](#).

Referenced by [Li2\(\)](#).

### 5.1.3.621 `Li2()`

```
const numeric GiNaC::Li2 (
    const numeric & x )
```

References [\\_num0\\_p](#), [Li2\\_\(\)](#), and [x](#).

Referenced by [Li2\\_conjugate\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_evalf\(\)](#), and [Li2\\_series\(\)](#).

### 5.1.3.622 `zeta()` [3/3]

```
const numeric GiNaC::zeta (
    const numeric & x )
```

Numeric evaluation of Riemann's Zeta function.

Currently works only for integer arguments.

References [x](#).

### 5.1.3.623 `guess_precision()`

```
static cln::float_format_t GiNaC::guess_precision (
    const cln::cl_N & x ) [static]
```

References [x](#).

Referenced by [lgamma\(\)](#), and [tgamma\(\)](#).

**5.1.3.624 lgamma()** [1/2]

```
const cln::cl_N GiNaC::lgamma (
    const cln::cl_N & x )
```

The Gamma function.

Use the Lanczos approximation. If the coefficients used here are not sufficiently many or sufficiently accurate, more can be calculated using the program `doc/examples/lanczos.cpp`. In that case, be sure to read the comments in that file.

References [GiNaC::lanczos\\_coeffs::calc\\_lanczos\\_A\(\)](#), [GiNaC::lanczos\\_coeffs::get\\_order\(\)](#), [guess\\_precision\(\)](#), [lgamma\(\)](#), [log\(\)](#), [sin\(\)](#), [GiNaC::lanczos\\_coeffs::sufficiently\\_accurate\(\)](#), and [x](#).

Referenced by [beta\\_evalf\(\)](#), [lgamma\(\)](#), [lgamma\\_conjugate\(\)](#), [lgamma\\_eval\(\)](#), [lgamma\\_evalf\(\)](#), and [lgamma\\_series\(\)](#).

**5.1.3.625 lgamma()** [2/2]

```
const numeric GiNaC::lgamma (
    const numeric & x )
```

References [lgamma\(\)](#), and [x](#).

**5.1.3.626 tgamma()** [1/2]

```
const cln::cl_N GiNaC::tgamma (
    const cln::cl_N & x )
```

References [GiNaC::lanczos\\_coeffs::calc\\_lanczos\\_A\(\)](#), [exp\(\)](#), [GiNaC::lanczos\\_coeffs::get\\_order\(\)](#), [guess\\_precision\(\)](#), [sqrt\(\)](#), [GiNaC::lanczos\\_coeffs::sufficiently\\_accurate\(\)](#), [tgamma\(\)](#), and [x](#).

Referenced by [beta\\_eval\(\)](#), [beta\\_series\(\)](#), [psi2\\_eval\(\)](#), [tgamma\(\)](#), [tgamma\\_conjugate\(\)](#), [tgamma\\_deriv\(\)](#), [tgamma\\_eval\(\)](#), [tgamma\\_evalf\(\)](#), and [tgamma\\_series\(\)](#).

**5.1.3.627 tgamma()** [2/2]

```
const numeric GiNaC::tgamma (
    const numeric & x )
```

References [tgamma\(\)](#), and [x](#).



**5.1.3.628 psi()** [3/4]

```
const numeric GiNaC::psi (
    const numeric & x )
```

The psi function (aka polygamma function).

This is only a stub!

**5.1.3.629 psi()** [4/4]

```
const numeric GiNaC::psi (
    const numeric & n,
    const numeric & x )
```

The psi functions (aka polygamma functions).

This is only a stub!

**5.1.3.630 factorial()**

```
const numeric GiNaC::factorial (
    const numeric & n )
```

Factorial combinatorial function.

**Parameters**

<i>n</i>	integer argument $\geq 0$
----------	---------------------------

**Exceptions**

<i>range_error</i>	(argument must be integer $\geq 0$ )
--------------------	--------------------------------------

References [n](#).

Referenced by [Bernoulli\\_polynomial\(\)](#), [factorial\\_conjugate\(\)](#), [factorial\\_eval\(\)](#), [factorial\\_evalf\(\)](#), [factorial\\_real\\_part\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [H\\_eval\(\)](#), [lgamma\\_eval\(\)](#), [multinomial\\_coefficient\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_series\(\)](#), [S\\_eval\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), [symm\(\)](#), [tgamma\\_eval\(\)](#), and [zeta1\\_eval\(\)](#).

**5.1.3.631 doublefactorial()**

```
const numeric GiNaC::doublefactorial (
    const numeric & n )
```

The double factorial combinatorial function.

(Scarcely used, but still useful in cases, like for exact results of  $tgamma(n+1/2)$  for instance.)

**Parameters**

<i>n</i>	integer argument $\geq -1$
----------	----------------------------

**Returns**

$n!! == n * (n-2) * (n-4) * \dots * (\{1|2\})$  with  $0!! == (-1)!! == 1$

**Exceptions**

<i>range_error</i>	(argument must be integer $\geq -1$ )
--------------------	---------------------------------------

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), and [n](#).

Referenced by [tgamma\\_eval\(\)](#).

**5.1.3.632 binomial()**

```
const numeric GiNaC::binomial (
    const numeric & n,
    const numeric & k )
```

The Binomial coefficients.

It computes the binomial coefficients. For integer  $n$  and  $k$  and positive  $n$  this is the number of ways of choosing  $k$  objects from  $n$  distinct objects. If  $n$  is a negative integer, the formula  $\text{binomial}(n,k) == (-1)^k * \text{binomial}(k-n-1,k)$  (if  $k \geq 0$ )  $\text{binomial}(n,k) == (-1)^{(n-k)} * \text{binomial}(-k-1,n-k)$  (otherwise) is used to compute the result.

References [\\_num0\\_p](#), [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [binomial\(\)](#), [k](#), [n](#), and [GiNaC::numeric::power\(\)](#).

Referenced by [binomial\(\)](#), [binomial\\_conjugate\(\)](#), [binomial\\_eval\(\)](#), [binomial\\_evalf\(\)](#), [binomial\\_real\\_part\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::imag\\_part\(\)](#), and [GiNaC::power::real\\_part\(\)](#).

**5.1.3.633 bernoulli()**

```
const numeric GiNaC::bernoulli (
    const numeric & nn )
```

Bernoulli number.

The  $n$ th Bernoulli number is the coefficient of  $x^n/n!$  in the expansion of the function  $x/(e^x-1)$ .

**Returns**

the  $n$ th Bernoulli number (a rational number).

## Exceptions

<code>range_error</code>	(argument must be integer $\geq 0$ )
--------------------------	--------------------------------------

References [\\_num1\\_p](#), [c](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [k](#), [n](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), and [zeta1\\_eval\(\)](#).

5.1.3.634 **fibonacci()**

```
const numeric GiNaC::fibonacci (
    const numeric & n )
```

Fibonacci number.

The nth Fibonacci number  $F(n)$  is defined by the recurrence formula  $F(n) = F(n-1) + F(n-2)$  with  $F(0) = 0$  and  $F(1) = 1$ .

## Parameters

$n$	an integer
-----	------------

## Returns

the nth Fibonacci number  $F(n)$  (an integer number)

## Exceptions

<code>range_error</code>	(argument must be an integer)
--------------------------	-------------------------------

References [\\_num0\\_p](#), [fibonacci\(\)](#), [m](#), and [n](#).

Referenced by [fibonacci\(\)](#).

5.1.3.635 **abs()**

```
const numeric GiNaC::abs (
    const numeric & x )
```

Absolute value.

References [x](#).

Referenced by [abs\\_conjugate\(\)](#), [abs\\_eval\(\)](#), [abs\\_evalf\(\)](#), [abs\\_expand\(\)](#), [abs\\_expl\\_derivative\(\)](#), [abs\\_power\(\)](#), [abs\\_real\\_part\(\)](#), [adaptivesimpson\(\)](#), [atan\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [GiNaC::numeric::integer\\_content\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\(\)](#), [log\\_real\\_part\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [GiNaC::numeric::max\\_coefficient\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::power::real\\_part\(\)](#), [tgamma\\_eval\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), and [zeta1\\_eval\(\)](#).

**5.1.3.636 mod()**

```
const numeric GiNaC::mod (
    const numeric & a,
    const numeric & b )
```

Modulus (in positive representation).

In general, `mod(a,b)` has the sign of `b` or is zero, and `rem(a,b)` has the sign of `a` or is zero. This is different from Maple's `modp`, where the sign of `b` is ignored. It is in agreement with Mathematica's `Mod`.

**Returns**

`a mod b` in the range `[0,abs(b)-1]` with sign of `b` if both are integer, 0 otherwise.

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [cos\\_eval\(\)](#), [exp\\_eval\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**5.1.3.637 smod()**

```
const numeric GiNaC::smod (
    const numeric & a_,
    const numeric & b_ )
```

Modulus (in symmetric representation).

**Returns**

`a mod b` in the range `[-iquo(abs(b),2), iquo(abs(b),2)]`.

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [m](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), and [GiNaC::numeric::smod\(\)](#).

**5.1.3.638 irem() [1/2]**

```
const numeric GiNaC::irem (
    const numeric & a,
    const numeric & b )
```

Numeric integer remainder.

Equivalent to Maple's `irem(a,b)` as far as sign conventions are concerned. In general, `mod(a,b)` has the sign of `b` or is zero, and `irem(a,b)` has the sign of `a` or is zero.

**Returns**

remainder of `a/b` if both are integer, 0 otherwise.

## Exceptions

<code>overflow_error</code>	(division by zero) if b is zero.
-----------------------------	----------------------------------

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), and [ifactor\(\)](#).

**5.1.3.639 irem()** [2/2]

```
const numeric GiNaC::irem (
    const numeric & a,
    const numeric & b,
    numeric & q )
```

Numeric integer remainder.

Equivalent to Maple's `irem(a,b,'q')` it obeys the relation `irem(a,b,q) == a - q*b`. In general, `mod(a,b)` has the sign of b or is zero, and `irem(a,b)` has the sign of a or is zero.

## Returns

remainder of a/b and quotient stored in q if both are integer, 0 otherwise.

## Exceptions

<code>overflow_error</code>	(division by zero) if b is zero.
-----------------------------	----------------------------------

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**5.1.3.640 iquo()** [1/2]

```
const numeric GiNaC::iquo (
    const numeric & a,
    const numeric & b )
```

Numeric integer quotient.

Equivalent to Maple's `iquo` as far as sign conventions are concerned.

## Returns

truncated quotient of a/b if both are integer, 0 otherwise.

## Exceptions

<code>overflow_error</code>	(division by zero) if b is zero.
-----------------------------	----------------------------------

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::power::eval\(\)](#), and [heur\\_gcd\\_z\(\)](#).

**5.1.3.641 iquo()** [2/2]

```
const numeric GiNaC::iquo (
    const numeric & a,
    const numeric & b,
    numeric & r )
```

Numeric integer quotient.

Equivalent to Maple's `iquo(a,b,'r')` it obeys the relation `r == a - iquo(a,b,r)*b`.

## Returns

truncated quotient of `a/b` and remainder stored in `r` if both are integer, 0 otherwise.

## Exceptions

<code>overflow_error</code>	(division by zero) if b is zero.
-----------------------------	----------------------------------

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), `r`, and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**5.1.3.642 gcd()** [2/2]

```
const numeric GiNaC::gcd (
    const numeric & a,
    const numeric & b )
```

Greatest Common Divisor.

## Returns

The GCD of two numbers if both are integer, a numerical 1 if they are not.

References [\\_num1\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**5.1.3.643 lcm()** [2/2]

```
const numeric GiNaC::lcm (
    const numeric & a,
    const numeric & b )
```

Least Common Multiple.

**Returns**

The LCM of two numbers if both are integer, the product of those two numbers if they are not.

References [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::mul\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**5.1.3.644 sqrt()** [1/2]

```
const numeric GiNaC::sqrt (
    const numeric & x )
```

Numeric square root.

If possible, sqrt(x) should respect squares of exact numbers, i.e. sqrt(4) should return integer 2.

**Parameters**

$x$	numeric argument
-----	------------------

**Returns**

square root of  $x$ . Branch cut along negative real axis, the negative real axis itself where  $\text{imag}(x)=0$  and  $\text{real}(x)<0$  belongs to the upper part where  $\text{imag}(x)>0$ .

References [x](#).

Referenced by [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [EllipticE\\_evalf\(\)](#), [EllipticK\\_evalf\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [tgamma\(\)](#), and [tgamma\\_eval\(\)](#).

**5.1.3.645 isqrt()**

```
const numeric GiNaC::isqrt (
    const numeric & x )
```

Integer numeric square root.

References [\\_num0\\_p](#), and [x](#).

Referenced by [heur\\_gcd\\_z\(\)](#).

#### 5.1.3.646 `PiEvalf()`

```
ex GiNaC::PiEvalf ( )
```

Floating point evaluation of Archimedes' constant Pi.

#### 5.1.3.647 `EulerEvalf()`

```
ex GiNaC::EulerEvalf ( )
```

Floating point evaluation of Euler's constant gamma.

#### 5.1.3.648 `CatalanEvalf()`

```
ex GiNaC::CatalanEvalf ( )
```

Floating point evaluation of Catalan's constant.

#### 5.1.3.649 `operator<<()` [6/16]

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const _numeric_digits & e )
```

References [GiNaC::\\_numeric\\_digits::print\(\)](#).

#### 5.1.3.650 `GINAC_DECLARE_UNARCHIVER()` [38/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    numeric )
```



**5.1.3.651 pow()** [1/3]

```
const numeric GiNaC::pow (
    const numeric & x,
    const numeric & y ) [inline]
```

References [x](#).

Referenced by [abs\\_eval\(\)](#), [abs\\_power\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [beta\\_eval\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_and\\_derivative\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::power::derivative\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::evalm\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [factor\(\)](#), [find\\_common\\_factor\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::get\\_numerical\\_value\(\)](#), [H\\_eval\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [interpolate\(\)](#), [kronecker\\_symbol\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [lcmcoeff\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [Li\\_series\(\)](#), [log\\_series\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::op\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [prem\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::q\\_expansion\(\)](#), [quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [rem\(\)](#), [replace\\_with\\_symbol\(\)](#), [S\\_eval\(\)](#), [S\\_series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), [sprem\(\)](#), [sqrfree\\_parfrac\(\)](#), [sr\\_gcd\(\)](#), [GiNaC::power::subs\(\)](#), [tgamma\\_eval\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::power::to\\_rational\(\)](#), and [zeta1\\_eval\(\)](#).

**5.1.3.652 inverse()** [3/3]

```
const numeric GiNaC::inverse (
    const numeric & x ) [inline]
```

References [x](#).

**5.1.3.653 step()**

```
numeric GiNaC::step (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [abs\\_eval\(\)](#), [H\\_eval\(\)](#), [step\\_conjugate\(\)](#), [step\\_eval\(\)](#), [step\\_evalf\(\)](#), [step\\_real\\_part\(\)](#), and [step\\_series\(\)](#).

**5.1.3.654 csgn()**

```
int GiNaC::csgn (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [csgn\\_conjugate\(\)](#), [csgn\\_eval\(\)](#), [csgn\\_evalf\(\)](#), [csgn\\_power\(\)](#), [csgn\\_real\\_part\(\)](#), [csgn\\_series\(\)](#), [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), and [log\\_series\(\)](#).

#### 5.1.3.655 `is_zero()` [2/2]

```
bool GiNaC::is_zero (
    const numeric & x ) [inline]
```

References [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

#### 5.1.3.656 `is_positive()`

```
bool GiNaC::is_positive (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [beta\\_eval\(\)](#).

#### 5.1.3.657 `is_negative()`

```
bool GiNaC::is_negative (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::power::eval\(\)](#), and [frac\\_cancel\(\)](#).

#### 5.1.3.658 `is_integer()`

```
bool GiNaC::is_integer (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [beta\\_eval\(\)](#), [binomial\\_eval\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::lddegree\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [replace\\_with\\_symbol\(\)](#), and [GiNaC::power::series\(\)](#).

#### 5.1.3.659 `is_pos_integer()`

```
bool GiNaC::is_pos_integer (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#), and [GiNaC::power::expand\\_add\\_2\(\)](#).

**5.1.3.660 is\_nonneg\_integer()**

```
bool GiNaC::is_nonneg_integer (
    const numeric & x ) [inline]
```

References [x](#).

**5.1.3.661 is\_even()**

```
bool GiNaC::is_even (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [abs\\_power\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

**5.1.3.662 is\_odd()**

```
bool GiNaC::is_odd (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [csgn\\_power\(\)](#).

**5.1.3.663 is\_prime()**

```
bool GiNaC::is_prime (
    const numeric & x ) [inline]
```

References [x](#).

**5.1.3.664 is\_rational()**

```
bool GiNaC::is_rational (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [beta\\_eval\(\)](#), [lgamma\\_eval\(\)](#), [replace\\_with\\_symbol\(\)](#), and [tgamma\\_eval\(\)](#).

#### 5.1.3.665 `is_real()`

```
bool GiNaC::is_real (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [beta\\_eval\(\)](#), [fsolve\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), and [Li2\\_series\(\)](#).

#### 5.1.3.666 `is_cinteger()`

```
bool GiNaC::is_cinteger (
    const numeric & x ) [inline]
```

References [x](#).

#### 5.1.3.667 `is_crational()`

```
bool GiNaC::is_crational (
    const numeric & x ) [inline]
```

References [x](#).

#### 5.1.3.668 `to_int()`

```
int GiNaC::to_int (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [color\\_trace\(\)](#), [H\\_evalf\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [S\\_eval\(\)](#), [GiNaC::power::series\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.669 `to_long()`

```
long GiNaC::to_long (
    const numeric & x ) [inline]
```

References [x](#).

#### 5.1.3.670 to\_double()

```
double GiNaC::to_double (
    const numeric & x ) [inline]
```

References [x](#).

#### 5.1.3.671 real()

```
const numeric GiNaC::real (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [cosh\\_eval\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

#### 5.1.3.672 imag()

```
const numeric GiNaC::imag (
    const numeric & x ) [inline]
```

References [x](#).

Referenced by [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [G2\\_eval\(\)](#), and [G2\\_evalf\(\)](#).

#### 5.1.3.673 numer() [2/2]

```
const numeric GiNaC::numer (
    const numeric & x ) [inline]
```

References [GiNaC::ex::numer\(\)](#), and [x](#).

#### 5.1.3.674 denom() [2/2]

```
const numeric GiNaC::denom (
    const numeric & x ) [inline]
```

References [GiNaC::ex::denom\(\)](#), and [x](#).

#### 5.1.3.675 `exadd()`

```
static const ex GiNaC::exadd (
    const ex & lh,
    const ex & rh ) [inline], [static]
```

Used internally by [operator+\(\)](#) to add two `ex` objects.

Referenced by [operator+\(\)](#), [operator++\(\)](#), [operator+=\(\)](#), [operator-\(\)](#), [operator--\(\)](#), and [operator-=\(\)](#).

#### 5.1.3.676 `exmul()`

```
static const ex GiNaC::exmul (
    const ex & lh,
    const ex & rh ) [inline], [static]
```

Used internally by [operator\\*\(\)](#) to multiply two `ex` objects.

References [GiNaC::return\\_types::commutative](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [operator\\*\(\)](#), [operator\\*=\(\(\)\)](#), [operator/\(\)](#), and [operator/=\(\(\)\)](#).

#### 5.1.3.677 `exminus()`

```
static const ex GiNaC::exminus (
    const ex & lh ) [inline], [static]
```

Used internally by [operator-\(\)](#) and friends to change the sign of an argument.

References [\\_ex\\_1](#).

Referenced by [operator-\(\)](#), and [operator-=\(\)](#).

#### 5.1.3.678 `operator+()` [1/4]

```
const ex GiNaC::operator+ (
    const ex & lh,
    const ex & rh )
```

References [exadd\(\)](#).

**5.1.3.679 operator-() [1/4]**

```
const ex GiNaC::operator- (
    const ex & lh,
    const ex & rh )
```

References [exadd\(\)](#), and [exminus\(\)](#).

**5.1.3.680 operator\*() [1/2]**

```
const ex GiNaC::operator* (
    const ex & lh,
    const ex & rh )
```

References [exmul\(\)](#).

**5.1.3.681 operator/() [1/2]**

```
const ex GiNaC::operator/ (
    const ex & lh,
    const ex & rh )
```

References [\\_ex\\_1](#), and [exmul\(\)](#).

**5.1.3.682 operator+() [2/4]**

```
const numeric GiNaC::operator+ (
    const numeric & lh,
    const numeric & rh )
```

References [GiNaC::numeric::add\(\)](#).

**5.1.3.683 operator-() [2/4]**

```
const numeric GiNaC::operator- (
    const numeric & lh,
    const numeric & rh )
```

References [GiNaC::numeric::sub\(\)](#).

**5.1.3.684 operator\*()** [2/2]

```
const numeric GiNaC::operator* (
    const numeric & lh,
    const numeric & rh )
```

References [GiNaC::numeric::mul\(\)](#).

**5.1.3.685 operator/()** [2/2]

```
const numeric GiNaC::operator/ (
    const numeric & lh,
    const numeric & rh )
```

References [GiNaC::numeric::div\(\)](#).

**5.1.3.686 operator+=()** [1/2]

```
ex & GiNaC::operator+= (
    ex & lh,
    const ex & rh )
```

References [exadd\(\)](#).

**5.1.3.687 operator-=()** [1/2]

```
ex & GiNaC::operator-= (
    ex & lh,
    const ex & rh )
```

References [exadd\(\)](#), and [exminus\(\)](#).

**5.1.3.688 operator\*=( )** [1/2]

```
ex & GiNaC::operator*= (
    ex & lh,
    const ex & rh )
```

References [exmul\(\)](#).



**5.1.3.689 operator/=( ) [1/2]**

```
ex & GiNaC::operator/= (
    ex & lh,
    const ex & rh )
```

References [\\_ex\\_1](#), and [exmul\(\)](#).

**5.1.3.690 operator+=( ) [2/2]**

```
numeric & GiNaC::operator+= (
    numeric & lh,
    const numeric & rh )
```

References [GiNaC::numeric::add\(\)](#).

**5.1.3.691 operator-=( ) [2/2]**

```
numeric & GiNaC::operator-= (
    numeric & lh,
    const numeric & rh )
```

References [GiNaC::numeric::sub\(\)](#).

**5.1.3.692 operator\*=( ) [2/2]**

```
numeric & GiNaC::operator*= (
    numeric & lh,
    const numeric & rh )
```

References [GiNaC::numeric::mul\(\)](#).

**5.1.3.693 operator/=( ) [2/2]**

```
numeric & GiNaC::operator/= (
    numeric & lh,
    const numeric & rh )
```

References [GiNaC::numeric::div\(\)](#).

**5.1.3.694 operator+()** [3/4]

```
const ex GiNaC::operator+ (
    const ex & lh )
```

**5.1.3.695 operator-()** [3/4]

```
const ex GiNaC::operator- (
    const ex & lh )
```

References [exminus\(\)](#).

**5.1.3.696 operator+()** [4/4]

```
const numeric GiNaC::operator+ (
    const numeric & lh )
```

**5.1.3.697 operator-()** [4/4]

```
const numeric GiNaC::operator- (
    const numeric & lh )
```

References [\\_num\\_1\\_p](#), and [GiNaC::numeric::mul\(\)](#).

**5.1.3.698 operator++()** [1/4]

```
ex & GiNaC::operator++ (
    ex & rh )
```

Expression prefix increment.

Adds 1 and returns incremented *ex*.

References [\\_ex1](#), and [exadd\(\)](#).

**5.1.3.699 operator--()** [1/4]

```
ex & GiNaC::operator-- (
    ex & rh )
```

Expression prefix decrement.

Subtracts 1 and returns decremented ex.

References [\\_ex\\_1](#), and [exadd\(\)](#).

**5.1.3.700 operator++()** [2/4]

```
const ex GiNaC::operator++ (
    ex & lh,
    int )
```

Expression postfix increment.

Returns the ex and leaves the original incremented by 1.

References [\\_ex1](#), and [exadd\(\)](#).

**5.1.3.701 operator--()** [2/4]

```
const ex GiNaC::operator-- (
    ex & lh,
    int )
```

Expression postfix decrement.

Returns the ex and leaves the original decremented by 1.

References [\\_ex\\_1](#), and [exadd\(\)](#).

**5.1.3.702 operator++()** [3/4]

```
numeric & GiNaC::operator++ (
    numeric & rh )
```

Numeric prefix increment.

Adds 1 and returns incremented number.

References [\\_num1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**5.1.3.703 operator--()** [3/4]

```
numeric & GiNaC::operator-- (
    numeric & rh )
```

Numeric prefix decrement.

Subtracts 1 and returns decremented number.

References [\\_num\\_1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**5.1.3.704 operator++()** [4/4]

```
const numeric GiNaC::operator++ (
    numeric & lh,
    int )
```

Numeric postfix increment.

Returns the number and leaves the original incremented by 1.

References [\\_num1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**5.1.3.705 operator--()** [4/4]

```
const numeric GiNaC::operator-- (
    numeric & lh,
    int )
```

Numeric postfix decrement.

Returns the number and leaves the original decremented by 1.

References [\\_num\\_1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**5.1.3.706 operator==()**

```
const relational GiNaC::operator== (
    const ex & lh,
    const ex & rh )
```

References [GiNaC::relational::equal](#).

#### 5.1.3.707 operator"!=()

```
const relational GiNaC::operator!= (
    const ex & lh,
    const ex & rh )
```

References [GiNaC::relational::not\\_equal](#).

#### 5.1.3.708 operator<()

```
const relational GiNaC::operator< (
    const ex & lh,
    const ex & rh )
```

References [GiNaC::relational::less](#).

#### 5.1.3.709 operator<=()

```
const relational GiNaC::operator<= (
    const ex & lh,
    const ex & rh )
```

References [GiNaC::relational::less\\_or\\_equal](#).

#### 5.1.3.710 operator>()

```
const relational GiNaC::operator> (
    const ex & lh,
    const ex & rh )
```

References [GiNaC::relational::greater](#).

#### 5.1.3.711 operator>=()

```
const relational GiNaC::operator>= (
    const ex & lh,
    const ex & rh )
```

References [GiNaC::relational::greater\\_or\\_equal](#).

#### 5.1.3.712 `my_ios_index()`

```
static int GiNaC::my_ios_index ( ) [static]
```

Referenced by [get\\_print\\_context\(\)](#), and [set\\_print\\_context\(\)](#).

#### 5.1.3.713 `my_ios_callback()`

```
static void GiNaC::my_ios_callback (
    std::ios_base::event ev,
    std::ios_base & s,
    int i ) [static]
```

Referenced by [set\\_print\\_context\(\)](#).

#### 5.1.3.714 `get_print_context()`

```
static print\_context * GiNaC::get_print_context (
    std::ios_base & s ) [inline], [static]
```

References [my\\_ios\\_index\(\)](#).

Referenced by [get\\_print\\_options\(\)](#), [operator<<\(\)](#), and [set\\_print\\_options\(\)](#).

#### 5.1.3.715 `set_print_context()`

```
static void GiNaC::set_print_context (
    std::ios_base & s,
    const print\_context & c ) [static]
```

References [c](#), [callback\\_registered](#), [my\\_ios\\_callback\(\)](#), [my\\_ios\\_index\(\)](#), [options](#), and [GiNaC::print\\_context::options](#).

Referenced by [csrc\(\)](#), [csrc\\_cl\\_N\(\)](#), [csrc\\_double\(\)](#), [csrc\\_float\(\)](#), [dflt\(\)](#), [latex\(\)](#), [python\(\)](#), [python\\_repr\(\)](#), [set\\_print\\_options\(\)](#), and [tree\(\)](#).

#### 5.1.3.716 `get_print_options()`

```
static unsigned GiNaC::get_print_options (
    std::ios_base & s ) [inline], [static]
```

References [get\\_print\\_context\(\)](#), and [GiNaC::print\\_context::options](#).

Referenced by [index\\_dimensions\(\)](#), and [no\\_index\\_dimensions\(\)](#).

**5.1.3.717 set\_print\_options()**

```
static void GiNaC::set_print_options (
    std::ostream & s,
    unsigned options ) [static]
```

References [get\\_print\\_context\(\)](#), [options](#), [GiNaC::print\\_context::options](#), and [set\\_print\\_context\(\)](#).

Referenced by [dflt\(\)](#), [index\\_dimensions\(\)](#), and [no\\_index\\_dimensions\(\)](#).

**5.1.3.718 operator<<() [7/16]**

```
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const ex & e )
```

References [get\\_print\\_context\(\)](#), and [GiNaC::ex::print\(\)](#).

**5.1.3.719 operator>>() [3/3]**

```
std::istream & GiNaC::operator>> (
    std::istream & is,
    ex & e )
```

**5.1.3.720 dflt()**

```
std::ostream & GiNaC::dflt (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#), and [set\\_print\\_options\(\)](#).

**5.1.3.721 latex()**

```
std::ostream & GiNaC::latex (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

#### 5.1.3.722 python()

```
std::ostream & GiNaC::python (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

#### 5.1.3.723 python\_repr()

```
std::ostream & GiNaC::python_repr (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

#### 5.1.3.724 tree()

```
std::ostream & GiNaC::tree (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

Referenced by [GiNaC::class\\_info< OPT >::dump\\_hierarchy\(\)](#).

#### 5.1.3.725 csrc()

```
std::ostream & GiNaC::csrc (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

#### 5.1.3.726 csrc\_float()

```
std::ostream & GiNaC::csrc_float (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).



**5.1.3.727 csrc\_double()**

```
std::ostream & GiNaC::csrc_double (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

**5.1.3.728 csrc\_cl\_N()**

```
std::ostream & GiNaC::csrc_cl_N (
    std::ostream & os )
```

References [set\\_print\\_context\(\)](#).

**5.1.3.729 index\_dimensions()**

```
std::ostream & GiNaC::index_dimensions (
    std::ostream & os )
```

References [get\\_print\\_options\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [set\\_print\\_options\(\)](#).

**5.1.3.730 no\_index\_dimensions()**

```
std::ostream & GiNaC::no_index_dimensions (
    std::ostream & os )
```

References [get\\_print\\_options\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [set\\_print\\_options\(\)](#).

**5.1.3.731 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [27/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    power ,
    basic ,
    print_func< print_dflt > &::do_print_dflt. print_func< print_latex > &::do_↵
_print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_python > &↵
::do_print_python. print_func< print_python_repr > &::do_print_python_repr. print_func<
print_csrc_cl_N > &::do_print_csrc_cl_N )
```

**5.1.3.732 print\_sym\_pow()**

```
static void GiNaC::print_sym_pow (
    const print_context & c,
    const symbol & x,
    int exp ) [static]
```

References [c](#), [exp\(\)](#), [GiNaC::ex::print\(\)](#), [print\\_sym\\_pow\(\)](#), and [x](#).

Referenced by [GiNaC::power::do\\_print\\_csrc\(\)](#), and [print\\_sym\\_pow\(\)](#).

**5.1.3.733 GINAC\_BIND\_UNARCHIVER() [37/49]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
    power )
```

**5.1.3.734 GINAC\_DECLARE\_UNARCHIVER() [39/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    power )
```

**5.1.3.735 pow() [2/3]**

```
ex GiNaC::pow (
    const ex & b,
    const ex & e ) [inline]
```

Symbolic exponentiation.

Returns a power-object as a new expression.

**Parameters**

<i>b</i>	the basis expression
<i>e</i>	the exponent expression

**5.1.3.736 pow() [3/3]**

```
template<typename T1 , typename T2 >
ex GiNaC::pow (
```

```
const T1 & b,
const T2 & e ) [inline]
```

### 5.1.3.737 `sqrt()` [2/2]

```
ex GiNaC::sqrt (
    const ex & a ) [inline]
```

Square root expression.

Returns a power-object with exponent 1/2.

References [\\_ex1\\_2](#).

### 5.1.3.738 `is_a()` [3/3]

```
template<class T >
bool GiNaC::is_a (
    const print_context & obj ) [inline]
```

Check if obj is a T, including base classes.

### 5.1.3.739 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [28/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    pseries ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python > &::do_↵
print_python. print_func< print_python_repr > &::do_print_python_repr )
```

### 5.1.3.740 `GINAC_BIND_UNARCHIVER()` [38/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    pseries )
```

### 5.1.3.741 `GINAC_DECLARE_UNARCHIVER()` [40/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    pseries )
```

### 5.1.3.742 `series_to_poly()`

```
ex GiNaC::series_to_poly (
    const ex & e ) [inline]
```

Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.

The result is undefined if the expression does not contain a pseries object at its top level.

## Parameters

<i>e</i>	expression
----------	------------

## Returns

polynomial expression

## See also

[is\\_a<>](#)

[pseries::convert\\_to\\_poly](#)

Referenced by [Bernoulli\\_polynomial\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), and [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#).

**5.1.3.743 is\_terminating()**

```
bool GiNaC::is_terminating (
    const pseries & s ) [inline]
```

References [GiNaC::pseries::is\\_terminating\(\)](#).

**5.1.3.744 make\_return\_type\_t()**

```
template<typename T >
return_type_t GiNaC::make_return_type_t (
    const unsigned rl = 0 ) [inline]
```

References [GiNaC::return\\_type\\_t::rl](#), and [GiNaC::return\\_type\\_t::tinfo](#).

**5.1.3.745 set\_print\_func()** [1/2]

```
template<class Alg , class Ctx , class T , class C >
void GiNaC::set_print_func (
    void fconst T &, const C &c, unsigned )
```

Add or replace a print method.

References [options](#).

**5.1.3.746 set\_print\_func()** [2/2]

```
template<class Alg , class Ctx , class T , class C >
void GiNaC::set_print_func (
    void(T::*)(const C &, unsigned) f )
```

Add or replace a print method.

References [options](#).

**5.1.3.747 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [29/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    relational ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print↵
    _tree. print_func< print_python_repr > &::do_print_python_repr )
```

**5.1.3.748 GINAC\_BIND\_UNARCHIVER()** [39/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    relational )
```

**5.1.3.749 print\_operator()**

```
static void GiNaC::print_operator (
    const print_context & c,
    relational::operators o ) [static]
```

References [c](#), [GiNaC::relational::equal](#), [GiNaC::relational::greater](#), [GiNaC::relational::greater\\_or\\_equal](#), [GiNaC::relational::less](#), [GiNaC::relational::less\\_or\\_equal](#), and [GiNaC::relational::not\\_equal](#).

Referenced by [GiNaC::relational::do\\_print\(\)](#), and [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#).

**5.1.3.750 GINAC\_DECLARE\_UNARCHIVER()** [41/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    relational )
```

**5.1.3.751 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [30/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    symbol ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr > &↵
::do_print_python_repr )
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**5.1.3.752 get\_default\_TeX\_name()**

```
static const std::string & GiNaC::get_default_TeX_name (
    const std::string & name ) [static]
```

Return default TeX name for symbol.

This recognizes some greek letters.

Referenced by [GiNaC::symbol::do\\_print\\_latex\(\)](#), and [GiNaC::symbol::get\\_TeX\\_name\(\)](#).

**5.1.3.753 GINAC\_BIND\_UNARCHIVER()** [40/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    symbol )
```

**5.1.3.754 GINAC\_BIND\_UNARCHIVER()** [41/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    realsymbol )
```

**5.1.3.755 GINAC\_BIND\_UNARCHIVER()** [42/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    possymbol )
```

**5.1.3.756 GINAC\_DECLARE\_UNARCHIVER()** [42/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    symbol )
```

**5.1.3.757 GINAC\_DECLARE\_UNARCHIVER()** [43/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    realsymbol )
```

**5.1.3.758 GINAC\_DECLARE\_UNARCHIVER()** [44/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    possymbol )
```

**5.1.3.759 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [31/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    symmetry ,
    basic ,
    print_func< print\_context > &::do_print.  print_func< print\_tree > &::do_print↵
    _tree )
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**5.1.3.760 GINAC\_BIND\_UNARCHIVER()** [43/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    symmetry )
```

**5.1.3.761 index0()**

```
static const symmetry & GiNaC::index0 ( ) [static]
```

Referenced by [antisymmetric2\(\)](#), [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [symmetric2\(\)](#), [symmetric3\(\)](#), and [symmetric4\(\)](#).

**5.1.3.762 index1()**

```
static const symmetry & GiNaC::index1 ( ) [static]
```

Referenced by [antisymmetric2\(\)](#), [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [symmetric2\(\)](#), [symmetric3\(\)](#), and [symmetric4\(\)](#).

**5.1.3.763 index2()**

```
static const symmetry & GiNaC::index2 ( ) [static]
```

Referenced by [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [symmetric3\(\)](#), and [symmetric4\(\)](#).

**5.1.3.764 index3()**

```
static const symmetry & GiNaC::index3 ( ) [static]
```

Referenced by [antisymmetric4\(\)](#), and [symmetric4\(\)](#).

**5.1.3.765 not\_symmetric()**

```
const symmetry & GiNaC::not_symmetric ( )
```

Referenced by [GiNaC::indexed::indexed\(\)](#), and [GiNaC::indexed::read\\_archive\(\)](#).

**5.1.3.766 symmetric2()**

```
const symmetry & GiNaC::symmetric2 ( )
```

References [index0\(\)](#), [index1\(\)](#), and [GiNaC::symmetry::symmetric](#).

Referenced by [delta\\_tensor\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [lorentz\\_g\(\)](#), and [metric\\_tensor\(\)](#).

**5.1.3.767 symmetric3()**

```
const symmetry & GiNaC::symmetric3 ( )
```

References [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), and [GiNaC::symmetry::symmetric](#).

Referenced by [color\\_d\(\)](#).



**5.1.3.768 symmetric4()**

```
const symmetry & GiNaC::symmetric4 ( )
```

References [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), [index3\(\)](#), and [GiNaC::symmetry::symmetric](#).

**5.1.3.769 antisymmetric2()**

```
const symmetry & GiNaC::antisymmetric2 ( )
```

References [GiNaC::symmetry::antisymmetric](#), [index0\(\)](#), and [index1\(\)](#).

Referenced by [epsilon\\_tensor\(\)](#), and [spinor\\_metric\(\)](#).

**5.1.3.770 antisymmetric3()**

```
const symmetry & GiNaC::antisymmetric3 ( )
```

References [GiNaC::symmetry::antisymmetric](#), [index0\(\)](#), [index1\(\)](#), and [index2\(\)](#).

Referenced by [color\\_f\(\)](#), and [epsilon\\_tensor\(\)](#).

**5.1.3.771 antisymmetric4()**

```
const symmetry & GiNaC::antisymmetric4 ( )
```

References [GiNaC::symmetry::antisymmetric](#), [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), and [index3\(\)](#).

Referenced by [lorentz\\_eps\(\)](#).

**5.1.3.772 canonicalize()**

```
int GiNaC::canonicalize (
    exvector::iterator v,
    const symmetry & symm )
```

Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.

**Parameters**

<i>v</i>	Start of expression vector
<i>symm</i>	Root node of symmetry tree

**Returns**

the overall sign introduced by the reordering (+1, -1 or 0) or `numeric_limits<int>::max()` if nothing changed

Referenced by [GiNaC::function::eval\(\)](#), and [GiNaC::indexed::eval\(\)](#).

**5.1.3.773 symm()**

```
static ex GiNaC::symm (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last,
    bool asymmetric ) [static]
```

References [GiNaC::container< C >::append\(\)](#), [factorial\(\)](#), [last](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::op\(\)](#), [permutation\\_sign\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [antisymmetrize\(\)](#), [GiNaC::ex::antisymmetrize\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [symmetrize\(\)](#), and [GiNaC::ex::symmetrize\(\)](#).

**5.1.3.774 symmetrize() [3/4]**

```
ex GiNaC::symmetrize (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last )
```

Symmetrize expression over a set of objects (symbols, indices).

References [last](#), and [symm\(\)](#).

**5.1.3.775 antisymmetrize() [3/4]**

```
ex GiNaC::antisymmetrize (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last )
```

Antisymmetrize expression over a set of objects (symbols, indices).

References [last](#), and [symm\(\)](#).

**5.1.3.776 symmetrize\_cyclic()** [3/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & e,
    exvector::const_iterator first,
    exvector::const_iterator last )
```

Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).

References [GiNaC::container< C >::append\(\)](#), [last](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::container< C >::remove\\_first\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.777 GINAC\_DECLARE\_UNARCHIVER()** [45/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    symmetry )
```

**5.1.3.778 sy\_none()** [1/4]

```
symmetry GiNaC::sy_none ( ) [inline]
```

**5.1.3.779 sy\_none()** [2/4]

```
symmetry GiNaC::sy_none (
    const symmetry & c1,
    const symmetry & c2 ) [inline]
```

References [GiNaC::symmetry::none](#).

**5.1.3.780 sy\_none()** [3/4]

```
symmetry GiNaC::sy_none (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3 ) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::none](#).

**5.1.3.781 sy\_none() [4/4]**

```

symmetry GiNaC::sy_none (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4 ) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::none](#).

**5.1.3.782 sy\_symm() [1/4]**

```

symmetry GiNaC::sy_symm ( ) [inline]

```

References [GiNaC::symmetry::set\\_type\(\)](#), and [GiNaC::symmetry::symmetric](#).

Referenced by [GiNaC::indexed::read\\_archive\(\)](#).

**5.1.3.783 sy\_symm() [2/4]**

```

symmetry GiNaC::sy_symm (
    const symmetry & c1,
    const symmetry & c2 ) [inline]

```

References [GiNaC::symmetry::symmetric](#).

**5.1.3.784 sy\_symm() [3/4]**

```

symmetry GiNaC::sy_symm (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3 ) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::symmetric](#).

**5.1.3.785 sy\_symm() [4/4]**

```

symmetry GiNaC::sy_symm (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4 ) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::symmetric](#).

**5.1.3.786 sy\_anti() [1/4]**

```
symmetry GiNaC::sy_anti ( ) [inline]
```

References [GiNaC::symmetry::antisymmetric](#), and [GiNaC::symmetry::set\\_type\(\)](#).

Referenced by [GiNaC::indexed::read\\_archive\(\)](#).

**5.1.3.787 sy\_anti() [2/4]**

```
symmetry GiNaC::sy_anti (
    const symmetry & c1,
    const symmetry & c2 ) [inline]
```

References [GiNaC::symmetry::antisymmetric](#).

**5.1.3.788 sy\_anti() [3/4]**

```
symmetry GiNaC::sy_anti (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3 ) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::antisymmetric](#).

**5.1.3.789 sy\_anti() [4/4]**

```
symmetry GiNaC::sy_anti (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4 ) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::antisymmetric](#).

**5.1.3.790 sy\_cycl() [1/4]**

```
symmetry GiNaC::sy_cycl ( ) [inline]
```

References [GiNaC::symmetry::cyclic](#), and [GiNaC::symmetry::set\\_type\(\)](#).

**5.1.3.791 sy\_cycl()** [2/4]

```

symmetry GiNaC::sy_cycl (
    const symmetry & c1,
    const symmetry & c2 ) [inline]

```

References [GiNaC::symmetry::cyclic](#).

**5.1.3.792 sy\_cycl()** [3/4]

```

symmetry GiNaC::sy_cycl (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3 ) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::cyclic](#).

**5.1.3.793 sy\_cycl()** [4/4]

```

symmetry GiNaC::sy_cycl (
    const symmetry & c1,
    const symmetry & c2,
    const symmetry & c3,
    const symmetry & c4 ) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::cyclic](#).

**5.1.3.794 symmetrize()** [4/4]

```

ex GiNaC::symmetrize (
    const ex & e,
    const exvector & v ) [inline]

```

Symmetrize expression over a set of objects (symbols, indices).

References [GiNaC::ex::begin\(\)](#), and [symmetrize\(\)](#).

**5.1.3.795 antisymmetrize()** [4/4]

```

ex GiNaC::antisymmetrize (
    const ex & e,
    const exvector & v ) [inline]

```

Antisymmetrize expression over a set of objects (symbols, indices).

References [antisymmetrize\(\)](#), and [GiNaC::ex::begin\(\)](#).

**5.1.3.796 symmetrize\_cyclic()** [4/4]

```
ex GiNaC::symmetrize_cyclic (
    const ex & e,
    const exvector & v ) [inline]
```

Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).

References [GiNaC::ex::begin\(\)](#), and [symmetrize\(\)](#).

**5.1.3.797 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [32/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    tensdelta ,
    tensor ,
    print_func< print_dflt > &::do_print. print_func< print_latex > &::do_print_↵
    latex )
```

**5.1.3.798 print\_func< print\_dflt >()** [3/3]

```
GiNaC::print_func< print_dflt > (
    &tensmetric::do_print ) &
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**5.1.3.799 GINAC\_BIND\_UNARCHIVER()** [44/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    minkmetric )
```

**5.1.3.800 GINAC\_BIND\_UNARCHIVER()** [45/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    tensepsilon )
```

**5.1.3.801 GINAC\_BIND\_UNARCHIVER()** [46/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    tensdelta )
```

**5.1.3.802 GINAC\_BIND\_UNARCHIVER()** [47/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    tensmetric )
```

**5.1.3.803 GINAC\_BIND\_UNARCHIVER()** [48/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    spinmetric )
```

**5.1.3.804 delta\_tensor()**

```
ex GiNaC::delta_tensor (
    const ex & i1,
    const ex & i2 )
```

Create a delta tensor with specified indices.

The indices must be of class `idx` or a subclass. The delta tensor is always symmetric and its trace is the dimension of the index space.

**Parameters**

<i>i1</i>	First index
<i>i2</i>	Second index

**Returns**

newly constructed delta tensor

References [symmetric2\(\)](#).

Referenced by [color\\_trace\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), and [GiNaC::tensmetric::eval\\_indexed\(\)](#).

**5.1.3.805 metric\_tensor()**

```
ex GiNaC::metric_tensor (
    const ex & i1,
    const ex & i2 )
```

Create a symmetric metric tensor with specified indices.

The indices must be of class `varidx` or a subclass. A metric tensor with one covariant and one contravariant index is equivalent to the delta tensor.



## Parameters

<i>i1</i>	First index
<i>i2</i>	Second index

## Returns

newly constructed metric tensor

References [symmetric2\(\)](#).

Referenced by [GiNaC::tensepsilon::contract\\_with\(\)](#).

**5.1.3.806 lorentz\_g()**

```
ex GiNaC::lorentz_g (
    const ex & i1,
    const ex & i2,
    bool pos_sig = false )
```

Create a Minkowski metric tensor with specified indices.

The indices must be of class `varidx` or a subclass. The Lorentz metric is a symmetric tensor with a matrix representation of `diag(1,-1,-1,...)` (negative signature, the default) or `diag(-1,1,1,...)` (positive signature).

## Parameters

<i>i1</i>	First index
<i>i2</i>	Second index
<i>pos_sig</i>	Whether the signature is positive

## Returns

newly constructed Lorentz metric tensor

References [symmetric2\(\)](#).

Referenced by [GiNaC::tensepsilon::contract\\_with\(\)](#).

**5.1.3.807 spinor\_metric()**

```
ex GiNaC::spinor_metric (
    const ex & i1,
    const ex & i2 )
```

Create a spinor metric tensor with specified indices.

The indices must be of class `spinidx` or a subclass and have a dimension of 2. The spinor metric is an antisymmetric tensor with a matrix representation of `[[ [ 0, 1 ], [ -1, 0 ] ]]`.

## Parameters

<i>i1</i>	First index
<i>i2</i>	Second index

## Returns

newly constructed spinor metric tensor

References [antisymmetric2\(\)](#).

5.1.3.808 `epsilon_tensor()` [1/2]

```
ex GiNaC::epsilon_tensor (
    const ex & i1,
    const ex & i2 )
```

Create an epsilon tensor in a Euclidean space with two indices.

The indices must be of class `idx` or a subclass, and have a dimension of 2.

## Parameters

<i>i1</i>	First index
<i>i2</i>	Second index

## Returns

newly constructed epsilon tensor

References [\\_ex2](#), [antisymmetric2\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

5.1.3.809 `epsilon_tensor()` [2/2]

```
ex GiNaC::epsilon_tensor (
    const ex & i1,
    const ex & i2,
    const ex & i3 )
```

Create an epsilon tensor in a Euclidean space with three indices.

The indices must be of class `idx` or a subclass, and have a dimension of 3.

## Parameters

<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index

**Returns**

newly constructed epsilon tensor

References [\\_ex3](#), [antisymmetric3\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

**5.1.3.810 lorentz\_eps()**

```
ex GiNaC::lorentz_eps (
    const ex & i1,
    const ex & i2,
    const ex & i3,
    const ex & i4,
    bool pos_sig = false )
```

Create an epsilon tensor in a Minkowski space with four indices.

The indices must be of class varidx or a subclass, and have a dimension of 4.

**Parameters**

<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index
<i>i4</i>	Fourth index
<i>pos_sig</i>	Whether the signature of the metric is positive

**Returns**

newly constructed epsilon tensor

References [\\_ex4](#), [antisymmetric4\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

**5.1.3.811 GINAC\_DECLARE\_UNARCHIVER() [46/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    tensdelta )
```

**5.1.3.812 GINAC\_DECLARE\_UNARCHIVER() [47/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    tensmetric )
```

**5.1.3.813 GINAC\_DECLARE\_UNARCHIVER()** [48/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    minkmetric )
```

**5.1.3.814 GINAC\_DECLARE\_UNARCHIVER()** [49/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    spinmetric )
```

**5.1.3.815 GINAC\_DECLARE\_UNARCHIVER()** [50/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    tensepsilon )
```

**5.1.3.816 log2()**

```
unsigned GiNaC::log2 (
    unsigned n )
```

Integer binary logarithm.

References [k](#), and [n](#).

Referenced by [GiNaC::remember\\_table::remember\\_table\(\)](#).

**5.1.3.817 multinomial\_coefficient()**

```
const numeric GiNaC::multinomial_coefficient (
    const std::vector< unsigned > & p )
```

Compute the multinomial coefficient  $n!/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.

*p* is a partition of *n*.

References [GiNaC::numeric::div\(\)](#), [factorial\(\)](#), and [n](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

#### 5.1.3.818 rotate\_left()

```
unsigned GiNaC::rotate_left (
    unsigned n ) [inline]
```

Rotate bits of unsigned value by one bit to the left.

This can be necessary if the user wants to define its own hashes.

References [n](#).

Referenced by [GiNaC::basic::calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), and [GiNaC::symmetry::calchash\(\)](#).

#### 5.1.3.819 compare\_pointers()

```
template<class T >
int GiNaC::compare_pointers (
    const T * a,
    const T * b ) [inline]
```

Compare two pointers (just to establish some sort of canonical order).

Returns

-1, 0, or 1

Referenced by [GiNaC::basic::compare\\_same\\_type\(\)](#).

#### 5.1.3.820 golden\_ratio\_hash()

```
unsigned GiNaC::golden_ratio_hash (
    uintptr_t n ) [inline]
```

Truncated multiplication with golden ratio, for computing hash values.

References [n](#).

Referenced by [GiNaC::constant::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::numeric::calchash\(\)](#), [GiNaC::symbol::calchash\(\)](#), [GiNaC::wildcard::calchash\(\)](#), and [make\\_hash\\_seed\(\)](#).

#### 5.1.3.821 permutation\_sign() [1/2]

```
template<class It >
int GiNaC::permutation_sign (
    It first,
    It last )
```

References [last](#), [swap\(\)](#), and [std::swap\(\)](#).

Referenced by [GiNaC::matrix::determinant\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), and [symm\(\)](#).

#### 5.1.3.822 permutation\_sign() [2/2]

```
template<class It , class Cmp , class Swap >
int GiNaC::permutation_sign (
    It first,
    It last,
    Cmp comp,
    Swap swapit )
```

References [last](#).

#### 5.1.3.823 shaker\_sort()

```
template<class It , class Cmp , class Swap >
void GiNaC::shaker_sort (
    It first,
    It last,
    Cmp comp,
    Swap swapit )
```

References [last](#).

Referenced by [find\\_free\\_and\\_dummy\(\)](#), and [rename\\_dummy\\_indices\(\)](#).

#### 5.1.3.824 cyclic\_permutation()

```
template<class It , class Swap >
void GiNaC::cyclic_permutation (
    It first,
    It last,
    It new_first,
    Swap swapit )
```

References [last](#).

**5.1.3.825 format\_index\_value()** [1/2]

```
template<typename T >
std::enable_if< has\_distance< T >::value, typename std::iterator_traits< T >::difference_type
>::type GiNaC::format_index_value (
    const T & a,
    const T & b )
```

For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.

However, we may print the difference to the starting point.

Referenced by [operator<<\(\)](#).

**5.1.3.826 format\_index\_value()** [2/2]

```
template<typename T >
std::enable_if<!has\_distance< T >::value, T >::type GiNaC::format_index_value (
    const T & a,
    const T & b )
```

For all other cases we simply print the value.

**5.1.3.827 operator<<()** [8/16]

```
template<class T >
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const basic\_multi\_iterator< T > & v ) [inline]
```

Output operator.

A multi\_iterator prints out as [basic\\_multi\\_iterator](#)(  $n_0, n_1, \dots$  ).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.828 operator<<()** [9/16]

```
template<class T >
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi\_iterator\_ordered< T > & v ) [inline]
```

Output operator.

A multi\_iterator\_ordered prints out as [multi\\_iterator\\_ordered](#)(  $n_0, n_1, \dots$  ).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.829 operator<<() [10/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq< T > & v ) [inline]
```

Output operator.

A `multi_iterator_ordered_eq` prints out as `multi_iterator_ordered_eq(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.830 operator<<() [11/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq_indv< T > & v ) [inline]
```

Output operator.

A `multi_iterator_ordered_eq_indv` prints out as `multi_iterator_ordered_eq_indv(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.831 operator<<() [12/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_counter< T > & v ) [inline]
```

Output operator.

A `multi_iterator_counter` prints out as `multi_iterator_counter(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.832 operator<<() [13/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_counter_indv< T > & v ) [inline]
```

Output operator.

A `multi_iterator_counter_indv` prints out as `multi_iterator_counter_indv(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).



**5.1.3.833 operator<<() [14/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_permutation< T > & v ) [inline]
```

Output operator.

A `multi_iterator_permutation` prints out as `multi_iterator_permutation(  $n_0, n_1, \dots$  )`.

References `GiNaC::basic_multi_iterator< T >::B`, `format_index_value()`, and `GiNaC::basic_multi_iterator< T >::size()`.

**5.1.3.834 operator<<() [15/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_shuffle< T > & v ) [inline]
```

Output operator.

A `multi_iterator_shuffle` prints out as `multi_iterator_shuffle(  $n_0, n_1, \dots$  )`.

References `GiNaC::basic_multi_iterator< T >::B`, `format_index_value()`, and `GiNaC::basic_multi_iterator< T >::size()`.

**5.1.3.835 operator<<() [16/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
    std::ostream & os,
    const multi_iterator_shuffle_prime< T > & v ) [inline]
```

Output operator.

A `multi_iterator_shuffle_prime` prints out as `multi_iterator_shuffle_prime(  $n_0, n_1, \dots$  )`.

References `GiNaC::basic_multi_iterator< T >::B`, `format_index_value()`, and `GiNaC::basic_multi_iterator< T >::size()`.

**5.1.3.836 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [33/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
    wildcard ,
    basic ,
    print_func< print_context > &::do_print. print_func< print_tree > &::do_print↔
    _tree. print_func< print_python_repr > &::do_print_python_repr )
```

References `GiNaC::status_flags::evaluated`, and `GiNaC::status_flags::expanded`.

#### 5.1.3.837 GINAC\_BIND\_UNARCHIVER() [49/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
    wildcard )
```

#### 5.1.3.838 haswild()

```
bool GiNaC::haswild (
    const ex & x )
```

Check whether x has a wildcard anywhere as a subexpression.

References [haswild\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [x](#).

Referenced by [GiNaC::integral::eval\(\)](#), and [haswild\(\)](#).

#### 5.1.3.839 GINAC\_DECLARE\_UNARCHIVER() [51/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
    wildcard )
```

#### 5.1.3.840 wild()

```
ex GiNaC::wild (
    unsigned label = 0 ) [inline]
```

Create a wildcard object with the specified label.

### 5.1.4 Variable Documentation

#### 5.1.4.1 unarch\_table\_instance

```
unarchive_table_t GiNaC::unarch_table_instance [static]
```

#### 5.1.4.2 map\_evalm

`GiNaC::evalm_map_function` `GiNaC::map_evalm`

Referenced by `GiNaC::basic::evalm()`.

#### 5.1.4.3 map\_eval\_integ

`GiNaC::eval_integ_map_function` `GiNaC::map_eval_integ`

Referenced by `GiNaC::basic::eval_integ()`.

#### 5.1.4.4 tensor

`GiNaC::tensor`

#### 5.1.4.5 Pi

```
const constant GiNaC::Pi (
    "Pi" ,
    PiEvalf ,
    "\\pi" ,
    domain::positive )
```

Pi.

(3.14159...) Diverts straight into CLN for `evalf()`.

Referenced by `acos_eval()`, `acosh_eval()`, `asin_eval()`, `atan2_eval()`, `atan_eval()`, `atan_series()`, `atanh_series()`, `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `cos_eval()`, `cosh_eval()`, `EllipticE_eval()`, `EllipticE_evalf()`, `EllipticE_series()`, `EllipticK_eval()`, `EllipticK_evalf()`, `EllipticK_series()`, `eta_eval()`, `eta_evalf()`, `exp_eval()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::Kronecker_dz_kernel::get_numerical_value()`, `H_evalf()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `log_eval()`, `log_series()`, `GiNaC::constant::read_archive()`, `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `sin_eval()`, `sinh_eval()`, `tan_eval()`, `tan_series()`, `tanh_eval()`, `tanh_series()`, `tgamma_eval()`, and `zeta1_eval()`.

#### 5.1.4.6 Euler

```
const constant GiNaC::Euler (
    "Euler" ,
    EulerEvalf ,
    "\\gamma_E" ,
    domain::positive )
```

Euler's constant.

(0.57721...) Sometimes called Euler-Mascheroni constant. Diverts straight into CLN for `evalf()`.

Referenced by `psi1_eval()`, and `GiNaC::constant::read_archive()`.

#### 5.1.4.7 Catalan

```
const constant GiNaC::Catalan (
    "Catalan" ,
    CatalanEvalf ,
    "G" ,
    domain::positive )
```

Catalan's constant.

(0.91597...) Diverts straight into CLN for [evalf\(\)](#).

Referenced by [Li2\\_eval\(\)](#), [Li\\_eval\(\)](#), and [GiNaC::constant::read\\_archive\(\)](#).

#### 5.1.4.8 crctab

```
unsigned const GiNaC::crctab[256] [static]
```

Referenced by [crc32\(\)](#).

#### 5.1.4.9 library\_initializer

```
library\_init GiNaC::library_initializer [static]
```

For construction of flyweights, etc.

#### 5.1.4.10 \_num0\_bp

```
const basic * GiNaC::_num0_bp
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.11 idx

```
GiNaC::idx
```

#### 5.1.4.12 force\_include\_tgamma

```
unsigned GiNaC::force_include_tgamma = tgamma_SERIAL::serial
```

#### 5.1.4.13 `force_include_zeta1`

```
unsigned GiNaC::force_include_zeta1 = zeta1_SERIAL::serial
```

#### 5.1.4.14 `GINAC_BIND_UNARCHIVER`

```
template<>
GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T(lst, basic, print_func< print_context >(&lst::do_print).
print_func< print_tree >(&lst::do_print_tree)) template<> bool lst GiNaC::GINAC_BIND_UNARCHIVER(lst)
(
    lst )
```

Specialization of `container::info()` for `lst`.

#### 5.1.4.15 `I`

```
const numeric GiNaC::I = numeric(cln::complex(cln::cl_I(0),cln::cl_I(1)))
```

Imaginary unit.

This is not a constant but a numeric since we are natively handing complex numbers anyways, so in each expression containing an `I` it is automatically eval'ed away anyhow.

Referenced by `acosh_eval()`, `atan_eval()`, `atan_series()`, `atanh_series()`, `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `color_h()`, `GiNaC::su3f::contract_with()`, `cosh_eval()`, `csgn_eval()`, `eta_eval()`, `eta_evalf()`, `eta_imag_part()`, `exp_eval()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::Kronecker_dz_kernel::get_numerical_value()`, `H_evalf()`, `GiNaC::numeric::has()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `log_eval()`, `log_series()`, `GiNaC::numeric::normal()`, `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `GiNaC::Kronecker_dz_kernel::series_coeff_impl()`, `sinh_eval()`, `step_eval()`, `tanh_eval()`, `tanh_series()`, `GiNaC::numeric::to_polynomial()` and `GiNaC::numeric::to_rational()`.

#### 5.1.4.16 `Digits`

```
_numeric_digits GiNaC::Digits
```

Accuracy in decimal digits.

Only object of this type! Can be set using assignment from C++ unsigned ints and evaluated like any built-in type.

Referenced by `GiNaC::integration_kernel::get_numerical_value_impl()`, `iterated_integral_evalf_impl()`, `GiNaC::numeric::numeric()`, `print_real_cl_N()`, and `zeta1_evalf()`.

#### 5.1.4.17 `next_print_context_id`

```
unsigned GiNaC::next_print_context_id = 0
```

Next free ID for [print\\_context](#) types.

#### 5.1.4.18 `version_major`

```
const int GiNaC::version_major = GINACLIB_MAJOR_VERSION
```

#### 5.1.4.19 `version_minor`

```
const int GiNaC::version_minor = GINACLIB_MINOR_VERSION
```

#### 5.1.4.20 `version_micro`

```
const int GiNaC::version_micro = GINACLIB_MICRO_VERSION
```

#### 5.1.4.21 `_num_120_p`

```
const numeric * GiNaC::_num_120_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.22 `_ex_120`

```
const ex GiNaC::_ex_120 = ex(*_num_120_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.23 `_num_60_p`

```
const numeric * GiNaC::_num_60_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.24 `_ex_60`

```
const ex GiNaC::_ex_60 = ex(*_num_60_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.25 `_num_48_p`

```
const numeric * GiNaC::_num_48_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.26 `_ex_48`

```
const ex GiNaC::_ex_48 = ex(*_num_48_p)
```

Referenced by [Li2\\_eval\(\)](#), [Li\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.27 `_num_30_p`

```
const numeric * GiNaC::_num_30_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.28 `_ex_30`

```
const ex GiNaC::_ex_30 = ex(*_num_30_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.29 `_num_25_p`

```
const numeric * GiNaC::_num_25_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.30 `_ex_25`

```
const ex GiNaC::_ex_25 = ex(*\_num\_25\_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.31 `_num_24_p`

```
const numeric * GiNaC::_num_24_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.32 `_ex_24`

```
const ex GiNaC::_ex_24 = ex(*\_num\_24\_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.33 `_num_20_p`

```
const numeric * GiNaC::_num_20_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.34 `_ex_20`

```
const ex GiNaC::_ex_20 = ex(*\_num\_20\_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.35 `_num_18_p`

```
const numeric * GiNaC::_num_18_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).



#### 5.1.4.36 `_ex_18`

```
const ex GiNaC::_ex_18 = ex(*_num_18_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.37 `_num_15_p`

```
const numeric * GiNaC::_num_15_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.38 `_ex_15`

```
const ex GiNaC::_ex_15 = ex(*_num_15_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.39 `_num_12_p`

```
const numeric * GiNaC::_num_12_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.40 `_ex_12`

```
const ex GiNaC::_ex_12 = ex(*_num_12_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.41 `_num_11_p`

```
const numeric * GiNaC::_num_11_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.42 `_ex_11`

```
const ex GiNaC::_ex_11 = ex(*\_num\_11\_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.43 `_num_10_p`

```
const numeric * GiNaC::_num_10_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.44 `_ex_10`

```
const ex GiNaC::_ex_10 = ex(*\_num\_10\_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.45 `_num_9_p`

```
const numeric * GiNaC::_num_9_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.46 `_ex_9`

```
const ex GiNaC::_ex_9 = ex(*\_num\_9\_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.47 `_num_8_p`

```
const numeric * GiNaC::_num_8_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.48 `_ex_8`

```
const ex GiNaC::_ex_8 = ex(*_num_8_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.49 `_num_7_p`

```
const numeric * GiNaC::_num_7_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.50 `_ex_7`

```
const ex GiNaC::_ex_7 = ex(*_num_7_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.51 `_num_6_p`

```
const numeric * GiNaC::_num_6_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.52 `_ex_6`

```
const ex GiNaC::_ex_6 = ex(*_num_6_p)
```

Referenced by [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.53 `_num_5_p`

```
const numeric * GiNaC::_num_5_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.54 `_ex_5`

```
const ex GiNaC::_ex_5 = ex(*_num_5_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.55 `_num_4_p`

```
const numeric * GiNaC::_num_4_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.56 `_ex_4`

```
const ex GiNaC::_ex_4 = ex(*_num_4_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.57 `_num_3_p`

```
const numeric * GiNaC::_num_3_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.58 `_ex_3`

```
const ex GiNaC::_ex_3 = ex(*_num_3_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.59 `_num_2_p`

```
const numeric * GiNaC::_num_2_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [tgamma\\_eval\(\)](#).

**5.1.4.60** `_ex_2`

```
const ex GiNaC::_ex_2 = ex(*_num_2_p)
```

Referenced by [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.61** `_num_1_p`

```
const numeric * GiNaC::_num_1_p
```

Referenced by [acos\\_conjugate\(\)](#), [asin\\_conjugate\(\)](#), [asinh\\_conjugate\(\)](#), [atan\\_conjugate\(\)](#), [atanh\\_conjugate\(\)](#), [beta\\_eval\(\)](#), [binomial\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::add::do\\_print\\_cscc\(\)](#), [doublefactorial\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [psi1\\_eval\(\)](#), and [psi2\\_eval\(\)](#).

**5.1.4.62** `_ex_1`

```
const ex GiNaC::_ex_1 = ex(*_num_1_p)
```

Referenced by [acos\\_eval\(\)](#), [acosh\\_deriv\(\)](#), [acosh\\_eval\(\)](#), [asin\\_eval\(\)](#), [atan2\\_deriv\(\)](#), [atan\\_deriv\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_deriv\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_series\(\)](#), [cos\\_eval\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::add::do\\_print\\_cscc\(\)](#), [GiNaC::mul::do\\_print\\_cscc\(\)](#), [GiNaC::power::do\\_print\\_cscc\(\)](#), [GiNaC::power::do\\_print\\_cscc\\_cl\\_N\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [exminus\(\)](#), [exp\\_eval\(\)](#), [exp\\_power\(\)](#), [GiNaC::power::expand\(\)](#), [frac\\_cancel\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [lgamma\\_eval\(\)](#), [Li2\\_eval\(\)](#), [Li\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_deriv\(\)](#), [log\\_expand\(\)](#), [log\\_series\(\)](#), [operator-\(\)](#), [operator/\(\)](#), [operator/=\(\(\)\)](#), [psi1\\_series\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_series\(\)](#), [replace\\_with\\_symbol\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::ex::unit\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.63** `_num_1_2_p`

```
const numeric * GiNaC::_num_1_2_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**5.1.4.64** `_ex_1_2`

```
const ex GiNaC::_ex_1_2 = ex(*_num_1_2_p)
```

Referenced by [acos\\_deriv\(\)](#), [acos\\_eval\(\)](#), [acosh\\_deriv\(\)](#), [asin\\_deriv\(\)](#), [asin\\_eval\(\)](#), [asinh\\_deriv\(\)](#), [atan2\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [zeta1\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.65 `_num_1_3_p`**

```
const numeric * GiNaC::_num_1_3_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**5.1.4.66 `_ex_1_3`**

```
const ex GiNaC::_ex_1_3 = ex(*_num_1_3_p)
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.67 `_num_1_4_p`**

```
const numeric * GiNaC::_num_1_4_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**5.1.4.68 `_ex_1_4`**

```
const ex GiNaC::_ex_1_4 = ex(*_num_1_4_p)
```

Referenced by [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.69 `_num0_p`**

```
const numeric * GiNaC::_num0_p
```

Referenced by [GiNaC::numeric::add\\_dyn\(\)](#), [atan\(\)](#), [binomial\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::power::eval\(\)](#), [exp\\_eval\(\)](#), [GiNaC::mul::expand\(\)](#), [fibonacci\(\)](#), [GiNaC::numeric::has\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [iquo\(\)](#), [irem\(\)](#), [isqrt\(\)](#), [Li2\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [mod\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::numeric::power\(\)](#), [GiNaC::numeric::power\\_dyn\(\)](#), [sin\\_eval\(\)](#), [smod\(\)](#), [GiNaC::numeric::sub\\_dyn\(\)](#), and [tan\\_eval\(\)](#).

5.1.4.70 `_ex0`

```
const ex GiNaC::_ex0 = ex(*\_num0\_p)
```

Referenced by [acos\\_eval\(\)](#), [acosh\\_eval\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [asinh\\_eval\(\)](#), [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_series\(\)](#), [beta\\_eval\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::relational::canonical\(\)](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::numeric::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [color\\_trace\(\)](#), [GiNaC::ex::content\(\)](#), [cos\\_eval\(\)](#), [csgn\\_series\(\)](#), [GiNaC::expairseq::default\\_overall\\_coeff\(\)](#), [GiNaC::basic::derivative\(\)](#), [GiNaC::constant::derivative\(\)](#), [GiNaC::idx::derivative\(\)](#), [GiNaC::indexed::derivative\(\)](#), [GiNaC::symbol::derivative\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_eval\(\)](#), [EllipticK\\_series\(\)](#), [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [eta\\_series\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::indexed::expand\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [gcd\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_deriv\(\)](#), [Li\\_eval\(\)](#), [Li\\_evalf\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [log\\_series\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [Order\\_eval\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [prem\(\)](#), [GiNaC::ex::primpart\(\)](#), [rem\(\)](#), [S\\_deriv\(\)](#), [S\\_eval\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::symbol::series\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [sprem\(\)](#), [sqrfree\(\)](#), [sr\\_gcd\(\)](#), [step\\_series\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [zeta1\\_deriv\(\)](#), [zeta1\\_eval\(\)](#), [zeta2\\_deriv\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

5.1.4.71 `_num1_4_p`

```
const numeric * GiNaC::_num1_4_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

5.1.4.72 `_ex1_4`

```
const ex GiNaC::_ex1_4 = ex(*\_num1\_4\_p)
```

Referenced by [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

5.1.4.73 `_num1_3_p`

```
const numeric * GiNaC::_num1_3_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.74 `_ex1_3`

```
const ex GiNaC::_ex1_3 = ex(*\_num1\_3\_p)
```

Referenced by [acos\\_eval\(\)](#), [cos\\_eval\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.75 `_num1_2_p`

```
const numeric * GiNaC::_num1_2_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), [psi2\\_eval\(\)](#), and [tgamma\\_eval\(\)](#).

#### 5.1.4.76 `_ex1_2`

```
const ex GiNaC::_ex1_2 = ex(*\_num1\_2\_p)
```

Referenced by [acos\\_eval\(\)](#), [asin\\_eval\(\)](#), [atan2\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [cos\\_eval\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [sin\\_eval\(\)](#), [sqrt\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.77 `_num1_p`

```
const numeric * GiNaC::_num1_p
```

Referenced by [acos\\_conjugate\(\)](#), [acosh\\_conjugate\(\)](#), [asin\\_conjugate\(\)](#), [asinh\\_conjugate\(\)](#), [atan\(\)](#), [atan\\_conjugate\(\)](#), [atanh\\_conjugate\(\)](#), [bernoulli\(\)](#), [binomial\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::numeric::div\\_dyn\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [doublefactorial\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [exp\\_eval\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [frac\\_cancel\(\)](#), [gcd\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::basic::integer\\_content\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [lcmcoeff\(\)](#), [Li2\\_conjugate\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [GiNaC::basic::max\\_coefficient\(\)](#), [GiNaC::numeric::mul\\_dyn\(\)](#), [multiply\\_lcm\(\)](#), [operator++\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::numeric::power\(\)](#), [GiNaC::numeric::power\\_dyn\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [tgamma\\_eval\(\)](#), and [zeta1\\_eval\(\)](#).



5.1.4.78 `_ex1`

```
const ex GiNaC::_ex1 = ex(*\_num1\_p)
```

Referenced by [acos\\_eval\(\)](#), [acosh\\_deriv\(\)](#), [acosh\\_eval\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [asin\\_eval\(\)](#), [asinh\\_deriv\(\)](#), [atan\\_deriv\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_deriv\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_series\(\)](#), [beta\\_eval\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [color\\_trace\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::su3t::contract\\_with\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [GiNaC::mul::default\\_overall\\_coeff\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::symbol::derivative\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_series\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::mul::evalm\(\)](#), [exp\\_eval\(\)](#), [GiNaC::expairseq::expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::mul::expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_mul\(\)](#), [find\\_common\\_factor\(\)](#), [frac\\_cancel\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::matrix::inverse\(\)](#), [lgamma\\_series\(\)](#), [Li2\\_deriv\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [Li\\_evalf\(\)](#), [Li\\_series\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [log\\_expand\(\)](#), [log\\_series\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::symbol::normal\(\)](#), [operator++\(\)](#), [Order\\_eval\(\)](#), [Order\\_series\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [prem\(\)](#), [GiNaC::ex::primpart\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [psi1\\_deriv\(\)](#), [psi1\\_series\(\)](#), [psi2\\_deriv\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_series\(\)](#), [quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_tensor\(\)](#), [GiNaC::tensor::replace\\_contr\\_index\(\)](#), [S\\_series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::symbol::series\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [GiNaC::expairseq::split\\_ex\\_to\\_pair\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [sprem\(\)](#), [sqrfree\\_parfrac\(\)](#), [sqrfree\\_yun\(\)](#), [sr\\_gcd\(\)](#), [tan\\_deriv\(\)](#), [tan\\_eval\(\)](#), [tanh\\_deriv\(\)](#), [tanh\\_eval\(\)](#), [tgamma\\_series\(\)](#), [GiNaC::expairseq::to\\_polynomial\(\)](#), [GiNaC::expairseq::to\\_rational\(\)](#), [GiNaC::ex::unit\(\)](#), [unit\\_matrix\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [zeta1\\_deriv\(\)](#), [zeta2\\_deriv\(\)](#), and [GiNaC::library\\_init::~library\\_init\(\)](#).

5.1.4.79 `_num2_p`

```
const numeric * GiNaC::_num2_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [exp\\_eval\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [Li2\\_series\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [GiNaC::matrix::pow\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [tgamma\\_eval\(\)](#), and [zeta1\\_eval\(\)](#).

5.1.4.80 `_ex2`

```
const ex GiNaC::_ex2 = ex(*\_num2\_p)
```

Referenced by [acos\\_deriv\(\)](#), [asin\\_deriv\(\)](#), [asinh\\_deriv\(\)](#), [atan2\\_deriv\(\)](#), [atan\\_deriv\(\)](#), [atanh\\_deriv\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [csgn\\_power\(\)](#), [epsilon\\_tensor\(\)](#), [exp\\_eval\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [product\\_to\\_exvector\(\)](#), [psi1\\_eval\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [tan\\_deriv\(\)](#), [tan\\_eval\(\)](#), [tanh\\_deriv\(\)](#), [tanh\\_eval\(\)](#), and [GiNaC::library\\_init::~library\\_init\(\)](#).

#### 5.1.4.81 `_num3_p`

```
const numeric * GiNaC::_num3_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [exp\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.82 `_ex3`

```
const ex GiNaC::_ex3 = ex(*_num3_p)
```

Referenced by [color\\_trace\(\)](#), [cos\\_eval\(\)](#), [epsilon\\_tensor\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.83 `_num4_p`

```
const numeric * GiNaC::_num4_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [exp\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.84 `_ex4`

```
const ex GiNaC::_ex4 = ex(*_num4_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), [lorentz\\_eps\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.85 `_num5_p`

```
const numeric * GiNaC::_num5_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.86 `_ex5`

```
const ex GiNaC::_ex5 = ex(*_num5_p)
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.87 `_num6_p`

```
const numeric * GiNaC::_num6_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [sin\\_eval\(\)](#).

#### 5.1.4.88 `_ex6`

```
const ex GiNaC::_ex6 = ex(*_num6_p)
```

Referenced by [cos\\_eval\(\)](#), [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.89 `_num7_p`

```
const numeric * GiNaC::_num7_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.90 `_ex7`

```
const ex GiNaC::_ex7 = ex(*_num7_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.91 `_num8_p`

```
const numeric * GiNaC::_num8_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.92 `_ex8`

```
const ex GiNaC::_ex8 = ex(*_num8_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.93 `_num9_p`

```
const numeric * GiNaC::_num9_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.94 `_ex9`

```
const ex GiNaC::_ex9 = ex(*_num9_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.95 `_num10_p`

```
const numeric * GiNaC::_num10_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.96 `_ex10`

```
const ex GiNaC::_ex10 = ex(*_num10_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.97 `_num11_p`

```
const numeric * GiNaC::_num11_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.98 `_ex11`

```
const ex GiNaC::_ex11 = ex(*_num11_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.99 `_num12_p`

```
const numeric * GiNaC::_num12_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.100 `_ex12`

```
const ex GiNaC::_ex12 = ex(*_num12_p)
```

Referenced by [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.101 `_num15_p`

```
const numeric * GiNaC::_num15_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.102 `_ex15`

```
const ex GiNaC::_ex15 = ex(*_num15_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.103 `_num18_p`

```
const numeric * GiNaC::_num18_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [sin\\_eval\(\)](#).

#### 5.1.4.104 `_ex18`

```
const ex GiNaC::_ex18 = ex(*_num18_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.105 `_num20_p`

```
const numeric * GiNaC::_num20_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.106 `_ex20`

```
const ex GiNaC::_ex20 = ex(*_num20_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.107 `_num24_p`

```
const numeric * GiNaC::_num24_p
```

Referenced by [cos\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.108 `_ex24`

```
const ex GiNaC::_ex24 = ex(*_num24_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.109 `_num25_p`

```
const numeric * GiNaC::_num25_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.110 `_ex25`

```
const ex GiNaC::_ex25 = ex(*_num25_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.111 `_num30_p`

```
const numeric * GiNaC::_num30_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.112 `_ex30`

```
const ex GiNaC::_ex30 = ex(*_num30_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.113 `_num48_p`

```
const numeric * GiNaC::_num48_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.114 `_ex48`

```
const ex GiNaC::_ex48 = ex(*_num48_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.115 `_num60_p`

```
const numeric * GiNaC::_num60_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.116 `_ex60`

```
const ex GiNaC::_ex60 = ex(*_num60_p)
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.117 `_num120_p`

`const numeric * GiNaC::_num120_p`

Referenced by `cos_eval()`, `GiNaC::library_init::library_init()`, and `sin_eval()`.

#### 5.1.4.118 `_ex120`

`const ex GiNaC::_ex120 = ex(*_num120_p)`

Referenced by `GiNaC::library_init::library_init()`, and `GiNaC::library_init::~~library_init()`.

## 5.2 GiNaC::internal Namespace Reference

### Classes

- struct `_iter_rep`

## 5.3 std Namespace Reference

### Classes

- struct `equal_to< GiNaC::ex >`  
*Specialization of `std::equal_to()` for `ex` objects.*
- struct `hash< GiNaC::ex >`  
*Specialization of `std::hash()` for `ex` objects.*
- struct `less< GiNaC::ptr< T > >`  
*Specialization of `std::less` for `ptr< T >` to enable ordering of `ptr< T >` objects (e.g.*

### Functions

- template<> void `swap (GiNaC::ex &a, GiNaC::ex &b)`  
*Specialization of `std::swap()` for `ex` objects.*

### 5.3.1 Function Documentation

#### 5.3.1.1 `swap()`

```
template<>
void std::swap (
    GiNaC::ex & a,
    GiNaC::ex & b ) [inline]
```

Specialization of `std::swap()` for `ex` objects.

References `GiNaC::ex::swap()`.

Referenced by `GiNaC::su3f::eval_indexed()`, `GiNaC::su3d::eval_indexed()`, `GiNaC::matrix::markowitz_elimination()`, and `GiNaC::permutation_sign()`.



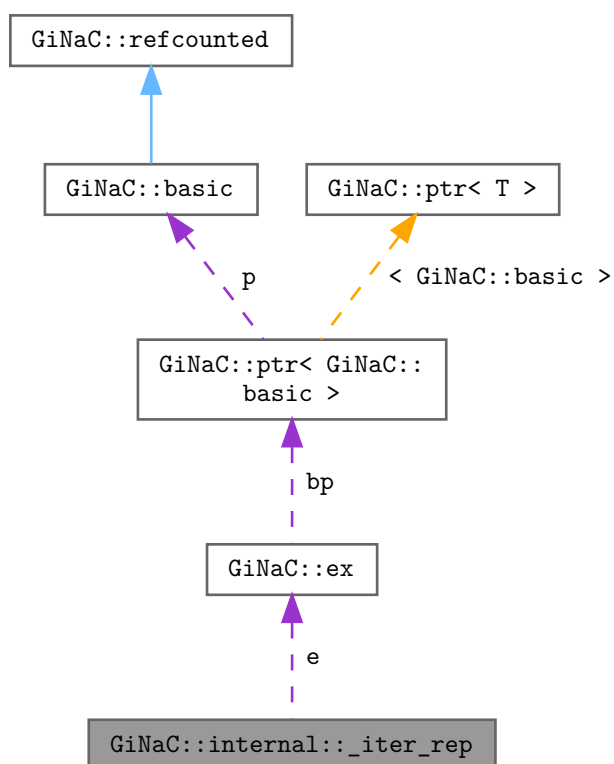
## Chapter 6

# Class Documentation

### 6.1 GiNaC::internal::\_iter\_rep Struct Reference

```
#include <ex.h>
```

Collaboration diagram for GiNaC::internal::\_iter\_rep:



## Public Member Functions

- [\\_iter\\_rep](#) (const [ex](#) &[e\\_](#), [size\\_t](#) [i\\_](#), [size\\_t](#) [i\\_end\\_](#))
- bool [operator==](#) (const [\\_iter\\_rep](#) &[other](#)) const noexcept
- bool [operator!=](#) (const [\\_iter\\_rep](#) &[other](#)) const noexcept

## Public Attributes

- [ex](#) [e](#)
- [size\\_t](#) [i](#)
- [size\\_t](#) [i\\_end](#)

## 6.1.1 Constructor & Destructor Documentation

### 6.1.1.1 [\\_iter\\_rep](#)()

```
GiNaC::internal::_iter_rep::_iter_rep (  
    const ex & e\_,  
    size\_t i\_,  
    size\_t i\_end\_ ) [inline]
```

## 6.1.2 Member Function Documentation

### 6.1.2.1 [operator==](#)()

```
bool GiNaC::internal::_iter_rep::operator== (  
    const \_iter\_rep & other ) const [inline], [noexcept]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [e](#), and [i](#).

### 6.1.2.2 [operator!=](#)()

```
bool GiNaC::internal::_iter_rep::operator!= (  
    const \_iter\_rep & other ) const [inline], [noexcept]
```

## 6.1.3 Member Data Documentation

### 6.1.3.1 e

```
ex GiNaC::internal::_iter_rep::e
```

Referenced by [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), and [operator==\(\)](#).

### 6.1.3.2 i

```
size_t GiNaC::internal::_iter_rep::i
```

Referenced by [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), and [operator==\(\)](#).

### 6.1.3.3 i\_end

```
size_t GiNaC::internal::_iter_rep::i_end
```

Referenced by [GiNaC::const\\_preorder\\_iterator::increment\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.2 GiNaC::\_numeric\_digits Class Reference

This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.

```
#include <numeric.h>
```

### Public Member Functions

- [\\_numeric\\_digits](#) ()  
*\_numeric\_digits* default ctor, checking for singleton invariance.
- [\\_numeric\\_digits](#) & [operator=](#) (long prec)  
*Assign a native long to global Digits object.*
- [operator long](#) ()  
*Convert global Digits object to native type long.*
- void [print](#) (std::ostream &os) const  
*Append global Digits object to ostream.*
- void [add\\_callback](#) ([digits\\_changed\\_callback](#) callback)  
*Add a new callback function.*

## Private Attributes

- long [digits](#)  
*Number of decimal digits.*
- `std::vector< digits\_changed\_callback > callbacklist`

## Static Private Attributes

- static bool [too\\_late](#) = false  
*Already one object present.*

### 6.2.1 Detailed Description

This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.

We need an object rather than a dumbier basic type since as a side-effect we let it change `cl_default_float_format` when it gets changed. The only other meaningful thing to do with it is converting it to an unsigned, for temporarily storing its value e.g. The user must not create an own working object of this class! Since C++ forces us to make the class definition visible in order to use an object we put in a flag which prevents other objects of that class to be created.

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 `_numeric_digits()`

```
GiNaC::_numeric_digits::_numeric_digits ( )
```

[\\_numeric\\_digits](#) default ctor, checking for singleton invariance.

References [too\\_late](#).

### 6.2.3 Member Function Documentation

#### 6.2.3.1 `operator=()`

```
\_numeric\_digits & GiNaC::_numeric_digits::operator= (
    long prec )
```

Assign a native long to global Digits object.

References [callbacklist](#), and [digits](#).

### 6.2.3.2 operator long()

```
GiNaC::_numeric_digits::operator long ( )
```

Convert global Digits object to native type long.

### 6.2.3.3 print()

```
void GiNaC::_numeric_digits::print (
    std::ostream & os ) const
```

Append global Digits object to ostream.

References [digits](#).

Referenced by [GiNaC::operator<<\(\)](#).

### 6.2.3.4 add\_callback()

```
void GiNaC::_numeric_digits::add_callback (
    digits_changed_callback callback )
```

Add a new callback function.

References [callbacklist](#).

## 6.2.4 Member Data Documentation

### 6.2.4.1 digits

```
long GiNaC::_numeric_digits::digits [private]
```

Number of decimal digits.

Referenced by [operator=\(\)](#), and [print\(\)](#).

### 6.2.4.2 too\_late

```
bool GiNaC::_numeric_digits::too_late = false [static], [private]
```

Already one object present.

Referenced by [\\_numeric\\_digits\(\)](#).

### 6.2.4.3 callbacklist

```
std::vector<digits_changed_callback> GiNaC::_numeric_digits::callbacklist [private]
```

Referenced by [add\\_callback\(\)](#), and [operator=\(\)](#).

The documentation for this class was generated from the following files:

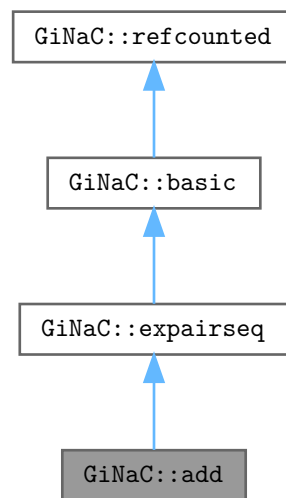
- [numeric.h](#)
- [numeric.cpp](#)

## 6.3 GiNaC::add Class Reference

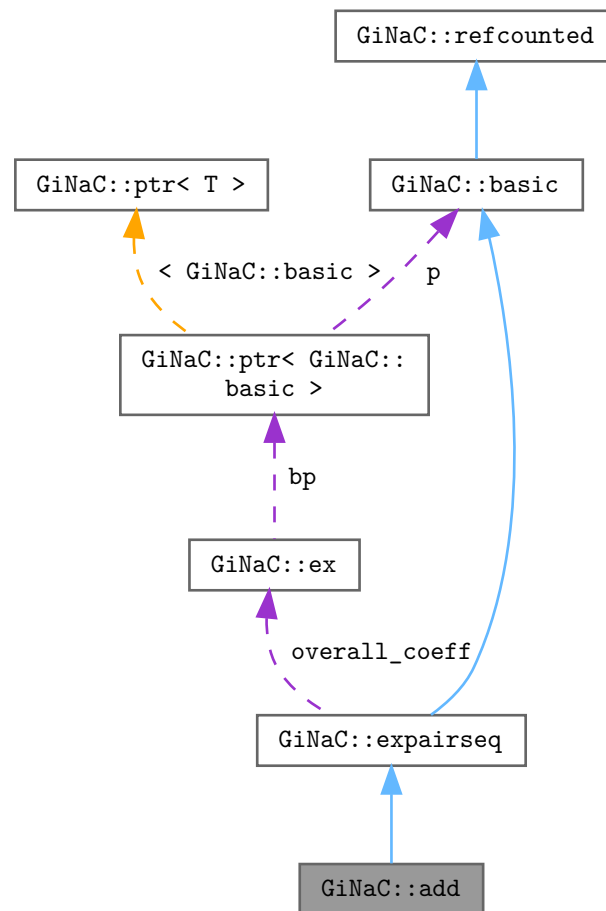
Sum of expressions.

```
#include <add.h>
```

Inheritance diagram for GiNaC::add:



Collaboration diagram for GiNaC::add:



## Public Member Functions

- `add` (const `ex` &lh, const `ex` &rh)
- `add` (const `exvector` &v)
- `add` (const `epvector` &v)
- `add` (const `epvector` &v, const `ex` &oc)
- `add` (`epvector` &&v)
- `add` (`epvector` &&v, const `ex` &oc)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*

- `int ldegree (const ex &s) const` override  
*Return degree of lowest power in object s.*
- `ex coeff (const ex &s, int n=1) const` override  
*Return coefficient of degree n in object s.*
- `ex eval ()` const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalm ()` const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex series (const relational &r, int order, unsigned options=0) const` override  
*Implementation of [ex::series\(\)](#) for sums.*
- `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const` override  
*Implementation of [ex::normal\(\)](#) for a sum.*
- `numeric integer_content ()` const override
- `ex smod (const numeric &xi) const` override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient ()` const override  
*Implementation [ex::max\\_coefficient\(\)](#).*
- `ex conjugate ()` const override
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `exvector get_free_indices ()` const override  
*Return a vector containing the free indices of an expression.*
- `ex eval_ncmul (const exvector &v) const` override

#### Public Member Functions inherited from [GiNaC::expairseq](#)

- `expairseq (const ex &lh, const ex &rh)`
- `expairseq (const exvector &v)`
- `expairseq (const epvector &v, const ex &oc, bool do_index_renaming=false)`
- `expairseq (epvector &&vp, const ex &oc, bool do_index_renaming=false)`
- `unsigned precedence ()` const override  
*Return relative operator precedence (for parenthezing output).*
- `bool info (unsigned inf) const` override  
*Information about the object.*
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i) const` override  
*Return operand/member at position i.*
- `ex map (map\_function &f) const` override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex eval ()` const override  
*Perform coefficient-wise automatic term rewriting rules in this class.*
- `ex to_rational (exmap &repl) const` override  
*Implementation of [ex::to\\_rational\(\)](#) for expairseqs.*
- `ex to_polynomial (exmap &repl) const` override  
*Implementation of [ex::to\\_polynomial\(\)](#) for expairseqs.*
- `bool match (const ex &pattern, exmap &repl_lst) const` override  
*Check whether the expression matches a given pattern.*
- `ex subs (const exmap &m, unsigned options=0) const` override  
*Substitute a set of objects by arbitrary expressions.*
- `ex conjugate ()` const override
- `void archive (archive\_node &n) const` override  
*Save (serialize) the object into archive node.*
- `void read_archive (const archive\_node &n, lst &syms) override`  
*Load (deserialize) the object from an archive node.*



Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

#### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

#### Protected Member Functions

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for a sum.*
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- [ex thisexpairseq](#) (const [epvector](#) &v, const [ex](#) &oc, bool do\_index\_renaming=false) const override  
*Create an object of this type.*
- [ex thisexpairseq](#) ([epvector](#) &&vp, const [ex](#) &oc, bool do\_index\_renaming=false) const override
- [expair split\\_ex\\_to\\_pair](#) (const [ex](#) &e) const override  
*Form an [expair](#) from an [ex](#), using the corresponding semantics.*
- [expair combine\\_ex\\_with\\_coeff\\_to\\_pair](#) (const [ex](#) &e, const [ex](#) &c) const override
- [expair combine\\_pair\\_with\\_coeff\\_to\\_pair](#) (const [expair](#) &p, const [ex](#) &c) const override
- [ex recombine\\_pair\\_to\\_ex](#) (const [expair](#) &p) const override  
*Form an [ex](#) out of an [expair](#), using the corresponding semantics.*
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- void [print\\_add](#) (const [print\\_context](#) &c, const char \*openbrace, const char \*closebrace, const char \*mul\_sym, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

#### Protected Member Functions inherited from [GiNaC::expairseq](#)

- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [return\\_type](#) () const override
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- virtual [ex thisexpairseq](#) (const [epvector](#) &v, const [ex](#) &oc, bool do\_index\_renaming=false) const

Create an object of this type.

- virtual `ex thisexpairseq` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false) const
- virtual void `printseq` (const `print_context` &c, char delim, unsigned this\_precedence, unsigned upper\_precedence) const
- virtual void `printpair` (const `print_context` &c, const `expair` &p, unsigned upper\_precedence) const
- virtual `expair split_ex_to_pair` (const `ex` &e) const

Form an expair from an ex, using the corresponding semantics.

- virtual `expair combine_ex_with_coeff_to_pair` (const `ex` &e, const `ex` &c) const
- virtual `expair combine_pair_with_coeff_to_pair` (const `expair` &p, const `ex` &c) const
- virtual `ex recombine_pair_to_ex` (const `expair` &p) const

Form an ex out of an expair, using the corresponding semantics.

- virtual bool `expair_needs_further_processing` (`epp` it)
- virtual `ex default_overall_coeff` () const
- virtual void `combine_overall_coeff` (const `ex` &c)
- virtual void `combine_overall_coeff` (const `ex` &c1, const `ex` &c2)
- virtual bool `can_make_flat` (const `expair` &p) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `construct_from_2_ex` (const `ex` &lh, const `ex` &rh)
- void `construct_from_2_expairseq` (const `expairseq` &s1, const `expairseq` &s2)
- void `construct_from_expairseq_ex` (const `expairseq` &s, const `ex` &e)
- void `construct_from_exvector` (const `exvector` &v)
- void `construct_from_epvector` (const `epvector` &v, bool do\_index\_renaming=false)
- void `construct_from_epvector` (`epvector` &&v, bool do\_index\_renaming=false)
- void `make_flat` (const `exvector` &v)

Combine this expairseq with argument exvector.

- void `make_flat` (const `epvector` &v, bool do\_index\_renaming=false)

Combine this expairseq with argument epvector.

- void `canonicalize` ()

Brings this expairseq into a sorted (canonical) form.

- void `combine_same_terms_sorted_seq` ()

Compact a presorted expairseq by combining all matching expairs to one each.

- bool `is_canonical` () const

Check if this expairseq is in sorted (canonical) form.

- `epvector expandchildren` (unsigned `options`) const

Member-wise expand the expairs in this sequence.

- `epvector evalchildren` () const

Member-wise evaluate the expairs in this sequence.

- `epvector subschildren` (const `exmap` &m, unsigned `options`=0) const

Member-wise substitute in this sequence.

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const

Returns true if the attributes of two objects are similar enough for a match.

- virtual `ex derivative` (const `symbol` &s) const

Default implementation of `ex::diff()`.

- virtual int `compare_same_type` (const `basic` &other) const

Returns order relation between two objects of same type.

- virtual bool `is_equal_same_type` (const `basic` &other) const

- Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Friends

- class `mul`
- class `power`

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::expairseq`

- `epvector seq`
- `ex overall_coeff`

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 6.3.1 Detailed Description

Sum of expressions.

## 6.3.2 Constructor & Destructor Documentation

### 6.3.2.1 `add()` [1/6]

```
GiNaC::add::add (
    const ex & lh,
    const ex & rh )
```

References `GiNaC::_ex0`, `GiNaC::expairseq::construct_from_2_ex()`, `GINAC_ASSERT`, `GiNaC::expairseq::is_canonical()`, and `GiNaC::expairseq::overall_coeff`.

**6.3.2.2 add()** [2/6]

```
GiNaC::add::add (
    const exvector & v )
```

References [GiNaC::\\_ex0](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.2.3 add()** [3/6]

```
GiNaC::add::add (
    const epvector & v )
```

References [GiNaC::\\_ex0](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.2.4 add()** [4/6]

```
GiNaC::add::add (
    const epvector & v,
    const ex & oc )
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.2.5 add()** [5/6]

```
GiNaC::add::add (
    epvector && v )
```

References [GiNaC::\\_ex0](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.2.6 add()** [6/6]

```
GiNaC::add::add (
    epvector && v,
    const ex & oc )
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

### 6.3.3 Member Function Documentation

#### 6.3.3.1 precedence()

```
unsigned GiNaC::add::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\\_csrc\(\)](#), and [print\\_add\(\)](#).

#### 6.3.3.2 info()

```
bool GiNaC::add::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::ex::info\(\)](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [info\(\)](#).

#### 6.3.3.3 is\_polynomial()

```
bool GiNaC::add::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

#### 6.3.3.4 degree()

```
int GiNaC::add::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

#### 6.3.3.5 ldegree()

```
int GiNaC::add::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

#### 6.3.3.6 coeff()

```
ex GiNaC::add::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::clifford\\_max\\_label\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::dirac\\_ONE\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

Referenced by [print\\_add\(\)](#), and [smod\(\)](#).

#### 6.3.3.7 eval()

```
ex GiNaC::add::eval ( ) const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following x stands for a symbolic variables of type [ex](#) and c stands for such an expression that contain a plain number.

- $+(;c) \rightarrow c$
- $+(x;0) \rightarrow x$

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [unlikely](#).



### 6.3.3.8 evalm()

```
ex GiNaC::add::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::matrix::add\(\)](#), [GiNaC::ex::evalm\(\)](#), [m](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.3.3.9 series()

```
ex GiNaC::add::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for sums.

This performs series addition when adding pseries objects.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::op\(\)](#), [options](#), [order](#), [GiNaC::expairseq::overall\\_coeff](#), [r](#), [GiNaC::expairseq::seq](#), and [GiNaC::ex::series\(\)](#).

### 6.3.3.10 normal()

```
ex GiNaC::add::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a sum.

It expands terms and performs fractional addition.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [expand\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GINAC\\_ASSERT](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.3.3.11 integer\_content()

```
numeric GiNaC::add::integer_content ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [c](#), [GiNaC::denom\(\)](#), [GiNaC::gcd\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::lcm\(\)](#), [GiNaC::numer\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 6.3.3.12 smod()

```
ex GiNaC::add::smod (
    const numeric & xi ) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

#### Parameters

$xi$	modulus
------	---------

#### Returns

mapped polynomial

#### See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::expairseq::seq](#), and [GiNaC::smod\(\)](#).

### 6.3.3.13 max\_coefficient()

```
numeric GiNaC::add::max_coefficient ( ) const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

#### See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

**6.3.3.14 conjugate()**

```
ex GiNaC::add::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

**6.3.3.15 real\_part()**

```
ex GiNaC::add::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

**6.3.3.16 imag\_part()**

```
ex GiNaC::add::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

**6.3.3.17 get\_free\_indices()**

```
exvector GiNaC::add::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::indices\\_consistent\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

**6.3.3.18 eval\_ncmul()**

```
ex GiNaC::add::eval_ncmul (
    const exvector & v ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

**6.3.3.19 derivative()**

```
ex GiNaC::add::derivative (
    const symbol & y ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a sum.

It differentiates each term.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

**6.3.3.20 return\_type()**

```
unsigned GiNaC::add::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), and [GiNaC::expairseq::seq](#).

**6.3.3.21 return\_type\_tinfo()**

```
return\_type\_t GiNaC::add::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

**6.3.3.22 thisexpairseq() [1/2]**

```
ex GiNaC::add::thisexpairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false ) const [override], [protected], [virtual]
```

Create an object of this type.

This method works similar to a constructor. It is useful because `expairseq` has (at least) two possible different semantics but we want to inherit methods thus avoiding code duplication. Sometimes a method in `expairseq` has to create a new one of the same semantics, which cannot be done by a ctor because the name (`add`, `mul`,...) is unknown on the `expairseq` level. In order for this trick to work a derived class must of course override this definition.

Reimplemented from [GiNaC::expairseq](#).

**6.3.3.23 thisexpairseq() [2/2]**

```
ex GiNaC::add::thisexpairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

**6.3.3.24 split\_ex\_to\_pair()**

```
expair GiNaC::add::split_ex_to_pair (
    const ex & e ) const [override], [protected], [virtual]
```

Form an expair from an ex, using the corresponding semantics.

See also

[expairseq::recombine\\_pair\\_to\\_ex\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

Referenced by [evalm\(\)](#), [imag\\_part\(\)](#), and [real\\_part\(\)](#).

**6.3.3.25 combine\_ex\_with\_coeff\_to\_pair()**

```
expair GiNaC::add::combine_ex_with_coeff_to_pair (
    const ex & e,
    const ex & c ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [c](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::ex::is\\_equal\(\)](#), [likely](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.3.26 combine\_pair\_with\_coeff\_to\_pair()**

```
expair GiNaC::add::combine_pair_with_coeff_to_pair (
    const expair & p,
    const ex & c ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [c](#), [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), and [GiNaC::expair::rest](#).

Referenced by [GiNaC::mul::eval\(\)](#), and [GiNaC::power::expand\\_add\\_2\(\)](#).

### 6.3.3.27 recombine\_pair\_to\_ex()

```
ex GiNaC::add::recombine_pair_to_ex (
    const expair & p ) const [override], [protected], [virtual]
```

Form an ex out of an expair, using the corresponding semantics.

See also

[expairseq::split\\_ex\\_to\\_pair\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_num1\\_p](#), [GiNaC::expair::coeff](#), and [GiNaC::expair::rest](#).

Referenced by [eval\(\)](#), [evalm\(\)](#), [GiNaC::power::expand\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [normal\(\)](#), and [real\\_part\(\)](#).

### 6.3.3.28 expand()

```
ex GiNaC::add::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [normal\(\)](#).

### 6.3.3.29 print\_add()

```
void GiNaC::add::print_add (
    const print\_context & c,
    const char * openbrace,
    const char * closebrace,
    const char * mul_sym,
    unsigned level ) const [protected]
```

References [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [precedence\(\)](#), [GiNaC::basic::print\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

**6.3.3.30 do\_print()**

```
void GiNaC::add::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_add\(\)](#).

**6.3.3.31 do\_print\_latex()**

```
void GiNaC::add::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_add\(\)](#).

**6.3.3.32 do\_print\_csrc()**

```
void GiNaC::add::do_print_csrc (
    const print\_csrc & c,
    unsigned level ) const [protected]
```

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::info\\_flags::real](#), and [GiNaC::expairseq::seq](#).

**6.3.3.33 do\_print\_python\_repr()**

```
void GiNaC::add::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), and [GiNaC::ex::print\(\)](#).

**6.3.4 Friends And Related Function Documentation****6.3.4.1 mul**

```
friend class mul [friend]
```

### 6.3.4.2 power

```
friend class power [friend]
```

The documentation for this class was generated from the following files:

- [add.h](#)
- [add.cpp](#)
- [indexed.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.4 GiNaC::archive Class Reference

This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).

```
#include <archive.h>
```

### Classes

- struct [archived\\_ex](#)  
*Archived expression descriptor.*

### Public Member Functions

- [archive](#) ()  
*Construct archive from expression using the default name "ex".*
- [~archive](#) ()
- [archive](#) (const [ex](#) &e)  
*Construct archive from expression using the specified name.*
- [archive](#) (const [ex](#) &e, const char \*n)  
*Construct archive from expression using the specified name.*
- void [archive\\_ex](#) (const [ex](#) &e, const char \*name)  
*Archive an expression.*
- [ex unarchive\\_ex](#) (const [lst](#) &sym\_lst, const char \*name) const  
*Retrieve expression from archive by name.*
- [ex unarchive\\_ex](#) (const [lst](#) &sym\_lst, unsigned index=0) const  
*Retrieve expression from archive by index.*
- [ex unarchive\\_ex](#) (const [lst](#) &sym\_lst, std::string &name, unsigned index=0) const  
*Retrieve expression and its name from archive by index.*
- unsigned [num\\_expressions](#) () const  
*Return number of archived expressions.*
- const [archive\\_node](#) & [get\\_top\\_node](#) (unsigned index=0) const  
*Return reference to top node of an expression specified by index.*
- void [clear](#) ()  
*Clear all archived expressions.*
- [archive\\_node\\_id](#) [add\\_node](#) (const [archive\\_node](#) &n)  
*Add [archive\\_node](#) to archive if the corresponding expression is not already archived.*
- [archive\\_node](#) & [get\\_node](#) ([archive\\_node\\_id](#) id)



- Retrieve [archive\\_node](#) by ID.
- void [forget](#) ()  
Delete cached unarchived expressions in all [archive\\_nodes](#) (mainly for debugging).
- void [prinraw](#) (std::ostream &os) const  
Print archive to stream in ugly raw format (for debugging).
- [archive\\_atom](#) [atomize](#) (const std::string &s) const  
Atomize a string (i.e.
- const std::string & [unatomize](#) ([archive\\_atom](#) id) const  
Unatomize a string (i.e.

## Private Types

- typedef std::map< std::string, [archive\\_atom](#) >::const\_iterator [inv\\_at\\_cit](#)  
The map of from strings to indices of the atoms vectors allows for faster archiving.

## Private Attributes

- std::vector< [archive\\_node](#) > [nodes](#)  
Vector of archived nodes.
- std::vector< [archived\\_ex](#) > [exprs](#)  
Vector of archived expression descriptors.
- std::vector< std::string > [atoms](#)  
Vector of atomized strings (using a vector allows faster unarchiving).
- std::map< std::string, [archive\\_atom](#) > [inverse\\_atoms](#)
- std::map< [ex](#), [archive\\_node\\_id](#), [ex\\_is\\_less](#) > [exprtable](#)  
Map of stored expressions to nodes for faster archiving.

## Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [archive](#) &ar)  
Write archive to binary data stream.
- std::istream & [operator>>](#) (std::istream &is, [archive](#) &ar)  
Read archive from binary data stream.

### 6.4.1 Detailed Description

This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).

An archive can be constructed from an expression and then written to a stream; or it can be read from a stream and then unarchived, yielding back the expression. Archives can hold multiple expressions which can be referred to by name or index number. The main component of the archive class is a vector of [archive\\_nodes](#) which each store one object of class [basic](#) (or a derived class).

### 6.4.2 Member Typedef Documentation

#### 6.4.2.1 inv\_at\_cit

```
typedef std::map<std::string,archive_atom>::const_iterator GiNaC::archive::inv_at_cit [private]
```

The map of from strings to indices of the atoms vectors allows for faster archiving.

### 6.4.3 Constructor & Destructor Documentation

#### 6.4.3.1 archive() [1/3]

```
GiNaC::archive::archive ( ) [inline]
```

#### 6.4.3.2 ~archive()

```
GiNaC::archive::~~archive ( ) [inline]
```

#### 6.4.3.3 archive() [2/3]

```
GiNaC::archive::archive (
    const ex & e ) [inline]
```

Construct archive from expression using the default name "ex".

References [archive\\_ex\(\)](#).

#### 6.4.3.4 archive() [3/3]

```
GiNaC::archive::archive (
    const ex & e,
    const char * n ) [inline]
```

Construct archive from expression using the specified name.

References [archive\\_ex\(\)](#), and [n](#).

### 6.4.4 Member Function Documentation

#### 6.4.4.1 archive\_ex()

```
void GiNaC::archive::archive_ex (
    const ex & e,
    const char * name )
```

Archive an expression.

## Parameters

<i>e</i>	the expression to be archived
<i>name</i>	name under which the expression is stored

References [add\\_node\(\)](#), [atomize\(\)](#), and [exprs](#).

Referenced by [archive\(\)](#).

#### 6.4.4.2 unarchive\_ex() [1/3]

```
ex GiNaC::archive::unarchive_ex (
    const lst & sym_lst,
    const char * name ) const
```

Retrieve expression from archive by name.

## Parameters

<i>sym_lst</i>	list of pre-defined symbols
<i>name</i>	name of expression

References [atomize\(\)](#), [exprs](#), and [nodes](#).

#### 6.4.4.3 unarchive\_ex() [2/3]

```
ex GiNaC::archive::unarchive_ex (
    const lst & sym_lst,
    unsigned index = 0 ) const
```

Retrieve expression from archive by index.

## Parameters

<i>sym_lst</i>	list of pre-defined symbols
<i>index</i>	index of expression

## See also

[count\\_expressions](#)

References [exprs](#), and [nodes](#).

#### 6.4.4.4 unarchive\_ex() [3/3]

```
ex GiNaC::archive::unarchive_ex (
    const lst & sym_lst,
    std::string & name,
    unsigned index = 0 ) const
```

Retrieve expression and its name from archive by index.

##### Parameters

<i>sym_lst</i>	list of pre-defined symbols
<i>name</i>	receives the name of the expression
<i>index</i>	index of expression

##### See also

`count_expressions`

References [exprs](#), [nodes](#), and [unatomize\(\)](#).

#### 6.4.4.5 num\_expressions()

```
unsigned GiNaC::archive::num_expressions ( ) const
```

Return number of archived expressions.

References [exprs](#).

#### 6.4.4.6 get\_top\_node()

```
const archive_node & GiNaC::archive::get_top_node (
    unsigned index = 0 ) const
```

Return reference to top node of an expression specified by index.

References [exprs](#), and [nodes](#).

#### 6.4.4.7 clear()

```
void GiNaC::archive::clear ( )
```

Clear all archived expressions.

References [atoms](#), [exprs](#), [exprtable](#), [inverse\\_atoms](#), and [nodes](#).

#### 6.4.4.8 add\_node()

```
archive_node_id GiNaC::archive::add_node (
    const archive_node & n )
```

Add [archive\\_node](#) to archive if the corresponding expression is not already archived.

##### Returns

ID of archived node

References [exprtable](#), [n](#), and [nodes](#).

Referenced by [GiNaC::archive\\_node::add\\_ex\(\)](#), and [archive\\_ex\(\)](#).

#### 6.4.4.9 get\_node()

```
archive_node & GiNaC::archive::get_node (
    archive_node_id id )
```

Retrieve [archive\\_node](#) by ID.

References [nodes](#).

Referenced by [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_ex\\_by\\_loc\(\)](#), and [GiNaC::archive\\_node::find\\_ex\\_node\(\)](#).

#### 6.4.4.10 forget()

```
void GiNaC::archive::forget ( )
```

Delete cached unarchived expressions in all [archive\\_nodes](#) (mainly for debugging).

References [GiNaC::archive\\_node::forget\(\)](#), and [nodes](#).

#### 6.4.4.11 printraw()

```
void GiNaC::archive::printraw (
    std::ostream & os ) const
```

Print archive to stream in ugly raw format (for debugging).

References [atoms](#), [exprs](#), [nodes](#), and [unatomize\(\)](#).

#### 6.4.4.12 atomize()

```
archive_atom GiNaC::archive::atomize (
    const std::string & s ) const
```

Atomize a string (i.e.

convert it into an ID number that uniquely represents the string).

References [atoms](#), and [inverse\\_atoms](#).

Referenced by [GiNaC::archive\\_node::add\\_bool\(\)](#), [GiNaC::archive\\_node::add\\_ex\(\)](#), [GiNaC::archive\\_node::add\\_string\(\)](#), [GiNaC::archive\\_node::add\\_unsigned\(\)](#), [archive\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_bool\(\)](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_ex\\_node\(\)](#), [GiNaC::archive\\_node::find\\_first\(\)](#), [GiNaC::archive\\_node::find\\_last\(\)](#), [GiNaC::archive\\_node::find\\_property\\_range\(\)](#), [GiNaC::archive\\_node::find\\_string\(\)](#), [GiNaC::archive\\_node::find\\_unsigned\(\)](#), and [unarchive\\_ex\(\)](#).

#### 6.4.4.13 unatomize()

```
const std::string & GiNaC::archive::unatomize (
    archive_atom id ) const
```

Unatomize a string (i.e.

convert the ID number back to the string).

References [atoms](#).

Referenced by [GiNaC::archive\\_node::find\\_string\(\)](#), [GiNaC::archive\\_node::get\\_properties\(\)](#), [GiNaC::archive\\_node::printraw\(\)](#), [printraw\(\)](#), and [unarchive\\_ex\(\)](#).

### 6.4.5 Friends And Related Function Documentation

#### 6.4.5.1 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const archive & ar ) [friend]
```

Write archive to binary data stream.

#### 6.4.5.2 operator>>

```
std::istream & operator>> (
    std::istream & is,
    archive & ar ) [friend]
```

Read archive from binary data stream.

## 6.4.6 Member Data Documentation

### 6.4.6.1 nodes

```
std::vector<archive_node> GiNaC::archive::nodes [private]
```

Vector of archived nodes.

Referenced by [add\\_node\(\)](#), [clear\(\)](#), [forget\(\)](#), [get\\_node\(\)](#), [get\\_top\\_node\(\)](#), [printraw\(\)](#), and [unarchive\\_ex\(\)](#).

### 6.4.6.2 exprs

```
std::vector<archived_ex> GiNaC::archive::exprs [private]
```

Vector of archived expression descriptors.

Referenced by [archive\\_ex\(\)](#), [clear\(\)](#), [get\\_top\\_node\(\)](#), [num\\_expressions\(\)](#), [printraw\(\)](#), and [unarchive\\_ex\(\)](#).

### 6.4.6.3 atoms

```
std::vector<std::string> GiNaC::archive::atoms [mutable], [private]
```

Vector of atomized strings (using a vector allows faster unarchiving).

Referenced by [atomize\(\)](#), [clear\(\)](#), [printraw\(\)](#), and [unatomize\(\)](#).

### 6.4.6.4 inverse\_atoms

```
std::map<std::string, archive_atom> GiNaC::archive::inverse_atoms [mutable], [private]
```

Referenced by [atomize\(\)](#), and [clear\(\)](#).

### 6.4.6.5 exprtable

```
std::map<ex, archive_node_id, ex_is_less> GiNaC::archive::exprtable [mutable], [private]
```

Map of stored expressions to nodes for faster archiving.

Referenced by [add\\_node\(\)](#), and [clear\(\)](#).

The documentation for this class was generated from the following files:

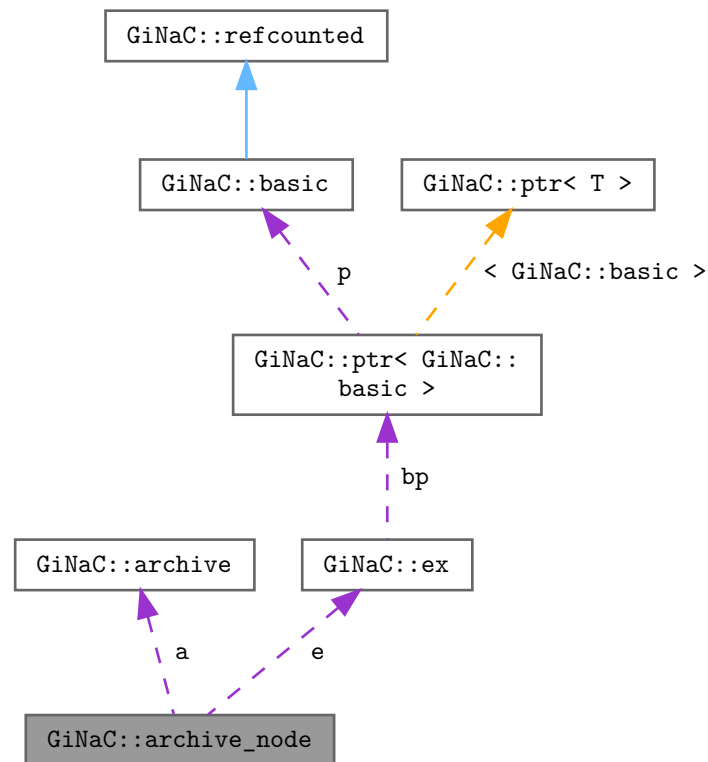
- [archive.h](#)
- [archive.cpp](#)

## 6.5 GiNaC::archive\_node Class Reference

This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).

```
#include <archive.h>
```

Collaboration diagram for GiNaC::archive\_node:



### Classes

- struct [archive\\_node\\_cit\\_range](#)
- struct [property](#)  
*Archived property (data type, name and associated data)*
- struct [property\\_info](#)  
*Information about a stored property.*

### Public Types

- enum [property\\_type](#) { `PTYPE_BOOL`, `PTYPE_UNSIGNED`, `PTYPE_STRING`, `PTYPE_NODE` }  
*Property data types.*
- typedef `std::vector< property\_info >` [propinfovector](#)
- typedef `std::vector< property >::const_iterator` [archive\\_node\\_cit](#)



## Public Member Functions

- [archive\\_node](#) ([archive](#) &ar)
- [archive\\_node](#) ([archive](#) &ar, const [ex](#) &expr)  
*Recursively construct archive node from expression.*
- const [archive\\_node](#) & [operator=](#) (const [archive\\_node](#) &other)  
*Assignment operator of [archive\\_node](#).*
- void [add\\_bool](#) (const std::string &name, bool [value](#))  
*Add property of type "bool" to node.*
- void [add\\_unsigned](#) (const std::string &name, unsigned [value](#))  
*Add property of type "unsigned int" to node.*
- void [add\\_string](#) (const std::string &name, const std::string &[value](#))  
*Add property of type "string" to node.*
- void [add\\_ex](#) (const std::string &name, const [ex](#) &[value](#))  
*Add property of type "ex" to node.*
- bool [find\\_bool](#) (const std::string &name, bool &ret, unsigned index=0) const  
*Retrieve property of type "bool" from node.*
- bool [find\\_unsigned](#) (const std::string &name, unsigned &ret, unsigned index=0) const  
*Retrieve property of type "unsigned" from node.*
- bool [find\\_string](#) (const std::string &name, std::string &ret, unsigned index=0) const  
*Retrieve property of type "string" from node.*
- [archive\\_node\\_cit](#) [find\\_first](#) (const std::string &name) const  
*Find the location in the vector of properties of the first/last property with a given name.*
- [archive\\_node\\_cit](#) [find\\_last](#) (const std::string &name) const
- [archive\\_node\\_cit\\_range](#) [find\\_property\\_range](#) (const std::string &name1, const std::string &name2) const  
*Find a range of locations in the vector of properties.*
- bool [find\\_ex](#) (const std::string &name, [ex](#) &ret, [lst](#) &sym\_lst, unsigned index=0) const  
*Retrieve property of type "ex" from node.*
- void [find\\_ex\\_by\\_loc](#) ([archive\\_node\\_cit](#) loc, [ex](#) &ret, [lst](#) &sym\_lst) const  
*Retrieve property of type "ex" from the node if it is known that this node in fact contains such a property at the given location.*
- const [archive\\_node](#) & [find\\_ex\\_node](#) (const std::string &name, unsigned index=0) const  
*Retrieve property of type "ex" from node, returning the node of the sub-expression.*
- void [get\\_properties](#) ([propinfovector](#) &v) const  
*Return vector of properties stored in node.*
- [ex](#) [unarchive](#) ([lst](#) &sym\_lst) const  
*Convert archive node to [GiNaC](#) expression.*
- bool [has\\_same\\_ex\\_as](#) (const [archive\\_node](#) &other) const  
*Check if the [archive\\_node](#) stores the same expression as another [archive\\_node](#).*
- bool [has\\_ex](#) () const
- [ex](#) [get\\_ex](#) () const
- void [forget](#) ()  
*Delete cached unarchived expressions from node (for debugging).*
- void [printraw](#) (std::ostream &os) const  
*Output [archive\\_node](#) to stream in ugly raw format (for debugging).*

## Private Attributes

- [archive](#) & [a](#)  
*Reference to the archive to which this node belongs.*
- `std::vector< property > props`  
*Vector of stored properties.*
- `bool has\_expression`  
*Flag indicating whether a cached unarchived representation of this node exists.*
- `ex e`  
*The cached unarchived representation of this node (if any).*

## Friends

- `std::ostream & operator<< (std::ostream &os, const archive\_node &ar)`  
*Write [archive\\_node](#) to binary data stream.*
- `std::istream & operator>> (std::istream &is, archive\_node &ar)`  
*Read [archive\\_node](#) from binary data stream.*

### 6.5.1 Detailed Description

This class stores all properties needed to record/retrieve the state of one object of class `basic` (or a derived class).

Each property is addressed by its name and data type.

### 6.5.2 Member Typedef Documentation

#### 6.5.2.1 `propinfovector`

```
typedef std::vector<property\_info> GiNaC::archive\_node::propinfovector
```

#### 6.5.2.2 `archive_node_cit`

```
typedef std::vector<property>::const_iterator GiNaC::archive\_node::archive\_node\_cit
```

### 6.5.3 Member Enumeration Documentation

#### 6.5.3.1 `property_type`

```
enum GiNaC::archive\_node::property\_type
```

Property data types.

## Enumerator

PTYPE_BOOL	
PTYPE_UNSIGNED	
PTYPE_STRING	
PTYPE_NODE	

## 6.5.4 Constructor & Destructor Documentation

### 6.5.4.1 archive\_node() [1/2]

```
GiNaC::archive_node::archive_node (
    archive & ar ) [inline]
```

### 6.5.4.2 archive\_node() [2/2]

```
GiNaC::archive_node::archive_node (
    archive & ar,
    const ex & expr )
```

Recursively construct archive node from expression.

References [GiNaC::ex::bp](#).

## 6.5.5 Member Function Documentation

### 6.5.5.1 operator=()

```
const archive_node & GiNaC::archive_node::operator= (
    const archive_node & other )
```

Assignment operator of [archive\\_node](#).

References [e](#), [has\\_expression](#), and [props](#).

#### 6.5.5.2 add\_bool()

```
void GiNaC::archive_node::add_bool (
    const std::string & name,
    bool value )
```

Add property of type "bool" to node.

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_BOOL](#), and [value](#).

#### 6.5.5.3 add\_unsigned()

```
void GiNaC::archive_node::add_unsigned (
    const std::string & name,
    unsigned value )
```

Add property of type "unsigned int" to node.

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_UNSIGNED](#), and [value](#).

#### 6.5.5.4 add\_string()

```
void GiNaC::archive_node::add_string (
    const std::string & name,
    const std::string & value )
```

Add property of type "string" to node.

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_STRING](#), and [value](#).

#### 6.5.5.5 add\_ex()

```
void GiNaC::archive_node::add_ex (
    const std::string & name,
    const ex & value )
```

Add property of type "ex" to node.

References [a](#), [GiNaC::archive::add\\_node\(\)](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_NODE](#), and [value](#).

Referenced by [GiNaC::container< C >::archive\(\)](#).

#### 6.5.5.6 find\_bool()

```
bool GiNaC::archive_node::find_bool (
    const std::string & name,
    bool & ret,
    unsigned index = 0 ) const
```

Retrieve property of type "bool" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), and [PTYPE\\_BOOL](#).

#### 6.5.5.7 find\_unsigned()

```
bool GiNaC::archive_node::find_unsigned (
    const std::string & name,
    unsigned & ret,
    unsigned index = 0 ) const
```

Retrieve property of type "unsigned" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), and [PTYPE\\_UNSIGNED](#).

#### 6.5.5.8 find\_string()

```
bool GiNaC::archive_node::find_string (
    const std::string & name,
    std::string & ret,
    unsigned index = 0 ) const
```

Retrieve property of type "string" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_STRING](#), and [GiNaC::archive::unatomize\(\)](#).

Referenced by [unarchive\(\)](#).

#### 6.5.5.9 find\_first()

```
archive_node::archive_node_cit GiNaC::archive_node::find_first (
    const std::string & name ) const
```

Find the location in the vector of properties of the first/last property with a given name.

References [a](#), [GiNaC::archive::atomize\(\)](#), and [props](#).

#### 6.5.5.10 find\_last()

```
archive_node::archive_node_cit GiNaC::archive_node::find_last (
    const std::string & name ) const
```

References [a](#), [GiNaC::archive::atomize\(\)](#), and [props](#).

#### 6.5.5.11 find\_property\_range()

```
archive_node::archive_node_cit_range GiNaC::archive_node::find_property_range (
    const std::string & name1,
    const std::string & name2 ) const
```

Find a range of locations in the vector of properties.

The result begins at the first property with name1 and ends one past the last property with name2.

References [a](#), [GiNaC::archive::atomize\(\)](#), [GiNaC::archive\\_node::archive\\_node\\_cit\\_range::begin](#), [GiNaC::archive\\_node::archive\\_node\\_cit\\_range::end](#), and [props](#).

#### 6.5.5.12 find\_ex()

```
bool GiNaC::archive_node::find_ex (
    const std::string & name,
    ex & ret,
    lst & sym_lst,
    unsigned index = 0 ) const
```

Retrieve property of type "ex" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [GiNaC::archive::atomize\(\)](#), [GiNaC::archive::get\\_node\(\)](#), [props](#), [PTYPE\\_NODE](#), and [unarchive\(\)](#).

#### 6.5.5.13 find\_ex\_by\_loc()

```
void GiNaC::archive_node::find_ex_by_loc (
    archive_node_cit loc,
    ex & ret,
    lst & sym_lst ) const
```

Retrieve property of type "ex" from the node if it is known that this node in fact contains such a property at the given location.

This is much more efficient than the preceding function.

References [a](#), [GiNaC::archive::get\\_node\(\)](#), and [unarchive\(\)](#).

#### 6.5.5.14 find\_ex\_node()

```
const archive_node & GiNaC::archive_node::find_ex_node (
    const std::string & name,
    unsigned index = 0 ) const
```

Retrieve property of type "ex" from node, returning the node of the sub-expression.

References [a](#), [GiNaC::archive::atomize\(\)](#), [GiNaC::archive::get\\_node\(\)](#), [props](#), and [PTYPE\\_NODE](#).

#### 6.5.5.15 get\_properties()

```
void GiNaC::archive_node::get_properties (
    propinfovector & v ) const
```

Return vector of properties stored in node.

References [a](#), [props](#), and [GiNaC::archive::unatomize\(\)](#).

#### 6.5.5.16 unarchive()

```
ex GiNaC::archive_node::unarchive (
    lst & sym_lst ) const
```

Convert archive node to [GiNaC](#) expression.

References [GiNaC::status\\_flags::dynallocated](#), [e](#), [GiNaC::find\\_factory\\_fcn\(\)](#), [find\\_string\(\)](#), and [has\\_expression](#).

Referenced by [find\\_ex\(\)](#), and [find\\_ex\\_by\\_loc\(\)](#).

#### 6.5.5.17 has\_same\_ex\_as()

```
bool GiNaC::archive_node::has_same_ex_as (
    const archive\_node & other ) const
```

Check if the [archive\\_node](#) stores the same expression as another [archive\\_node](#).

##### Returns

"true" if expressions are the same

References [GiNaC::ex::bp](#), [e](#), and [has\\_expression](#).

#### 6.5.5.18 has\_ex()

```
bool GiNaC::archive_node::has_ex ( ) const [inline]
```

References [has\\_expression](#).

#### 6.5.5.19 get\_ex()

```
ex GiNaC::archive_node::get_ex ( ) const [inline]
```

References [e](#).

#### 6.5.5.20 forget()

```
void GiNaC::archive_node::forget ( )
```

Delete cached unarchived expressions from node (for debugging).

References [e](#), and [has\\_expression](#).

Referenced by [GiNaC::archive::forget\(\)](#).

#### 6.5.5.21 printraw()

```
void GiNaC::archive_node::printraw (
    std::ostream & os ) const
```

Output [archive\\_node](#) to stream in ugly raw format (for debugging).

References [a](#), [GiNaC::ex::bp](#), [e](#), [has\\_expression](#), [props](#), [PTYPE\\_BOOL](#), [PTYPE\\_NODE](#), [PTYPE\\_STRING](#), [PTYPE\\_UNSIGNED](#), and [GiNaC::archive::unatomize\(\)](#).



## 6.5.6 Friends And Related Function Documentation

### 6.5.6.1 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const archive_node & ar ) [friend]
```

Write [archive\\_node](#) to binary data stream.

### 6.5.6.2 operator>>

```
std::istream & operator>> (
    std::istream & is,
    archive_node & ar ) [friend]
```

Read [archive\\_node](#) from binary data stream.

## 6.5.7 Member Data Documentation

### 6.5.7.1 a

```
archive& GiNaC::archive_node::a [private]
```

Reference to the archive to which this node belongs.

Referenced by [add\\_bool\(\)](#), [add\\_ex\(\)](#), [add\\_string\(\)](#), [add\\_unsigned\(\)](#), [find\\_bool\(\)](#), [find\\_ex\(\)](#), [find\\_ex\\_by\\_loc\(\)](#), [find\\_ex\\_node\(\)](#), [find\\_first\(\)](#), [find\\_last\(\)](#), [find\\_property\\_range\(\)](#), [find\\_string\(\)](#), [find\\_unsigned\(\)](#), [get\\_properties\(\)](#), and [printraw\(\)](#).

### 6.5.7.2 props

```
std::vector<property> GiNaC::archive_node::props [private]
```

Vector of stored properties.

Referenced by [add\\_bool\(\)](#), [add\\_ex\(\)](#), [add\\_string\(\)](#), [add\\_unsigned\(\)](#), [find\\_bool\(\)](#), [find\\_ex\(\)](#), [find\\_ex\\_node\(\)](#), [find\\_first\(\)](#), [find\\_last\(\)](#), [find\\_property\\_range\(\)](#), [find\\_string\(\)](#), [find\\_unsigned\(\)](#), [get\\_properties\(\)](#), [operator=\(\)](#), and [printraw\(\)](#).

### 6.5.7.3 has\_expression

```
bool GiNaC::archive_node::has_expression [mutable], [private]
```

Flag indicating whether a cached unarchived representation of this node exists.

Referenced by [forget\(\)](#), [has\\_ex\(\)](#), [has\\_same\\_ex\\_as\(\)](#), [operator=\(\)](#), [prinraw\(\)](#), and [unarchive\(\)](#).

### 6.5.7.4 e

```
ex GiNaC::archive_node::e [mutable], [private]
```

The cached unarchived representation of this node (if any).

Referenced by [forget\(\)](#), [get\\_ex\(\)](#), [has\\_same\\_ex\\_as\(\)](#), [operator=\(\)](#), [prinraw\(\)](#), and [unarchive\(\)](#).

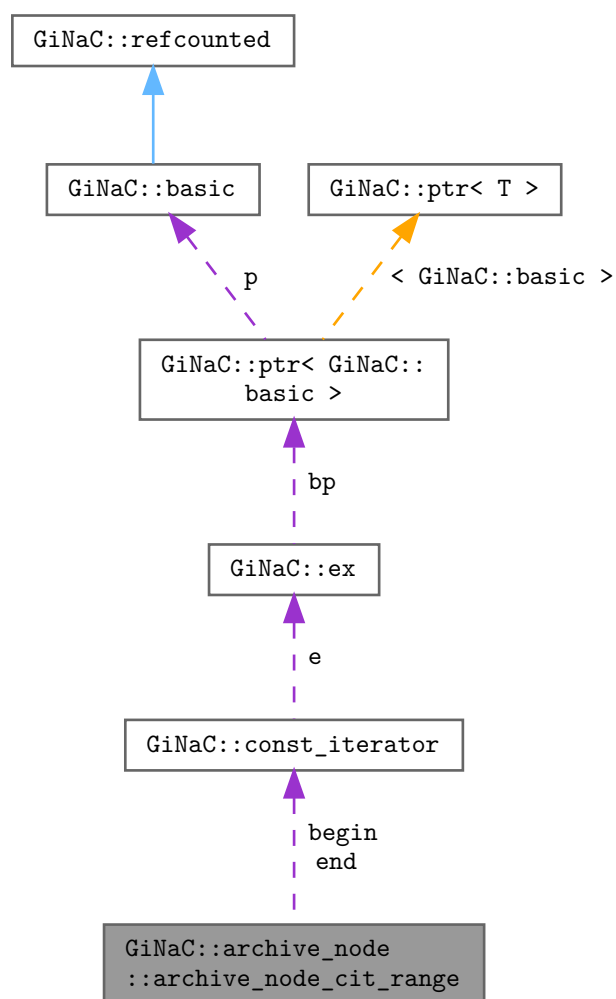
The documentation for this class was generated from the following files:

- [archive.h](#)
- [archive.cpp](#)

## 6.6 GiNaC::archive\_node::archive\_node\_cit\_range Struct Reference

```
#include <archive.h>
```

Collaboration diagram for GiNaC::archive\_node::archive\_node\_cit\_range:



## Public Attributes

- [archive\\_node\\_cit begin](#)
- [archive\\_node\\_cit end](#)

## 6.6.1 Member Data Documentation

### 6.6.1.1 begin

[archive\\_node\\_cit](#) `GiNaC::archive_node::archive_node_cit_range::begin`

Referenced by [GiNaC::archive\\_node::find\\_property\\_range\(\)](#).

### 6.6.1.2 end

`archive_node_cit` `GiNaC::archive_node::archive_node_cit_range::end`

Referenced by `GiNaC::archive_node::find_property_range()`.

The documentation for this struct was generated from the following file:

- `archive.h`

## 6.7 GiNaC::archive::archived\_ex Struct Reference

Archived expression descriptor.

### Public Member Functions

- `archived_ex` ()
- `archived_ex` (`archive_atom` `n`, `archive_node_id` `node`)

### Public Attributes

- `archive_atom` `name`  
*Name of expression.*
- `archive_node_id` `root`  
*ID of root node.*

### 6.7.1 Detailed Description

Archived expression descriptor.

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 archived\_ex() [1/2]

`GiNaC::archive::archived_ex::archived_ex ( )` [inline]

#### 6.7.2.2 archived\_ex() [2/2]

`GiNaC::archive::archived_ex::archived_ex (`  
`archive_atom n,`  
`archive_node_id node )` [inline]

## 6.7.3 Member Data Documentation

### 6.7.3.1 name

`archive_atom` GiNaC::archive::archived\_ex::name

Name of expression.

### 6.7.3.2 root

`archive_node_id` GiNaC::archive::archived\_ex::root

ID of root node.

The documentation for this struct was generated from the following file:

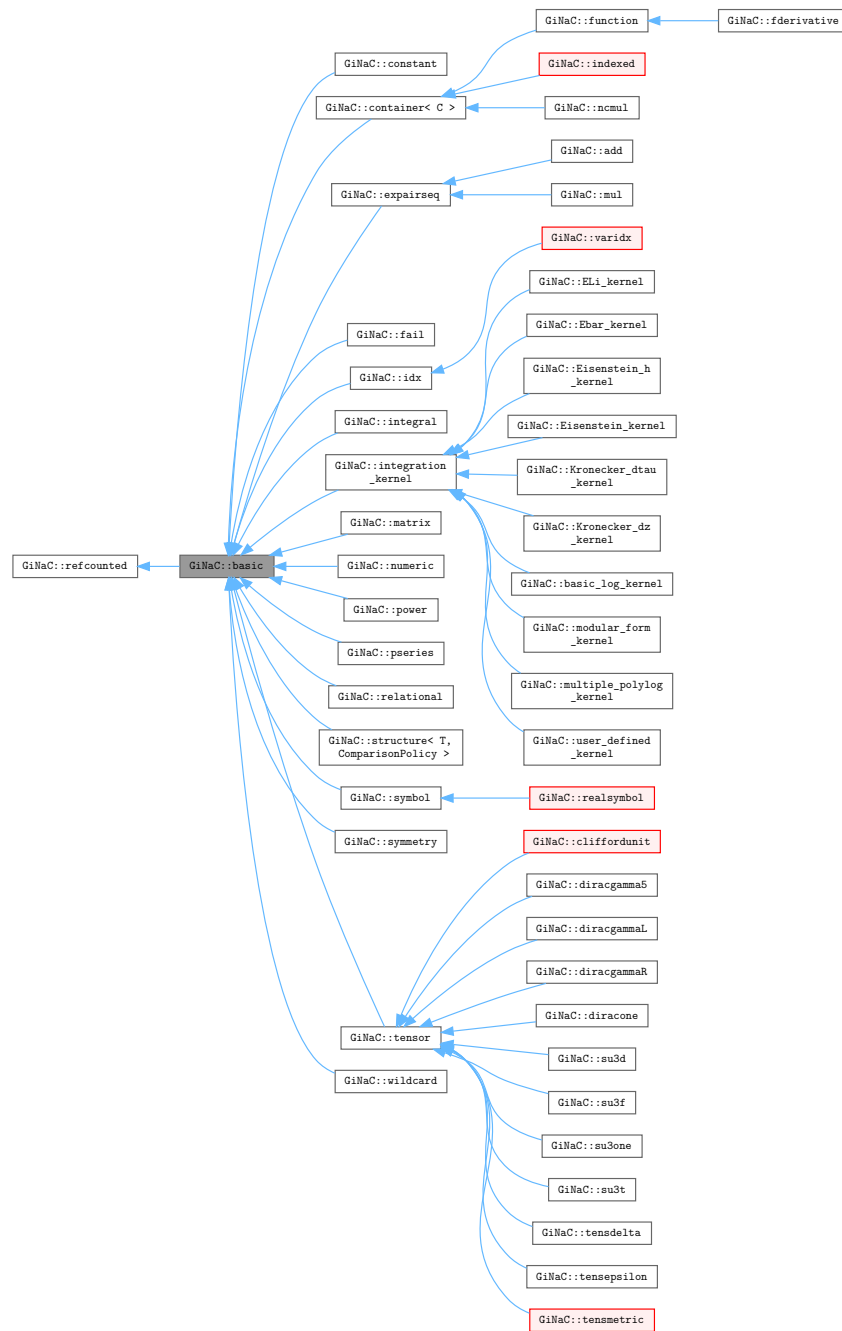
- [archive.h](#)

## 6.8 GiNaC::basic Class Reference

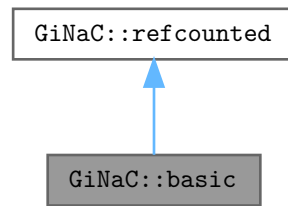
This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.

```
#include <basic.h>
```

Inheritance diagram for GiNaC::basic:



Collaboration diagram for GiNaC::basic:



## Public Member Functions

- virtual `~basic()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic(const basic &other)`
- `const basic &operator=(const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate() const`  
*Create a clone of this object on the heap.*
- virtual `ex eval() const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf() const`  
*Evaluate object numerically.*
- virtual `ex evalm() const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ() const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed(const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print(const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint() const`  
*Little wrapper around `print` to be called within a debugger.*
- virtual `void dbgprntree() const`  
*Little wrapper around `prntree` to be called within a debugger.*
- virtual `unsigned precedence() const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `bool info(unsigned inf) const`  
*Information about the object.*
- virtual `size_t nops() const`  
*Number of operands/members.*
- virtual `ex op(size_t i) const`  
*Return operand/member at position `i`.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`

- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0) const`  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0) const`  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl) const`
- virtual `numeric integer_content () const`
- virtual `ex smod (const numeric &xi) const`  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient () const`  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices () const`  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other) const`  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other) const`  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const`  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `unsigned return_type () const`
- virtual `return_type_t return_type_tinfo () const`
- virtual `ex conjugate () const`
- virtual `ex real_part () const`
- virtual `ex imag_part () const`



- `template<class T >`  
`void print_dispatch (const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `virtual void archive (archive_node &n) const`  
*Save (serialize) the object into archive node.*
- `virtual void read_archive (const archive_node &n, lst &syms)`  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level (const exmap &m, unsigned options) const`  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1) const`  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare (const basic &other) const`  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal (const basic &other) const`  
*Test for syntactic equality.*
- `const basic & hold () const`  
*Stop further evaluation.*
- `unsigned gethash () const`
- `const basic & setflag (unsigned f) const`  
*Set some `status_flags`.*
- `const basic & clearflag (unsigned f) const`  
*Clear some `status_flags`.*

#### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted () noexcept`
- `unsigned int add_reference () noexcept`
- `unsigned int remove_reference () noexcept`
- `unsigned int get_refcount () const noexcept`
- `void set_refcount (unsigned int r) noexcept`

#### Protected Member Functions

- `basic ()`
- `virtual ex eval_ncmul (const exvector &v) const`
- `virtual bool match_same_type (const basic &other) const`  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual ex derivative (const symbol &s) const`  
*Default implementation of `ex::diff()`.*
- `virtual int compare_same_type (const basic &other) const`  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type (const basic &other) const`  
*Returns true if two objects of same type are equal.*
- `virtual unsigned calchash () const`  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void ensure_if_modifiable () const`  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- `void do_print (const print_context &c, unsigned level) const`  
*Default output to stream.*
- `void do_print_tree (const print_tree &c, unsigned level) const`  
*Tree output to stream.*
- `void do_print_python_repr (const print_python_repr &c, unsigned level) const`  
*Python parsable output to stream.*

## Protected Attributes

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## Friends

- class [ex](#)

### 6.8.1 Detailed Description

This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.

### 6.8.2 Constructor & Destructor Documentation

#### 6.8.2.1 `basic()` [1/2]

```
GiNaC::basic::basic ( ) [inline], [protected]
```

Referenced by [duplicate\(\)](#).

#### 6.8.2.2 `~basic()`

```
virtual GiNaC::basic::~~basic ( ) [inline], [virtual]
```

basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.

References [GiNaC::status\\_flags::dynallocated](#), [flags](#), [GiNaC::refcounted::get\\_refcount\(\)](#), and [GINAC\\_ASSERT](#).

#### 6.8.2.3 `basic()` [2/2]

```
GiNaC::basic::basic (
    const basic & other )
```

### 6.8.3 Member Function Documentation

### 6.8.3.1 operator=()

```
const basic & GiNaC::basic::operator= (
    const basic & other )
```

basic assignment operator: the other object might be of a derived class.

References [flags](#), and [hashvalue](#).

### 6.8.3.2 duplicate()

```
virtual basic * GiNaC::basic::duplicate ( ) const [inline], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented in [GiNaC::realsymbol](#), and [GiNaC::possymbol](#).

References [basic\(\)](#), [GiNaC::status\\_flags::dynallocated](#), and [setflag\(\)](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#), [GiNaC::idx::map\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::spinidx::toggle\\_dot\(\)](#), [GiNaC::varidx::toggle\\_variance\(\)](#), and [GiNaC::spinidx::toggle\\_variance\\_dot\(\)](#).

### 6.8.3.3 eval()

```
ex GiNaC::basic::eval ( ) const [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented in [GiNaC::add](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy>](#), and [GiNaC::symbol](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#).

### 6.8.3.4 evalf()

```
ex GiNaC::basic::evalf ( ) const [virtual]
```

Evaluate object numerically.

Reimplemented in [GiNaC::constant](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::symbol](#).

References [GiNaC::nops\(\)](#).

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 6.8.3.5 evalm()

```
ex GiNaC::basic::evalm ( ) const [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented in [GiNaC::add](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::map\\_evalm](#), and [GiNaC::nops\(\)](#).

### 6.8.3.6 eval\_integ()

```
ex GiNaC::basic::eval_integ ( ) const [virtual]
```

Evaluate integrals, if result is known.

Reimplemented in [GiNaC::integral](#), and [GiNaC::pseries](#).

References [GiNaC::map\\_eval\\_integ](#), and [GiNaC::nops\(\)](#).

### 6.8.3.7 eval\_ncmul()

```
ex GiNaC::basic::eval_ncmul (
    const exvector & v ) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::function](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::power](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::hold\\_ncmul\(\)](#).

### 6.8.3.8 eval\_indexed()

```
ex GiNaC::basic::eval_indexed (
    const basic & i ) const [virtual]
```

Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.

Reimplemented in [GiNaC::su3f](#), [GiNaC::su3d](#), [GiNaC::matrix](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::tensdelta](#), [GiNaC::tensmetric](#), [GiNaC::minkmetric](#), [GiNaC::spinmetric](#), and [GiNaC::tensepsilon](#).

### 6.8.3.9 print()

```
void GiNaC::basic::print (
    const print\_context & c,
    unsigned level = 0 ) const [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

## Parameters

<i>c</i>	print context object that describes the output formatting
<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented in [GiNaC::fderivative](#), [GiNaC::function](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [c](#).

Referenced by [GiNaC::fderivative::print\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), and [GiNaC::pseries::print\\_series\(\)](#).

### 6.8.3.10 dbgprint()

```
void GiNaC::basic::dbgprint ( ) const [virtual]
```

Little wrapper around print to be called within a debugger.

This is needed because you cannot call `foo.print(cout)` from within the debugger because it might not know what `cout` is. This method can be invoked with no argument and it will simply print to `stdout`.

See also

[basic::print](#)  
[basic::dbgprinttree](#)

### 6.8.3.11 dbgprinttree()

```
void GiNaC::basic::dbgprinttree ( ) const [virtual]
```

Little wrapper around printtree to be called within a debugger.

See also

[basic::dbgprint](#)

### 6.8.3.12 precedence()

```
unsigned GiNaC::basic::precedence ( ) const [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

### 6.8.3.13 info()

```
bool GiNaC::basic::info (
    unsigned int ) const [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::indexed](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::symbol](#), [GiNaC::tensdelta](#), [GiNaC::tensmetric](#), [GiNaC::minkmetric](#), [GiNaC::spinmetric](#), and [GiNaC::tensepsilon](#).

Referenced by [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::function::info\(\)](#), and [GiNaC::matrix::solve\(\)](#).

### 6.8.3.14 nops()

```
size_t GiNaC::basic::nops ( ) const [virtual]
```

Number of operands/members.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::integral](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::user\\_defined\\_kernel](#), [GiNaC::matrix](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

Referenced by [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), and [normal\(\)](#).

### 6.8.3.15 op()

```
ex GiNaC::basic::op (
    size_t i ) const [virtual]
```

Return operand/member at position i.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::integral](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::user\\_defined\\_kernel](#), [GiNaC::matrix](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

Referenced by [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), and [GiNaC::tensepsilon::eval\\_indexed\(\)](#).

**6.8.3.16 operator[]() [1/4]**

```
ex GiNaC::basic::operator[] (
    const ex & index ) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::op\(\)](#), and [GiNaC::to\\_int\(\)](#).

**6.8.3.17 operator[]() [2/4]**

```
ex GiNaC::basic::operator[] (
    size_t i ) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::op\(\)](#).

**6.8.3.18 let\_op()**

```
ex & GiNaC::basic::let_op (
    size_t i ) [virtual]
```

Return modifiable operand/member at position i.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::integral](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::user\\_defined\\_kernel](#), [GiNaC::matrix](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

Referenced by [map\(\)](#), and [subs\(\)](#).

**6.8.3.19 operator[]() [3/4]**

```
ex & GiNaC::basic::operator[] (
    const ex & index ) [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::to\\_int\(\)](#).

### 6.8.3.20 operator[]() [4/4]

```
ex & GiNaC::basic::operator[] (
    size_t i ) [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

### 6.8.3.21 has()

```
bool GiNaC::basic::has (
    const ex & pattern,
    unsigned options = 0 ) const [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented in [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::ex::has\(\)](#), [GiNaC::match\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), and [options](#).

Referenced by [GiNaC::mul::has\(\)](#), [GiNaC::power::has\(\)](#), and [series\(\)](#).

### 6.8.3.22 match()

```
bool GiNaC::basic::match (
    const ex & pattern,
    exmap & repl_lst ) const [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to `repl_lst`.

Reimplemented in [GiNaC::expairseq](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::wildcard](#).

References [GiNaC::ex::match\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), and [GiNaC::ex::op\(\)](#).



### 6.8.3.23 match\_same\_type()

```
bool GiNaC::basic::match_same_type (
    const basic & other ) const [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::varidx](#), [GiNaC::spinidx](#), [GiNaC::matrix](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

### 6.8.3.24 subs()

```
ex GiNaC::basic::subs (
    const exmap & m,
    unsigned options = 0 ) const [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::matrix](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [clearflag\(\)](#), [let\\_op\(\)](#), [m](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), [options](#), [GiNaC::ex::subs\(\)](#), and [subs\\_one\\_level\(\)](#).

Referenced by [GiNaC::tensmetric::eval\\_indexed\(\)](#).

### 6.8.3.25 map()

```
ex GiNaC::basic::map (
    map_function & f ) const [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented in [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::power](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [clearflag\(\)](#), [let\\_op\(\)](#), [n](#), [GiNaC::nops\(\)](#), and [GiNaC::op\(\)](#).

Referenced by [normal\(\)](#).

#### 6.8.3.26 `accept()`

```
virtual void GiNaC::basic::accept (
    GiNaC::visitor & v ) const [inline], [virtual]
```

#### 6.8.3.27 `is_polynomial()`

```
bool GiNaC::basic::is_polynomial (
    const ex & var ) const [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), and [GiNaC::symbol](#).

References [GiNaC::has\(\)](#).

#### 6.8.3.28 `degree()`

```
int GiNaC::basic::degree (
    const ex & s ) const [virtual]
```

Return degree of highest power in object s.

Reimplemented in [GiNaC::add](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

#### 6.8.3.29 `ldegree()`

```
int GiNaC::basic::ldegree (
    const ex & s ) const [virtual]
```

Return degree of lowest power in object s.

Reimplemented in [GiNaC::add](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

**6.8.3.30 coeff()**

```
ex GiNaC::basic::coeff (
    const ex & s,
    int n = 1 ) const [virtual]
```

Return coefficient of degree n in object s.

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), and [n](#).

Referenced by [GiNaC::expairseq::read\\_archive\(\)](#), [series\(\)](#), and [GiNaC::sqrfree\\_parfrac\(\)](#).

**6.8.3.31 expand()**

```
ex GiNaC::basic::expand (
    unsigned options = 0 ) const [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented in [GiNaC::add](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::nops\(\)](#), and [options](#).

Referenced by [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), and [GiNaC::matrix::pivot\(\)](#).

**6.8.3.32 collect()**

```
ex GiNaC::basic::collect (
    const ex & s,
    bool distributed = false ) const [virtual]
```

Sort expanded expression in terms of powers of some object(s).

**Parameters**

<i>s</i>	object(s) to sort in
<i>distributed</i>	recursive or distributed form (only used when s is a list)

Reimplemented in [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::coeff\(\)](#), [GiNaC::collect\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::degree\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::ldegree\(\)](#), [n](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::pow\(\)](#), and [x](#).

### 6.8.3.33 derivative()

```
ex GiNaC::basic::derivative (
    const symbol & s ) const [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy>](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex0](#), and [GiNaC::nops\(\)](#).

### 6.8.3.34 series()

```
ex GiNaC::basic::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented in [GiNaC::add](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::integral](#), [GiNaC::integration\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy>](#), [GiNaC::mul](#), [GiNaC::power](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [GiNaC::ex::diff\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::ex::expand\(\)](#), [has\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [order](#), [r](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::lgamma\\_series\(\)](#), [GiNaC::fderivative::series\(\)](#), [GiNaC::function::series\(\)](#), and [GiNaC::power::series\(\)](#).

**6.8.3.35 normal()**

```
ex GiNaC::basic::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [virtual]
```

Default implementation of [ex::normal\(\)](#).

It normalizes the children and replaces the object with a temporary symbol.

See also

[ex::normal](#)

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::info\(\)](#), [map\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [nops\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::matrix::markowitz\\_elimination\(\)](#), and [GiNaC::matrix::solve\(\)](#).

**6.8.3.36 to\_rational()**

```
ex GiNaC::basic::to_rational (
    exmap & repl ) const [virtual]
```

Default implementation of [ex::to\\_rational\(\)](#).

This replaces the object with a temporary symbol.

Reimplemented in [GiNaC::expairseq](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.8.3.37 to\_polynomial()**

```
ex GiNaC::basic::to_polynomial (
    exmap & repl ) const [virtual]
```

Reimplemented in [GiNaC::expairseq](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.8.3.38 integer\_content()**

```
numeric GiNaC::basic::integer_content ( ) const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_num1\\_p](#).

**6.8.3.39 smod()**

```
ex GiNaC::basic::smod (
    const numeric & xi ) const [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

**Parameters**

$xi$	modulus
------	---------

**Returns**

mapped polynomial

**See also**

[heur\\_gcd](#)

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

**6.8.3.40 max\_coefficient()**

```
numeric GiNaC::basic::max_coefficient ( ) const [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

**See also**

[heur\\_gcd](#)

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_num1\\_p](#).

**6.8.3.41 get\_free\_indices()**

```
exvector GiNaC::basic::get_free_indices ( ) const [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented in [GiNaC::add](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

**6.8.3.42 add\_indexed()**

```
ex GiNaC::basic::add_indexed (
    const ex & self,
    const ex & other ) const [virtual]
```

Add two indexed expressions.

They are guaranteed to be of class [indexed](#) (or a subclass) and their indices are compatible. This function is used internally by [simplify\\_indexed\(\)](#).

**Parameters**

<i>self</i>	First indexed expression; its base object is *this
<i>other</i>	Second indexed expression

**Returns**

sum of self and other

**See also**

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::matrix](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

**6.8.3.43 scalar\_mul\_indexed()**

```
ex GiNaC::basic::scalar_mul_indexed (
    const ex & self,
    const numeric & other ) const [virtual]
```

Multiply an indexed expression with a scalar.

This function is used internally by [simplify\\_indexed\(\)](#).

## Parameters

<i>self</i>	Indexed expression; its base object is <i>*this</i>
<i>other</i>	Numeric value

## Returns

product of *self* and *other*

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::matrix](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

**6.8.3.44 contract\_with()**

```
bool GiNaC::basic::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [virtual]
```

Try to contract two indexed expressions that appear in the same product.

If a contraction exists, the function overwrites one or both of the expressions and returns true. Otherwise it returns false. It is guaranteed that both expressions are of class indexed (or a subclass) and that at least one dummy index has been found. This functions is used internally by [simplify\\_indexed\(\)](#).

## Parameters

<i>self</i>	Pointer to first indexed expression; its base object is <i>*this</i>
<i>other</i>	Pointer to second indexed expression
<i>v</i>	The complete vector of factors

## Returns

true if the contraction was successful, false otherwise

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::cliffordunit](#), [GiNaC::diracgamma](#), [GiNaC::su3t](#), [GiNaC::su3f](#), [GiNaC::su3d](#), [GiNaC::matrix](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::tensdelta](#), [GiNaC::tensmetric](#), [GiNaC::spinmetric](#), and [GiNaC::tensepsilon](#).



**6.8.3.45 return\_type()**

```
unsigned GiNaC::basic::return_type ( ) const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::su3f](#), [GiNaC::su3d](#), [GiNaC::expairseq](#), [GiNaC::fail](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::tensor](#), [GiNaC::tensdelta](#), [GiNaC::tensmetric](#), [GiNaC::minkmetric](#), and [GiNaC::tensepsilon](#).

**6.8.3.46 return\_type\_tinfo()**

```
return_type_t GiNaC::basic::return_type_tinfo ( ) const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::return\\_type\\_t::rl](#), and [GiNaC::return\\_type\\_t::tinfo](#).

**6.8.3.47 conjugate()**

```
ex GiNaC::basic::conjugate ( ) const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::diracgamma5](#), [GiNaC::diracgammaL](#), [GiNaC::diracgammaR](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::spinidx](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::symbol](#), and [GiNaC::realsymbol](#).

**6.8.3.48 real\_part()**

```
ex GiNaC::basic::real_part ( ) const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::symbol](#), and [GiNaC::realsymbol](#).

Referenced by [GiNaC::function::real\\_part\(\)](#), and [GiNaC::ncmul::real\\_part\(\)](#).

**6.8.3.49 imag\_part()**

```
ex GiNaC::basic::imag_part ( ) const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::symbol](#), and [GiNaC::realsymbol](#).

Referenced by [GiNaC::function::imag\\_part\(\)](#), and [GiNaC::ncmul::imag\\_part\(\)](#).

### 6.8.3.50 `compare_same_type()`

```
int GiNaC::basic::compare_same_type (
    const basic & other ) const [protected], [virtual]
```

Returns order relation between two objects of same type.

This needs to be implemented by each class. It may never return anything else than 0, signalling equality, or +1 and -1 signalling inequality and determining the canonical ordering. (Perl hackers will wonder why C++ doesn't feature the spaceship operator `<=>` for denoting just this.)

References [GiNaC::compare\\_pointers\(\)](#).

Referenced by [GiNaC::symbol::derivative\(\)](#).

### 6.8.3.51 `is_equal_same_type()`

```
bool GiNaC::basic::is_equal_same_type (
    const basic & other ) const [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented in [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::numeric](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

### 6.8.3.52 `calchash()`

```
unsigned GiNaC::basic::calchash ( ) const [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented in [GiNaC::constant](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::numeric](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::symbol](#), [GiNaC::symmetry](#), and [GiNaC::wildcard](#).

References [GiNaC::ex::gethash\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), and [GiNaC::rotate\\_left\(\)](#).

Referenced by [gethash\(\)](#).

**6.8.3.53 print\_dispatch() [1/2]**

```
template<class T >
void GiNaC::basic::print_dispatch (
    const print\_context & c,
    unsigned level ) const [inline]
```

Like [print\(\)](#), but dispatch to the specified class.

Can be used by implementations of print methods to dispatch to the method of the superclass.

See also

[basic::print](#)

References [c](#), and [print\\_dispatch\(\)](#).

Referenced by [print\\_dispatch\(\)](#).

**6.8.3.54 print\_dispatch() [2/2]**

```
void GiNaC::basic::print_dispatch (
    const registered\_class\_info & ri,
    const print\_context & c,
    unsigned level ) const
```

Like [print\(\)](#), but dispatch to the specified class.

Can be used by implementations of print methods to dispatch to the method of the superclass.

See also

[basic::print](#)

References [c](#), [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), and [GiNaC::class\\_info< OPT >::options](#).

**6.8.3.55 archive()**

```
void GiNaC::basic::archive (
    archive\_node & n ) const [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::varidx](#), [GiNaC::spinidx](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::symbol](#), [GiNaC::symmetry](#), [GiNaC::minkmetric](#), [GiNaC::tensepsilon](#), and [GiNaC::wildcard](#).

References [n](#).

### 6.8.3.56 read\_archive()

```
void GiNaC::basic::read_archive (
    const archive_node & n,
    lst & syms ) [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::container< C >](#), [GiNaC::constant](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::varidx](#), [GiNaC::spinidx](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::symbol](#), [GiNaC::symmetry](#), [GiNaC::minkmetric](#), [GiNaC::tensepsilon](#), and [GiNaC::wildcard](#).

### 6.8.3.57 subs\_one\_level()

```
ex GiNaC::basic::subs_one_level (
    const exmap & m,
    unsigned options ) const
```

Helper function for [subs\(\)](#).

Does not recurse into subexpressions.

References [m](#), [GiNaC::match\(\)](#), [options](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [subs\(\)](#), [GiNaC::expairseq::subs\(\)](#), [GiNaC::matrix::subs\(\)](#), [GiNaC::numeric::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::relational::subs\(\)](#), and [GiNaC::symbol::subs\(\)](#).

### 6.8.3.58 diff()

```
ex GiNaC::basic::diff (
    const symbol & s,
    unsigned nth = 1 ) const
```

Default interface of nth derivative `ex::diff(s, n)`.

It should be called instead of `::derivative(s)` for first derivatives and for nth derivatives it just recurses down.

#### Parameters

<i>s</i>	symbol to differentiate in
<i>nth</i>	order of differentiation

See also

[ex::diff](#)

References [GiNaC::ex::diff\(\)](#), and [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [GiNaC::mul::derivative\(\)](#).

### 6.8.3.59 compare()

```
int GiNaC::basic::compare (
    const basic & other ) const
```

Compare objects syntactically to establish canonical ordering.

All compare functions return: -1 for \*this less than other, 0 equal, 1 greater.

References [gethash\(\)](#).

### 6.8.3.60 is\_equal()

```
bool GiNaC::basic::is_equal (
    const basic & other ) const
```

Test for syntactic equality.

This is only a quick test, meaning objects should be in the same domain. You might have to [.expand\(\)](#), [.normal\(\)](#) objects first, depending on the domain of your computation, to get a more reliable answer.

See also

[is\\_equal\\_same\\_type](#)

References [gethash\(\)](#).

Referenced by [GiNaC::ncmul::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::ncmul::degree\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::ncmul::ldegree\(\)](#), [GiNaC::power::ldegree\(\)](#), and [GiNaC::wildcard::match\(\)](#).

### 6.8.3.61 hold()

```
const basic & GiNaC::basic::hold ( ) const
```

Stop further evaluation.

See also

[basic::eval](#)

Referenced by [GiNaC::abs\\_conjugate\(\)](#), [GiNaC::abs\\_eval\(\)](#), [GiNaC::abs\\_evalf\(\)](#), [GiNaC::abs\\_expand\(\)](#), [GiNaC::abs\\_power\(\)](#), [GiNaC::abs\\_real\\_part\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acos\\_evalf\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::acosh\\_evalf\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asin\\_evalf\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::asinh\\_evalf\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_evalf\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::atanh\\_evalf\(\)](#), [GiNaC::binomial\\_conjugate\(\)](#), [GiNaC::binomial\\_eval\(\)](#), [GiNaC::binomial\\_evalf\(\)](#), [GiNaC::binomial\\_real\\_part\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::conjugate\\_expl\\_derivative\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cos\\_evalf\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::cosh\\_evalf\(\)](#), [GiNaC::csgn\\_power\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::eta\\_imag\\_part\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::expairseq::eval\(\)](#), [GiNaC::fderivative::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::numeric::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::pseries::eval\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::eval\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::function::evalf\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::exp\\_evalf\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::exp\\_power\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::factorial\\_conjugate\(\)](#), [GiNaC::factorial\\_eval\(\)](#), [GiNaC::factorial\\_evalf\(\)](#), [GiNaC::factorial\\_real\\_part\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::imag\\_part\\_expl\\_derivative\(\)](#), [GiNaC::iterated\\_integral2\\_eval\(\)](#), [GiNaC::iterated\\_integral3\\_eval\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_evalf\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_evalf\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::log\\_real\\_part\(\)](#), [GiNaC::lorentz\\_eps\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi1\\_evalf\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_evalf\(\)](#), [GiNaC::real\\_part\\_expl\\_derivative\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sin\\_evalf\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::sinh\\_evalf\(\)](#), [GiNaC::step\\_conjugate\(\)](#), [GiNaC::step\\_eval\(\)](#), [GiNaC::step\\_evalf\(\)](#), [GiNaC::step\\_real\\_part\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tan\\_evalf\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [GiNaC::tanh\\_evalf\(\)](#), [GiNaC::zeta1\\_eval\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), [GiNaC::zeta2\\_eval\(\)](#), [GiNaC::zeta2\\_evalf\(\)](#), and [GiNaC::zetaderiv\\_eval\(\)](#).

### 6.8.3.62 gethash()

```
unsigned GiNaC::basic::gethash ( ) const [inline]
```

References [calchash\(\)](#), [flags](#), [GiNaC::status\\_flags::hash\\_calculated](#), and [hashvalue](#).

Referenced by [GiNaC::remember\\_table::add\\_entry\(\)](#), [compare\(\)](#), [is\\_equal\(\)](#), [GiNaC::remember\\_table\\_entry::is\\_equal\(\)](#), and [GiNaC::remember\\_table::lookup\\_entry\(\)](#).

### 6.8.3.63 setflag()

```
const basic & GiNaC::basic::setflag (
    unsigned f ) const [inline]
```

Set some [status\\_flags](#).

References [flags](#).

Referenced by [GiNaC::constant::calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::numeric::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symbol::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), [GiNaC::wildcard::calchash\(\)](#), [GiNaC::constant::constant\(\)](#), [GiNaC::container< C >::container\(\)](#), [duplicate\(\)](#), [GiNaC::realsymbol::duplicate\(\)](#), [GiNaC::possymbol::duplicate\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::add::expand\(\)](#), [GiNaC::expairseq::expand\(\)](#), [GiNaC::function::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASSES\(\)](#), [GiNaC::expairseq::info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::power::info\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::numeric::numeric\(\)](#), [GiNaC::container< C >::read\\_archive\(\)](#), [GiNaC::numeric::read\\_archive\(\)](#), [GiNaC::symbol::read\\_archive\(\)](#), [GiNaC::wildcard::read\\_archive\(\)](#), [GiNaC::reduced\\_matrix\(\)](#), [GiNaC::sub\\_matrix\(\)](#), [GiNaC::symbol::symbol\(\)](#), [GiNaC::symbolic\\_matrix\(\)](#), [GiNaC::symmetry::symmetry\(\)](#), [GiNaC::unit\\_matrix\(\)](#), and [GiNaC::wildcard::wildcard\(\)](#).

### 6.8.3.64 clearflag()

```
const basic & GiNaC::basic::clearflag (
    unsigned f ) const [inline]
```

Clear some [status\\_flags](#).

References [flags](#).

Referenced by [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::function::function\(\)](#), [GiNaC::expairseq::info\(\)](#), [GiNaC::power::info\(\)](#), [map\(\)](#), [GiNaC::idx::map\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), [GiNaC::mul::smod\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::spinidx::toggle\\_dot\(\)](#), [GiNaC::varidx::toggle\\_variance\(\)](#), and [GiNaC::spinidx::toggle\\_variance\\_dot\(\)](#).

### 6.8.3.65 ensure\_if\_modifiable()

```
void GiNaC::basic::ensure_if_modifiable ( ) const [protected]
```

Ensure the object may be modified without hurting others, throws if this is not the case.

Referenced by [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::clifford::let\\_op\(\)](#), [GiNaC::integral::let\\_op\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::let\\_op\(\)](#), [GiNaC::ELi\\_kernel::let\\_op\(\)](#), [GiNaC::Ebar\\_kernel::let\\_op\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::let\\_op\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::let\\_op\(\)](#), [GiNaC::Eisenstein\\_kernel::let\\_op\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::let\\_op\(\)](#), [GiNaC::modular\\_form\\_kernel::let\\_op\(\)](#), [GiNaC::user\\_defined\\_kernel::let\\_op\(\)](#), [GiNaC::matrix::let\\_op\(\)](#), [GiNaC::matrix::operator\(\)\(\)](#), and [GiNaC::matrix::pivot\(\)](#).

#### 6.8.3.66 do\_print()

```
void GiNaC::basic::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

Default output to stream.

References [c](#).

#### 6.8.3.67 do\_print\_tree()

```
void GiNaC::basic::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

Tree output to stream.

References [c](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), and [GiNaC::ex::print\(\)](#).

#### 6.8.3.68 do\_print\_python\_repr()

```
void GiNaC::basic::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

Python parsable output to stream.

References [c](#).

### 6.8.4 Friends And Related Function Documentation

#### 6.8.4.1 ex

```
friend class ex [friend]
```

Referenced by [GiNaC::matrix::charpoly\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), and [GiNaC::structure< T, ComparisonPolicy >::scalar\\_mul\\_indexed\(\)](#).

### 6.8.5 Member Data Documentation



### 6.8.5.1 flags

`unsigned GiNaC::basic::flags [mutable], [protected]`

of type [status\\_flags](#)

Referenced by [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), [clearflag\(\)](#), [GiNaC::ex::construct\\_from\\_basic\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::expairseq::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::integral::expand\(\)](#), [gethash\(\)](#), [GiNaC::expairseq::info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::power::info\(\)](#), [operator=\(\)](#), [GiNaC::function::print\(\)](#), [setflag\(\)](#), and [~basic\(\)](#).

### 6.8.5.2 hashvalue

`unsigned GiNaC::basic::hashvalue [mutable], [protected]`

hash value

Referenced by [GiNaC::constant::calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::numeric::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symbol::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), [GiNaC::wildcard::calchash\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [gethash\(\)](#), [operator=\(\)](#), and [GiNaC::function::print\(\)](#).

The documentation for this class was generated from the following files:

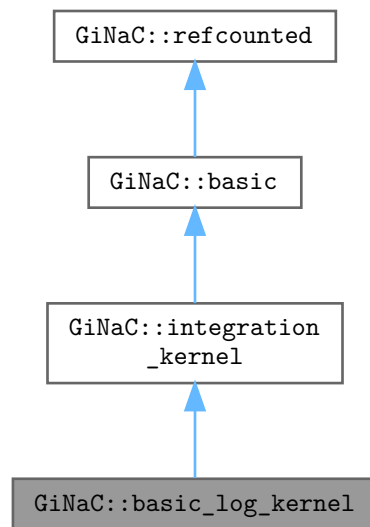
- [basic.h](#)
- [basic.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.9 GiNaC::basic\_log\_kernel Class Reference

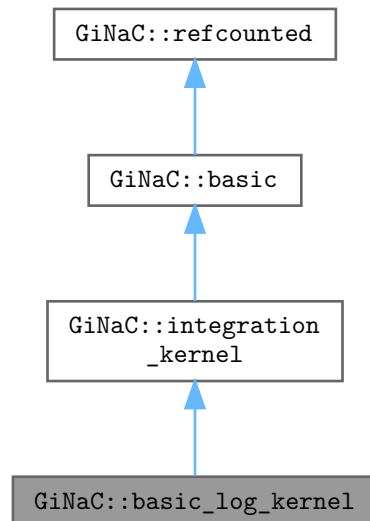
The basic integration kernel with a logarithmic singularity at the origin.

```
#include <integration_kernel.h>
```

Inheritance diagram for `GiNaC::basic_log_kernel`:



Collaboration diagram for `GiNaC::basic_log_kernel`:



## Protected Member Functions

- `cln::cl_N` [series\\_coeff\\_impl](#) (int i) const override

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

- void `do_print` (const `print_context` &c, unsigned level) const

#### Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `uses_Laurent_series` () const  
Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).
- virtual `cln::cl_N` `series_coeff_impl` (int i) const  
For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.
- `ex` `get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
The actual implementation for computing a numerical value for the integrand.
- void `do_print` (const `print_context` &c, unsigned level) const

#### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex` `eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
Returns true if the attributes of two objects are similar enough for a match.
- virtual `ex` `derivative` (const `symbol` &s) const  
Default implementation of `ex::diff()`.
- virtual int `compare_same_type` (const `basic` &other) const  
Returns order relation between two objects of same type.
- virtual bool `is_equal_same_type` (const `basic` &other) const  
Returns true if two objects of same type are equal.
- virtual unsigned `calchash` () const  
Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.
- void `ensure_if_modifiable` () const  
Ensure the object may be modified without hurting others, throws if this is not the case.
- void `do_print` (const `print_context` &c, unsigned level) const  
Default output to stream.
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
Tree output to stream.
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
Python parsable output to stream.

### Additional Inherited Members

#### Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex` `series` (const `relational` &r, int order, unsigned options=0) const override  
Default implementation of `ex::series()`.
- virtual bool `has_trailing_zero` (void) const  
This routine returns true, if the integration kernel has a trailing zero.
- virtual bool `is_numeric` (void) const  
This routine returns true, if the integration kernel can be evaluated numerically.
- virtual `ex` `Laurent_series` (const `ex` &x, int order) const  
Returns the Laurent series, starting possibly with the pole term.

- virtual `ex get_numerical_value` (const `ex` &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- `size_t get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- `cln::cl_N series_coeff` (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual `size_t nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)

- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_info` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*

- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

#### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

#### Protected Attributes inherited from `GiNaC::integration_kernel`

- int `cache_step_size`
- `std::vector< cln::cl_N >` `series_vec`

#### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.9.1 Detailed Description

The basic integration kernel with a logarithmic singularity at the origin.

This class represents the differential one-form

$$L_0 = \frac{d\lambda}{\lambda}$$

## 6.9.2 Member Function Documentation

### 6.9.2.1 series\_coeff\_impl()

```
cln::cl_N GiNaC::basic_log_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.9.2.2 do\_print()

```
void GiNaC::basic_log_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#).

The documentation for this class was generated from the following files:

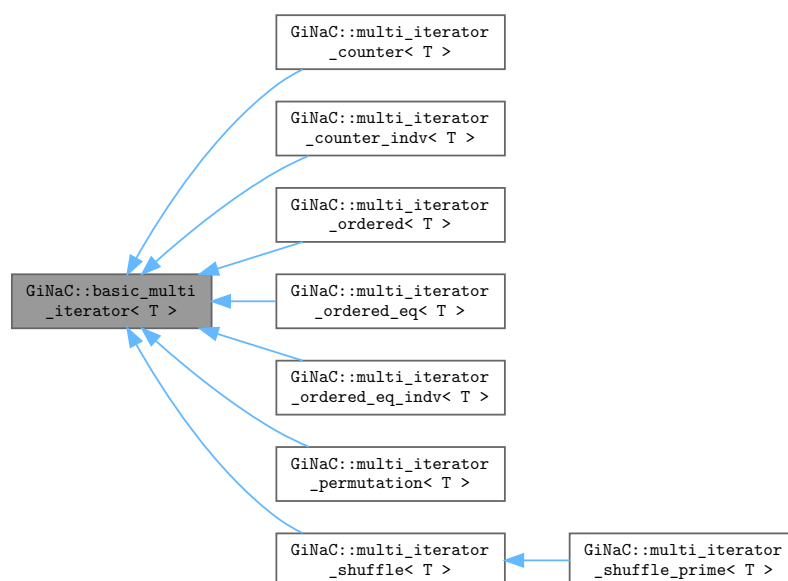
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.10 GiNaC::basic\_multi\_iterator< T > Class Template Reference

[basic\\_multi\\_iterator](#) is a base class.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::basic\_multi\_iterator< T >:



## Public Member Functions

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), size\_t [k](#))  
*Construct a multi\_iterator with upper limit N, lower limit B and size k.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > &[get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*
- virtual [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- virtual [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*No effect for basic\_multi\_iterator.*

## Protected Attributes

- T [N](#)
- T [B](#)
- std::vector< T > [v](#)
- bool [flag\\_overflow](#)

## Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [basic\\_multi\\_iterator](#)< TT > &v)



### 6.10.1 Detailed Description

```
template<class T>
class GiNaC::basic_multi_iterator< T >
```

[basic\\_multi\\_iterator](#) is a base class.

The base class itself does not do anything useful. A typical use of a class derived from [basic\\_multi\\_iterator](#) is

```
multi_iterator_ordered<int> k(0,4,2);
```

```
for( k.init(); !k.overflow(); k++) { std::cout << k << std::endl; }
```

which prints out

```
multi_iterator_ordered(0,1) multi_iterator_ordered(0,2) multi_iterator_ordered(0,3) multi_iterator_ordered(1,2)
multi_iterator_ordered(1,3) multi_iterator_ordered(2,3)
```

Individual components of k can be accessed with `k[i]` or `k(i)`.

All classes derived from [basic\\_multi\\_iterator](#) follow the same syntax.

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 basic\_multi\_iterator() [1/3]

```
template<class T >
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
    void ) [inline]
```

Default constructor.

#### 6.10.2.2 basic\_multi\_iterator() [2/3]

```
template<class T >
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N, lower limit B and size k .

### 6.10.2.3 `basic_multi_iterator()` [3/3]

```
template<class T >
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

### 6.10.2.4 `~basic_multi_iterator()`

```
template<class T >
GiNaC::basic_multi_iterator< T >::~~basic_multi_iterator [inline], [virtual]
```

Destructor.

## 6.10.3 Member Function Documentation

### 6.10.3.1 `size()`

```
template<class T >
size_t GiNaC::basic_multi_iterator< T >::size (
    void ) const [inline]
```

Returns the size of a multi\_iterator.

Referenced by [GiNaC::operator<<\(\)](#).

### 6.10.3.2 `overflow()`

```
template<class T >
bool GiNaC::basic_multi_iterator< T >::overflow (
    void ) const [inline]
```

Return the overflow flag.

### 6.10.3.3 get\_vector()

```
template<class T >
const std::vector< T > & GiNaC::basic_multi_iterator< T >::get_vector (
    void ) const [inline]
```

Returns a reference to the vector v.

### 6.10.3.4 operator[]() [1/2]

```
template<class T >
T GiNaC::basic_multi_iterator< T >::operator[] (
    size_t i ) const [inline]
```

Subscription via [].

### 6.10.3.5 operator[]() [2/2]

```
template<class T >
T & GiNaC::basic_multi_iterator< T >::operator[] (
    size_t i ) [inline]
```

Subscription via [].

### 6.10.3.6 operator()() [1/2]

```
template<class T >
T GiNaC::basic_multi_iterator< T >::operator() (
    size_t i ) const [inline]
```

Subscription via ()

### 6.10.3.7 operator()() [2/2]

```
template<class T >
T & GiNaC::basic_multi_iterator< T >::operator() (
    size_t i ) [inline]
```

Subscription via ()

### 6.10.3.8 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::basic_multi_iterator< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented in [GiNaC::multi\\_iterator\\_ordered< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_counter< T >](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#), [GiNaC::multi\\_iterator\\_permutation< T >](#), [GiNaC::multi\\_iterator\\_shuffle< T >](#), and [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#).

### 6.10.3.9 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::basic_multi_iterator< T >::operator++ (
    int ) [inline], [virtual]
```

No effect for [basic\\_multi\\_iterator](#).

Reimplemented in [GiNaC::multi\\_iterator\\_ordered< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_counter< T >](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#), [GiNaC::multi\\_iterator\\_permutation< T >](#), and [GiNaC::multi\\_iterator\\_shuffle< T >](#).

## 6.10.4 Friends And Related Function Documentation

### 6.10.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
    std::ostream & os,
    const basic_multi_iterator< TT > & v ) [friend]
```

## 6.10.5 Member Data Documentation

### 6.10.5.1 N

```
template<class T >
T GiNaC::basic_multi_iterator< T >::N [protected]
```

## 6.10.5.2 B

```
template<class T >
T GiNaC::basic_multi_iterator< T >::B [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#), and [GiNaC::operator<<\(\)](#).

## 6.10.5.3 v

```
template<class T >
std::vector<T> GiNaC::basic_multi_iterator< T >::v [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

## 6.10.5.4 flag\_overflow

```
template<class T >
bool GiNaC::basic_multi_iterator< T >::flag_overflow [protected]
```

The documentation for this class was generated from the following file:

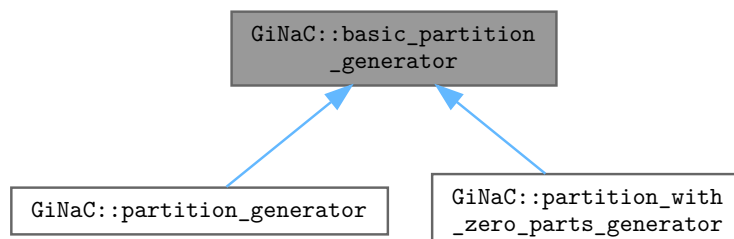
- [utils\\_multi\\_iterator.h](#)

## 6.11 GiNaC::basic\_partition\_generator Class Reference

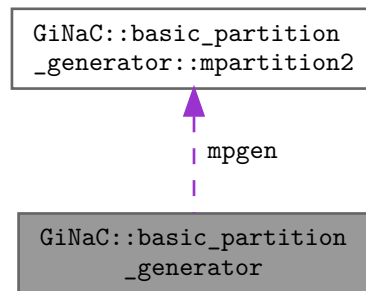
Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for GiNaC::basic\_partition\_generator:



Collaboration diagram for GiNaC::basic\_partition\_generator:



## Classes

- struct [mpartition2](#)

## Protected Member Functions

- [basic\\_partition\\_generator](#) (unsigned *n\_*, unsigned *m\_*)

## Protected Attributes

- [mpartition2](#) [mpgen](#)

### 6.11.1 Detailed Description

Base class for generating all bounded combinatorial partitions of an integer *n* with exactly *m* parts in non-decreasing order.

### 6.11.2 Constructor & Destructor Documentation

#### 6.11.2.1 basic\_partition\_generator()

```

GiNaC::basic_partition_generator::basic_partition_generator (
    unsigned n_,
    unsigned m_ ) [inline], [protected]
  
```

### 6.11.3 Member Data Documentation

#### 6.11.3.1 mpngen

`mpartition2` GiNaC::basic\_partition\_generator::mpngen [protected]

Referenced by [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::partition\\_generator::get\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#) and [GiNaC::partition\\_generator::next\(\)](#).

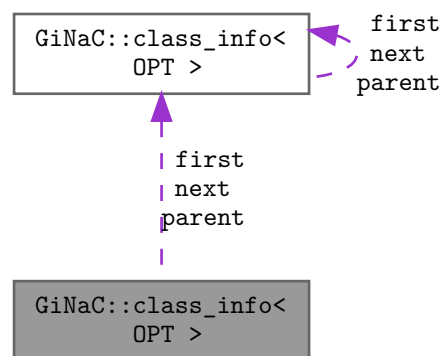
The documentation for this class was generated from the following file:

- [utils.h](#)

## 6.12 GiNaC::class\_info< OPT > Class Template Reference

```
#include <class_info.h>
```

Collaboration diagram for GiNaC::class\_info< OPT >:



### Classes

- struct [tree\\_node](#)

### Public Member Functions

- [class\\_info](#) (const OPT &o)
- [class\\_info](#) \* [get\\_parent](#) () const  
Get pointer to [class\\_info](#) of parent class (or nullptr).

## Static Public Member Functions

- static const [class\\_info](#) \* [find](#) (const std::string &class\_name)  
*Find [class\\_info](#) by name.*
- static void [dump\\_hierarchy](#) (bool verbose=false)  
*Dump class hierarchy to std::cout.*

## Public Attributes

- OPT [options](#)

## Static Private Member Functions

- static void [dump\\_tree](#) ([tree\\_node](#) \*n, const std::string &prefix, bool verbose)
- static void [identify\\_parents](#) ()

## Private Attributes

- [class\\_info](#) \* [next](#)
- [class\\_info](#) \* [parent](#)

## Static Private Attributes

- static [class\\_info](#) \* [first](#) = nullptr
- static bool [parents\\_identified](#) = false

## 6.12.1 Constructor & Destructor Documentation

### 6.12.1.1 [class\\_info](#)()

```
template<class OPT >
GiNaC::class_info< OPT >::class_info (
    const OPT & o ) [inline]
```

References [GiNaC::class\\_info< OPT >::first](#), and [GiNaC::class\\_info< OPT >::parents\\_identified](#).

## 6.12.2 Member Function Documentation



### 6.12.2.1 get\_parent()

```
template<class OPT >
class_info * GiNaC::class_info< OPT >::get_parent ( ) const [inline]
```

Get pointer to [class\\_info](#) of parent class (or nullptr).

References [GiNaC::class\\_info< OPT >::identify\\_parents\(\)](#), and [GiNaC::class\\_info< OPT >::parent](#).

Referenced by [GiNaC::class\\_info< OPT >::dump\\_hierarchy\(\)](#), [GiNaC::function::print\(\)](#), and [GiNaC::basic::print\\_dispatch\(\)](#).

### 6.12.2.2 find()

```
template<class OPT >
const class_info< OPT > * GiNaC::class_info< OPT >::find (
    const std::string & class_name ) [static]
```

Find [class\\_info](#) by name.

References [GiNaC::class\\_info< OPT >::find\(\)](#), [GiNaC::class\\_info< OPT >::next](#), and [GiNaC::class\\_info< OPT >::options](#).

Referenced by [GiNaC::class\\_info< OPT >::find\(\)](#).

### 6.12.2.3 dump\_hierarchy()

```
template<class OPT >
void GiNaC::class_info< OPT >::dump_hierarchy (
    bool verbose = false ) [static]
```

Dump class hierarchy to `std::cout`.

References [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), [GiNaC::class\\_info< OPT >::next](#), and [GiNaC::tree\(\)](#).

### 6.12.2.4 dump\_tree()

```
template<class OPT >
void GiNaC::class_info< OPT >::dump_tree (
    tree_node * n,
    const std::string & prefix,
    bool verbose ) [static], [private]
```

References [n](#).

#### 6.12.2.5 identify\_parents()

```
template<class OPT >
void GiNaC::class_info< OPT >::identify_parents [static], [private]
```

References [GiNaC::class\\_info< OPT >::next](#).

Referenced by [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#).

### 6.12.3 Member Data Documentation

#### 6.12.3.1 options

```
template<class OPT >
OPT GiNaC::class_info< OPT >::options
```

Referenced by [GiNaC::class\\_info< OPT >::find\(\)](#), [GiNaC::function::print\(\)](#), and [GiNaC::basic::print\\_dispatch\(\)](#).

#### 6.12.3.2 first

```
template<class OPT >
class_info< OPT > * GiNaC::class_info< OPT >::first = nullptr [static], [private]
```

Referenced by [GiNaC::class\\_info< OPT >::class\\_info\(\)](#).

#### 6.12.3.3 next

```
template<class OPT >
class_info* GiNaC::class_info< OPT >::next [private]
```

Referenced by [GiNaC::class\\_info< OPT >::dump\\_hierarchy\(\)](#), [GiNaC::class\\_info< OPT >::find\(\)](#), and [GiNaC::class\\_info< OPT >::i](#)

#### 6.12.3.4 parent

```
template<class OPT >
class_info* GiNaC::class_info< OPT >::parent [mutable], [private]
```

Referenced by [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#).

### 6.12.3.5 parents\_identified

```
template<class OPT >
bool GiNaC::class_info< OPT >::parents_identified = false [static], [private]
```

Referenced by [GiNaC::class\\_info< OPT >::class\\_info\(\)](#).

The documentation for this class was generated from the following file:

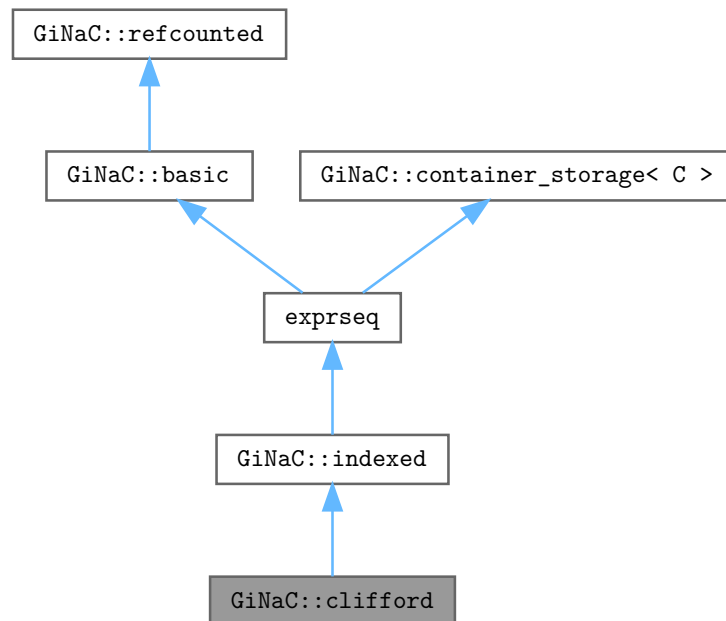
- [class\\_info.h](#)

## 6.13 GiNaC::clifford Class Reference

This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::clifford:





*Return modifiable operand/member at position  $i$ .*

- `ex subs` (const `exmap` &`m`, unsigned `options`=0) const override

*Substitute a set of objects by arbitrary expressions.*

### Public Member Functions inherited from `GiNaC::indexed`

- `indexed` (const `ex` &`b`)  
*Construct indexed object with no index.*
- `indexed` (const `ex` &`b`, const `ex` &`i1`)  
*Construct indexed object with one index.*
- `indexed` (const `ex` &`b`, const `ex` &`i1`, const `ex` &`i2`)  
*Construct indexed object with two indices.*
- `indexed` (const `ex` &`b`, const `ex` &`i1`, const `ex` &`i2`, const `ex` &`i3`)  
*Construct indexed object with three indices.*
- `indexed` (const `ex` &`b`, const `ex` &`i1`, const `ex` &`i2`, const `ex` &`i3`, const `ex` &`i4`)  
*Construct indexed object with four indices.*
- `indexed` (const `ex` &`b`, const `symmetry` &`symm`, const `ex` &`i1`, const `ex` &`i2`)  
*Construct indexed object with two indices and a specified symmetry.*
- `indexed` (const `ex` &`b`, const `symmetry` &`symm`, const `ex` &`i1`, const `ex` &`i2`, const `ex` &`i3`)  
*Construct indexed object with three indices and a specified symmetry.*
- `indexed` (const `ex` &`b`, const `symmetry` &`symm`, const `ex` &`i1`, const `ex` &`i2`, const `ex` &`i3`, const `ex` &`i4`)  
*Construct indexed object with four indices and a specified symmetry.*
- `indexed` (const `ex` &`b`, const `exvector` &`iv`)  
*Construct indexed object with a specified vector of indices.*
- `indexed` (const `ex` &`b`, const `symmetry` &`symm`, const `exvector` &`iv`)  
*Construct indexed object with a specified vector of indices and symmetry.*
- `indexed` (const `symmetry` &`symm`, const `exprseq` &`es`)
- `indexed` (const `symmetry` &`symm`, const `exvector` &`v`)
- `indexed` (const `symmetry` &`symm`, `exvector` &&`v`)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned `inf`) const override  
*Information about the object.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex real_part` () const override
- `ex imag_part` () const override
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- void `archive` (`archive_node` &`n`) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
*Read (a.k.a.*
- bool `all_index_values_are` (unsigned `inf`) const  
*Check whether all index values have a certain property.*
- `exvector get_indices` () const  
*Return a vector containing the object's indices.*
- `exvector get_dummy_indices` () const  
*Return a vector containing the dummy indices of the object, if any.*
- `exvector get_dummy_indices` (const `indexed` &`other`) const

*Return a vector containing the dummy indices in the contraction with another indexed object.*

- `bool has_dummy_index_for (const ex &i) const`

*Check whether the object has an index that forms a dummy index pair with a given index.*

- `ex get_symmetry () const`

*Return symmetry properties.*

## Public Member Functions inherited from [GiNaC::container< C >](#)

- `container (STLT const &s)`
- `container (STLT &&v)`
- `container (exvector::const_iterator b, exvector::const_iterator e)`
- `container (std::initializer_list< ex > il)`
- `bool info (unsigned inf) const override`  
*Information about the object.*
- `unsigned precedence () const override`  
*Return relative operator precedence (for parenthezing output).*
- `size_t nops () const override`  
*Number of operands/members.*
- `ex op (size_t i) const override`  
*Return operand/member at position i.*
- `ex & let\_op (size_t i) override`  
*Return modifiable operand/member at position i.*
- `ex subs (const exmap &m, unsigned options=0) const override`  
*Substitute a set of objects by arbitrary expressions.*
- `void read\_archive (const archive\_node &n, lst &sym_lst) override`  
*Load (deserialize) the object from an archive node.*
- `void archive (archive\_node &n) const override`  
*Archive the object.*
- `container & prepend (const ex &b)`  
*Add element at front.*
- `container & append (const ex &b)`  
*Add element at back.*
- `container & remove\_first ()`  
*Remove first element.*
- `container & remove\_last ()`  
*Remove last element.*
- `container & remove\_all ()`  
*Remove all elements.*
- `container & sort ()`  
*Sort elements.*
- `container & unique ()`  
*Remove adjacent duplicate elements.*
- `const_iterator begin () const`
- `const_iterator end () const`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rend () const`

Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*



- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

#### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

#### Protected Member Functions

- [ex eval\\_ncmul](#) (const [exvector](#) &v) const override  
*Perform automatic simplification on noncommutative product of clifford objects.*
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- [ex thiscontainer](#) (const [exvector](#) &v) const override
- [ex thiscontainer](#) ([exvector](#) &&v) const override
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- void [do\\_print\\_dflt](#) (const [print\\_dflt](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

#### Protected Member Functions inherited from [GiNaC::indexed](#)

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for an indexed object always returns 0.*
- [ex thiscontainer](#) (const [exvector](#) &v) const override
- [ex thiscontainer](#) ([exvector](#) &&v) const override
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- void [printindices](#) (const [print\\_context](#) &c, unsigned level) const
- void [print\\_indexed](#) (const [print\\_context](#) &c, const char \*openbrace, const char \*closebrace, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [validate](#) () const  
*Check whether all indices are of class [idx](#) and validate the symmetry tree.*

### Protected Member Functions inherited from [GiNaC::container< C >](#)

- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- virtual [ex thiscontainer](#) (const [STLT](#) &v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- virtual [ex thiscontainer](#) ([STLT](#) &&v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this↔\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned [options](#)=0) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

## Protected Attributes

- unsigned char [representation\\_label](#)  
*Representation label to distinguish independent spin lines.*
- [ex metric](#)  
*Metric of the space, all constructors make it an indexed object.*
- int [commutator\\_sign](#)  
*It is the sign in the definition  $e \sim i \ e \sim j \ +/- \ e \sim j \ e \sim i = B(i, j) + B(j, i)$*

## Protected Attributes inherited from [GiNaC::indexed](#)

- [ex symtree](#)  
*Index symmetry (tree of symmetry objects)*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)

## Additional Inherited Members

### Public Types inherited from [GiNaC::container< C >](#)

- typedef [STLT::const\\_iterator](#) [const\\_iterator](#)
- typedef [STLT::const\\_reverse\\_iterator](#) [const\\_reverse\\_iterator](#)

### Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

### Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef [C< ex >](#) [STLT](#)

### Static Protected Member Functions inherited from [GiNaC::container< C >](#)

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for [lst](#).*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for [lst](#).*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for [lst](#).*

## Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) (STLT &, size\_t)

### 6.13.1 Detailed Description

This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).

These objects only carry Lorentz indices. Spinor indices are hidden. A representation label (an unsigned 8-bit integer) is used to distinguish elements from different Clifford algebras (objects with different labels commute).

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 [clifford\(\)](#) [1/4]

```
GiNaC::clifford::clifford (
    const ex & b,
    unsigned char rl = 0 )
```

Construct object without any indices.

This constructor is for internal use only. Use the [dirac\\_ONE\(\)](#) function instead.

See also

[dirac\\_ONE](#)

#### 6.13.2.2 [clifford\(\)](#) [2/4]

```
GiNaC::clifford::clifford (
    const ex & b,
    const ex & mu,
    const ex & metr,
    unsigned char rl = 0,
    int comm_sign = -1 )
```

Construct object with one Lorentz index.

This constructor is for internal use only. Use the [clifford\\_unit\(\)](#) or [dirac\\_gamma\(\)](#) functions instead.

See also

[clifford\\_unit](#)

[dirac\\_gamma](#)

References [GINAC\\_ASSERT](#).

**6.13.2.3 clifford()** [3/4]

```
GiNaC::clifford::clifford (
    unsigned char rl,
    const ex & metr,
    int comm_sign,
    const exvector & v )
```

**6.13.2.4 clifford()** [4/4]

```
GiNaC::clifford::clifford (
    unsigned char rl,
    const ex & metr,
    int comm_sign,
    exvector && v )
```

**6.13.3 Member Function Documentation****6.13.3.1 precedence()**

```
unsigned GiNaC::clifford::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\\_dflt\(\)](#), and [do\\_print\\_latex\(\)](#).

**6.13.3.2 archive()**

```
void GiNaC::clifford::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [commutator\\_sign](#), [metric](#), [n](#), and [representation\\_label](#).

### 6.13.3.3 read\_archive()

```
void GiNaC::clifford::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [commutator\\_sign](#), [metric](#), [n](#), and [representation\\_label](#).

### 6.13.3.4 eval\_ncmul()

```
ex GiNaC::clifford::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Perform automatic simplification on noncommutative product of clifford objects.

This removes superfluous ONEs, permutes gamma5/L/R's to the front and removes squares of gamma objects.

Reimplemented from [GiNaC::basic](#).

### 6.13.3.5 match\_same\_type()

```
bool GiNaC::clifford::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

#### See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [commutator\\_sign](#), [get\\_commutator\\_sign\(\)](#), [GINAC\\_ASSERT](#), [representation\\_label](#), and [same\\_metric\(\)](#).

#### 6.13.3.6 thiscontainer() [1/2]

```
ex GiNaC::clifford::thiscontainer (
    const exvector & v ) const [override], [protected]
```

#### 6.13.3.7 thiscontainer() [2/2]

```
ex GiNaC::clifford::thiscontainer (
    exvector && v ) const [override], [protected]
```

#### 6.13.3.8 return\_type()

```
unsigned GiNaC::clifford::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

#### 6.13.3.9 return\_type\_tinfo()

```
return\_type\_t GiNaC::clifford::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [representation\\_label](#).

#### 6.13.3.10 get\_representation\_label()

```
unsigned char GiNaC::clifford::get_representation_label ( ) const [inline]
```

References [representation\\_label](#).

#### 6.13.3.11 get\_metric() [1/2]

```
ex GiNaC::clifford::get_metric ( ) const [inline]
```

References [metric](#).

Referenced by [same\\_metric\(\)](#).

**6.13.3.12 `get_metric()` [2/2]**

```
ex GiNaC::clifford::get_metric (
    const ex & i,
    const ex & j,
    bool symmetrised = false ) const [virtual]
```

References [GiNaC::\\_ex1\\_2](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::indexed::get\\_symmetry\(\)](#), [metric](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::symmetric2\(\)](#).

**6.13.3.13 `same_metric()`**

```
bool GiNaC::clifford::same_metric (
    const ex & other ) const
```

References [GiNaC::ex::get\\_free\\_indices\(\)](#), [get\\_metric\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), and [GiNaC::indexed::simplify\\_indexed](#).

Referenced by [match\\_same\\_type\(\)](#).

**6.13.3.14 `get_commutator_sign()`**

```
int GiNaC::clifford::get_commutator_sign ( ) const [inline]
```

References [commutator\\_sign](#).

Referenced by [match\\_same\\_type\(\)](#).

**6.13.3.15 `nops()`**

```
size_t GiNaC::clifford::nops ( ) const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

Referenced by [let\\_op\(\)](#), and [op\(\)](#).



**6.13.3.16 op()**

```
ex GiNaC::clifford::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [nops\(\)](#), and [representation\\_label](#).

Referenced by [same\\_metric\(\)](#).

**6.13.3.17 let\_op()**

```
ex & GiNaC::clifford::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [nops\(\)](#), and [representation\\_label](#).

**6.13.3.18 subs()**

```
ex GiNaC::clifford::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [c](#), [m](#), [options](#), and [GiNaC::ex::subs\(\)](#).

**6.13.3.19 do\_print\_dflt()**

```
void GiNaC::clifford::do_print_dflt (
    const print_dflt & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::is\\_dirac\\_slash\(\)](#), [precedence\(\)](#), [GiNaC::indexed::printindices\(\)](#), [representation\\_label](#), and [GiNaC::container\\_storage< C >::seq](#).

### 6.13.3.20 do\_print\_latex()

```
void GiNaC::clifford::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::is\\_dirac\\_slash\(\)](#), [precedence\(\)](#), [representation\\_label](#), and [GiNaC::container\\_storage< C >::seq](#).

### 6.13.3.21 do\_print\_tree()

```
void GiNaC::clifford::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [metric](#), [GiNaC::ex::print\(\)](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::indexed::symtree](#).

## 6.13.4 Member Data Documentation

### 6.13.4.1 representation\_label

```
unsigned char GiNaC::clifford::representation_label [protected]
```

Representation label to distinguish independent spin lines.

Referenced by [archive\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [get\\_representation\\_label\(\)](#), [let\\_op\(\)](#), [match\\_same\\_type\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [return\\_type\\_tinfo\(\)](#).

### 6.13.4.2 metric

```
ex GiNaC::clifford::metric [protected]
```

Metric of the space, all constructors make it an indexed object.

Referenced by [archive\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_metric\(\)](#), and [read\\_archive\(\)](#).

### 6.13.4.3 commutator\_sign

```
int GiNaC::clifford::commutator_sign [protected]
```

It is the sign in the definition  $e_i e_j \pm e_j e_i = B(i, j) + B(j, i)$

Referenced by [archive\(\)](#), [get\\_commutator\\_sign\(\)](#), [match\\_same\\_type\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

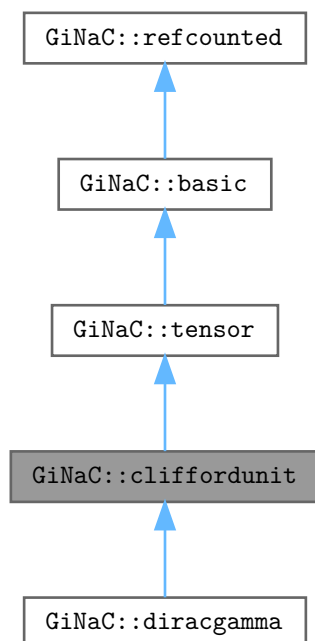
- [clifford.h](#)
- [clifford.cpp](#)

## 6.14 GiNaC::cliffordunit Class Reference

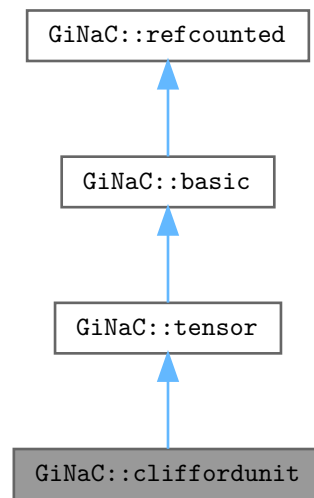
This class represents the Clifford algebra generators (units).

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::cliffordunit:



Collaboration diagram for `GiNaC::cliffordunit`:



## Public Member Functions

- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of a Clifford unit with something else.*

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*

- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex &let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex &operator[] (const ex &index)`
- virtual `ex &operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.14.1 Detailed Description

This class represents the Clifford algebra generators (units).

### 6.14.2 Member Function Documentation

#### 6.14.2.1 `contract_with()`

```
bool GiNaC::cliffordunit::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of a Clifford unit with something else.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::diracgamma](#).

#### 6.14.2.2 `do_print()`

```
void GiNaC::cliffordunit::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

#### 6.14.2.3 `do_print_latex()`

```
void GiNaC::cliffordunit::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

- [clifford.h](#)
- [clifford.cpp](#)

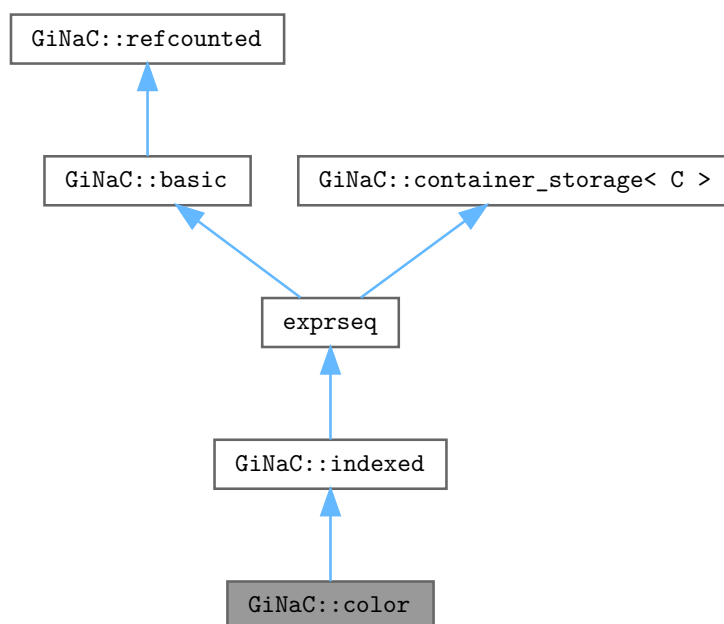
## 6.15 `GiNaC::color` Class Reference

This class holds a generator `T_a` or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.

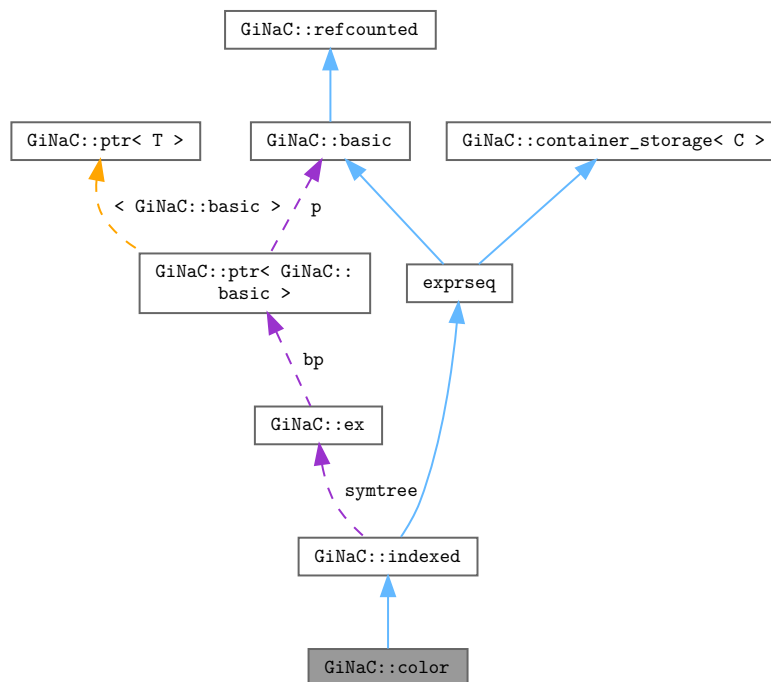
```
#include <color.h>
```



Inheritance diagram for GiNaC::color:



Collaboration diagram for `GiNaC::color`:



## Public Member Functions

- `color` (const `ex` &b, unsigned char rl=0)  
*Construct object without any color index.*
- `color` (const `ex` &b, const `ex` &i1, unsigned char rl=0)  
*Construct object with one color index.*
- `color` (unsigned char rl, const `exvector` &v)
- `color` (unsigned char rl, `exvector` &&v)
- void `archive` (`archive_node` &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- unsigned char `get_representation_label` () const

## Public Member Functions inherited from `GiNaC::indexed`

- `indexed` (const `ex` &b)  
*Construct indexed object with no index.*
- `indexed` (const `ex` &b, const `ex` &i1)  
*Construct indexed object with one index.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2)  
*Construct indexed object with two indices.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3)

- Construct indexed object with three indices.
  - `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)
  - Construct indexed object with four indices.
  - `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2)
  - Construct indexed object with two indices and a specified symmetry.
  - `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3)
  - Construct indexed object with three indices and a specified symmetry.
  - `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)
  - Construct indexed object with four indices and a specified symmetry.
  - `indexed` (const `ex` &b, const `exvector` &iv)
  - Construct indexed object with a specified vector of indices.
  - `indexed` (const `ex` &b, const `symmetry` &symm, const `exvector` &iv)
  - Construct indexed object with a specified vector of indices and symmetry.
  - `indexed` (const `symmetry` &symm, const `exprseq` &es)
  - `indexed` (const `symmetry` &symm, const `exvector` &v)
  - `indexed` (const `symmetry` &symm, `exvector` &&v)
  - unsigned `precedence` () const override
  - Return relative operator precedence (for parenthezing output).
  - bool `info` (unsigned inf) const override
  - Information about the object.
  - `ex eval` () const override
  - Perform automatic non-interruptive term rewriting rules.
  - `ex real_part` () const override
  - `ex imag_part` () const override
  - `exvector get_free_indices` () const override
  - Return a vector containing the free indices of an expression.
  - void `archive` (`archive_node` &n) const override
  - Save (a.k.a.
  - void `read_archive` (const `archive_node` &n, `lst` &sylms) override
  - Read (a.k.a.
  - bool `all_index_values_are` (unsigned inf) const
  - Check whether all index values have a certain property.
  - `exvector get_indices` () const
  - Return a vector containing the object's indices.
  - `exvector get_dummy_indices` () const
  - Return a vector containing the dummy indices of the object, if any.
  - `exvector get_dummy_indices` (const `indexed` &other) const
  - Return a vector containing the dummy indices in the contraction with another indexed object.
  - bool `has_dummy_index_for` (const `ex` &i) const
  - Check whether the object has an index that forms a dummy index pair with a given index.
  - `ex get_symmetry` () const
  - Return symmetry properties.

#### Public Member Functions inherited from `GiNaC::container< C >`

- `container` (STLT const &s)
- `container` (STLT &&v)
- `container` (`exvector::const_iterator` b, `exvector::const_iterator` e)
- `container` (std::initializer\_list< `ex` > il)
- bool `info` (unsigned inf) const override
- Information about the object.

- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- size\_t `nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex subs` (const `exmap` &m, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- void `read_archive` (const `archive_node` &n, `lst` &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- void `archive` (`archive_node` &n) const override  
*Archive the object.*
- `container & prepend` (const `ex` &b)  
*Add element at front.*
- `container & append` (const `ex` &b)  
*Add element at back.*
- `container & remove_first` ()  
*Remove first element.*
- `container & remove_last` ()  
*Remove last element.*
- `container & remove_all` ()  
*Remove all elements.*
- `container & sort` ()  
*Sort elements.*
- `container & unique` ()  
*Remove adjacent duplicate elements.*
- `const_iterator begin` () const
- `const_iterator end` () const
- `const_reverse_iterator rbegin` () const
- `const_reverse_iterator rend` () const

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic & operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*

- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const

- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &`v`) const override  
*Perform automatic simplification on noncommutative product of color objects.*
- `bool match_same_type` (const `basic` &`other`) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex thiscontainer` (const `exvector` &`v`) const override
- `ex thiscontainer` (`exvector` &&`v`) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override

## Protected Member Functions inherited from `GiNaC::indexed`

- `ex derivative` (const `symbol` &`s`) const override  
*Implementation of `ex::diff()` for an indexed object always returns 0.*
- `ex thiscontainer` (const `exvector` &`v`) const override
- `ex thiscontainer` (`exvector` &&`v`) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- void `printindices` (const `print_context` &`c`, unsigned level) const
- void `print_indexed` (const `print_context` &`c`, const char \*`openbrace`, const char \*`closebrace`, unsigned level) const
- void `do_print` (const `print_context` &`c`, unsigned level) const
- void `do_print_latex` (const `print_latex` &`c`, unsigned level) const
- void `do_print_tree` (const `print_tree` &`c`, unsigned level) const
- void `validate` () const  
*Check whether all indices are of class `idx` and validate the symmetry tree.*

## Protected Member Functions inherited from `GiNaC::container< C >`

- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- `bool is_equal_same_type` (const `basic` &`other`) const override  
*Returns true if two objects of same type are equal.*
- virtual `ex thiscontainer` (const `STLT` &`v`) const  
*Similar to `duplicate()`, but with a preset sequence.*
- virtual `ex thiscontainer` (`STLT` &&`v`) const  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- virtual void `printseq` (const `print_context` &`c`, char `openbracket`, char `delim`, char `closebracket`, unsigned `this↔_precedence`, unsigned `upper_precedence`=0) const  
*Print sequence of contained elements.*
- void `do_print` (const `print_context` &`c`, unsigned level) const
- void `do_print_tree` (const `print_tree` &`c`, unsigned level) const
- void `do_print_python` (const `print_python` &`c`, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &`c`, unsigned level) const
- `STLT subchildren` (const `exmap` &`m`, unsigned `options`=0) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

### Private Attributes

- unsigned char [representation\\_label](#)  
*Representation label to distinguish independent color matrices coming from separated fermion lines.*

### Additional Inherited Members

#### Public Types inherited from [GiNaC::container< C >](#)

- typedef STLT::const\_iterator [const\\_iterator](#)
- typedef STLT::const\_reverse\_iterator [const\\_reverse\\_iterator](#)

#### Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >::STLT](#) [STLT](#)



**Protected Types inherited from [GiNaC::container\\_storage< C >](#)**

- typedef C< [ex](#) > [STLT](#)

**Static Protected Member Functions inherited from [GiNaC::container< C >](#)**

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for *lst*.*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for *lst*.*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for *lst*.*

**Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)**

- static void [reserve](#) ([STLT](#) &, size\_t)

**Protected Attributes inherited from [GiNaC::indexed](#)**

- [ex symtree](#)  
*Index symmetry (tree of symmetry objects)*

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

**Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)**

- [STLT seq](#)

**6.15.1 Detailed Description**

This class holds a generator `T_a` or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.

A representation label (an unsigned 8-bit integer) is used to distinguish elements from different Lie algebras (objects with different labels commute). These objects implement an abstract representation of the group, not a specific matrix representation. The indices used for color objects should not have a variance.

**6.15.2 Constructor & Destructor Documentation**

#### 6.15.2.1 `color()` [1/4]

```
GiNaC::color::color (
    const ex & b,
    unsigned char rl = 0 )
```

Construct object without any color index.

This constructor is for internal use only. Use the `color_ONE()` function instead.

See also

[color\\_ONE](#)

#### 6.15.2.2 `color()` [2/4]

```
GiNaC::color::color (
    const ex & b,
    const ex & il,
    unsigned char rl = 0 )
```

Construct object with one color index.

This constructor is for internal use only. Use the `color_T()` function instead.

See also

[color\\_T](#)

#### 6.15.2.3 `color()` [3/4]

```
GiNaC::color::color (
    unsigned char rl,
    const exvector & v )
```

#### 6.15.2.4 `color()` [4/4]

```
GiNaC::color::color (
    unsigned char rl,
    exvector && v )
```

### 6.15.3 Member Function Documentation

### 6.15.3.1 archive()

```
void GiNaC::color::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Loosely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [n](#), and [representation\\_label](#).

### 6.15.3.2 read\_archive()

```
void GiNaC::color::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [n](#), and [representation\\_label](#).

### 6.15.3.3 eval\_ncmul()

```
ex GiNaC::color::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Perform automatic simplification on noncommutative product of color objects.

This removes superfluous ONES.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::hold\\_ncmul\(\)](#), and [GiNaC::container\\_storage< C >::reserve\(\)](#).

#### 6.15.3.4 match\_same\_type()

```
bool GiNaC::color::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [representation\\_label](#).

#### 6.15.3.5 thiscontainer() [1/2]

```
ex GiNaC::color::thiscontainer (
    const exvector & v ) const [override], [protected]
```

References [representation\\_label](#).

#### 6.15.3.6 thiscontainer() [2/2]

```
ex GiNaC::color::thiscontainer (
    exvector && v ) const [override], [protected]
```

References [representation\\_label](#).

#### 6.15.3.7 return\_type()

```
unsigned GiNaC::color::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

#### 6.15.3.8 return\_type\_tinfo()

```
return_type_t GiNaC::color::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [representation\\_label](#).

#### 6.15.3.9 get\_representation\_label()

```
unsigned char GiNaC::color::get_representation_label ( ) const [inline]
```

References [representation\\_label](#).

### 6.15.4 Member Data Documentation

#### 6.15.4.1 representation\_label

```
unsigned char GiNaC::color::representation_label [private]
```

Representation label to distinguish independent color matrices coming from separated fermion lines.

Referenced by [archive\(\)](#), [get\\_representation\\_label\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [return\\_type\\_tinfo\(\)](#), and [thiscontainer\(\)](#).

The documentation for this class was generated from the following files:

- [color.h](#)
- [color.cpp](#)

## 6.16 GiNaC::compare\_all\_equal< T > Class Template Reference

Comparison policy: all structures of one type are equal.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_all\\_equal\(\)](#)

## Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 6.16.1 Detailed Description

```
template<class T>
class GiNaC::compare_all_equal< T >
```

Comparison policy: all structures of one type are equal.

### 6.16.2 Constructor & Destructor Documentation

#### 6.16.2.1 ~compare\_all\_equal()

```
template<class T >
GiNaC::compare_all_equal< T >::~~compare_all_equal ( ) [inline], [protected]
```

### 6.16.3 Member Function Documentation

#### 6.16.3.1 struct\_is\_equal()

```
template<class T >
static bool GiNaC::compare_all_equal< T >::struct_is_equal (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

#### 6.16.3.2 struct\_compare()

```
template<class T >
static int GiNaC::compare_all_equal< T >::struct_compare (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

- [structure.h](#)

## 6.17 GiNaC::compare\_bitwise< T > Class Template Reference

Comparison policy: use bit-wise comparison to compare structures.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_bitwise](#) ()

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

#### 6.17.1 Detailed Description

```
template<class T>  
class GiNaC::compare_bitwise< T >
```

Comparison policy: use bit-wise comparison to compare structures.

#### 6.17.2 Constructor & Destructor Documentation

##### 6.17.2.1 ~compare\_bitwise()

```
template<class T >  
GiNaC::compare\_bitwise< T >::~~compare\_bitwise ( ) [inline], [protected]
```

#### 6.17.3 Member Function Documentation

##### 6.17.3.1 struct\_is\_equal()

```
template<class T >  
static bool GiNaC::compare\_bitwise< T >::struct_is_equal (   
    const T * t1,  
    const T * t2 ) [inline], [static], [protected]
```

### 6.17.3.2 struct\_compare()

```
template<class T >
static int GiNaC::compare_bitwise< T >::struct_compare (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

- [structure.h](#)

## 6.18 GiNaC::compare\_std\_less< T > Class Template Reference

Comparison policy: use std::equal\_to/std::less (defaults to operators == and <) to compare structures.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_std\\_less\(\)](#)

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 6.18.1 Detailed Description

```
template<class T>
class GiNaC::compare_std_less< T >
```

Comparison policy: use std::equal\_to/std::less (defaults to operators == and <) to compare structures.

### 6.18.2 Constructor & Destructor Documentation

#### 6.18.2.1 ~compare\_std\_less()

```
template<class T >
GiNaC::compare_std_less< T >::~~compare_std_less ( ) [inline], [protected]
```

### 6.18.3 Member Function Documentation



## 6.18.3.1 struct\_is\_equal()

```
template<class T >
static bool GiNaC::compare_std_less< T >::struct_is_equal (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

## 6.18.3.2 struct\_compare()

```
template<class T >
static int GiNaC::compare_std_less< T >::struct_compare (
    const T * t1,
    const T * t2 ) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

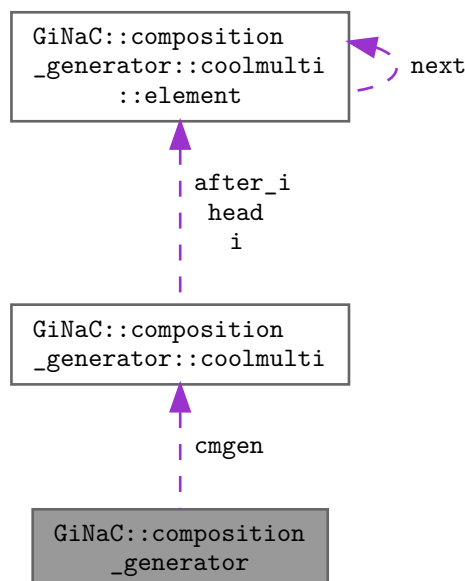
- [structure.h](#)

## 6.19 GiNaC::composition\_generator Class Reference

Generate all compositions of a partition of an integer n, starting with the compositions which has non-decreasing order.

```
#include <utils.h>
```

Collaboration diagram for GiNaC::composition\_generator:



## Classes

- struct [coolmulti](#)

## Public Member Functions

- [composition\\_generator](#) (const std::vector< unsigned > &partition)
- const std::vector< unsigned > & [get](#) () const
- bool [next](#) ()

## Private Attributes

- struct [GiNaC::composition\\_generator::coolmulti](#) cmgen
- bool [atend](#)
- bool [trivial](#)
- std::vector< unsigned > [composition](#)
- bool [current\\_updated](#)

### 6.19.1 Detailed Description

Generate all compositions of a partition of an integer n, starting with the compositions which has non-decreasing order.

### 6.19.2 Constructor & Destructor Documentation

#### 6.19.2.1 composition\_generator()

```
GiNaC::composition_generator::composition_generator (
    const std::vector< unsigned > & partition ) [inline], [explicit]
```

References [trivial](#).

### 6.19.3 Member Function Documentation

#### 6.19.3.1 get()

```
const std::vector< unsigned > & GiNaC::composition_generator::get ( ) const [inline]
```

References [cmgen](#), [composition](#), [current\\_updated](#), [GiNaC::composition\\_generator::coolmulti::head](#), [GiNaC::composition\\_generator::coolmulti::element::value](#), and [GiNaC::composition\\_generator::coolmulti::element::value](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

### 6.19.3.2 next()

```
bool GiNaC::composition_generator::next ( ) [inline]
```

References [attend](#), [cmgen](#), [current\\_updated](#), [GiNaC::composition\\_generator::coolmulti::finished\(\)](#), [GiNaC::composition\\_generator::coolmulti::next\(\)](#) and [trivial](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

## 6.19.4 Member Data Documentation

### 6.19.4.1 cmgen

```
struct GiNaC::composition_generator::coolmulti GiNaC::composition_generator::cmgen [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

### 6.19.4.2 attend

```
bool GiNaC::composition_generator::attend [private]
```

Referenced by [next\(\)](#).

### 6.19.4.3 trivial

```
bool GiNaC::composition_generator::trivial [private]
```

Referenced by [composition\\_generator\(\)](#), and [next\(\)](#).

### 6.19.4.4 composition

```
std::vector<unsigned> GiNaC::composition_generator::composition [mutable], [private]
```

Referenced by [get\(\)](#).

#### 6.19.4.5 current\_updated

```
bool GiNaC::composition_generator::current_updated [mutable], [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

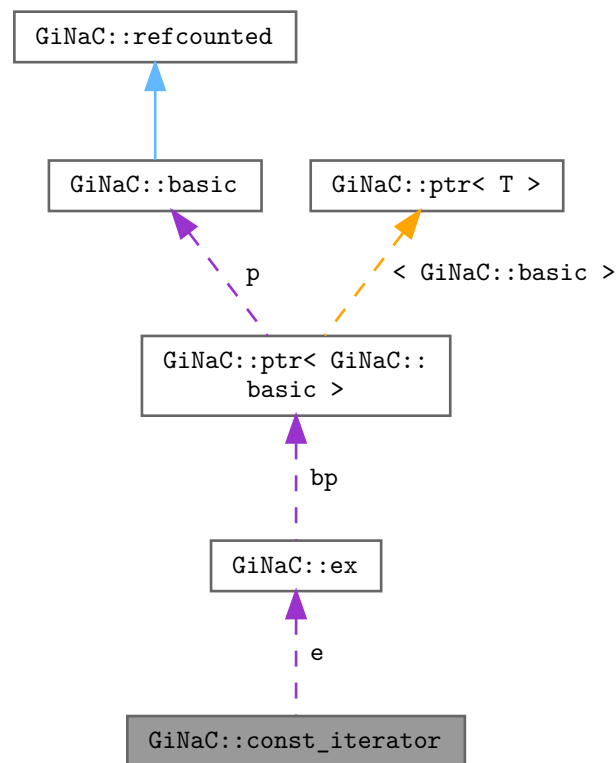
The documentation for this class was generated from the following file:

- [utils.h](#)

## 6.20 GiNaC::const\_iterator Class Reference

```
#include <ex.h>
```

Collaboration diagram for GiNaC::const\_iterator:



### Public Types

- using `iterator_category` = `std::random_access_iterator_tag`
- using `value_type` = `ex`
- using `difference_type` = `ptrdiff_t`
- using `pointer` = `const ex *`
- using `reference` = `const ex &`

## Public Member Functions

- `const_iterator` () noexcept
- `ex operator*` () const
- `std::unique_ptr< ex > operator->` () const
- `ex operator[]` (difference\_type n) const
- `const_iterator & operator++` () noexcept
- `const_iterator operator++` (int) noexcept
- `const_iterator & operator+=` (difference\_type n) noexcept
- `const_iterator operator+` (difference\_type n) const noexcept
- `const_iterator & operator--` () noexcept
- `const_iterator operator--` (int) noexcept
- `const_iterator & operator-=` (difference\_type n) noexcept
- `const_iterator operator-` (difference\_type n) const noexcept
- `bool operator==` (const `const_iterator` &other) const noexcept
- `bool operator!=` (const `const_iterator` &other) const noexcept
- `bool operator<` (const `const_iterator` &other) const noexcept
- `bool operator>` (const `const_iterator` &other) const noexcept
- `bool operator<=` (const `const_iterator` &other) const noexcept
- `bool operator>=` (const `const_iterator` &other) const noexcept

## Protected Attributes

- `ex e`
- `size_t i`

## Private Member Functions

- `const_iterator` (const `ex` &e\_, size\_t i\_) noexcept

## Friends

- class `ex`
- class `const_preorder_iterator`
- class `const_postorder_iterator`
- `const_iterator operator+` (difference\_type n, const `const_iterator` &it) noexcept
- `difference_type operator-` (const `const_iterator` &lhs, const `const_iterator` &rhs) noexcept

### 6.20.1 Member Typedef Documentation

#### 6.20.1.1 iterator\_category

```
using GiNaC::const_iterator::iterator_category = std::random_access_iterator_tag
```

### 6.20.1.2 value\_type

```
using GiNaC::const_iterator::value_type = ex
```

### 6.20.1.3 difference\_type

```
using GiNaC::const_iterator::difference_type = ptrdiff_t
```

### 6.20.1.4 pointer

```
using GiNaC::const_iterator::pointer = const ex *
```

### 6.20.1.5 reference

```
using GiNaC::const_iterator::reference = const ex &
```

## 6.20.2 Constructor & Destructor Documentation

### 6.20.2.1 const\_iterator() [1/2]

```
GiNaC::const_iterator::const_iterator ( ) [inline], [noexcept]
```

Referenced by [operator+\(\)](#), and [operator-\(\)](#).

### 6.20.2.2 const\_iterator() [2/2]

```
GiNaC::const_iterator::const_iterator (
    const ex & e_,
    size_t i_ ) [inline], [private], [noexcept]
```

## 6.20.3 Member Function Documentation

### 6.20.3.1 operator\*()

```
ex GiNaC::const_iterator::operator* ( ) const [inline]
```

References [e](#), [i](#), and [GiNaC::ex::op\(\)](#).

### 6.20.3.2 operator->()

```
std::unique_ptr< ex > GiNaC::const_iterator::operator-> ( ) const [inline]
```

References [ex](#).

### 6.20.3.3 operator[]()

```
ex GiNaC::const_iterator::operator[] (
    difference_type n ) const [inline]
```

References [e](#), [i](#), [n](#), and [GiNaC::ex::op\(\)](#).

### 6.20.3.4 operator++() [1/2]

```
const_iterator & GiNaC::const_iterator::operator++ ( ) [inline], [noexcept]
```

References [i](#).

### 6.20.3.5 operator++() [2/2]

```
const_iterator GiNaC::const_iterator::operator++ (
    int ) [inline], [noexcept]
```

References [i](#).

### 6.20.3.6 operator+=()

```
const_iterator & GiNaC::const_iterator::operator+= (
    difference_type n ) [inline], [noexcept]
```

References [i](#), and [n](#).

### 6.20.3.7 operator+()

```
const_iterator GiNaC::const_iterator::operator+ (
    difference_type n ) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#), [e](#), [i](#), and [n](#).

### 6.20.3.8 operator--() [1/2]

```
const_iterator & GiNaC::const_iterator::operator-- ( ) [inline], [noexcept]
```

References [i](#).

### 6.20.3.9 operator--() [2/2]

```
const_iterator GiNaC::const_iterator::operator-- (
    int ) [inline], [noexcept]
```

References [i](#).

### 6.20.3.10 operator-=()

```
const_iterator & GiNaC::const_iterator::operator-= (
    difference_type n ) [inline], [noexcept]
```

References [i](#), and [n](#).

### 6.20.3.11 operator-()

```
const_iterator GiNaC::const_iterator::operator- (
    difference_type n ) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#), [e](#), [i](#), and [n](#).

### 6.20.3.12 operator==()

```
bool GiNaC::const_iterator::operator== (
    const const_iterator & other ) const [inline], [noexcept]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [e](#), and [i](#).



### 6.20.3.13 operator"!=()

```
bool GiNaC::const_iterator::operator!= (
    const const\_iterator & other ) const    [inline], [noexcept]
```

### 6.20.3.14 operator<()

```
bool GiNaC::const_iterator::operator< (
    const const\_iterator & other ) const    [inline], [noexcept]
```

References [i](#).

### 6.20.3.15 operator>()

```
bool GiNaC::const_iterator::operator> (
    const const\_iterator & other ) const    [inline], [noexcept]
```

### 6.20.3.16 operator<=()

```
bool GiNaC::const_iterator::operator<= (
    const const\_iterator & other ) const    [inline], [noexcept]
```

### 6.20.3.17 operator>=()

```
bool GiNaC::const_iterator::operator>= (
    const const\_iterator & other ) const    [inline], [noexcept]
```

## 6.20.4 Friends And Related Function Documentation

### 6.20.4.1 ex

```
friend class ex    [friend]
```

Referenced by [operator->\(\)](#).

#### 6.20.4.2 `const_preorder_iterator`

```
friend class const_preorder_iterator [friend]
```

#### 6.20.4.3 `const_postorder_iterator`

```
friend class const_postorder_iterator [friend]
```

#### 6.20.4.4 `operator+`

```
const_iterator operator+ (  
    difference_type n,  
    const const_iterator & it ) [friend]
```

#### 6.20.4.5 `operator-`

```
difference_type operator- (  
    const const_iterator & lhs,  
    const const_iterator & rhs ) [friend]
```

### 6.20.5 Member Data Documentation

#### 6.20.5.1 `e`

```
ex GiNaC::const_iterator::e [protected]
```

Referenced by [operator\\*\(\)](#), [operator+\(\)](#), [operator-\(\)](#), [operator==\(\)](#), and [operator\[\]\(\)](#).

#### 6.20.5.2 `i`

```
size_t GiNaC::const_iterator::i [protected]
```

Referenced by [operator\\*\(\)](#), [operator+\(\)](#), [operator++\(\)](#), [operator+=\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [operator-=\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [operator\[\]\(\)](#).

The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.21 GiNaC::const\_postorder\_iterator Class Reference

```
#include <ex.h>
```

### Public Types

- using `iterator_category` = `std::forward_iterator_tag`
- using `value_type` = `ex`
- using `difference_type` = `ptrdiff_t`
- using `pointer` = `const ex *`
- using `reference` = `const ex &`

### Public Member Functions

- `const_postorder_iterator` () noexcept
- `const_postorder_iterator` (const `ex` &`e`, `size_t` `n`)
- `reference operator*` () const
- `pointer operator->` () const
- `const_postorder_iterator & operator++` ()
- `const_postorder_iterator operator++` (int)
- bool `operator==` (const `const_postorder_iterator` &`other`) const noexcept
- bool `operator!=` (const `const_postorder_iterator` &`other`) const noexcept

### Private Member Functions

- void `descend` ()
- void `increment` ()

### Private Attributes

- `std::stack< internal::_iter_rep, std::vector< internal::_iter_rep > >` `s`

### 6.21.1 Member Typedef Documentation

#### 6.21.1.1 iterator\_category

```
using GiNaC::const_postorder_iterator::iterator_category = std::forward_iterator_tag
```

#### 6.21.1.2 value\_type

```
using GiNaC::const_postorder_iterator::value_type = ex
```

### 6.21.1.3 difference\_type

```
using GiNaC::const_postorder_iterator::difference_type = ptrdiff_t
```

### 6.21.1.4 pointer

```
using GiNaC::const_postorder_iterator::pointer = const ex *
```

### 6.21.1.5 reference

```
using GiNaC::const_postorder_iterator::reference = const ex &
```

## 6.21.2 Constructor & Destructor Documentation

### 6.21.2.1 const\_postorder\_iterator() [1/2]

```
GiNaC::const_postorder_iterator::const_postorder_iterator ( ) [inline], [noexcept]
```

### 6.21.2.2 const\_postorder\_iterator() [2/2]

```
GiNaC::const_postorder_iterator::const_postorder_iterator (
    const ex & e,
    size_t n ) [inline]
```

References [descend\(\)](#), [n](#), and [s](#).

## 6.21.3 Member Function Documentation

### 6.21.3.1 operator\*()

```
reference GiNaC::const_postorder_iterator::operator* ( ) const [inline]
```

References [s](#).

### 6.21.3.2 operator->()

```
pointer GiNaC::const_postorder_iterator::operator-> ( ) const [inline]
```

References [s](#).

### 6.21.3.3 operator++() [1/2]

```
const_postorder_iterator & GiNaC::const_postorder_iterator::operator++ ( ) [inline]
```

References [increment\(\)](#).

### 6.21.3.4 operator++() [2/2]

```
const_postorder_iterator GiNaC::const_postorder_iterator::operator++ (
    int ) [inline]
```

References [increment\(\)](#).

### 6.21.3.5 operator==()

```
bool GiNaC::const_postorder_iterator::operator== (
    const const_postorder_iterator & other ) const [inline], [noexcept]
```

References [s](#).

### 6.21.3.6 operator!=(=)

```
bool GiNaC::const_postorder_iterator::operator!= (
    const const_postorder_iterator & other ) const [inline], [noexcept]
```

### 6.21.3.7 descend()

```
void GiNaC::const_postorder_iterator::descend ( ) [inline], [private]
```

References [GiNaC::internal::\\_iter\\_rep::e](#), [GiNaC::internal::\\_iter\\_rep::i](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [s](#).

Referenced by [const\\_postorder\\_iterator\(\)](#), and [increment\(\)](#).

### 6.21.3.8 increment()

```
void GiNaC::const_postorder_iterator::increment ( ) [inline], [private]
```

References [descend\(\)](#), and [s](#).

Referenced by [operator++\(\)](#).

## 6.21.4 Member Data Documentation

### 6.21.4.1 s

```
std::stack<internal::_iter_rep, std::vector<internal::_iter_rep> > GiNaC::const_postorder_iterator::s [private]
```

Referenced by [const\\_postorder\\_iterator\(\)](#), [descend\(\)](#), [increment\(\)](#), [operator\\*\(\)](#), [operator->\(\)](#), and [operator==\(\)](#).

The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.22 GiNaC::const\_preorder\_iterator Class Reference

```
#include <ex.h>
```

### Public Types

- using [iterator\\_category](#) = std::forward\_iterator\_tag
- using [value\\_type](#) = ex
- using [difference\\_type](#) = ptrdiff\_t
- using [pointer](#) = const ex \*
- using [reference](#) = const ex &

### Public Member Functions

- [const\\_preorder\\_iterator](#) () noexcept
- [const\\_preorder\\_iterator](#) (const ex &e, size\_t n)
- [reference operator\\*](#) () const
- [pointer operator->](#) () const
- [const\\_preorder\\_iterator](#) & [operator++](#) ()
- [const\\_preorder\\_iterator](#) [operator++](#) (int)
- bool [operator==](#) (const [const\\_preorder\\_iterator](#) &other) const noexcept
- bool [operator!=](#) (const [const\\_preorder\\_iterator](#) &other) const noexcept

## Private Member Functions

- void [increment](#) ()

## Private Attributes

- std::stack< [internal::\\_iter\\_rep](#), std::vector< [internal::\\_iter\\_rep](#) > > s

## 6.22.1 Member Typedef Documentation

### 6.22.1.1 iterator\_category

```
using GiNaC::const_preorder_iterator::iterator_category = std::forward_iterator_tag
```

### 6.22.1.2 value\_type

```
using GiNaC::const_preorder_iterator::value_type = ex
```

### 6.22.1.3 difference\_type

```
using GiNaC::const_preorder_iterator::difference_type = ptrdiff_t
```

### 6.22.1.4 pointer

```
using GiNaC::const_preorder_iterator::pointer = const ex *
```

### 6.22.1.5 reference

```
using GiNaC::const_preorder_iterator::reference = const ex &
```

## 6.22.2 Constructor & Destructor Documentation

#### 6.22.2.1 `const_preorder_iterator()` [1/2]

```
GiNaC::const_preorder_iterator::const_preorder_iterator ( ) [inline], [noexcept]
```

#### 6.22.2.2 `const_preorder_iterator()` [2/2]

```
GiNaC::const_preorder_iterator::const_preorder_iterator (
    const ex & e,
    size_t n ) [inline]
```

References [n](#), and [s](#).

### 6.22.3 Member Function Documentation

#### 6.22.3.1 `operator*()`

```
reference GiNaC::const_preorder_iterator::operator* ( ) const [inline]
```

References [s](#).

#### 6.22.3.2 `operator->()`

```
pointer GiNaC::const_preorder_iterator::operator-> ( ) const [inline]
```

References [s](#).

#### 6.22.3.3 `operator++()` [1/2]

```
const\_preorder\_iterator & GiNaC::const_preorder_iterator::operator++ ( ) [inline]
```

References [increment\(\)](#).

#### 6.22.3.4 `operator++()` [2/2]

```
const\_preorder\_iterator GiNaC::const_preorder_iterator::operator++ (
    int ) [inline]
```

References [increment\(\)](#).



### 6.22.3.5 operator==()

```
bool GiNaC::const_preorder_iterator::operator== (
    const const\_preorder\_iterator & other ) const [inline], [noexcept]
```

References [s](#).

### 6.22.3.6 operator!=(())

```
bool GiNaC::const_preorder_iterator::operator!= (
    const const\_preorder\_iterator & other ) const [inline], [noexcept]
```

### 6.22.3.7 increment()

```
void GiNaC::const_preorder_iterator::increment ( ) [inline], [private]
```

References [GiNaC::internal::\\_iter\\_rep::e](#), [GiNaC::internal::\\_iter\\_rep::i](#), [GiNaC::internal::\\_iter\\_rep::i\\_end](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [s](#).

Referenced by [operator++\(\)](#).

## 6.22.4 Member Data Documentation

### 6.22.4.1 s

```
std::stack<internal::\_iter\_rep, std::vector<internal::\_iter\_rep> > GiNaC::const_preorder_↵
iterator::s [private]
```

Referenced by [const\\_preorder\\_iterator\(\)](#), [increment\(\)](#), [operator\\*\(\)](#), [operator->\(\)](#), and [operator==\(\)](#).

The documentation for this class was generated from the following file:

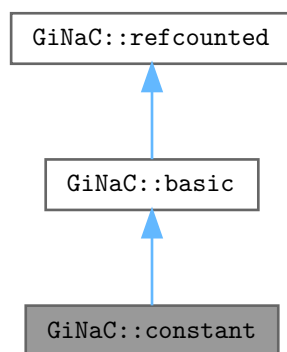
- [ex.h](#)

## 6.23 GiNaC::constant Class Reference

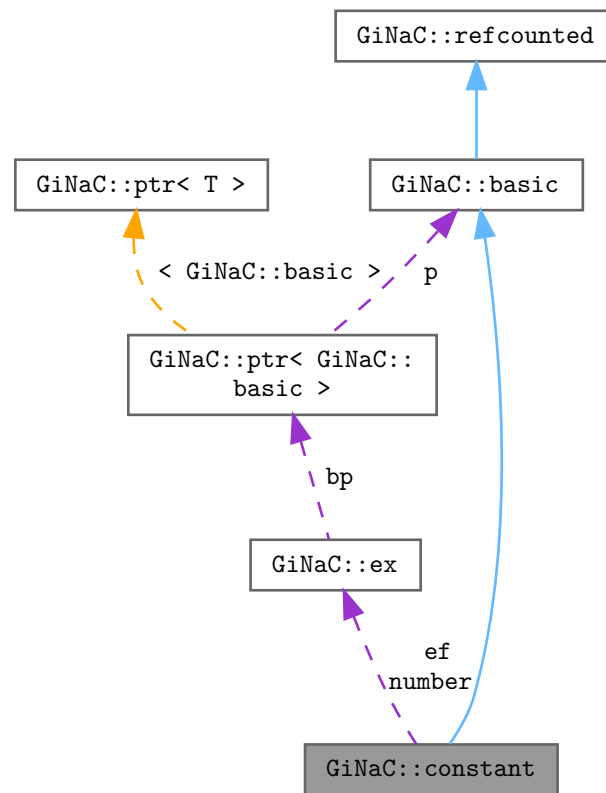
This class holds constants, symbols with specific numerical value.

```
#include <constant.h>
```

Inheritance diagram for GiNaC::constant:



Collaboration diagram for GiNaC::constant:



## Public Member Functions

- `constant` (const std::string &initname, evalffunctype efun=nullptr, const std::string &texname=std::string(), unsigned domain=domain::complex)
- `constant` (const std::string &initname, const numeric &initnumber, const std::string &texname=std::string(), unsigned domain=domain::complex)
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- void `archive` (archive\_node &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const archive\_node &n, lst &syms) override  
*Load (deserialize) the object from an archive node.*

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const

*Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

*Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const

*Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const

*Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const

*Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for a constant always returns 0.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex\\_derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- `std::string name`  
*printrname of this constant*
- `std::string TeX_name`  
*LaTeX name.*
- `evalfunctype ef`
- `ex number`  
*numerical value this constant `evalf()`s to*
- `unsigned serial`  
*unique serial number for comparison*
- `unsigned domain`  
*numerical value this constant `evalf()`s to*

## Static Private Attributes

- static `unsigned next_serial = 0`

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- `unsigned flags`  
*of type `status_flags`*
- `unsigned hashvalue`  
*hash value*

## 6.23.1 Detailed Description

This class holds constants, symbols with specific numerical value.

Each object of this class must either provide their own function to evaluate it to class numeric or provide the constant as a numeric (if it's an exact number).

## 6.23.2 Constructor & Destructor Documentation

### 6.23.2.1 `constant()` [1/2]

```
GiNaC::constant::constant (
    const std::string & initname,
    evalfunctype efun = nullptr,
    const std::string & texname = std::string(),
    unsigned domain = domain::complex )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [name](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

### 6.23.2.2 `constant()` [2/2]

```
GiNaC::constant::constant (
    const std::string & initname,
    const numeric & initnumber,
    const std::string & texname = std::string(),
    unsigned domain = domain::complex )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [name](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

## 6.23.3 Member Function Documentation

### 6.23.3.1 `info()`

```
bool GiNaC::constant::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::domain::positive](#), [GiNaC::info\\_flags::positive](#), [GiNaC::domain::real](#), and [GiNaC::info\\_flags::real](#).

### 6.23.3.2 `evalf()`

```
ex GiNaC::constant::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [ef](#), [GiNaC::ex::evalf\(\)](#), and [number](#).

Referenced by [GiNaC::EllipticE\\_eval\(\)](#), and [GiNaC::EllipticK\\_eval\(\)](#).



### 6.23.3.3 is\_polynomial()

```
bool GiNaC::constant::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

### 6.23.3.4 conjugate()

```
ex GiNaC::constant::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

### 6.23.3.5 real\_part()

```
ex GiNaC::constant::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

### 6.23.3.6 imag\_part()

```
ex GiNaC::constant::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

### 6.23.3.7 archive()

```
void GiNaC::constant::archive (
    archive\_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [n](#), and [name](#).

### 6.23.3.8 read\_archive()

```
void GiNaC::constant::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::Catalan](#), [GiNaC::Euler](#), [n](#), [name](#), and [GiNaC::Pi](#).

### 6.23.3.9 derivative()

```
ex GiNaC::constant::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a constant always returns 0.

#### See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#).

### 6.23.3.10 is\_equal\_same\_type()

```
bool GiNaC::constant::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [serial](#).

### 6.23.3.11 calchash()

```
unsigned GiNaC::constant::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

### 6.23.3.12 do\_print()

```
void GiNaC::constant::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [name](#).

### 6.23.3.13 do\_print\_tree()

```
void GiNaC::constant::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), and [name](#).

### 6.23.3.14 do\_print\_latex()

```
void GiNaC::constant::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [TeX\\_name](#).

### 6.23.3.15 do\_print\_python\_repr()

```
void GiNaC::constant::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), [name](#), and [TeX\\_name](#).

## 6.23.4 Member Data Documentation

### 6.23.4.1 name

`std::string GiNaC::constant::name [private]`

printname of this constant

Referenced by [archive\(\)](#), [constant\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), and [read\\_archive\(\)](#).

### 6.23.4.2 TeX\_name

`std::string GiNaC::constant::TeX_name [private]`

LaTeX name.

Referenced by [constant\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

### 6.23.4.3 ef

`evalffunctype GiNaC::constant::ef [private]`

Referenced by [evalf\(\)](#).

### 6.23.4.4 number

`ex GiNaC::constant::number [private]`

numerical value this constant [evalf\(\)](#)s to

Referenced by [evalf\(\)](#).

### 6.23.4.5 serial

`unsigned GiNaC::constant::serial [private]`

unique serial number for comparison

Referenced by [calchash\(\)](#), and [is\\_equal\\_same\\_type\(\)](#).

## 6.23.4.6 next\_serial

```
unsigned GiNaC::constant::next_serial = 0 [static], [private]
```

## 6.23.4.7 domain

```
unsigned GiNaC::constant::domain [private]
```

numerical value this constant [evalf\(\)](#)s to

The documentation for this class was generated from the following files:

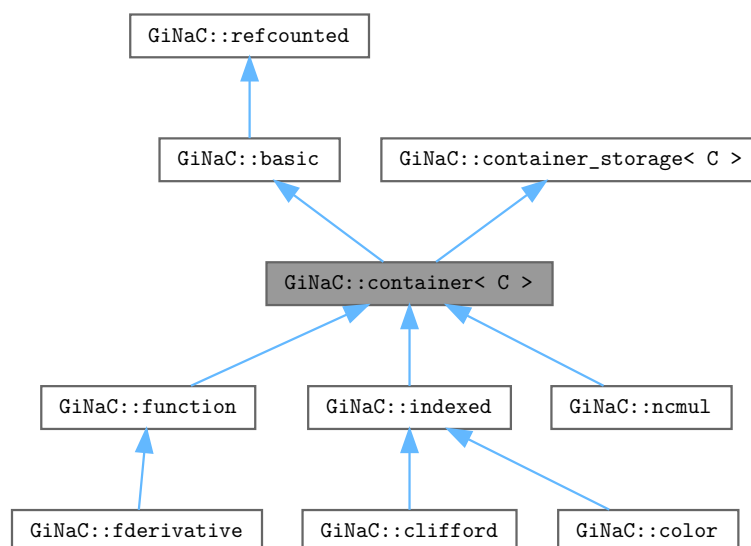
- [constant.h](#)
- [constant.cpp](#)

## 6.24 GiNaC::container&lt; C &gt; Class Template Reference

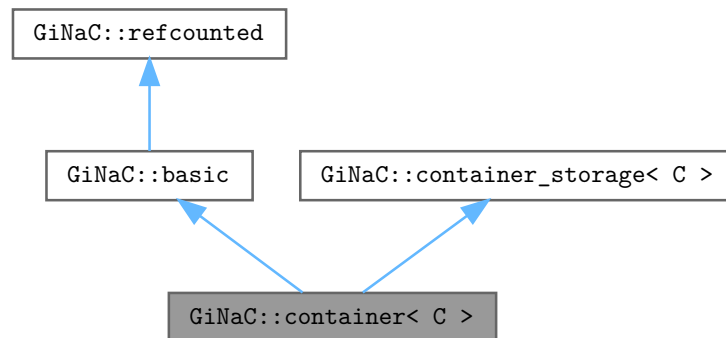
Wrapper template for making [GiNaC](#) classes out of STL containers.

```
#include <container.h>
```

Inheritance diagram for GiNaC::container< C >:



Collaboration diagram for `GiNaC::container< C >`:



## Public Types

- typedef `STLT::const_iterator` [const\\_iterator](#)
- typedef `STLT::const_reverse_iterator` [const\\_reverse\\_iterator](#)

## Public Member Functions

- [container](#) (`STLT const &s`)
- [container](#) (`STLT &&v`)
- [container](#) (`exvector::const_iterator b, exvector::const_iterator e`)
- [container](#) (`std::initializer_list< ex > il`)
- `bool` [info](#) (`unsigned inf`) const override  
*Information about the object.*
- `unsigned` [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- `size_t` [nops](#) () const override  
*Number of operands/members.*
- `ex op` (`size_t i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (`size_t i`) override  
*Return modifiable operand/member at position i.*
- `ex subs` (`const exmap &m, unsigned options=0`) const override  
*Substitute a set of objects by arbitrary expressions.*
- `void` [read\\_archive](#) (`const archive_node &n, lst &sym_lst`) override  
*Load (deserialize) the object from an archive node.*
- `void` [archive](#) (`archive_node &n`) const override  
*Archive the object.*
- `container &` [prepend](#) (`const ex &b`)  
*Add element at front.*
- `container &` [append](#) (`const ex &b`)  
*Add element at back.*

- `container & remove_first ()`  
*Remove first element.*
- `container & remove_last ()`  
*Remove last element.*
- `container & remove_all ()`  
*Remove all elements.*
- `container & sort ()`  
*Sort elements.*
- `container & unique ()`  
*Remove adjacent duplicate elements.*
- `const_iterator begin () const`
- `const_iterator end () const`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rend () const`

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual void `dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf) const`  
*Information about the object.*
- virtual size\_t `nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`

- Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const
- Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
- Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
- void `print_dispatch` (const `print_context` &c, unsigned level) const



- Like `print()`, but dispatch to the specified class.
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.
- virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.
- int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.
- bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.
- const `basic` & `hold` () const
- Stop further evaluation.
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- Set some `status_flags`.
- const `basic` & `clearflag` (unsigned f) const
- Clear some `status_flags`.

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Types

- typedef `container_storage`< C >::STLT STLT

### Protected Types inherited from `GiNaC::container_storage< C >`

- typedef C< `ex` > STLT

## Protected Member Functions

- `ex conjugate ()` const override
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `bool is_equal_same_type (const basic &other)` const override  
*Returns true if two objects of same type are equal.*
- virtual `ex thiscontainer (const STLT &v)` const  
*Similar to `duplicate()`, but with a preset sequence.*
- virtual `ex thiscontainer (STLT &&v)` const  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- virtual void `printseq (const print_context &c, char openbracket, char delim, char closebracket, unsigned this←_precedence, unsigned upper_precedence=0)` const  
*Print sequence of contained elements.*
- void `do_print (const print_context &c, unsigned level)` const
- void `do_print_tree (const print_tree &c, unsigned level)` const
- void `do_print_python (const print_python &c, unsigned level)` const
- void `do_print_python_repr (const print_python_repr &c, unsigned level)` const
- `STLT subchildren (const exmap &m, unsigned options=0)` const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic ()`
- virtual `ex eval_ncmul (const exvector &v)` const
- virtual `bool match_same_type (const basic &other)` const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative (const symbol &s)` const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type (const basic &other)` const  
*Returns order relation between two objects of same type.*
- virtual `bool is_equal_same_type (const basic &other)` const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash ()` const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable ()` const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print (const print_context &c, unsigned level)` const  
*Default output to stream.*
- void `do_print_tree (const print_tree &c, unsigned level)` const  
*Tree output to stream.*
- void `do_print_python_repr (const print_python_repr &c, unsigned level)` const  
*Python parsable output to stream.*

## Protected Member Functions inherited from `GiNaC::container_storage< C >`

- `container_storage ()`
- `container_storage (size_t n, const ex &e)`
- `container_storage (std::initializer_list< ex > il)`
- template<class In >  
  `container_storage (In b, In e)`
- void `reserve (size_t)`
- `~container_storage ()`
- void `reserve (size_t n)`
- void `reserve (std::vector< ex > &v, size_t n)`

## Static Protected Member Functions

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of `container::get_default_flags()` for `lst`.*
- static char [get\\_open\\_delim](#) ()  
*Specialization of `container::get_open_delim()` for `lst`.*
- static char [get\\_close\\_delim](#) ()  
*Specialization of `container::get_close_delim()` for `lst`.*

## Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) (STLT &, size\_t)

## Private Member Functions

- void [sort\\_](#) (std::random\_access\_iterator\_tag)
- void [sort\\_](#) (std::input\_iterator\_tag)
- void [unique\\_](#) ()
- void [unique\\_](#) ()  
*Specialization of `container::unique_()` for `std::list`.*

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type `status_flags`*
- unsigned [hashvalue](#)  
*hash value*

### Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)

### 6.24.1 Detailed Description

```
template<template< class T, class=std::allocator< T > > class C>
class GiNaC::container< C >
```

Wrapper template for making [GiNaC](#) classes out of STL containers.

### 6.24.2 Member Typedef Documentation

### 6.24.2.1 STL

```
template<template< class T, class=std::allocator< T > > class C>
typedef container\_storage<C>::STLT GiNaC::container< C >::STLT [protected]
```

### 6.24.2.2 const\_iterator

```
template<template< class T, class=std::allocator< T > > class C>
typedef STLT::const_iterator GiNaC::container< C >::const_iterator
```

### 6.24.2.3 const\_reverse\_iterator

```
template<template< class T, class=std::allocator< T > > class C>
typedef STLT::const_reverse_iterator GiNaC::container< C >::const_reverse_iterator
```

## 6.24.3 Constructor & Destructor Documentation

### 6.24.3.1 container() [1/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
    STLT const & s ) [inline]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

### 6.24.3.2 container() [2/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
    STLT && v ) [inline], [explicit]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

**6.24.3.3 container() [3/4]**

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
    exvector::const_iterator b,
    exvector::const_iterator e ) [inline]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**6.24.3.4 container() [4/4]**

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
    std::initializer_list< ex > il ) [inline]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**6.24.4 Member Function Documentation****6.24.4.1 get\_default\_flags()**

```
unsigned GiNaC::lst::get_default_flags [inline], [static], [protected]
```

Specialization of [container::get\\_default\\_flags\(\)](#) for `lst`.

Referenced by [GiNaC::container< C >::container\(\)](#), and [GiNaC::container< C >::read\\_archive\(\)](#).

**6.24.4.2 get\_open\_delim()**

```
char GiNaC::lst::get_open_delim [inline], [static], [protected]
```

Specialization of [container::get\\_open\\_delim\(\)](#) for `lst`.

**6.24.4.3 get\_close\_delim()**

```
char GiNaC::lst::get_close_delim [inline], [static], [protected]
```

Specialization of [container::get\\_close\\_delim\(\)](#) for `lst`.

#### 6.24.4.4 info()

```
template bool GiNaC::container< C >::info (
    unsigned inf ) const [inline], [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

#### 6.24.4.5 precedence()

```
template<template< class T, class=std::allocator< T > > class C>
unsigned GiNaC::container< C >::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

Referenced by [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), and [GiNaC::function::print\(\)](#).

#### 6.24.4.6 nops()

```
template<template< class T, class=std::allocator< T > > class C>
size_t GiNaC::container< C >::nops ( ) const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::function::calchash\(\)](#), [GiNaC::ncmul::conjugate\(\)](#), [GiNaC::ex::denom\(\)](#), [GiNaC::diag\\_matrix\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::ncmul::get\\_free\\_indices\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::lst\\_to\\_matrix\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::ex::numer\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::container< C >::real\\_part\(\)](#), [GiNaC::sqrfree\(\)](#), and [GiNaC::ex::subs\(\)](#).

## 6.24.4.7 op()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::nops\(\)](#).

Referenced by [GiNaC::function::calchash\(\)](#), [GiNaC::ex::denom\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::ncmul::get\\_free\\_indices\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::indexed::imag\\_part\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::ex::numer\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::indexed::real\\_part\(\)](#), [GiNaC::rename\\_dummy\\_index\(\)](#), [GiNaC::indexed::return\\_type\(\)](#), [GiNaC::indexed::return\\_type\\_tinfo\(\)](#), [GiNaC::sqrfree\(\)](#), [GiNaC::symm\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

## 6.24.4.8 let\_op()

```
template<template< class T, class=std::allocator< T > > class C>
ex & GiNaC::container< C >::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::nops\(\)](#).

## 6.24.4.9 subs()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::is\\_a\(\)](#), [m](#), and [options](#).

#### 6.24.4.10 read\_archive()

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::read_archive (
    const archive_node & n,
    lst & syms ) [inline], [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

##### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::fderivative](#), [GiNaC::function](#), and [GiNaC::indexed](#).

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), [n](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

#### 6.24.4.11 archive()

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::archive (
    archive_node & n ) const [inline], [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::fderivative](#), [GiNaC::function](#), and [GiNaC::indexed](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [n](#), and [GiNaC::container\\_storage< C >::seq](#).

#### 6.24.4.12 conjugate()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::conjugate ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), and [GiNaC::ncmul](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [GiNaC::container< C >::thiscontainer\(\)](#), and [x](#).

Referenced by [GiNaC::ncmul::conjugate\(\)](#).



#### 6.24.4.13 real\_part()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::real_part ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

References [GiNaC::container< C >::begin\(\)](#), [cont](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), and [GiNaC::container< C >::thiscontainer\(\)](#).

#### 6.24.4.14 imag\_part()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::imag_part ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

References [GiNaC::container< C >::begin\(\)](#), [cont](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), and [GiNaC::container< C >::thiscontainer\(\)](#).

#### 6.24.4.15 is\_equal\_same\_type()

```
template<template< class T, class=std::allocator< T > > class C>
bool GiNaC::container< C >::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::fderivative](#), and [GiNaC::function](#).

References [GINAC\\_ASSERT](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::function::is\\_equal\\_same\\_type\(\)](#).

**6.24.4.16 thiscontainer() [1/2]**

```
template<template< class T, class=std::allocator< T > > class C>
virtual ex GiNaC::container< C >::thiscontainer (
    const STLT & v ) const [inline], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

Referenced by [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), and [GiNaC::container< C >::real\\_part\(\)](#).

**6.24.4.17 thiscontainer() [2/2]**

```
template<template< class T, class=std::allocator< T > > class C>
virtual ex GiNaC::container< C >::thiscontainer (
    STLT && v ) const [inline], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

**6.24.4.18 printseq()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::printseq (
    const print\_context & c,
    char openbracket,
    char delim,
    char closebracket,
    unsigned this_precedence,
    unsigned upper_precedence = 0 ) const [protected], [virtual]
```

Print sequence of contained elements.

References [c](#).

Referenced by [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::ncmul::do\\_print\(\)](#), [GiNaC::ncmul::do\\_print\\_src\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), and [GiNaC::function::print\(\)](#).

**6.24.4.19 sort\_() [1/2]**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::sort_ (
    std::random_access_iterator_tag ) [inline], [private]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.20 sort\_()** [2/2]

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::sort_ (
    std::input_iterator_tag ) [inline], [private]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.21 unique\_()** [1/2]

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::unique_ ( ) [inline], [private]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.22 prepend()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::prepend (
    const ex & b )
```

Add element at front.

**6.24.4.23 append()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::append (
    const ex & b )
```

Add element at back.

Referenced by [GiNaC::ifactor\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::symbol::read\\_archive\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::sqrfree\(\)](#), [GiNaC::symm\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

**6.24.4.24 remove\_first()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::remove_first
```

Remove first element.

Referenced by [GiNaC::sqrfree\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

**6.24.4.25 remove\_last()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::remove_last
```

Remove last element.

**6.24.4.26 remove\_all()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::remove_all
```

Remove all elements.

**6.24.4.27 sort()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::sort
```

Sort elements.

**6.24.4.28 unique()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::unique
```

Remove adjacent duplicate elements.

**6.24.4.29 begin()**

```
template<template< class T, class=std::allocator< T > > class C>
const_iterator GiNaC::container< C >::begin ( ) const [inline]
```

References [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::ex::antisymmetrize\(\)](#), [GiNaC::ncmul::conjugate\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::container< C >::real\\_part\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::symmetrize\(\)](#), [GiNaC::ex::symmetrize\\_cyclic\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), [GiNaC::zeta2\\_evalf\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

**6.24.4.30 end()**

```
template<template< class T, class=std::allocator< T > > class C>
const_iterator GiNaC::container< C >::end ( ) const [inline]
```

References [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::ex::antisymmetrize\(\)](#), [GiNaC::fderivative::archive\(\)](#), [GiNaC::ncmul::conjugate\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::container< C >::real\\_part\(\)](#), [GiNaC::ncmul::return\\_type\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::symmetrize\(\)](#), and [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

**6.24.4.31 rbegin()**

```
template<template< class T, class=std::allocator< T > > class C>
const_reverse_iterator GiNaC::container< C >::rbegin ( ) const [inline]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.32 rend()**

```
template<template< class T, class=std::allocator< T > > class C>
const_reverse_iterator GiNaC::container< C >::rend ( ) const [inline]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.33 do\_print()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#).

**6.24.4.34 do\_print\_tree()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print_tree (
    const print_tree & c,
    unsigned level ) const [protected]
```

References [c](#), and [GiNaC::nops\(\)](#).

**6.24.4.35 do\_print\_python()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print_python (
    const print_python & c,
    unsigned level ) const [protected]
```

References [c](#).

**6.24.4.36 do\_print\_python\_repr()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print_python_repr (
    const print_python_repr & c,
    unsigned level ) const [protected]
```

References [c](#).

**6.24.4.37 subschildren()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C >::STLT GiNaC::container< C >::subschildren (
    const exmap & m,
    unsigned options = 0 ) const [protected]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [m](#), [options](#), and [GiNaC::ex::subs\(\)](#).

**6.24.4.38 unique\_() [2/2]**

```
void GiNaC::container< std::list >::unique_ ( ) [inline], [private]
```

Specialization of [container::unique\\_\(\)](#) for `std::list`.

The documentation for this class was generated from the following files:

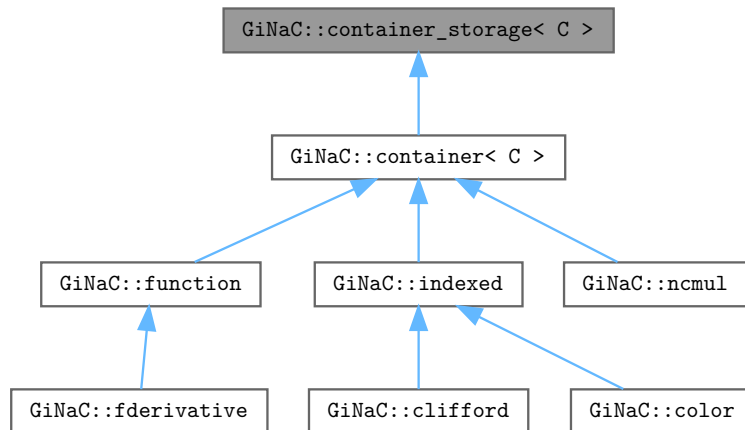
- [container.h](#)
- [exprseq.h](#)
- [lst.h](#)

## 6.25 GiNaC::container\_storage< C > Class Template Reference

Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.

```
#include <container.h>
```

Inheritance diagram for GiNaC::container\_storage< C >:



### Protected Types

- typedef C< [ex](#) > [STLT](#)

### Protected Member Functions

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

### Static Protected Member Functions

- static void [reserve](#) ([STLT](#) &, size\_t)

### Protected Attributes

- [STLT seq](#)

### 6.25.1 Detailed Description

```
template<template< class T, class=std::allocator< T > > class C>
class GiNaC::container_storage< C >
```

Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.

### 6.25.2 Member Typedef Documentation

#### 6.25.2.1 STLT

```
template<template< class T, class=std::allocator< T > > class C>
typedef C<ex> GiNaC::container_storage< C >::STLT [protected]
```

### 6.25.3 Constructor & Destructor Documentation

#### 6.25.3.1 container\_storage() [1/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::container_storage ( ) [inline], [protected]
```

#### 6.25.3.2 container\_storage() [2/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::container_storage (
    size_t n,
    const ex & e ) [inline], [protected]
```

#### 6.25.3.3 container\_storage() [3/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::container_storage (
    std::initializer_list< ex > il ) [inline], [protected]
```



**6.25.3.4 container\_storage() [4/4]**

```
template<template< class T, class=std::allocator< T > > class C>
template<class In >
GiNaC::container_storage< C >::container_storage (
    In b,
    In e ) [inline], [protected]
```

**6.25.3.5 ~container\_storage()**

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::~~container_storage ( ) [inline], [protected]
```

**6.25.4 Member Function Documentation****6.25.4.1 reserve() [1/4]**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container_storage< C >::reserve (
    size_t ) [inline], [protected]
```

Referenced by [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::color::eval\\_ncmul\(\)](#), [GiNaC::container< C >::imag](#), [GiNaC::container< C >::read\\_archive\(\)](#), [GiNaC::container< C >::real\\_part\(\)](#), and [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#).

**6.25.4.2 reserve() [2/4]**

```
template<template< class T, class=std::allocator< T > > class C>
static void GiNaC::container_storage< C >::reserve (
    STLT & ,
    size_t ) [inline], [static], [protected]
```

**6.25.4.3 reserve() [3/4]**

```
void GiNaC::container_storage< std::vector >::reserve (
    size_t n ) [inline], [protected]
```

References [n](#).

#### 6.25.4.4 `reserve()` [4/4]

```
void GiNaC::container_storage< std::vector >::reserve (
    std::vector< ex > & v,
    size_t n ) [inline], [protected]
```

References [n](#).

### 6.25.5 Member Data Documentation

#### 6.25.5.1 `seq`

```
template<template< class T, class=std::allocator< T > > class C>
STLT GiNaC::container_storage< C >::seq [protected]
```

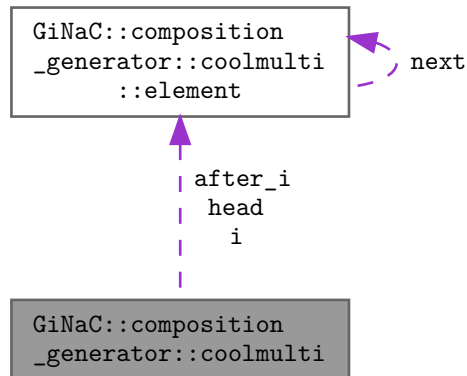
Referenced by [GiNaC::indexed::all\\_index\\_values\\_are\(\)](#), [GiNaC::container< C >::archive\(\)](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::function::conjugate\(\)](#), [GiNaC::container< C >::container\(\)](#), [GiNaC::ncmul::degree\(\)](#), [GiNaC::fderivative::derivative\(\)](#), [GiNaC::function::derivative\(\)](#), [GiNaC::ncmul::derivative\(\)](#), [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::clifford::do\\_print\\_latex\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::fderivative::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::function::eval\\_ncmul\(\)](#), [GiNaC::function::evalf\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [GiNaC::function::expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::indexed::get\\_dummy\\_indices\(\)](#), [GiNaC::ncmul::get\\_factors\(\)](#), [GiNaC::indexed::get\\_free\\_indices\(\)](#), [GiNaC::indexed::get\\_indices\(\)](#), [GiNaC::indexed::has\\_dummy\\_index\\_for\(\)](#), [GiNaC::function::imag\\_part\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::function::info\(\)](#), [GiNaC::indexed::info\(\)](#), [GiNaC::remember\\_table\\_entry::is\\_equal\(\)](#), [GiNaC::container< C >::is\\_equal\\_same\\_type\(\)](#), [GiNaC::ncmul::ldegree\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::indexed::print\\_indexed\(\)](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::container< C >::rbegin\(\)](#), [GiNaC::container< C >::read\\_archive\(\)](#), [GiNaC::function::read\\_archive\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [GiNaC::function::real\\_part\(\)](#), [GiNaC::container< C >::rend\(\)](#), [GiNaC::function::return\\_type\(\)](#), [GiNaC::ncmul::return\\_type\(\)](#), [GiNaC::function::return\\_type\\_tinfo\(\)](#), [GiNaC::ncmul::return\\_type\\_tinfo\(\)](#), [GiNaC::function::series\(\)](#), [GiNaC::container< C >::sort\\_\(\)](#), [GiNaC::container< C >::unique\\_\(\)](#), and [GiNaC::indexed::validate\(\)](#).

The documentation for this class was generated from the following file:

- [container.h](#)

## 6.26 GiNaC::composition\_generator::coolmulti Struct Reference

Collaboration diagram for GiNaC::composition\_generator::coolmulti:



### Classes

- struct [element](#)

### Public Member Functions

- [coolmulti](#) (const std::vector< unsigned > &partition)
- [~coolmulti](#) ()
- void [next\\_permutation](#) ()
- bool [finished](#) () const

### Public Attributes

- [element](#) \* [head](#)
- [element](#) \* [i](#)
- [element](#) \* [after\\_i](#)

### 6.26.1 Constructor & Destructor Documentation

#### 6.26.1.1 coolmulti()

```
GiNaC::composition_generator::coolmulti::coolmulti (
    const std::vector< unsigned > & partition ) [inline], [explicit]
```

References [after\\_i](#), [head](#), [i](#), [n](#), and [GiNaC::composition\\_generator::coolmulti::element::next](#).

### 6.26.1.2 ~coolmulti()

`GiNaC::composition_generator::coolmulti::~~coolmulti ( ) [inline]`

References [head](#).

## 6.26.2 Member Function Documentation

### 6.26.2.1 next\_permutation()

`void GiNaC::composition_generator::coolmulti::next_permutation ( ) [inline]`

References [after\\_i](#), [head](#), [i](#), [k](#), [GiNaC::composition\\_generator::coolmulti::element::next](#), and [GiNaC::composition\\_generator::coolmulti::next\\_permutation](#).

Referenced by [GiNaC::composition\\_generator::next\(\)](#).

### 6.26.2.2 finished()

`bool GiNaC::composition_generator::coolmulti::finished ( ) const [inline]`

References [after\\_i](#), [head](#), [GiNaC::composition\\_generator::coolmulti::element::next](#), and [GiNaC::composition\\_generator::coolmulti::element::finished](#).

Referenced by [GiNaC::composition\\_generator::next\(\)](#).

## 6.26.3 Member Data Documentation

### 6.26.3.1 head

`element* GiNaC::composition_generator::coolmulti::head`

Referenced by [coolmulti\(\)](#), [finished\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), [next\\_permutation\(\)](#), and [~coolmulti\(\)](#).

### 6.26.3.2 i

`element * GiNaC::composition_generator::coolmulti::i`

Referenced by [coolmulti\(\)](#), and [next\\_permutation\(\)](#).

### 6.26.3.3 after\_i

`element` \* `GiNaC::composition_generator::coolmulti::after_i`

Referenced by `coolmulti()`, `finished()`, and `next_permutation()`.

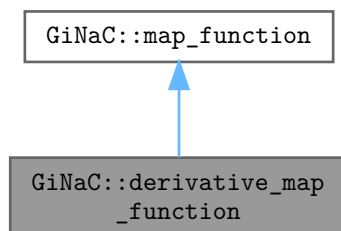
The documentation for this struct was generated from the following file:

- [utils.h](#)

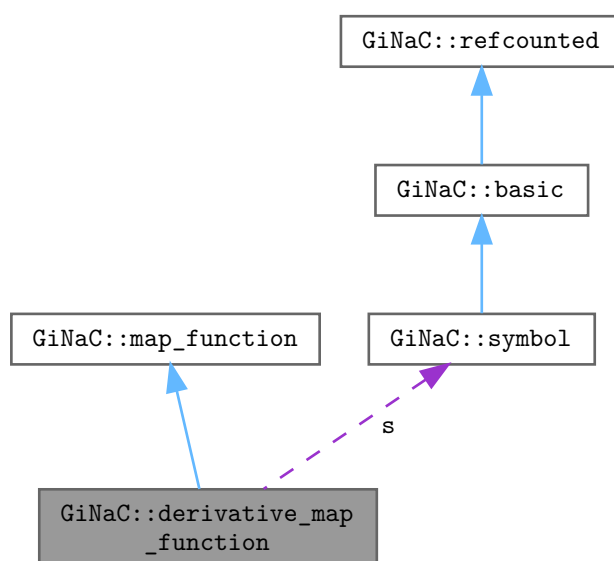
## 6.27 GiNaC::derivative\_map\_function Struct Reference

Function object to be applied by `basic::derivative()`.

Inheritance diagram for `GiNaC::derivative_map_function`:



Collaboration diagram for `GiNaC::derivative_map_function`:



## Public Member Functions

- [derivative\\_map\\_function](#) (const [symbol](#) &sym)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()
- virtual [ex operator\(\)](#) (const [ex](#) &e)=0

## Public Attributes

- const [symbol](#) & s

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 6.27.1 Detailed Description

Function object to be applied by [basic::derivative\(\)](#).

### 6.27.2 Constructor & Destructor Documentation

#### 6.27.2.1 [derivative\\_map\\_function\(\)](#)

```
GiNaC::derivative_map_function::derivative_map_function (
    const symbol & sym ) [inline]
```

### 6.27.3 Member Function Documentation

#### 6.27.3.1 [operator\(\)](#)

```
ex GiNaC::derivative_map_function::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::diff\(\)](#).

## 6.27.4 Member Data Documentation

### 6.27.4.1 s

```
const symbol& GiNaC::derivative_map_function::s
```

The documentation for this struct was generated from the following file:

- [basic.cpp](#)

## 6.28 GiNaC::determinant\_algo Class Reference

Switch to control algorithm for determinant computation.

```
#include <flags.h>
```

### Public Types

- enum {  
    [automatic](#) , [gauss](#) , [divfree](#) , [laplace](#) ,  
    [bareiss](#) }

### 6.28.1 Detailed Description

Switch to control algorithm for determinant computation.

### 6.28.2 Member Enumeration Documentation

---

**Enumerator**


---

**6.28.2.1 anonymous enum**

anonymous enum

**Enumerator**

automatic	Let the system choose. A heuristics is applied for automatic determination of a suitable algorithm.
gauss	<p>Gauss elimination. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$ <p>The determinant is then just the product of diagonal elements. Choose this algorithm only for purely numerical matrices.</p>
divfree	<p>Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$ <p>The determinant can later be computed by inspecting the diagonal elements only. This algorithm is only there for the purpose of cross-checks. It is never fast.</p>
laplace	Laplace elimination. This is plain recursive elimination along minors although multiple minors are avoided by the algorithm. Although the algorithm is exponential in complexity it is frequently the fastest one when the matrix is populated by complicated symbolic expressions.
bareiss	<p>Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is <i>delayed</i> until it can be carried out without computing GCDs. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = (m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}) / m_{k-1,k-1}^{(k-1)}$ <p>(We have set <math>m_{-1,-1}^{(-1)} = 1</math> in order to avoid a case distinction in above formula.) It can be shown that nothing more than polynomial long division is needed for carrying out the division. The determinant can then be read of from the lower right entry. This algorithm is rarely fast for computing determinants.</p>

The documentation for this class was generated from the following file:

- [flags.h](#)

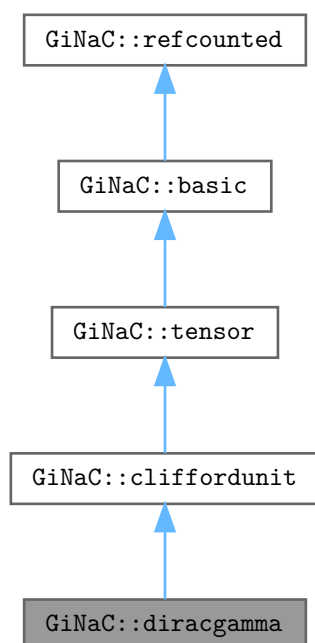
**6.29 GiNaC::diracgamma Class Reference**

This class represents the Dirac gamma Lorentz vector.

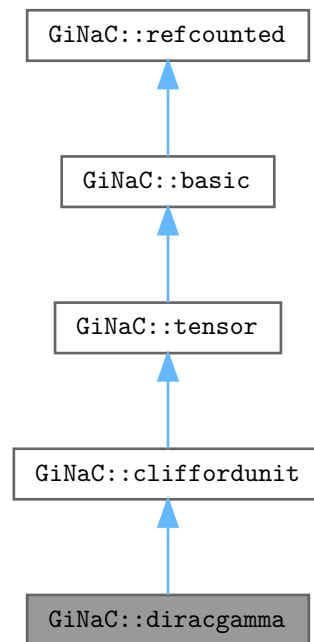
```
#include <clifford.h>
```



Inheritance diagram for GiNaC::diracgamma:



Collaboration diagram for `GiNaC::diracgamma`:



## Public Member Functions

- `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const` override  
*Contraction of a gamma matrix with something else.*
- `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const` override  
*Contraction of a Clifford unit with something else.*

## Public Member Functions inherited from [GiNaC::tensor](#)

- `bool replace_contr_index (exvector::iterator self, exvector::iterator other) const`  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from [GiNaC::basic](#)

- `virtual ~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate () const`

- Create a clone of this object on the heap.*

  - virtual `ex eval` () const
- Perform automatic non-interruptive term rewriting rules.*

  - virtual `ex evalf` () const
- Evaluate object numerically.*

  - virtual `ex evalm` () const
- Evaluate sums, products and integer powers of matrices.*

  - virtual `ex eval_integ` () const
- Evaluate integrals, if result is known.*

  - virtual `ex eval_indexed` (const `basic` &i) const
- Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*

  - virtual void `print` (const `print_context` &c, unsigned level=0) const
- Output to stream.*

  - virtual void `dbgprint` () const
- Little wrapper around print to be called within a debugger.*

  - virtual void `dbgprinttree` () const
- Little wrapper around printtree to be called within a debugger.*

  - virtual unsigned `precedence` () const
- Return relative operator precedence (for parenthezing output).*

  - virtual bool `info` (unsigned inf) const
- Information about the object.*

  - virtual size\_t `nops` () const
- Number of operands/members.*

  - virtual `ex op` (size\_t i) const
- Return operand/member at position i.*

  - virtual `ex operator[]` (const `ex` &index) const
  - virtual `ex operator[]` (size\_t i) const
  - virtual `ex & let_op` (size\_t i)
- Return modifiable operand/member at position i.*

  - virtual `ex & operator[]` (const `ex` &index)
  - virtual `ex & operator[]` (size\_t i)
- Test for occurrence of a pattern.*

  - virtual bool `has` (const `ex` &other, unsigned `options`=0) const
- Check whether the expression matches a given pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual `ex map` (`map_function` &f) const
- Check whether this is a polynomial in the given variables.*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Return degree of highest power in object s.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return coefficient of degree n in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Expand expression, i.e.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const

- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const

*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const

*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const

*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const

*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const

*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*
- const `basic` & `hold` () const

*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const

*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const

*Clear some `status_flags`.*

**Public Member Functions inherited from [GiNaC::refcounted](#)**

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

**Protected Member Functions**

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

**Protected Member Functions inherited from [GiNaC::cliffordunit](#)**

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

**Protected Member Functions inherited from [GiNaC::tensor](#)**

- unsigned [return\\_type](#) () const override

**Protected Member Functions inherited from [GiNaC::basic](#)**

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.29.1 Detailed Description

This class represents the Dirac gamma Lorentz vector.

## 6.29.2 Member Function Documentation

### 6.29.2.1 `contract_with()`

```
bool GiNaC::diracgamma::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of a gamma matrix with something else.

Reimplemented from [GiNaC::cliffordunit](#).

### 6.29.2.2 `do_print()`

```
void GiNaC::diracgamma::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

### 6.29.2.3 `do_print_latex()`

```
void GiNaC::diracgamma::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

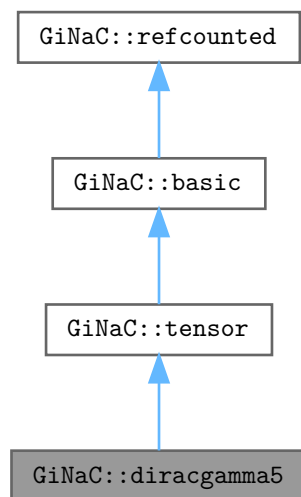
- [clifford.h](#)
- [clifford.cpp](#)

## 6.30 GiNaC::diracgamma5 Class Reference

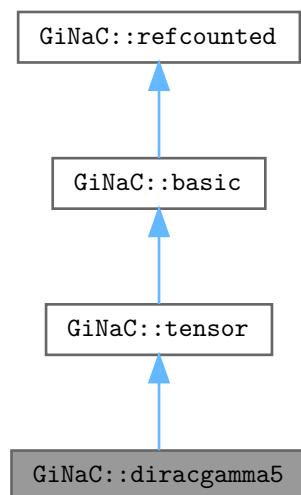
This class represents the Dirac gamma5 object which anticommutes with all other gammas.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgamma5:



Collaboration diagram for GiNaC::diracgamma5:



## Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Private Member Functions

- `ex conjugate` () const override

## Additional Inherited Members

## Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (`exvector::iterator` self, `exvector::iterator` other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*



Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

#### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

#### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.30.1 Detailed Description

This class represents the Dirac gamma5 object which anticommutes with all other gammas.

### 6.30.2 Member Function Documentation

#### 6.30.2.1 conjugate()

```
ex GiNaC::diracgamma5::conjugate ( ) const [override], [private], [virtual]
```

Reimplemented from [GiNaC::basic](#).

#### 6.30.2.2 do\_print()

```
void GiNaC::diracgamma5::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

### 6.30.2.3 do\_print\_latex()

```
void GiNaC::diracgamma5::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

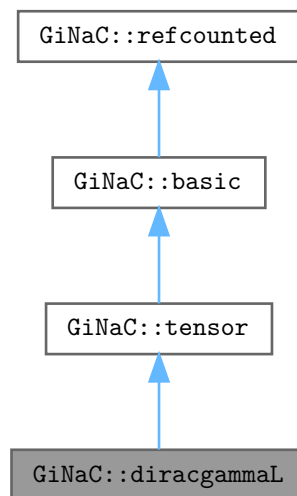
- [clifford.h](#)
- [clifford.cpp](#)

## 6.31 GiNaC::diracgammaL Class Reference

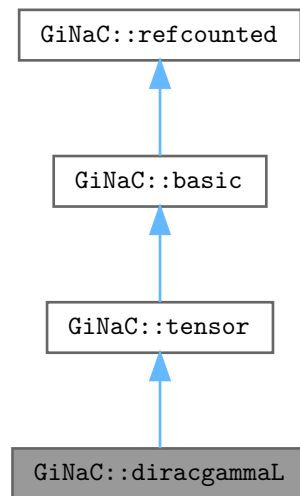
This class represents the Dirac gammaL object which behaves like  $1/2 (1-\gamma_5)$ .

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgammaL:



Collaboration diagram for GiNaC::diracgammaL:



## Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Private Member Functions

- `ex conjugate ()` const override

## Additional Inherited Members

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic & operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate ()` const  
*Create a clone of this object on the heap.*
- virtual `ex eval ()` const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf ()` const  
*Evaluate object numerically.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)

- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_info` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

- Like `print()`, but dispatch to the specified class.
- virtual void `archive (archive_node &n)` const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive (const archive_node &n, lst &syms)`  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level (const exmap &m, unsigned options)` const  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1)` const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare (const basic &other)` const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal (const basic &other)` const  
*Test for syntactic equality.*
- const `basic & hold ()` const  
*Stop further evaluation.*
- unsigned `gethash ()` const
- const `basic & setflag (unsigned f)` const  
*Set some `status_flags`.*
- const `basic & clearflag (unsigned f)` const  
*Clear some `status_flags`.*

#### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted ()` noexcept
- unsigned int `add_reference ()` noexcept
- unsigned int `remove_reference ()` noexcept
- unsigned int `get_refcount ()` const noexcept
- void `set_refcount (unsigned int r)` noexcept

#### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.31.1 Detailed Description

This class represents the Dirac gammaL object which behaves like  $1/2 (1-\gamma_5)$ .

### 6.31.2 Member Function Documentation



### 6.31.2.1 conjugate()

```
ex GiNaC::diracgammaL::conjugate ( ) const [override], [private], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.31.2.2 do\_print()

```
void GiNaC::diracgammaL::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.31.2.3 do\_print\_latex()

```
void GiNaC::diracgammaL::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

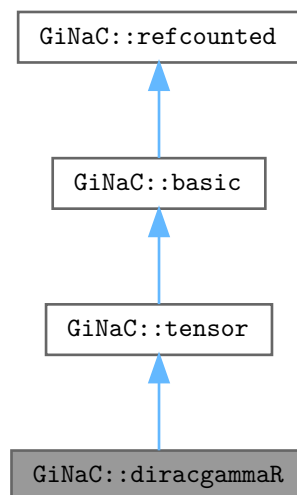
- [clifford.h](#)
- [clifford.cpp](#)

## 6.32 GiNaC::diracgammaR Class Reference

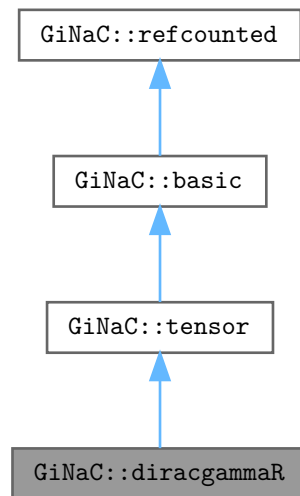
This class represents the Dirac gammaL object which behaves like  $\frac{1}{2}(1+\gamma_5)$ .

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgammaR:



Collaboration diagram for `GiNaC::diracgammaR`:



## Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Private Member Functions

- `ex conjugate ()` const override

## Additional Inherited Members

### Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index (exvector::iterator self, exvector::iterator other) const`  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate () const`  
*Create a clone of this object on the heap.*
- `virtual ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf () const`  
*Evaluate object numerically.*
- `virtual ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- `virtual ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- `virtual void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- `virtual void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- `virtual unsigned precedence () const`  
*Return relative operator precedence (for parenthesizing output).*
- `virtual bool info (unsigned inf) const`  
*Information about the object.*
- `virtual size_t nops () const`  
*Number of operands/members.*
- `virtual ex op (size_t i) const`  
*Return operand/member at position i.*
- `virtual ex operator[] (const ex &index) const`
- `virtual ex operator[] (size_t i) const`
- `virtual ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- `virtual ex & operator[] (const ex &index)`

- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_info` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

- Like `print()`, but dispatch to the specified class.
- virtual void `archive (archive_node &n)` const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive (const archive_node &n, lst &syms)`  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level (const exmap &m, unsigned options)` const  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1)` const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare (const basic &other)` const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal (const basic &other)` const  
*Test for syntactic equality.*
- const `basic & hold ()` const  
*Stop further evaluation.*
- unsigned `gethash ()` const
- const `basic & setflag (unsigned f)` const  
*Set some `status_flags`.*
- const `basic & clearflag (unsigned f)` const  
*Clear some `status_flags`.*

#### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted ()` noexcept
- unsigned int `add_reference ()` noexcept
- unsigned int `remove_reference ()` noexcept
- unsigned int `get_refcount ()` const noexcept
- void `set_refcount (unsigned int r)` noexcept

#### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.32.1 Detailed Description

This class represents the Dirac gammaL object which behaves like  $1/2 (1+\gamma_5)$ .

### 6.32.2 Member Function Documentation

### 6.32.2.1 conjugate()

```
ex GiNaC::diracgammaR::conjugate ( ) const [override], [private], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.32.2.2 do\_print()

```
void GiNaC::diracgammaR::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

### 6.32.2.3 do\_print\_latex()

```
void GiNaC::diracgammaR::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

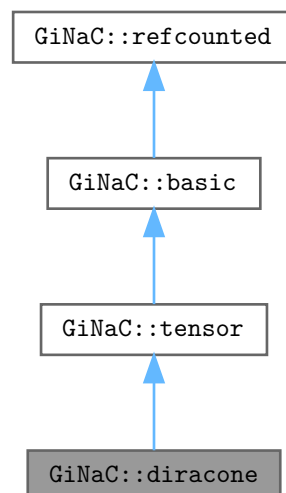
- [clifford.h](#)
- [clifford.cpp](#)

## 6.33 GiNaC::diracone Class Reference

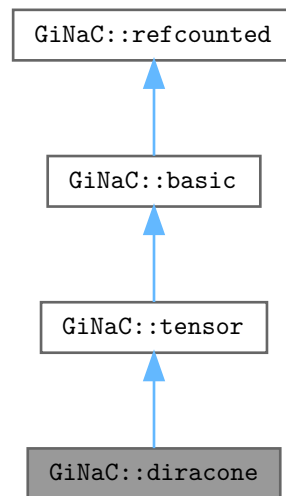
This class represents the Clifford algebra unity element.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracone:



Collaboration diagram for GiNaC::diracone:



## Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Public Member Functions inherited from [GiNaC::tensor](#)

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*



- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*

- `ex subs_one_level` (const `exmap` &`m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &`s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &`other`) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &`other`) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- `const basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 6.33.1 Detailed Description

This class represents the Clifford algebra unity element.

## 6.33.2 Member Function Documentation

### 6.33.2.1 `do_print()`

```
void GiNaC::diracone::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.33.2.2 do\_print\_latex()

```
void GiNaC::diracone::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following file:

- [clifford.h](#)

## 6.34 GiNaC::do\_taylor Class Reference

Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.

```
#include <function.h>
```

### 6.34.1 Detailed Description

Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.

The documentation for this class was generated from the following file:

- [function.h](#)

## 6.35 GiNaC::domain Class Reference

Domain of an object.

```
#include <flags.h>
```

### Public Types

- enum { [complex](#) , [real](#) , [positive](#) }

### 6.35.1 Detailed Description

Domain of an object.

### 6.35.2 Member Enumeration Documentation

#### 6.35.2.1 anonymous enum

```
anonymous enum
```

## Enumerator

complex	
real	
positive	

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.36 GiNaC::dunno Class Reference

Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold()

```
#include <utils.h>
```

### 6.36.1 Detailed Description

Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold()

The documentation for this class was generated from the following file:

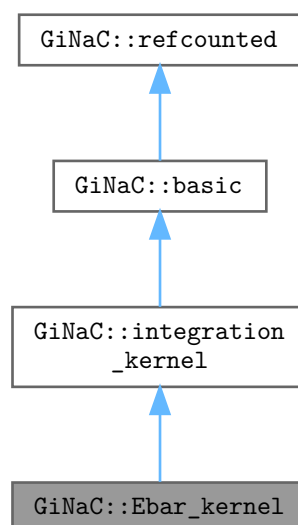
- [utils.h](#)

## 6.37 GiNaC::Ebar\_kernel Class Reference

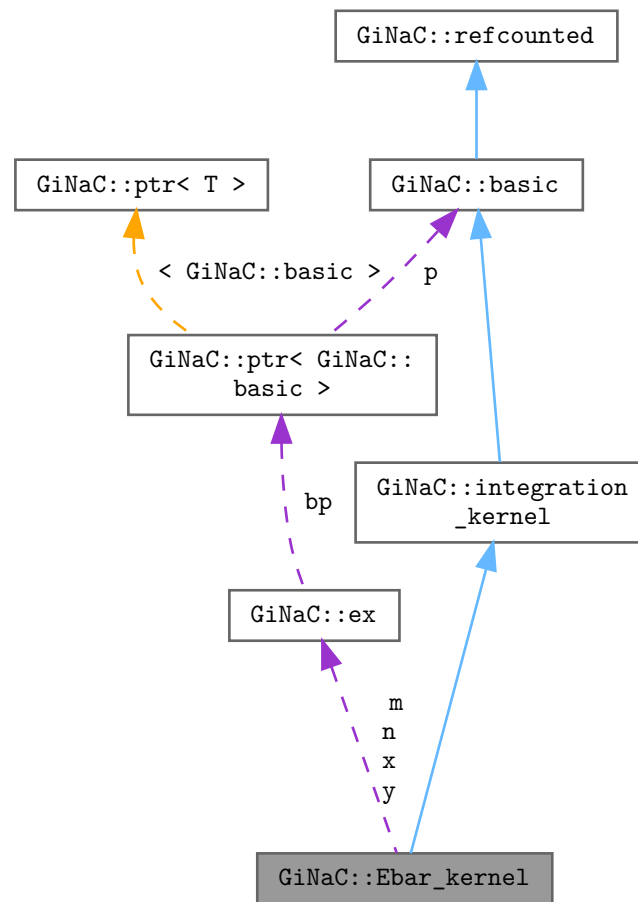
The Ebar-kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Ebar\_kernel:



Collaboration diagram for GiNaC::Ebar\_kernel:



## Public Member Functions

- `Ebar_kernel` (const `ex` &`n`, const `ex` &`m`, const `ex` &`x`, const `ex` &`y`)
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position i.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position i.*
- `bool is_numeric (void)` const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value (const ex &qbar, int N_trunc=0)` const override  
*Returns the value of `Ebar_{n,m}(x,y,qbar)`*

### Public Member Functions inherited from [GiNaC::integration\\_kernel](#)

- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Default implementation of [ex::series\(\)](#).*
- virtual bool [has\\_trailing\\_zero](#) (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool [is\\_numeric](#) (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual [ex Laurent\\_series](#) (const [ex](#) &x, int [order](#)) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual [ex get\\_numerical\\_value](#) (const [ex](#) &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t [get\\_cache\\_size](#) (void) const  
*Returns the current size of the cache.*
- void [set\\_cache\\_step](#) (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- [ex get\\_series\\_coeff](#) (int i) const  
*Wrapper around [series\\_coeff\(i\)](#), converts [cl\\_N](#) to numeric.*
- [cln::cl\\_N series\\_coeff](#) (int i) const  
*Subclasses have either to implement [series\\_coeff\\_impl](#) or the two methods [Laurent\\_series](#) and [uses\\_Laurent\\_series](#).*

### Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class [ex](#) will delete objects of derived classes via a [basic\\*](#).*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic \\* duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around [print](#) to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around [printtree](#) to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*

- virtual `size_t nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v)` const
- virtual `bool is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl)` const
- virtual `numeric integer_content ()` const
- virtual `ex smod (const numeric &xi)` const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*

- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- virtual `cln::cl_N series_coeff_impl` (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const



### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- [ex n](#)
- [ex m](#)
- [ex x](#)
- [ex y](#)

### Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.37.1 Detailed Description

The Ebar-kernel.

This class represents the differential one-form

$$\omega_{n;m}^{\bar{E}}(x; y) = \bar{E}_{n;m}(x; y; \bar{q}) \frac{d\bar{q}}{\bar{q}}$$

## 6.37.2 Constructor & Destructor Documentation

### 6.37.2.1 Ebar\_kernel()

```
GiNaC::Ebar_kernel::Ebar_kernel (
    const ex & n,
    const ex & m,
    const ex & x,
    const ex & y )
```

## 6.37.3 Member Function Documentation

### 6.37.3.1 nops()

```
size_t GiNaC::Ebar_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.37.3.2 op()

```
ex GiNaC::Ebar_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [m](#), [n](#), [x](#), and [y](#).

### 6.37.3.3 let\_op()

```
ex & GiNaC::Ebar_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [m](#), [n](#), [x](#), and [y](#).

### 6.37.3.4 is\_numeric()

```
bool GiNaC::Ebar_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [x](#), and [y](#).

### 6.37.3.5 get\_numerical\_value()

```
ex GiNaC::Ebar_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of  $Ebar_{\{n,m\}}(x,y,qbar)$

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 6.37.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::Ebar_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [k](#), [m](#), [n](#), [x](#), and [y](#).

### 6.37.3.7 do\_print()

```
void GiNaC::Ebar_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [m](#), [n](#), [GiNaC::ex::print\(\)](#), [x](#), and [y](#).

## 6.37.4 Member Data Documentation

### 6.37.4.1 `n`

`ex` `GiNaC::Ebar_kernel::n` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.37.4.2 `m`

`ex` `GiNaC::Ebar_kernel::m` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.37.4.3 `x`

`ex` `GiNaC::Ebar_kernel::x` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.37.4.4 `y`

`ex` `GiNaC::Ebar_kernel::y` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

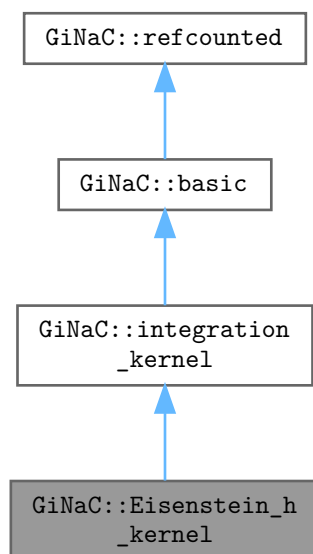
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.38 GiNaC::Eisenstein\_h\_kernel Class Reference

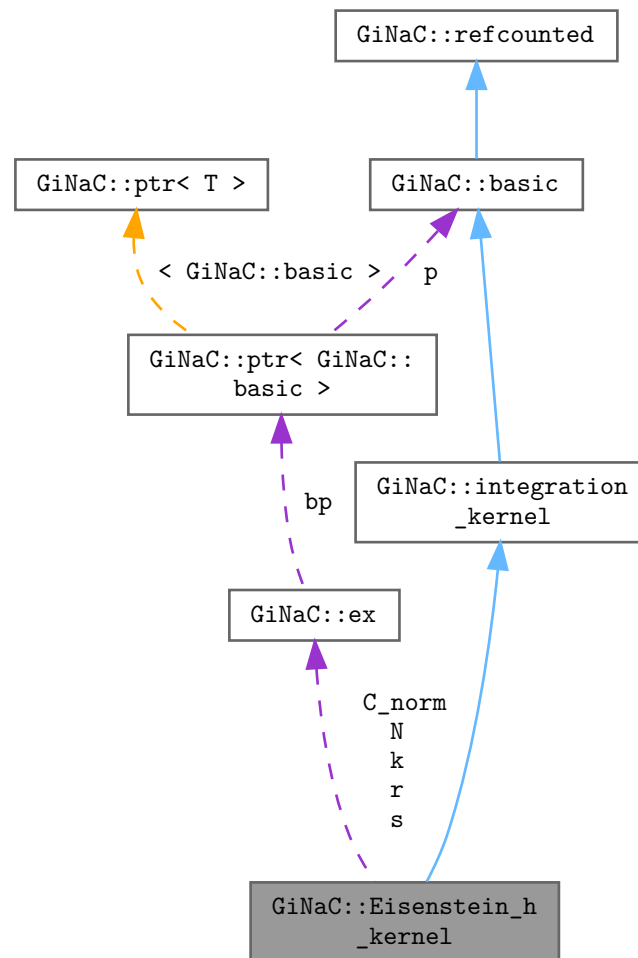
The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Eisenstein\_h\_kernel:



Collaboration diagram for `GiNaC::Eisenstein_h_kernel`:



## Public Member Functions

- `Eisenstein_h_kernel` (const `ex` &`k`, const `ex` &`N`, const `ex` &`r`, const `ex` &`s`, const `ex` &`C_norm=numeric(1)`)
- `ex series` (const `relational` &`r`, int `order`, unsigned `options=0`) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of  $C\_norm/qbar$ .*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position  $i$ .*
- `ex &let_op` (size\_t `i`) override  
*Return modifiable operand/member at position  $i$ .*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series` (const `ex` &`x`, int `order`) const override

- Returns the Laurent series, starting possibly with the pole term.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc`=0) const override  
*Returns the value of the modular form.*
- `ex coefficient_a0` (const `numeric` &`k`, const `numeric` &`r`, const `numeric` &`s`, const `numeric` &`N`) const  
*The constant coefficient in the Fourier expansion.*
- `ex coefficient_an` (const `numeric` &`n`, const `numeric` &`k`, const `numeric` &`r`, const `numeric` &`s`, const `numeric` &`N`) const  
*The higher coefficients in the Fourier expansion.*
- `ex q_expansion_modular_form` (const `ex` &`q`, int `order`) const

#### Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool `is_numeric` (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual `ex Laurent_series` (const `ex` &`x`, int `order`) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex` &`lambda`, int `N_trunc`=0) const  
*Evaluates the integrand at `lambda`.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int `cache_steps`) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int `i`) const  
*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- `cln::cl_N series_coeff` (int `i`) const  
*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

#### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &`i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*

- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprintree` () const  
*Little wrapper around printree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*



- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `unsigned return_type ()` const
- virtual `return_type_t return_type_tinfo ()` const
- virtual `ex conjugate ()` const
- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- `template<class T >`  
void `print_dispatch (const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive (archive_node &n)` const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive (const archive_node &n, lst &syms)`  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level (const exmap &m, unsigned options)` const  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1)` const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare (const basic &other)` const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal (const basic &other)` const  
*Test for syntactic equality.*
- const `basic & hold ()` const  
*Stop further evaluation.*
- unsigned `gethash ()` const
- const `basic & setflag (unsigned f)` const  
*Set some `status_flags`.*
- const `basic & clearflag (unsigned f)` const  
*Clear some `status_flags`.*

#### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted ()` noexcept
- unsigned int `add_reference ()` noexcept
- unsigned int `remove_reference ()` noexcept
- unsigned int `get_refcount ()` const noexcept
- void `set_refcount (unsigned int r)` noexcept

## Protected Member Functions

- bool [uses\\_Laurent\\_series](#) () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::integration\\_kernel](#)

- virtual bool [uses\\_Laurent\\_series](#) () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- virtual `cln::cl_N` [series\\_coeff\\_impl](#) (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex` [get\\_numerical\\_value\\_impl](#) (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual `ex` [eval\\_ncmul](#) (const `exvector` &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex` [derivative](#) (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- `ex` k
- `ex` N
- `ex` r
- `ex` s
- `ex` C\_norm

**Protected Attributes inherited from [GiNaC::integration\\_kernel](#)**

- int [cache\\_step\\_size](#)
- std::vector< cln::cl\_N > [series\\_vec](#)

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

**6.38.1 Detailed Description**

The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .

This class represents the differential one-form

$$\omega_{k,N,r,s}^{\text{Eisenstein,h}} = C_k h_{k,N,r,s}(\tau) \frac{d\bar{q}_N}{\bar{q}_N}$$

**6.38.2 Constructor & Destructor Documentation****6.38.2.1 Eisenstein\_h\_kernel()**

```
GiNaC::Eisenstein_h_kernel::Eisenstein_h_kernel (
    const ex & k,
    const ex & N,
    const ex & r,
    const ex & s,
    const ex & C_norm = numeric(1) )
```

**6.38.3 Member Function Documentation****6.38.3.1 series()**

```
ex GiNaC::Eisenstein_h_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

This allows for easy use in the class [modular\\_form\\_kernel](#).

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::lhs\(\)](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [r](#), [GiNaC::ex::rhs\(\)](#), and [GiNaC::ex::series\(\)](#).

### 6.38.3.2 nops()

```
size_t GiNaC::Eisenstein_h_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.38.3.3 op()

```
ex GiNaC::Eisenstein_h_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [k](#), [N](#), [r](#), and [s](#).

### 6.38.3.4 let\_op()

```
ex & GiNaC::Eisenstein_h_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [k](#), [N](#), [r](#), and [s](#).

### 6.38.3.5 is\_numeric()

```
bool GiNaC::Eisenstein_h_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [k](#), [N](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [r](#), and [s](#).

### 6.38.3.6 Laurent\_series()

```
ex GiNaC::Eisenstein_h_kernel::Laurent_series (
    const ex & x,
    int order ) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::ex::series\(\)](#), and [x](#).

### 6.38.3.7 get\_numerical\_value()

```
ex GiNaC::Eisenstein_h_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

### 6.38.3.8 uses\_Laurent\_series()

```
bool GiNaC::Eisenstein_h_kernel::uses_Laurent_series ( ) const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.38.3.9 coefficient\_a0()

```
ex GiNaC::Eisenstein_h_kernel::coefficient_a0 (
    const numeric & k,
    const numeric & r,
    const numeric & s,
    const numeric & N ) const
```

The constant coefficient in the Fourier expansion.

References [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::cos\(\)](#), [GiNaC::I](#), [GiNaC::irem\(\)](#), [k](#), [GiNaC::mod\(\)](#), [N](#), [GiNaC::Pi](#), [r](#), [s](#), and [GiNaC::sin\(\)](#).

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

### 6.38.3.10 coefficient\_an()

```
ex GiNaC::Eisenstein_h_kernel::coefficient_an (
    const numeric & n,
    const numeric & k,
    const numeric & r,
    const numeric & s,
    const numeric & N ) const
```

The higher coefficients in the Fourier expansion.

References [GiNaC::exp\(\)](#), [GiNaC::l](#), [GiNaC::irem\(\)](#), [k](#), [m](#), [GiNaC::mod\(\)](#), [n](#), [N](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), [r](#), and [s](#).

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

### 6.38.3.11 q\_expansion\_modular\_form()

```
ex GiNaC::Eisenstein_h_kernel::q_expansion_modular_form (
    const ex & q,
    int order ) const
```

References [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [k](#), [N](#), [GiNaC::pow\(\)](#), [r](#), [s](#), and [GiNaC::ex::series\(\)](#).

Referenced by [Laurent\\_series\(\)](#), and [series\(\)](#).

### 6.38.3.12 do\_print()

```
void GiNaC::Eisenstein_h_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [C\\_norm](#), [k](#), [N](#), [GiNaC::ex::print\(\)](#), [r](#), and [s](#).

## 6.38.4 Member Data Documentation

### 6.38.4.1 k

```
ex GiNaC::Eisenstein_h_kernel::k [protected]
```

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

#### 6.38.4.2 N

`ex` GiNaC::Eisenstein\_h\_kernel::N [protected]

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

#### 6.38.4.3 r

`ex` GiNaC::Eisenstein\_h\_kernel::r [protected]

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [q\\_expansion\\_modular\\_form\(\)](#), and [series\(\)](#).

#### 6.38.4.4 s

`ex` GiNaC::Eisenstein\_h\_kernel::s [protected]

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

#### 6.38.4.5 C\_norm

`ex` GiNaC::Eisenstein\_h\_kernel::C\_norm [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

The documentation for this class was generated from the following files:

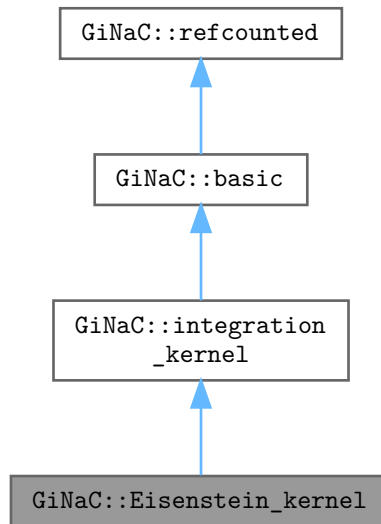
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.39 GiNaC::Eisenstein\_kernel Class Reference

The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .

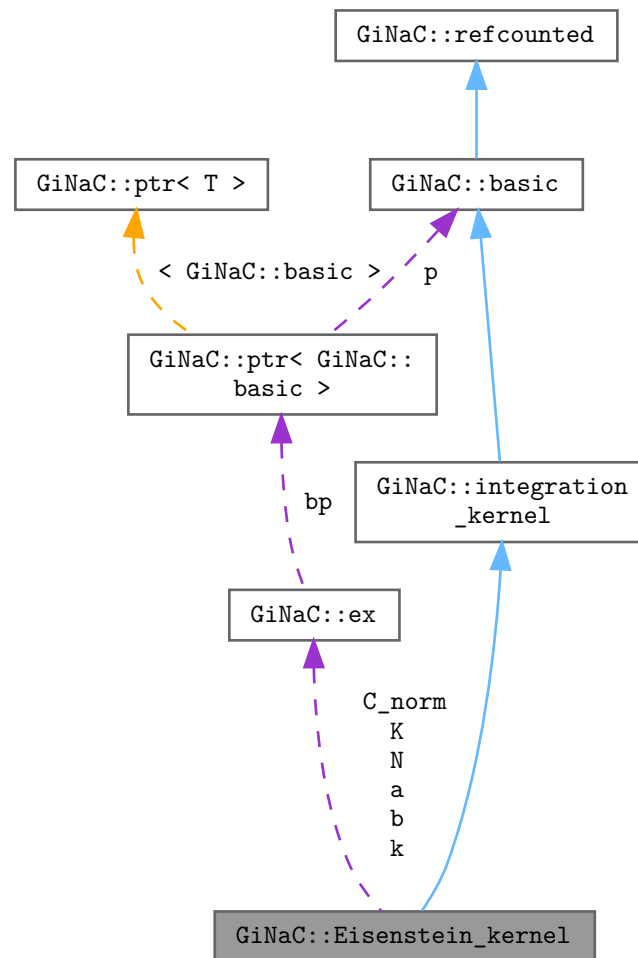
```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Eisenstein\_kernel:





Collaboration diagram for GiNaC::Eisenstein\_kernel:



## Public Member Functions

- `Eisenstein_kernel` (const `ex` &`k`, const `ex` &`N`, const `ex` &`a`, const `ex` &`b`, const `ex` &`K`, const `ex` &`C_norm`=numeric(1))
- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position i.*
- `ex &let_op` (size\_t `i`) override  
*Return modifiable operand/member at position i.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*

- `ex Laurent_series` (const `ex &x`, int `order`) const override  
*Returns the Laurent series, starting possibly with the pole term.*
- `ex get_numerical_value` (const `ex &qbar`, int `N_trunc=0`) const override  
*Returns the value of the modular form.*
- `ex q_expansion_modular_form` (const `ex &q`, int `order`) const

#### Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational &r`, int `order`, unsigned `options=0`) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool `is_numeric` (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual `ex Laurent_series` (const `ex &x`, int `order`) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex &lambda`, int `N_trunc=0`) const  
*Evaluates the integrand at `lambda`.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int `cache_steps`) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int `i`) const  
*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- `cln::cl_N series_coeff` (int `i`) const  
*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

#### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic &other`)
- const `basic & operator=` (const `basic &other`)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic &i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context &c`, unsigned `level=0`) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*

- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex & let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex & operator\[\]](#) (const [ex](#) &index)
- virtual [ex & operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) (GiNaC::visitor &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned [options](#)=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*

- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t` `return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- bool `uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- void `do_print` (const `print_context` &c, unsigned level) const

**Protected Member Functions inherited from [GiNaC::integration\\_kernel](#)**

- virtual bool [uses\\_Laurent\\_series](#) () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method [Laurent\\_series](#) needs to be implemented).*
- virtual [cln::cl\\_N](#) [series\\_coeff\\_impl](#) (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- [ex](#) [get\\_numerical\\_value\\_impl](#) (const [ex](#) &lambda, const [ex](#) &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

**Protected Member Functions inherited from [GiNaC::basic](#)**

- [basic](#) ()
- virtual [ex](#) [eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex](#) [derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

**Protected Attributes**

- [ex](#) k
- [ex](#) N
- [ex](#) a
- [ex](#) b
- [ex](#) K
- [ex](#) C\_norm

**Protected Attributes inherited from [GiNaC::integration\\_kernel](#)**

- int [cache\\_step\\_size](#)
- [std::vector](#)< [cln::cl\\_N](#) > [series\\_vec](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.39.1 Detailed Description

The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .

This class represents the differential one-form

$$\omega_{k,N,a,b,K}^{\text{Eisenstein}} = C_k E_{k,N,a,b,K}(\tau) \frac{d\bar{q}_N}{\bar{q}_N}$$

The integers a and b are either one or the discriminant of a quadratic number field. This class represents Eisenstein series, which can be defined by primitive Dirichlet characters from the Kronecker symbol. This implies that the characters take the values -1,0,1, i.e. no higher roots of unity occur. The

$$\bar{q}$$

-expansion has then rational coefficients.

Ref.: W. Stein, Modular Forms: A Computational Approach, Chapter 5

## 6.39.2 Constructor & Destructor Documentation

### 6.39.2.1 Eisenstein\_kernel()

```
GiNaC::Eisenstein_kernel::Eisenstein_kernel (
    const ex & k,
    const ex & N,
    const ex & a,
    const ex & b,
    const ex & K,
    const ex & C_norm = numeric(1) )
```

## 6.39.3 Member Function Documentation

### 6.39.3.1 series()

```
ex GiNaC::Eisenstein_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of  $C\_norm/qbar$ .

This allows for easy use in the class [modular\\_form\\_kernel](#).

Reimplemented from [GiNaC::integration\\_kernel](#).

References [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [r](#), and [GiNaC::ex::series\(\)](#).

### 6.39.3.2 nops()

```
size_t GiNaC::Eisenstein_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.39.3.3 op()

```
ex GiNaC::Eisenstein_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position  $i$ .

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [C\\_norm](#), [k](#), [K](#), and [N](#).

### 6.39.3.4 let\_op()

```
ex & GiNaC::Eisenstein_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position  $i$ .

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [k](#), [K](#), and [N](#).

### 6.39.3.5 is\_numeric()

```
bool GiNaC::Eisenstein_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [a](#), [b](#), [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [k](#), [K](#), [N](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), and [GiNaC::info\\_flags::posint](#).

### 6.39.3.6 Laurent\_series()

```
ex GiNaC::Eisenstein_kernel::Laurent_series (
    const ex & x,
    int order ) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::ex::series\(\)](#), and [x](#).

### 6.39.3.7 get\_numerical\_value()

```
ex GiNaC::Eisenstein_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

### 6.39.3.8 uses\_Laurent\_series()

```
bool GiNaC::Eisenstein_kernel::uses_Laurent_series ( ) const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).



### 6.39.3.9 q\_expansion\_modular\_form()

```
ex GiNaC::Eisenstein_kernel::q_expansion_modular_form (
    const ex & q,
    int order ) const
```

References [a](#), [b](#), [k](#), [K](#), [N](#), and [order](#).

Referenced by [Laurent\\_series\(\)](#), and [series\(\)](#).

### 6.39.3.10 do\_print()

```
void GiNaC::Eisenstein_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [a](#), [b](#), [c](#), [C\\_norm](#), [k](#), [K](#), [N](#), and [GiNaC::ex::print\(\)](#).

## 6.39.4 Member Data Documentation

### 6.39.4.1 k

```
ex GiNaC::Eisenstein_kernel::k [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

### 6.39.4.2 N

```
ex GiNaC::Eisenstein_kernel::N [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

### 6.39.4.3 a

```
ex GiNaC::Eisenstein_kernel::a [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

#### 6.39.4.4 **b**

`ex GiNaC::Eisenstein_kernel::b [protected]`

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

#### 6.39.4.5 **K**

`ex GiNaC::Eisenstein_kernel::K [protected]`

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

#### 6.39.4.6 **C\_norm**

`ex GiNaC::Eisenstein_kernel::C_norm [protected]`

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

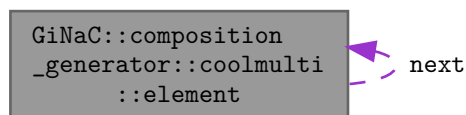
The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.40 GiNaC::composition\_generator::coolmulti::element Struct Reference

```
#include <utils.h>
```

Collaboration diagram for GiNaC::composition\_generator::coolmulti::element:



### Public Member Functions

- [element](#) (unsigned val, [element](#) \*n)
- [~element](#) ()

## Public Attributes

- unsigned [value](#)
- [element](#) \* [next](#)

## 6.40.1 Constructor & Destructor Documentation

### 6.40.1.1 [element\(\)](#)

```
GiNaC::composition_generator::coolmulti::element::element (
    unsigned val,
    element * n ) [inline]
```

### 6.40.1.2 [~element\(\)](#)

```
GiNaC::composition_generator::coolmulti::element::~~element ( ) [inline]
```

References [next](#).

## 6.40.2 Member Data Documentation

### 6.40.2.1 [value](#)

```
unsigned GiNaC::composition_generator::coolmulti::element::value
```

Referenced by [GiNaC::composition\\_generator::coolmulti::finished\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), and [GiNaC::composition\\_generator::coolmulti::next\\_permutation\(\)](#).

### 6.40.2.2 [next](#)

```
element* GiNaC::composition_generator::coolmulti::element::next
```

Referenced by [GiNaC::composition\\_generator::coolmulti::coolmulti\(\)](#), [GiNaC::composition\\_generator::coolmulti::finished\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), [GiNaC::composition\\_generator::coolmulti::next\\_permutation\(\)](#), and [~element\(\)](#).

The documentation for this struct was generated from the following file:

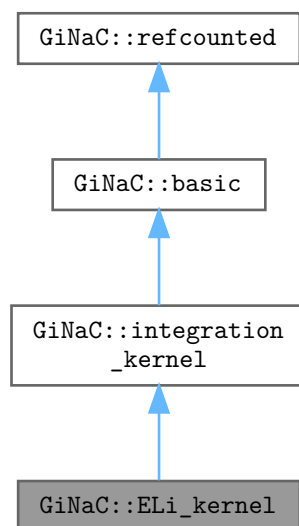
- [utils.h](#)

## 6.41 GiNaC::ELi\_kernel Class Reference

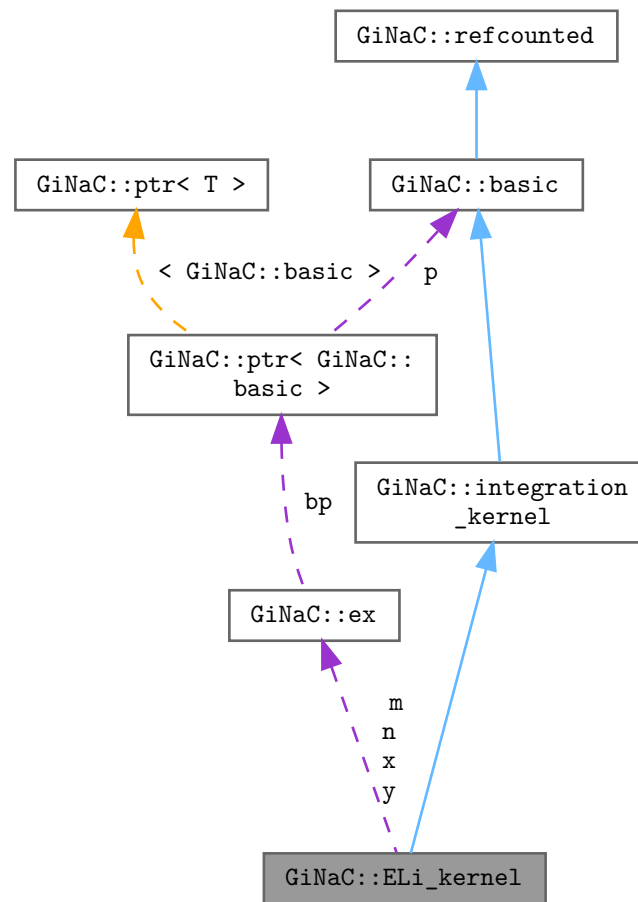
The ELi-kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::ELi\_kernel:



Collaboration diagram for GiNaC::ELi\_kernel:



## Public Member Functions

- `ELi_kernel` (const `ex` &`n`, const `ex` &`m`, const `ex` &`x`, const `ex` &`y`)
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position i.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position i.*
- `bool is_numeric (void)` const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value (const ex &qbar, int N_trunc=0)` const override  
*Returns the value of  $ELi_{\{n,m\}}(x,y,qbar)$*

### Public Member Functions inherited from [GiNaC::integration\\_kernel](#)

- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Default implementation of [ex::series\(\)](#).*
- virtual bool [has\\_trailing\\_zero](#) (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool [is\\_numeric](#) (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual [ex Laurent\\_series](#) (const [ex](#) &x, int [order](#)) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual [ex get\\_numerical\\_value](#) (const [ex](#) &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t [get\\_cache\\_size](#) (void) const  
*Returns the current size of the cache.*
- void [set\\_cache\\_step](#) (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- [ex get\\_series\\_coeff](#) (int i) const  
*Wrapper around [series\\_coeff\(i\)](#), converts [cl\\_N](#) to numeric.*
- [cln::cl\\_N series\\_coeff](#) (int i) const  
*Subclasses have either to implement [series\\_coeff\\_impl](#) or the two methods [Laurent\\_series](#) and [uses\\_Laurent\\_series](#).*

### Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class [ex](#) will delete objects of derived classes via a [basic\\*](#).*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic \\* duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around [print](#) to be called within a debugger.*
- virtual void [dbgprnttree](#) () const  
*Little wrapper around [prnttree](#) to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*

- virtual `size_t nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v)` const
- virtual `bool is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl)` const
- virtual `numeric integer_content ()` const
- virtual `ex smod (const numeric &xi)` const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*

- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- virtual `cln::cl_N series_coeff_impl` (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const



**Protected Member Functions inherited from GiNaC::basic**

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

**Protected Attributes**

- [ex n](#)
- [ex m](#)
- [ex x](#)
- [ex y](#)

**Protected Attributes inherited from GiNaC::integration\_kernel**

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

**Protected Attributes inherited from GiNaC::basic**

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

**6.41.1 Detailed Description**

The ELi-kernel.

This class represents the differential one-form

$$\omega_{n;m}^{\text{ELi}}(x; y) = \text{ELi}_{n;m}(x; y; \bar{q}) \frac{d\bar{q}}{\bar{q}}$$

## 6.41.2 Constructor & Destructor Documentation

### 6.41.2.1 ELi\_kernel()

```
GiNaC::ELi_kernel::ELi_kernel (
    const ex & n,
    const ex & m,
    const ex & x,
    const ex & y )
```

## 6.41.3 Member Function Documentation

### 6.41.3.1 nops()

```
size_t GiNaC::ELi_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.41.3.2 op()

```
ex GiNaC::ELi_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [m](#), [n](#), [x](#), and [y](#).

### 6.41.3.3 let\_op()

```
ex & GiNaC::ELi_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [m](#), [n](#), [x](#), and [y](#).

#### 6.41.3.4 is\_numeric()

```
bool GiNaC::ELi_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [x](#), and [y](#).

#### 6.41.3.5 get\_numerical\_value()

```
ex GiNaC::ELi_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of  $\text{ELi}_{\{n,m\}}(x,y,qbar)$

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

#### 6.41.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::ELi_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [k](#), [m](#), [n](#), [x](#), and [y](#).

#### 6.41.3.7 do\_print()

```
void GiNaC::ELi_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [m](#), [n](#), [GiNaC::ex::print\(\)](#), [x](#), and [y](#).

## 6.41.4 Member Data Documentation

### 6.41.4.1 `n`

`ex GiNaC::ELi_kernel::n` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.41.4.2 `m`

`ex GiNaC::ELi_kernel::m` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.41.4.3 `x`

`ex GiNaC::ELi_kernel::x` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.41.4.4 `y`

`ex GiNaC::ELi_kernel::y` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.42 `std::equal_to< GiNaC::ex >` Struct Reference

Specialization of `std::equal_to()` for `ex` objects.

```
#include <ex.h>
```

## Public Member Functions

- bool [operator\(\)](#) (const [GiNaC::ex](#) &e1, const [GiNaC::ex](#) &e2) const noexcept

### 6.42.1 Detailed Description

Specialization of std::equal\_to() for ex objects.

### 6.42.2 Member Function Documentation

#### 6.42.2.1 operator()

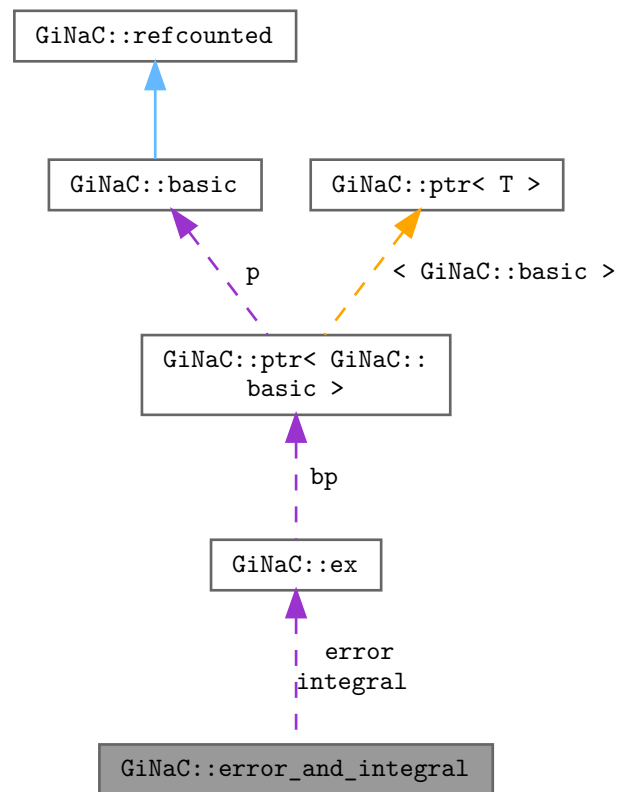
```
bool std::equal_to< GiNaC::ex >::operator() (
    const GiNaC::ex & e1,
    const GiNaC::ex & e2 ) const [inline], [noexcept]
```

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.43 GiNaC::error\_and\_integral Struct Reference

Collaboration diagram for GiNaC::error\_and\_integral:



### Public Member Functions

- [error\\_and\\_integral](#) (const [ex](#) &err, const [ex](#) &integ)

### Public Attributes

- [ex error](#)
- [ex integral](#)

#### 6.43.1 Constructor & Destructor Documentation

### 6.43.1.1 error\_and\_integral()

```
GiNaC::error_and_integral::error_and_integral (
    const ex & err,
    const ex & integ ) [inline]
```

## 6.43.2 Member Data Documentation

### 6.43.2.1 error

[ex](#) [GiNaC::error\\_and\\_integral::error](#)

Referenced by [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)](#)).

### 6.43.2.2 integral

[ex](#) [GiNaC::error\\_and\\_integral::integral](#)

Referenced by [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)](#)).

The documentation for this struct was generated from the following file:

- [integral.cpp](#)

## 6.44 GiNaC::error\_and\_integral\_is\_less Struct Reference

### Public Member Functions

- [bool operator\(\)](#) (const [error\\_and\\_integral](#) &e1, const [error\\_and\\_integral](#) &e2) const

### 6.44.1 Member Function Documentation

#### 6.44.1.1 operator()

```
bool GiNaC::error_and_integral_is_less::operator() (
    const error\_and\_integral & e1,
    const error\_and\_integral & e2 ) const [inline]
```

References [c](#), [GiNaC::ex::compare\(\)](#), [GiNaC::error\\_and\\_integral::error](#), and [GiNaC::error\\_and\\_integral::integral](#).

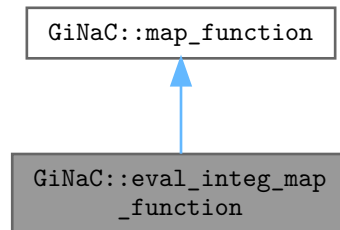
The documentation for this struct was generated from the following file:

- [integral.cpp](#)

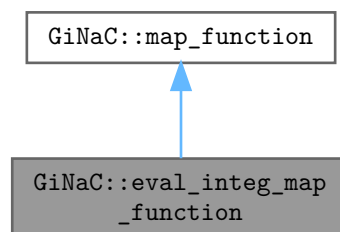
## 6.45 GiNaC::eval\_integ\_map\_function Struct Reference

Function object to be applied by [basic::eval\\_integ\(\)](#).

Inheritance diagram for GiNaC::eval\_integ\_map\_function:



Collaboration diagram for GiNaC::eval\_integ\_map\_function:



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()
- virtual [ex operator\(\)](#) (const [ex](#) &e)=0

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)



### 6.45.1 Detailed Description

Function object to be applied by [basic::eval\\_integ\(\)](#).

### 6.45.2 Member Function Documentation

#### 6.45.2.1 operator()

```
ex GiNaC::eval_integ_map_function::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::eval\\_integ\(\)](#).

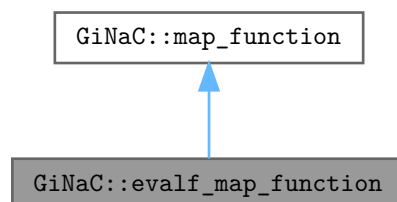
The documentation for this struct was generated from the following file:

- [basic.cpp](#)

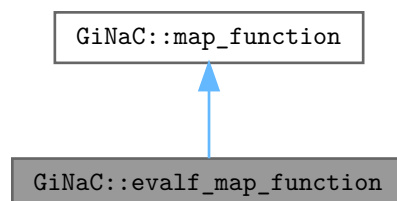
## 6.46 GiNaC::evalf\_map\_function Struct Reference

Function object to be applied by [basic::evalf\(\)](#).

Inheritance diagram for GiNaC::evalf\_map\_function:



Collaboration diagram for GiNaC::evalf\_map\_function:



## Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()
- virtual [ex operator\(\)](#) (const [ex](#) &e)=0

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 6.46.1 Detailed Description

Function object to be applied by [basic::evalf\(\)](#).

### 6.46.2 Member Function Documentation

#### 6.46.2.1 [operator\(\)](#)()

```
ex GiNaC::evalf_map_function::operator() (  
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::evalf\(\)](#).

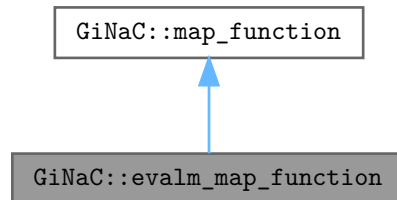
The documentation for this struct was generated from the following file:

- [basic.cpp](#)

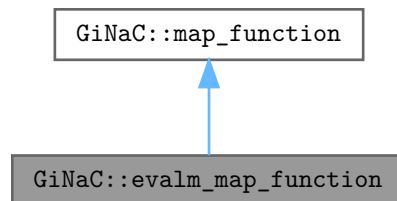
## 6.47 GiNaC::evalm\_map\_function Struct Reference

Function object to be applied by [basic::evalm\(\)](#).

Inheritance diagram for GiNaC::evalm\_map\_function:



Collaboration diagram for GiNaC::evalm\_map\_function:



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &*e*) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()
- virtual [ex operator\(\)](#) (const [ex](#) &*e*)=0

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 6.47.1 Detailed Description

Function object to be applied by [basic::evalm\(\)](#).

### 6.47.2 Member Function Documentation

#### 6.47.2.1 operator()

```
ex GiNaC::evalm_map_function::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::evalm\(\)](#).

The documentation for this struct was generated from the following file:

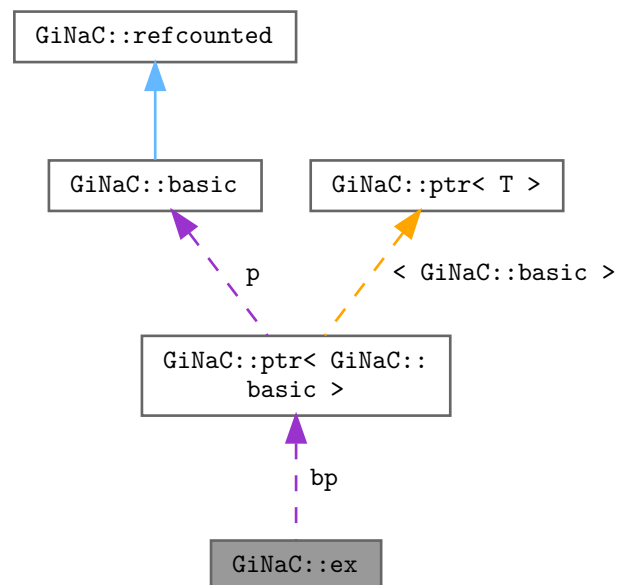
- [basic.cpp](#)

## 6.48 GiNaC::ex Class Reference

Lightweight wrapper for [GiNaC](#)'s symbolic objects.

```
#include <ex.h>
```

Collaboration diagram for GiNaC::ex:



## Public Member Functions

- `ex ()` noexcept
- `ex (const basic &other)`
- `ex (int i)`
- `ex (unsigned int i)`
- `ex (long i)`
- `ex (unsigned long i)`
- `ex (long long i)`
- `ex (unsigned long long i)`
- `ex (double const d)`
- `ex (const std::string &s, const ex &l)`  
*Construct ex from string and a list of symbols.*
- `void swap (ex &other)` noexcept  
*Efficiently swap the contents of two expressions.*
- `const_iterator begin ()` const noexcept
- `const_iterator end ()` const noexcept
- `const_preorder_iterator preorder_begin ()` const
- `const_preorder_iterator preorder_end ()` const noexcept
- `const_postorder_iterator postorder_begin ()` const
- `const_postorder_iterator postorder_end ()` const noexcept
- `ex eval ()` const
- `ex evalf ()` const
- `ex evalm ()` const
- `ex eval_ncmul (const exvector &v)` const
- `ex eval_integ ()` const
- `void print (const print_context &c, unsigned level=0)` const  
*Print expression to stream.*
- `void dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- `void dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- `bool info (unsigned inf)` const
- `size_t nops ()` const
- `ex op (size_t i)` const
- `ex operator[] (const ex &index)` const
- `ex operator[] (size_t i)` const
- `ex &let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- `ex &operator[] (const ex &index)`
- `ex &operator[] (size_t i)`
- `ex lhs ()` const  
*Left hand side of relational expression.*
- `ex rhs ()` const  
*Right hand side of relational expression.*
- `ex conjugate ()` const
- `ex real_part ()` const
- `ex imag_part ()` const
- `bool has (const ex &pattern, unsigned options=0)` const
- `bool find (const ex &pattern, exset &found)` const  
*Find all occurrences of a pattern.*
- `bool match (const ex &pattern)` const  
*Check whether expression matches a specified pattern.*

- `bool match` (const `ex` &pattern, `exmap` &repls) const
- `ex subs` (const `exmap` &m, unsigned `options`=0) const
- `ex subs` (const `lst` &ls, const `lst` &lr, unsigned `options`=0) const
 

*Substitute objects in an expression (syntactic substitution) and return the result as a new expression.*
- `ex subs` (const `ex` &e, unsigned `options`=0) const
 

*Substitute objects in an expression (syntactic substitution) and return the result as a new expression.*
- `ex map` (`map_function` &f) const
- `ex map` (`ex`(\*f)(const `ex` &e)) const
- `void accept` (`visitor` &v) const
- `void traverse_preorder` (`visitor` &v) const
 

*Traverse expression tree with given visitor, preorder traversal.*
- `void traverse_postorder` (`visitor` &v) const
 

*Traverse expression tree with given visitor, postorder traversal.*
- `void traverse` (`visitor` &v) const
- `bool is_polynomial` (const `ex` &vars) const
 

*Check whether expression is a polynomial.*
- `int degree` (const `ex` &s) const
- `int ldegree` (const `ex` &s) const
- `ex coeff` (const `ex` &s, int `n`=1) const
- `ex lcoeff` (const `ex` &s) const
- `ex tcoeff` (const `ex` &s) const
- `ex expand` (unsigned `options`=0) const
- `ex collect` (const `ex` &s, bool distributed=false) const
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const
 

*Compute partial derivative of an expression.*
- `ex series` (const `ex` &r, int `order`, unsigned `options`=0) const
 

*Compute the truncated series expansion of an expression.*
- `ex normal` () const
 

*Normalization of rational functions.*
- `ex to_rational` (`exmap` &repl) const
 

*Rationalization of non-rational functions.*
- `ex to_polynomial` (`exmap` &repl) const
- `ex numer` () const
 

*Get numerator of an expression.*
- `ex denom` () const
 

*Get denominator of an expression.*
- `ex numer_denom` () const
 

*Get numerator and denominator of an expression.*
- `ex unit` (const `ex` &x) const
 

*Compute unit part (= sign of leading coefficient) of a multivariate polynomial in  $Q[x]$ .*
- `ex content` (const `ex` &x) const
 

*Compute content part (= unit normal GCD of all coefficients) of a multivariate polynomial in  $Q[x]$ .*
- `numeric integer_content` () const
 

*Compute the integer content (= GCD of all numeric coefficients) of an expanded polynomial.*
- `ex primpart` (const `ex` &x) const
 

*Compute primitive part of a multivariate polynomial in  $Q[x]$ .*
- `ex primpart` (const `ex` &x, const `ex` &cont) const
 

*Compute primitive part of a multivariate polynomial in  $Q[x]$  when the content part is already known.*
- `void unitcontprim` (const `ex` &x, `ex` &u, `ex` &c, `ex` &p) const
 

*Compute unit part, content part, and primitive part of a multivariate polynomial in  $Q[x]$ .*
- `ex smod` (const `numeric` &xi) const
- `numeric max_coefficient` () const

- *Return maximum (absolute value) coefficient of a polynomial.*
- `exvector get_free_indices ()` const
- `ex simplify_indexed` (unsigned `options=0`) const
- *Simplify/canonicalize expression containing indexed objects.*
- `ex simplify_indexed` (const `scalar_products` &sp, unsigned `options=0`) const
- *Simplify/canonicalize expression containing indexed objects.*
- `int compare` (const `ex` &other) const
- `bool is_equal` (const `ex` &other) const
- `bool is_zero ()` const
- `bool is_zero_matrix ()` const
- *Check whether expression is zero or zero matrix.*
- `ex symmetrize ()` const
- *Symmetrize expression over its free indices.*
- `ex symmetrize` (const `lst` &l) const
- *Symmetrize expression over a list of objects (symbols, indices).*
- `ex antisymmetrize ()` const
- *Antisymmetrize expression over its free indices.*
- `ex antisymmetrize` (const `lst` &l) const
- *Antisymmetrize expression over a list of objects (symbols, indices).*
- `ex symmetrize_cyclic ()` const
- *Symmetrize expression by cyclic permutation over its free indices.*
- `ex symmetrize_cyclic` (const `lst` &l) const
- *Symmetrize expression by cyclic permutation over a list of objects (symbols, indices).*
- `unsigned return_type ()` const
- `return_type_t return_type_tinfo ()` const
- `unsigned gethash ()` const

## Private Member Functions

- `void makewriteable ()`
- *Make this ex writable (if more than one ex handle the same basic) by unlinking the object and creating an unshared copy of it.*
- `void share` (const `ex` &other) const
- *Share equal objects between expressions.*

## Static Private Member Functions

- `static ptr< basic > construct_from_basic` (const `basic` &other)
- *Helper function for the ex-from-basic constructor.*
- `static basic & construct_from_int` (int i)
- `static basic & construct_from_uint` (unsigned int i)
- `static basic & construct_from_long` (long i)
- `static basic & construct_from_ulong` (unsigned long i)
- `static basic & construct_from_longlong` (long long i)
- `static basic & construct_from_ulonglong` (unsigned long long i)
- `static basic & construct_from_double` (double d)
- `static ptr< basic > construct_from_string_and_lst` (const std::string &s, const `ex` &l)

## Private Attributes

- `ptr< basic > bp`  
*pointer to basic object managed by this*

## Friends

- class `archive_node`
- bool `are_ex_trivially_equal` (const `ex` &, const `ex` &)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- template<class T >  
const T & `ex_to` (const `ex` &)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- template<class T >  
bool `is_a` (const `ex` &)  
*Check if ex is a handle to a T, including base classes.*
- template<class T >  
bool `is_exactly_a` (const `ex` &)  
*Check if ex is a handle to a T, not including base classes.*

### 6.48.1 Detailed Description

Lightweight wrapper for `GiNaC`'s symbolic objects.

It holds a pointer to the other object in order to do garbage collection by the method of reference counting. I.e., it is a smart pointer. Also, the constructor `ex::ex(const basic & other)` calls the methods that do automatic evaluation. E.g., `x-x` turns automatically into 0.

### 6.48.2 Constructor & Destructor Documentation

#### 6.48.2.1 `ex()` [1/10]

```
GiNaC::ex::ex ( ) [inline], [noexcept]
```

References `bp`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.

#### 6.48.2.2 `ex()` [2/10]

```
GiNaC::ex::ex (
    const basic & other ) [inline]
```

References `bp`, `GiNaC::status_flags::dynallocated`, and `GINAC_ASSERT`.



**6.48.2.3 ex()** [3/10]

```
GiNaC::ex::ex (
    int i ) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.4 ex()** [4/10]

```
GiNaC::ex::ex (
    unsigned int i ) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.5 ex()** [5/10]

```
GiNaC::ex::ex (
    long i ) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.6 ex()** [6/10]

```
GiNaC::ex::ex (
    unsigned long i ) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.7 ex()** [7/10]

```
GiNaC::ex::ex (
    long long i ) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.8 ex()** [8/10]

```
GiNaC::ex::ex (
    unsigned long long i ) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

### 6.48.2.9 `ex()` [9/10]

```
GiNaC::ex::ex (
    double const d ) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

### 6.48.2.10 `ex()` [10/10]

```
GiNaC::ex::ex (
    const std::string & s,
    const ex & l ) [inline]
```

Construct `ex` from string and a list of symbols.

The input grammar is similar to the [GiNaC](#) output format. All symbols and indices to be used in the expression must be specified in a list in the second argument. Undefined symbols and other parser errors will throw an exception.

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

## 6.48.3 Member Function Documentation

### 6.48.3.1 `swap()`

```
void GiNaC::ex::swap (
    ex & other ) [inline], [noexcept]
```

Efficiently swap the contents of two expressions.

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

Referenced by [GiNaC::ncmul::derivative\(\)](#), [GiNaC::ex\\_swap::operator\(\)\(\)](#), [GiNaC::swap\(\)](#), [GiNaC::expair::swap\(\)](#), and [std::swap\(\)](#).

### 6.48.3.2 `begin()`

```
const\_iterator GiNaC::ex::begin ( ) const [inline], [noexcept]
```

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::antisymmetrize\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::symmetrize\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

### 6.48.3.3 end()

```
const_iterator GiNaC::ex::end ( ) const [inline], [noexcept]
```

References [nops\(\)](#).

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::log\\_expand\(\)](#), and [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#).

### 6.48.3.4 preorder\_begin()

```
const_preorder_iterator GiNaC::ex::preorder_begin ( ) const [inline]
```

References [nops\(\)](#).

### 6.48.3.5 preorder\_end()

```
const_preorder_iterator GiNaC::ex::preorder_end ( ) const [inline], [noexcept]
```

### 6.48.3.6 postorder\_begin()

```
const_postorder_iterator GiNaC::ex::postorder_begin ( ) const [inline]
```

References [nops\(\)](#).

### 6.48.3.7 postorder\_end()

```
const_postorder_iterator GiNaC::ex::postorder_end ( ) const [inline], [noexcept]
```

### 6.48.3.8 eval()

```
ex GiNaC::ex::eval ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::eval\(\)](#).

### 6.48.3.9 evalf()

```
ex GiNaC::ex::evalf ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::adaptivesimpson\(\)](#), [GiNaC::EllipticE\\_evalf\(\)](#), [GiNaC::EllipticK\\_evalf\(\)](#), [GiNaC::constant::evalf\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::mul::evalf\(\)](#), [GiNaC::power::evalf\(\)](#), [GiNaC::evalf\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::is\\_numeric\(\)](#), [GiNaC::ELi\\_kernel::is\\_numeric\(\)](#), [GiNaC::Ebar\\_kernel::is\\_numeric\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::is\\_numeric\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::is\\_numeric\(\)](#), [GiNaC::Eisenstein\\_kernel::is\\_numeric\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::is\\_numeric\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::user\\_defined\\_kernel::is\\_numeric\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::integration\\_kernel::series\\_coeff\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::ELi\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Ebar\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::subsvalue\(\)](#).

### 6.48.3.10 evalm()

```
ex GiNaC::ex::evalm ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::power::evalm\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::evalm\(\)](#), and [is\\_zero\\_matrix\(\)](#).

### 6.48.3.11 eval\_ncmul()

```
ex GiNaC::ex::eval_ncmul (
    const exvector & v ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::integral::eval\\_ncmul\(\)](#), and [GiNaC::relational::eval\\_ncmul\(\)](#).

### 6.48.3.12 eval\_integ()

```
ex GiNaC::ex::eval_integ ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), and [GiNaC::eval\\_integ\(\)](#).

### 6.48.3.13 print()

```
void GiNaC::ex::print (
    const print\_context & c,
    unsigned level = 0 ) const
```

Print expression to stream.

The formatting of the output is determined by the kind of [print\\_context](#) object that is passed. Possible formattings include ginsh-parsable output (the default), tree-like output for debugging, and C++ source.

See also

[print\\_context](#)

References [bp](#), and [c](#).

Referenced by [GiNaC::abs\\_print\\_csrc\\_float\(\)](#), [GiNaC::abs\\_print\\_latex\(\)](#), [GiNaC::conjugate\\_print\\_latex\(\)](#), [GiNaC::integral::do\\_print\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::do\\_print\(\)](#), [GiNaC::ELi\\_kernel::do\\_print\(\)](#), [GiNaC::Ebar\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::do\\_print\(\)](#), [GiNaC::modular\\_form\\_kernel::do\\_print\(\)](#), [GiNaC::user\\_defined\\_kernel::do\\_print\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::relational::do\\_print\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::integral::do\\_print\\_latex\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::power::do\\_print\\_python\\_repr\(\)](#), [GiNaC::pseries::do\\_print\\_python\\_repr\(\)](#), [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::factorial\\_print\\_dflt\\_latex\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::imag\\_part\\_print\\_latex\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::expair::print\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::idx::print\\_index\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::power::print\\_power\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::expairseq::printseq\(\)](#), [GiNaC::real\\_part\\_print\\_latex\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), and [GiNaC::zeta1\\_print\\_latex\(\)](#).

### 6.48.3.14 dbgprint()

```
void GiNaC::ex::dbgprint ( ) const
```

Little wrapper around print to be called within a debugger.

References [bp](#).

### 6.48.3.15 dbgprinttree()

```
void GiNaC::ex::dbgprinttree ( ) const
```

Little wrapper around printtree to be called within a debugger.

References [bp](#).

### 6.48.3.16 info()

```
bool GiNaC::ex::info (
    unsigned int ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::abs\\_info\(\)](#), [GiNaC::abs\\_power\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asin\\_info\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan2\\_info\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_info\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [GiNaC::mul::can\\_make\\_flat\(\)](#), [GiNaC::power::conjugate\(\)](#), [GiNaC::pseries::conjugate\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [content\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::csgn\\_series\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [GiNaC::eta\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::exp\\_info\(\)](#), [GiNaC::exp\\_power\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::func\\_arg\\_info\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::heur\\_gcd\(\)](#), [GiNaC::idx::idx\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::pseries::imag\\_part\(\)](#), [GiNaC::add::info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::power::info\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::is\\_numeric\(\)](#), [GiNaC::ELi\\_kernel::is\\_numeric\(\)](#), [GiNaC::Ebar\\_kernel::is\\_numeric\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::is\\_numeric\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::is\\_numeric\(\)](#), [GiNaC::Eisenstein\\_kernel::is\\_numeric\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::is\\_numeric\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::user\\_defined\\_kernel::is\\_numeric\(\)](#), [GiNaC::power::is\\_polynomial\(\)](#), [GiNaC::iterated\\_integral2\\_eval\(\)](#), [GiNaC::iterated\\_integral3\\_eval\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::lgamma\\_eval\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::log\\_imag\\_part\(\)](#), [GiNaC::log\\_info\(\)](#), [GiNaC::log\\_real\\_part\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::Order\\_imag\\_part\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::pseries::real\\_part\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::step\\_series\(\)](#), [subs\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [GiNaC::tgamma\\_series\(\)](#), [GiNaC::expairseq::to\\_polynomial\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::expairseq::to\\_rational\(\)](#), [GiNaC::power::to\\_rational\(\)](#), [GiNaC::matrix::trace\(\)](#), [GiNaC::trig\\_info\(\)](#), [GiNaC::tryfactsubs\(\)](#), [unitcontprim\(\)](#), [GiNaC::zeta2\\_deriv\(\)](#), and [GiNaC::zeta2\\_eval\(\)](#).

### 6.48.3.17 nops()

```
size_t GiNaC::ex::nops ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::ncmul::append\\_factors\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::collect\\_symbols\(\)](#), [GiNaC::color\\_trace\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::divide\(\)](#), [end\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [find\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::gcd\\_pf\\_mul\(\)](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::hasindex\(\)](#), [GiNaC::haswild\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [GiNaC::nops\(\)](#), [postorder\\_begin\(\)](#), [preorder\\_begin\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::step\\_eval\(\)](#), [GiNaC::symminfo::symminfo\(\)](#), [traverse\\_postorder\(\)](#), [traverse\\_preorder\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), and [GiNaC::zeta2\\_evalf\(\)](#).

**6.48.3.18 op()**

```
ex GiNaC::ex::op (
    size_t i ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::ncmul::append\\_factors\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::collect\\_symbols\(\)](#), [GiNaC::color\\_trace\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::decomp\\_rational\(\)](#), [denom\(\)](#), [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [find\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::gcd\\_pf\\_mul\(\)](#), [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::gcd\\_pf\\_pow\\_pow\(\)](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::hasindex\(\)](#), [GiNaC::haswild\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::lorentz\\_eps\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::power::normal\(\)](#), [numer\(\)](#), [GiNaC::op\(\)](#), [GiNaC::op0\\_is\\_equal::operator\(\)\(\)](#), [GiNaC::ex\\_base\\_is\\_less::operator\(\)\(\)](#), [GiNaC::const\\_iterator::operator\\*\(\)](#), [GiNaC::const\\_iterator::operator\[\]\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::spmapkey::spmapkey\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::step\\_eval\(\)](#), [subs\(\)](#), [GiNaC::symminfo::symminfo\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [traverse\\_postorder\(\)](#), [traverse\\_preorder\(\)](#), [GiNaC::tryfactsubs\(\)](#), and [GiNaC::zeta2\\_deriv\(\)](#).

**6.48.3.19 operator[]() [1/4]**

```
ex GiNaC::ex::operator[] (
    const ex & index ) const [inline]
```

References [bp](#).

**6.48.3.20 operator[]() [2/4]**

```
ex GiNaC::ex::operator[] (
    size_t i ) const [inline]
```

References [bp](#).

**6.48.3.21 let\_op()**

```
ex & GiNaC::ex::let_op (
    size_t i )
```

Return modifiable operand/member at position i.

References [bp](#), and [makewritable\(\)](#).

**6.48.3.22 operator[]() [3/4]**

```
ex & GiNaC::ex::operator[] (
    const ex & index )
```

References [bp](#), and [makewriteable\(\)](#).

**6.48.3.23 operator[]() [4/4]**

```
ex & GiNaC::ex::operator[] (
    size_t i )
```

References [bp](#), and [makewriteable\(\)](#).

**6.48.3.24 lhs()**

```
ex GiNaC::ex::lhs ( ) const
```

Left hand side of relational expression.

References [bp](#).

Referenced by [GiNaC::fsolve\(\)](#), [GiNaC::lhs\(\)](#), [GiNaC::pseries::pseries\(\)](#), and [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#).

**6.48.3.25 rhs()**

```
ex GiNaC::ex::rhs ( ) const
```

Right hand side of relational expression.

References [bp](#).

Referenced by [GiNaC::fsolve\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::rhs\(\)](#), and [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#).

**6.48.3.26 conjugate()**

```
ex GiNaC::ex::conjugate ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::abs\\_expl\\_derivative\(\)](#), [GiNaC::abs\\_power\(\)](#), [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::expair::conjugate\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::expairseq::conjugate\(\)](#), [GiNaC::integral::conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::power::conjugate\(\)](#), [GiNaC::pseries::conjugate\(\)](#), [GiNaC::conjugate\(\)](#), [GiNaC::conjugate\\_eval\(\)](#), [GiNaC::conjugate\\_evalf\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::cos\\_conjugate\(\)](#), [GiNaC::cosh\\_conjugate\(\)](#), [GiNaC::exp\\_conjugate\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::sin\\_conjugate\(\)](#), [GiNaC::sinh\\_conjugate\(\)](#), [GiNaC::tan\\_conjugate\(\)](#), [GiNaC::tanh\\_conjugate\(\)](#), and [GiNaC::tgamma\\_conjugate\(\)](#).



### 6.48.3.27 real\_part()

```
ex GiNaC::ex::real_part ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::conjugate\\_real\\_part\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::pseries::imag\\_part\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::pseries::real\\_part\(\)](#), [GiNaC::real\\_part\(\)](#), and [GiNaC::real\\_part\\_eval\(\)](#).

### 6.48.3.28 imag\_part()

```
ex GiNaC::ex::imag_part ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::conjugate\\_imag\\_part\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::imag\\_part\(\)](#), [GiNaC::imag\\_part\\_eval\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::log\\_conjugate\(\)](#), and [GiNaC::power::real\\_part\(\)](#).

### 6.48.3.29 has()

```
bool GiNaC::ex::has (
    const ex & pattern,
    unsigned options = 0 ) const [inline]
```

References [bp](#), and [options](#).

Referenced by [GiNaC::power::degree\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::basic::has\(\)](#), [GiNaC::has\(\)](#), [GiNaC::power::is\\_polynomial\(\)](#), and [GiNaC::power::ldegree\(\)](#).

### 6.48.3.30 find()

```
bool GiNaC::ex::find (
    const ex & pattern,
    exset & found ) const
```

Find all occurrences of a pattern.

The found matches are appended to the "found" list. If the expression itself matches the pattern, the children are not further examined. This function returns true when any matches were found.

References [find\(\)](#), [match\(\)](#), [nops\(\)](#), and [op\(\)](#).

Referenced by [GiNaC::divide\\_in\\_z\(\)](#), [find\(\)](#), [GiNaC::find\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.48.3.31 match()** [1/2]

```
bool GiNaC::ex::match (
    const ex & pattern ) const
```

Check whether expression matches a specified pattern.

References [bp](#).

Referenced by [find\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::match\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

**6.48.3.32 match()** [2/2]

```
bool GiNaC::ex::match (
    const ex & pattern,
    exmap & repls ) const [inline]
```

References [bp](#).

**6.48.3.33 subs()** [1/3]

```
ex GiNaC::ex::subs (
    const exmap & m,
    unsigned options = 0 ) const [inline]
```

References [bp](#), [m](#), and [options](#).

Referenced by [GiNaC::adaptivesimpson\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::collect\\_common\\_factors\(\)](#), [GiNaC::integral::conjugate\(\)](#), [GiNaC::csgn\\_series\(\)](#), [denom\(\)](#), [GiNaC::integral::derivative\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::eta\\_series\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::expand\\_dummy\\_sum\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::user\\_defined\\_kernel::is\\_numeric\(\)](#), [GiNaC::user\\_defined\\_kernel::Laurent\\_series\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::power::normal\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::tensor::replace\\_contr\\_index\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::step\\_series\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::basic::subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::pseries::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [GiNaC::container< C >::subchildren\(\)](#), [GiNaC::expairseq::subchildren\(\)](#), [GiNaC::subsvalue\(\)](#), [GiNaC::symm\(\)](#), [GiNaC::symmetrize\\_cyclic\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_series\(\)](#), and [GiNaC::tgamma\\_series\(\)](#).

**6.48.3.34 subs()** [2/3]

```
ex GiNaC::ex::subs (
    const lst & ls,
    const lst & lr,
    unsigned options = 0 ) const
```

Substitute objects in an expression (syntactic substitution) and return the result as a new expression.

References [GiNaC::container< C >::begin\(\)](#), [bp](#), [GiNaC::container< C >::end\(\)](#), [GINAC\\_ASSERT](#), [lr](#), [m](#), [GiNaC::container< C >::nops\(\)](#), [options](#), [GiNaC::subs\\_options::pattern\\_is\\_not\\_product](#), and [GiNaC::subs\\_options::pattern\\_is\\_product](#).

**6.48.3.35 subs()** [3/3]

```
ex GiNaC::ex::subs (
    const ex & e,
    unsigned options = 0 ) const
```

Substitute objects in an expression (syntactic substitution) and return the result as a new expression.

There are two valid types of replacement arguments: 1) a relational like `object==ex` and 2) a list of relationals `lst{object1==ex1,object2==ex2,...}`.

References [bp](#), [GINAC\\_ASSERT](#), [info\(\)](#), [GiNaC::info\\_flags::list](#), [m](#), [op\(\)](#), [options](#), [GiNaC::subs\\_options::pattern\\_is\\_not\\_product](#), [GiNaC::subs\\_options::pattern\\_is\\_product](#), [r](#), and [GiNaC::info\\_flags::relation\\_equal](#).

**6.48.3.36 map()** [1/2]

```
ex GiNaC::ex::map (
    map_function & f ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::color\\_trace\(\)](#), and [GiNaC::expand\\_dummy\\_sum\(\)](#).

**6.48.3.37 map()** [2/2]

```
ex GiNaC::ex::map (
    ex(*) (const ex &e) f ) const
```

#### 6.48.3.38 `accept()`

```
void GiNaC::ex::accept (
    visitor & v ) const [inline]
```

References [bp](#).

Referenced by [traverse\\_postorder\(\)](#), and [traverse\\_preorder\(\)](#).

#### 6.48.3.39 `traverse_preorder()`

```
void GiNaC::ex::traverse_preorder (
    visitor & v ) const
```

Traverse expression tree with given visitor, preorder traversal.

References [accept\(\)](#), [n](#), [nops\(\)](#), [op\(\)](#), and [traverse\\_preorder\(\)](#).

Referenced by [traverse\(\)](#), and [traverse\\_preorder\(\)](#).

#### 6.48.3.40 `traverse_postorder()`

```
void GiNaC::ex::traverse_postorder (
    visitor & v ) const
```

Traverse expression tree with given visitor, postorder traversal.

References [accept\(\)](#), [n](#), [nops\(\)](#), [op\(\)](#), and [traverse\\_postorder\(\)](#).

Referenced by [traverse\\_postorder\(\)](#).

#### 6.48.3.41 `traverse()`

```
void GiNaC::ex::traverse (
    visitor & v ) const [inline]
```

References [traverse\\_preorder\(\)](#).

**6.48.3.42 is\_polynomial()**

```
bool GiNaC::ex::is_polynomial (
    const ex & vars ) const
```

Check whether expression is a polynomial.

References [bp](#).

Referenced by [GiNaC::is\\_polynomial\(\)](#), and [GiNaC::power::is\\_polynomial\(\)](#).

**6.48.3.43 degree()**

```
int GiNaC::ex::degree (
    const ex & s ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::basic::collect\(\)](#), [GiNaC::mul::degree\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::degree\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [lcoeff\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::sprem\(\)](#), and [GiNaC::sr\\_gcd\(\)](#).

**6.48.3.44 ldegree()**

```
int GiNaC::ex::ldegree (
    const ex & s ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::mul::ldegree\(\)](#), [GiNaC::power::ldegree\(\)](#), [GiNaC::ldegree\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), and [tcoeff\(\)](#).

**6.48.3.45 coeff()**

```
ex GiNaC::ex::coeff (
    const ex & s,
    int n = 1 ) const [inline]
```

References [bp](#), and [n](#).

Referenced by [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::generalised\\_Bernoulli\\_num\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [lcoeff\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::integration\\_kernel::series\\_coeff\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqfree\\_parfrac\(\)](#), [GiNaC::expairseq::subschildren\(\)](#), and [tcoeff\(\)](#).

**6.48.3.46 lcoeff()**

```
ex GiNaC::ex::lcoeff (
    const ex & s ) const [inline]
```

References [coeff\(\)](#), and [degree\(\)](#).

Referenced by [content\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), and [unit\(\)](#).

**6.48.3.47 tcoeff()**

```
ex GiNaC::ex::tcoeff (
    const ex & s ) const [inline]
```

References [coeff\(\)](#), and [ldegree\(\)](#).

**6.48.3.48 expand()**

```
ex GiNaC::ex::expand (
    unsigned options = 0 ) const
```

References [bp](#), [GiNaC::status\\_flags::expanded](#), and [options](#).

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::color\\_trace\(\)](#), [content\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::expand\\_dummy\\_sum\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expand\\_map\\_function::op\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqfree\\_parfrac\(\)](#), [GiNaC::matrix::trace\(\)](#), [unit\(\)](#), and [unitcontprim\(\)](#).

**6.48.3.49 collect()**

```
ex GiNaC::ex::collect (
    const ex & s,
    bool distributed = false ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::matrix::charpoly\(\)](#), and [GiNaC::collect\(\)](#).

**6.48.3.50 diff()**

```
ex GiNaC::ex::diff (
    const symbol & s,
    unsigned nth = 1 ) const
```

Compute partial derivative of an expression.

## Parameters

<i>s</i>	symbol by which the expression is derived
<i>nth</i>	order of derivative (default 1)

## Returns

partial derivative as a new expression

References [bp](#).

Referenced by [GiNaC::abs\\_expl\\_derivative\(\)](#), [GiNaC::conjugate\\_expl\\_derivative\(\)](#), [GiNaC::integral::derivative\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::diff\(\)](#), [GiNaC::basic::diff\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::imag\\_part\\_expl\\_derivative\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::Order\\_expl\\_derivative\(\)](#), [GiNaC::real\\_part\\_expl\\_derivative\(\)](#), [GiNaC::basic::series\(\)](#), and [GiNaC::sqrfree\\_yun\(\)](#).

**6.48.3.51 series()**

```
ex GiNaC::ex::series (
    const ex & r,
    int order,
    unsigned options = 0 ) const
```

Compute the truncated series expansion of an expression.

This function returns an expression containing an object of class pseries to represent the series. If the series does not terminate within the given truncation order, the last term of the series will be an order term.

## Parameters

<i>r</i>	expansion relation, lhs holds variable and rhs holds point
<i>order</i>	truncation order of series calculations
<i>options</i>	of class <a href="#">series_options</a>

## Returns

an expression holding a pseries object

References [GiNaC::\\_ex0](#), [bp](#), [options](#), [order](#), and [r](#).

Referenced by [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::generalised\\_Bernoulli\\_nu](#), [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_kernel::Laurent\\_ser](#), [GiNaC::Eisenstein\\_h\\_kernel::Laurent\\_series\(\)](#), [GiNaC::user\\_defined\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::ser](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), and [GiNaC::tgamma\\_series\(\)](#).

### 6.48.3.52 `normal()`

```
ex GiNaC::ex::normal ( ) const
```

Normalization of rational functions.

This function converts an expression to its normal form "numerator/denominator", where numerator and denominator are (relatively prime) polynomials. Any subexpressions which are not rational functions (like non-rational numbers, non-integer powers or functions like `sin()`, `cos()` etc.) are replaced by temporary symbols which are re-substituted by the (normalized) subexpressions before `normal()` returns (this way, any expression can be treated as a rational function). `normal()` is applied recursively to arguments of functions etc.

#### Returns

normalized expression

References `bp`, `GINAC_ASSERT`, `GiNaC::subs_options::no_pattern`, `GiNaC::container< C >::nops()`, `op()`, `GiNaC::container< C >::op()`, and `subs()`.

Referenced by `GiNaC::matrix::determinant()`, `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::matrix::gauss_elimination()`, `GiNaC::normal()`, `GiNaC::power::normal()`, `GiNaC::pseries::normal()`, and `GiNaC::matrix::trace()`.

### 6.48.3.53 `to_rational()`

```
ex GiNaC::ex::to_rational (
    exmap & repl ) const
```

Rationalization of non-rational functions.

This function converts a general expression to a rational function by replacing all non-rational subexpressions (like non-rational numbers, non-integer powers or functions like `sin()`, `cos()` etc.) to temporary symbols. This makes it possible to use functions like `gcd()` and `divide()` on non-rational functions by applying `to_rational()` on the arguments, calling the desired function and re-substituting the temporary symbols in the result. To make the last step possible, all temporary symbols and their associated expressions are collected in the map specified by the `repl` parameter, ready to be passed as an argument to `ex::subs()`.

#### Parameters

<i>repl</i>	collects all temporary symbols and their replacements
-------------	---

#### Returns

rationalized expression

References `bp`.

Referenced by `GiNaC::matrix::fraction_free_elimination()`, `GiNaC::to_rational()`, `GiNaC::expairseq::to_rational()`, and `GiNaC::power::to_rational()`.



### 6.48.3.54 to\_polynomial()

```
ex GiNaC::ex::to_polynomial (
    exmap & repl ) const
```

References [bp](#).

Referenced by [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::to\\_polynomial\(\)](#), [GiNaC::expairseq::to\\_polynomial\(\)](#), and [GiNaC::power::to\\_polynomial\(\)](#).

### 6.48.3.55 numer()

```
ex GiNaC::ex::numer ( ) const
```

Get numerator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then the numerator is returned.

See also

[ex::normal](#)

Returns

numerator

References [bp](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [op\(\)](#), [GiNaC::container< C >::op\(\)](#), and [subs\(\)](#).

Referenced by [GiNaC::numer\(\)](#).

### 6.48.3.56 denom()

```
ex GiNaC::ex::denom ( ) const
```

Get denominator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then the denominator is returned.

See also

[ex::normal](#)

Returns

denominator

References [bp](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [op\(\)](#), [GiNaC::container< C >::op\(\)](#), and [subs\(\)](#).

Referenced by [GiNaC::denom\(\)](#).

### 6.48.3.57 numer\_denom()

```
ex GiNaC::ex::numer_denom ( ) const
```

Get numerator and denominator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then a list [numerator, denominator] is returned.

See also

[ex::normal](#)

Returns

a list [numerator, denominator]

References [bp](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), and [subs\(\)](#).

Referenced by [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), and [GiNaC::numer\\_denom\(\)](#).

### 6.48.3.58 unit()

```
ex GiNaC::ex::unit (
    const ex & x ) const
```

Compute unit part (= sign of leading coefficient) of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of unit part, content part, and primitive part is the polynomial itself.

Parameters

$x$	main variable
-----	---------------

Returns

unit part

See also

[ex::content](#), [ex::primpart](#), [ex::unitcontprim](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [c](#), [expand\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [lcoeff\(\)](#), [GiNaC::info\\_flags::negative](#), and [x](#).

Referenced by [content\(\)](#), and [GiNaC::frac\\_cancel\(\)](#).

**6.48.3.59 content()**

```
ex GiNaC::ex::content (
    const ex & x ) const
```

Compute content part (= unit normal GCD of all coefficients) of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of unit part, content part, and primitive part is the polynomial itself.

**Parameters**

<code>x</code>	main variable
----------------	---------------

**Returns**

content part

**See also**

[ex::unit](#), [ex::primpart](#), [ex::unitcontprim](#)

References [GiNaC::\\_ex0](#), [c](#), [cont](#), [expand\(\)](#), [GiNaC::gcd\(\)](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [integer\\_content\(\)](#), [is\\_zero\(\)](#), [lcoeff\(\)](#), [GiNaC::info\\_flags::negative](#), [r](#), [unit\(\)](#), and [x](#).

Referenced by [GiNaC::sr\\_gcd\(\)](#), and [unitcontprim\(\)](#).

**6.48.3.60 integer\_content()**

```
numeric GiNaC::ex::integer_content ( ) const
```

Compute the integer content (= GCD of all numeric coefficients) of an expanded polynomial.

For a polynomial with rational coefficients, this returns  $g/l$  where  $g$  is the GCD of the coefficients' numerators and  $l$  is the LCM of the coefficients' denominators.

**Returns**

integer content

References [bp](#).

Referenced by [content\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::gcd\(\)](#), and [GiNaC::heur\\_gcd\\_z\(\)](#).

**6.48.3.61 primpart() [1/2]**

```
ex GiNaC::ex::primpart (
    const ex & x ) const
```

Compute primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The result will be a unit-normal polynomial with a content part of 1. The product of unit part, content part, and primitive part is the polynomial itself.

## Parameters

<code>x</code>	main variable
----------------	---------------

## Returns

primitive part

## See also

[ex::unit](#), [ex::content](#), [ex::unitcontprim](#)

References [c](#), [unitcontprim\(\)](#), and [x](#).

Referenced by [GiNaC::sr\\_gcd\(\)](#).

**6.48.3.62 primpart() [2/2]**

```
ex GiNaC::ex::primpart (
    const ex & x,
    const ex & c ) const
```

Compute primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$  when the content part is already known.

This function is faster in computing the primitive part than the previous function.

## Parameters

<code>x</code>	main variable
<code>c</code>	previously computed content part

## Returns

primitive part

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [is\\_zero\(\)](#), [GiNaC::quo\(\)](#), [unit](#), and [x](#).

**6.48.3.63 unitcontprim()**

```
void GiNaC::ex::unitcontprim (
    const ex & x,
    ex & u,
    ex & c,
    ex & p ) const
```

Compute unit part, content part, and primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of the three parts is the polynomial itself.

## Parameters

$x$	main variable
$u$	unit part (returned)
$c$	content part (returned)
$p$	primitive part (returned)

## See also

[ex::unit](#), [ex::content](#), [ex::primpart](#)

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::abs\(\)](#), [c](#), [content\(\)](#), [expand\(\)](#), [info\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::quo\(\)](#), [unit](#), and [x](#).

Referenced by [GiNaC::gcd\(\)](#), and [primpart\(\)](#).

**6.48.3.64 smod()**

```
ex GiNaC::ex::smod (
    const numeric & xi ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::interpolate\(\)](#).

**6.48.3.65 max\_coefficient()**

```
numeric GiNaC::ex::max_coefficient ( ) const
```

Return maximum (absolute value) coefficient of a polynomial.

This function is used internally by [heur\\_gcd\(\)](#).

## Returns

maximum coefficient

## See also

[heur\\_gcd](#)

References [bp](#).

Referenced by [GiNaC::heur\\_gcd\\_z\(\)](#).

**6.48.3.66 `get_free_indices()`**

```
exvector GiNaC::ex::get_free_indices ( ) const [inline]
```

References [bp](#).

Referenced by [antisymmetrize\(\)](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::integral::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [GiNaC::ncmul::get\\_free\\_indices\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [symmetrize\(\)](#), and [symmetrize\\_cyclic\(\)](#).

**6.48.3.67 `simplify_indexed()` [1/2]**

```
ex GiNaC::ex::simplify_indexed (
    unsigned options = 0 ) const
```

Simplify/canonicalize expression containing indexed objects.

This performs contraction of dummy indices where possible and checks whether the free indices in sums are consistent.

Parameters

<i>options</i>	Simplification options (currently unused)
----------------	---

Returns

simplified expression

References [GiNaC::simplify\\_indexed\(\)](#).

Referenced by [GiNaC::tensepsilon::contract\\_with\(\)](#), and [GiNaC::simplify\\_indexed\(\)](#).

**6.48.3.68 `simplify_indexed()` [2/2]**

```
ex GiNaC::ex::simplify_indexed (
    const scalar_products & sp,
    unsigned options = 0 ) const
```

Simplify/canonicalize expression containing indexed objects.

This performs contraction of dummy indices where possible, checks whether the free indices in sums are consistent, and automatically replaces scalar products by known values if desired.

Parameters

<i>sp</i>	Scalar products to be replaced automatically
<i>options</i>	Simplification options (currently unused)

## Returns

simplified expression

References [GiNaC::simplify\\_indexed\(\)](#).

**6.48.3.69 compare()**

```
int GiNaC::ex::compare (
    const ex & other ) const [inline]
```

References [bp](#), and [share\(\)](#).

Referenced by [GiNaC::expair::compare\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::expair::is\\_less\(\)](#), [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)](#), [GiNaC::ex\\_is\\_less::operator\(\)](#), [GiNaC::expair\\_rest\\_is\\_less::operator\(\)](#), [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm::operator\(\)](#), [GiNaC::symminfo\\_is\\_less\\_by\\_orig::operator\(\)](#), [GiNaC::terminfo\\_is\\_less::operator\(\)](#), [GiNaC::spmapkey::operator<\(\)](#), and [GiNaC::spmapkey::spmapkey\(\)](#).

**6.48.3.70 is\_equal()**

```
bool GiNaC::ex::is_equal (
    const ex & other ) const [inline]
```

References [bp](#), and [share\(\)](#).

Referenced by [GiNaC::abs\\_power\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::conjugateevector\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::pseries::degree\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::gcd\\_pf\\_pow\\_pow\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::expair::is\\_canonical\\_numeric\(\)](#), [GiNaC::pseries::is\\_compatible\\_to\(\)](#), [GiNaC::idx::is\\_dummy\\_pair\\_same\\_type\(\)](#), [GiNaC::expair::is\\_equal\(\)](#), [GiNaC::expairseq::is\\_equal\\_same\\_type\(\)](#), [is\\_zero\(\)](#), [GiNaC::power::ldegree\(\)](#), [GiNaC::pseries::ldegree\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::lorentz\\_eps\(\)](#), [GiNaC::expairseq::map\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::idx::match\\_same\\_type\(\)](#), [GiNaC::minimal\\_dim\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::expairseq::op\(\)](#), [GiNaC::ex\\_is\\_equal::operator\(\)](#), [GiNaC::op0\\_is\\_equal::operator\(\)](#), [GiNaC::idx\\_is\\_equal\\_ignore\\_dim::operator\(\)](#), [GiNaC::spmapkey::operator==\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::expairseq::printseq\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), and [GiNaC::sqfree\\_yun\(\)](#).

**6.48.3.71 is\_zero()**

```
bool GiNaC::ex::is_zero ( ) const [inline]
```

References [GiNaC::\\_ex0](#), and [is\\_equal\(\)](#).

Referenced by [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::add::coeff\(\)](#), [content\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::add::degree\(\)](#), [GiNaC::fderivative::derivative\(\)](#), [GiNaC::function::derivative\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::basic::diff\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::add::info\(\)](#), [GiNaC::interpolate\(\)](#), [GiNaC::is\\_zero\(\)](#), [is\\_zero\\_matrix\(\)](#), [GiNaC::add::lddegree\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li3\\_eval\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::Order\\_eval\(\)](#), [GiNaC::prem\(\)](#), [primpart\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\\_yun\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [unitcontprim\(\)](#), and [GiNaC::indexed::validate\(\)](#).

**6.48.3.72 is\_zero\_matrix()**

```
bool GiNaC::ex::is_zero_matrix ( ) const
```

Check whether expression is zero or zero matrix.

References [evalm\(\)](#), and [is\\_zero\(\)](#).

**6.48.3.73 symmetrize() [1/2]**

```
ex GiNaC::ex::symmetrize ( ) const
```

Symmetrize expression over its free indices.

References [get\\_free\\_indices\(\)](#), and [GiNaC::symmetrize\(\)](#).

Referenced by [GiNaC::symmetrize\(\)](#).

**6.48.3.74 symmetrize() [2/2]**

```
ex GiNaC::ex::symmetrize (
    const lst & l ) const
```

Symmetrize expression over a list of objects (symbols, indices).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), and [GiNaC::symm\(\)](#).



**6.48.3.75 antisymmetrize()** [1/2]

```
ex GiNaC::ex::antisymmetrize ( ) const
```

Antisymmetrize expression over its free indices.

References [GiNaC::antisymmetrize\(\)](#), and [get\\_free\\_indices\(\)](#).

Referenced by [GiNaC::antisymmetrize\(\)](#).

**6.48.3.76 antisymmetrize()** [2/2]

```
ex GiNaC::ex::antisymmetrize (
    const lst & l ) const
```

Antisymmetrize expression over a list of objects (symbols, indices).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), and [GiNaC::symm\(\)](#).

**6.48.3.77 symmetrize\_cyclic()** [1/2]

```
ex GiNaC::ex::symmetrize_cyclic ( ) const
```

Symmetrize expression by cyclic permutation over its free indices.

References [get\\_free\\_indices\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

Referenced by [GiNaC::symmetrize\\_cyclic\(\)](#).

**6.48.3.78 symmetrize\_cyclic()** [2/2]

```
ex GiNaC::ex::symmetrize_cyclic (
    const lst & l ) const
```

Symmetrize expression by cyclic permutation over a list of objects (symbols, indices).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

**6.48.3.79 return\_type()**

```
unsigned GiNaC::ex::return_type ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::ncmul::append\\_factors\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::exmul\(\)](#), [GiNaC::matrix::mul\\_scalar\(\)](#), [GiNaC::indexed::return\\_type\(\)](#), [GiNaC::integral::return\\_type\(\)](#), [GiNaC::power::return\\_type\(\)](#), and [GiNaC::relational::return\\_type\(\)](#).

**6.48.3.80 return\_type\_tinfo()**

```
return_type_t GiNaC::ex::return_type_tinfo ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::color\\_trace\(\)](#), [GiNaC::indexed::return\\_type\\_tinfo\(\)](#), [GiNaC::integral::return\\_type\\_tinfo\(\)](#), [GiNaC::power::return\\_type\\_tinfo\(\)](#), and [GiNaC::relational::return\\_type\\_tinfo\(\)](#).

**6.48.3.81 gethash()**

```
unsigned GiNaC::ex::gethash ( ) const [inline]
```

References [bp](#).

Referenced by [GiNaC::basic::calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), and [GiNaC::relational::calchash\(\)](#).

**6.48.3.82 construct\_from\_basic()**

```
ptr< basic > GiNaC::ex::construct_from_basic (
    const basic & other ) [static], [private]
```

Helper function for the ex-from-basic constructor.

This is where [GiNaC](#)'s automatic evaluator and memory management are implemented.

See also

[ex::ex\(const basic &\)](#)

References [bp](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::dynallocated](#), [GiNaC::basic::eval\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::refcounted::get\\_refcount\(\)](#), and [GINAC\\_ASSERT](#).

### 6.48.3.83 `construct_from_int()`

```
basic & GiNaC::ex::construct_from_int (
    int i ) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), [GiNaC::\\_num\\_10\\_p](#), [GiNaC::\\_num\\_11\\_p](#), [GiNaC::\\_num\\_12\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::\\_num\\_2\\_p](#), [GiNaC::\\_num\\_3\\_p](#), [GiNaC::\\_num\\_4\\_p](#), [GiNaC::\\_num\\_5\\_p](#), [GiNaC::\\_num\\_6\\_p](#), [GiNaC::\\_num\\_7\\_p](#), [GiNaC::\\_num\\_8\\_p](#), and [GiNaC::\\_num\\_9\\_p](#).

Referenced by [construct\\_from\\_longlong\(\)](#).

### 6.48.3.84 `construct_from_uint()`

```
basic & GiNaC::ex::construct_from_uint (
    unsigned int i ) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), and [GiNaC::\\_num9\\_p](#).

Referenced by [construct\\_from\\_ulonglong\(\)](#).

### 6.48.3.85 `construct_from_long()`

```
basic & GiNaC::ex::construct_from_long (
    long i ) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), [GiNaC::\\_num\\_10\\_p](#), [GiNaC::\\_num\\_11\\_p](#), [GiNaC::\\_num\\_12\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::\\_num\\_2\\_p](#), [GiNaC::\\_num\\_3\\_p](#), [GiNaC::\\_num\\_4\\_p](#), [GiNaC::\\_num\\_5\\_p](#), [GiNaC::\\_num\\_6\\_p](#), [GiNaC::\\_num\\_7\\_p](#), [GiNaC::\\_num\\_8\\_p](#), and [GiNaC::\\_num\\_9\\_p](#).

### 6.48.3.86 `construct_from_ulong()`

```
basic & GiNaC::ex::construct_from_ulong (
    unsigned long i ) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), and [GiNaC::\\_num9\\_p](#).

#### 6.48.3.87 `construct_from_longlong()`

```
basic & GiNaC::ex::construct_from_longlong (
    long long i ) [static], [private]
```

References [construct\\_from\\_int\(\)](#).

#### 6.48.3.88 `construct_from_ulonglong()`

```
basic & GiNaC::ex::construct_from_ulonglong (
    unsigned long long i ) [static], [private]
```

References [construct\\_from\\_uint\(\)](#).

#### 6.48.3.89 `construct_from_double()`

```
basic & GiNaC::ex::construct_from_double (
    double d ) [static], [private]
```

#### 6.48.3.90 `construct_from_string_and_lst()`

```
static ptr< basic > GiNaC::ex::construct_from_string_and_lst (
    const std::string & s,
    const ex & l ) [static], [private]
```

#### 6.48.3.91 `makewriteable()`

```
void GiNaC::ex::makewriteable ( ) [private]
```

Make this ex writable (if more than one ex handle the same basic) by unlinking the object and creating an unshared copy of it.

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

Referenced by [let\\_op\(\)](#), and [operator\[\]\(\)](#).

### 6.48.3.92 share()

```
void GiNaC::ex::share (
    const ex & other ) const [private]
```

Share equal objects between expressions.

See also

`ex::compare(const ex &)`

References [bp](#), and [GiNaC::status\\_flags::not\\_shareable](#).

Referenced by [compare\(\)](#), and [is\\_equal\(\)](#).

## 6.48.4 Friends And Related Function Documentation

### 6.48.4.1 archive\_node

```
friend class archive\_node [friend]
```

### 6.48.4.2 are\_ex\_trivially\_equal

```
bool are_ex_trivially_equal (
    const ex & e1,
    const ex & e2 ) [friend]
```

Compare two objects of class quickly without doing a deep tree traversal.

Returns

"true" if they are equal "false" if equality cannot be established quickly (e1 and e2 may still be equal, in this case).

### 6.48.4.3 ex\_to

```
template<class T >
const T & ex_to (
    const ex & e ) [friend]
```

Return a reference to the basic-derived class T object embedded in an expression.

This is fast but unsafe: the result is undefined if the expression does not contain a T object at its top level. Hence, you should generally check the type of e first. Also, you shouldn't cache the returned reference because [GiNaC](#)'s garbage collector may destroy the referenced object any time it's used in another expression.

## Parameters

<i>e</i>	expression
----------	------------

## Returns

reference to object of class T

## See also

[is\\_exactly\\_a<class T>\(\)](#)

#### 6.48.4.4 is\_a

```
template<class T >
bool is_a (
    const ex & obj ) [friend]
```

Check if *ex* is a handle to a T, including base classes.

#### 6.48.4.5 is\_exactly\_a

```
template<class T >
bool is_exactly_a (
    const ex & obj ) [friend]
```

Check if *ex* is a handle to a T, not including base classes.

### 6.48.5 Member Data Documentation

#### 6.48.5.1 bp

```
ptr<basic> GiNaC::ex::bp [mutable], [private]
```

pointer to basic object managed by this

Referenced by [accept\(\)](#), [GiNaC::archive\\_node::archive\\_node\(\)](#), [coeff\(\)](#), [collect\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [construct\\_from\\_basic\(\)](#), [dbgprint\(\)](#), [dbgprinttree\(\)](#), [degree\(\)](#), [denom\(\)](#), [diff\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [eval\\_ncmul\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [ex\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [gethash\(\)](#), [has\(\)](#), [GiNaC::archive\\_node::has\\_same\\_ex\\_as\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [integer\\_content\(\)](#), [is\\_equal\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [lhs\(\)](#), [makewriteable\(\)](#), [map\(\)](#), [match\(\)](#), [max\\_coefficient\(\)](#), [nops\(\)](#), [normal\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [op\(\)](#), [operator\[\]\(\)](#), [print\(\)](#), [GiNaC::archive\\_node::printraw\(\)](#), [real\\_part\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [rhs\(\)](#), [series\(\)](#), [share\(\)](#), [smod\(\)](#), [subs\(\)](#), [swap\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

The documentation for this class was generated from the following files:

- [ex.h](#)
- [ex.cpp](#)
- [indexed.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)
- [symmetry.cpp](#)

## 6.49 GiNaC::ex\_base\_is\_less Struct Reference

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

### 6.49.1 Member Function Documentation

#### 6.49.1.1 operator()

```
bool GiNaC::ex_base_is_less::operator() (
    const ex & lh,
    const ex & rh ) const    [inline]
```

References [GiNaC::ex::op\(\)](#).

The documentation for this struct was generated from the following file:

- [indexed.cpp](#)

## 6.50 GiNaC::ex\_is\_equal Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

### 6.50.1 Member Function Documentation

#### 6.50.1.1 operator()

```
bool GiNaC::ex_is_equal::operator() (
    const ex & lh,
    const ex & rh ) const    [inline]
```

References [GiNaC::ex::is\\_equal\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.51 GiNaC::ex\_is\_less Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

#### 6.51.1 Member Function Documentation

##### 6.51.1.1 operator()

```
bool GiNaC::ex_is_less::operator() (
    const ex & lh,
    const ex & rh ) const [inline]
```

References [GiNaC::ex::compare\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.52 GiNaC::ex\_swap Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- void [operator\(\)](#) ([ex](#) &lh, [ex](#) &rh) const

#### 6.52.1 Member Function Documentation

##### 6.52.1.1 operator()

```
void GiNaC::ex_swap::operator() (
    ex & lh,
    ex & rh ) const [inline]
```

References [GiNaC::ex::swap\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

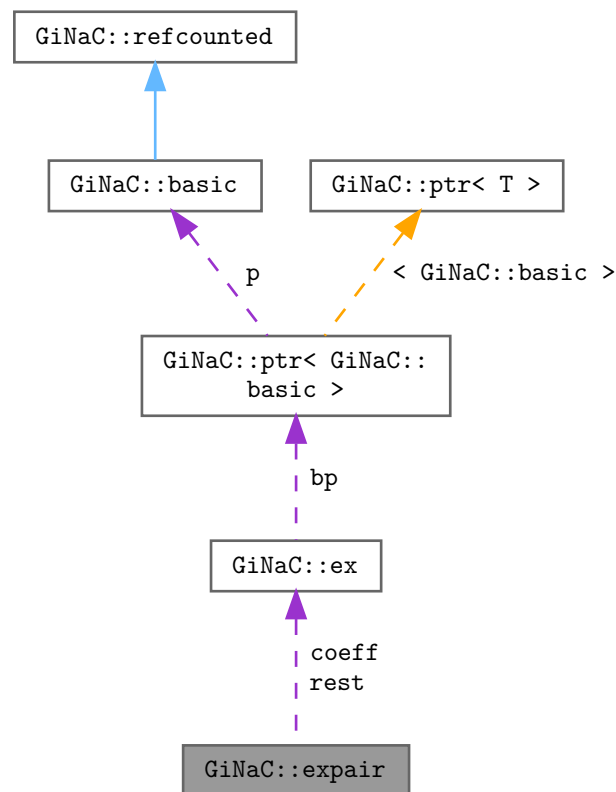


## 6.53 GiNaC::expair Class Reference

A pair of expressions.

```
#include <expair.h>
```

Collaboration diagram for GiNaC::expair:



### Public Member Functions

- `expair ()`
- `expair (const ex &r, const ex &c)`  
Construct an `expair` from two `ex`.
- `bool is_equal (const expair &other) const`  
Member-wise check for canonical ordering equality.
- `bool is_less (const expair &other) const`  
Member-wise check for canonical ordering lessness.
- `int compare (const expair &other) const`  
Member-wise check for canonical ordering.
- `void print (std::ostream &os) const`
- `bool is_canonical_numeric () const`  
True if this is of the form `(numeric,ex(1))`.
- `void swap (expair &other)`  
Swap contents with other `expair`.
- `const expair conjugate () const`

## Public Attributes

- [ex rest](#)  
*first member of pair, an arbitrary expression*
- [ex coeff](#)  
*second member of pair, must be numeric*

### 6.53.1 Detailed Description

A pair of expressions.

This is similar to STL's `pair<>`. It is slightly extended since we need to account for methods like `.compare()`. Also, since this is meant for use by class `expairseq` it must satisfy the invariance that the member `coeff` must be of type `numeric`.

### 6.53.2 Constructor & Destructor Documentation

#### 6.53.2.1 `expair()` [1/2]

```
GiNaC::expair::expair ( ) [inline]
```

Referenced by [conjugate\(\)](#).

#### 6.53.2.2 `expair()` [2/2]

```
GiNaC::expair::expair (
    const ex & r,
    const ex & c ) [inline]
```

Construct an `expair` from two `ex`.

References [coeff](#), and [GINAC\\_ASSERT](#).

### 6.53.3 Member Function Documentation

#### 6.53.3.1 `is_equal()`

```
bool GiNaC::expair::is_equal (
    const expair & other ) const [inline]
```

Member-wise check for canonical ordering equality.

References [coeff](#), [GiNaC::ex::is\\_equal\(\)](#), and [rest](#).

Referenced by [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::mul::expair\\_needs\\_further\\_processing\(\)](#), and [GiNaC::expairseq::subschildre](#)

### 6.53.3.2 is\_less()

```
bool GiNaC::expair::is_less (
    const expair & other ) const [inline]
```

Member-wise check for canonical ordering lessness.

References [coeff](#), [GiNaC::ex::compare\(\)](#), and [rest](#).

Referenced by [GiNaC::expair\\_is\\_less::operator\(\)](#).

### 6.53.3.3 compare()

```
int GiNaC::expair::compare (
    const expair & other ) const [inline]
```

Member-wise check for canonical ordering.

References [coeff](#), [GiNaC::ex::compare\(\)](#), and [rest](#).

### 6.53.3.4 print()

```
void GiNaC::expair::print (
    std::ostream & os ) const
```

References [c](#), [coeff](#), [GiNaC::ex::print\(\)](#), and [rest](#).

### 6.53.3.5 is\_canonical\_numeric()

```
bool GiNaC::expair::is_canonical_numeric ( ) const [inline]
```

True if this is of the form (numeric,ex(1)).

References [coeff](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), and [rest](#).

### 6.53.3.6 swap()

```
void GiNaC::expair::swap (
    expair & other ) [inline]
```

Swap contents with other expair.

References [coeff](#), [rest](#), and [GiNaC::ex::swap\(\)](#).

Referenced by [GiNaC::mul::derivative\(\)](#), [GiNaC::expair\\_swap::operator\(\)](#), and [GiNaC::swap\(\)](#).

### 6.53.3.7 conjugate()

```
const expair GiNaC::expair::conjugate ( ) const
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [coeff](#), [GiNaC::ex::conjugate\(\)](#), [expair\(\)](#), and [rest](#).

## 6.53.4 Member Data Documentation

### 6.53.4.1 rest

```
ex GiNaC::expair::rest
```

first member of pair, an arbitrary expression

Referenced by [GiNaC::expairseq::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [is\\_canonical\\_numeric\(\)](#), [is\\_equal\(\)](#), [is\\_less\(\)](#), [GiNaC::expair\\_rest\\_is\\_less::operator\(\)](#), [print\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::expairseq::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), and [swap\(\)](#).

### 6.53.4.2 coeff

```
ex GiNaC::expair::coeff
```

second member of pair, must be numeric

Referenced by [GiNaC::mul::can\\_make\\_flat\(\)](#), [GiNaC::expairseq::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [expair\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [is\\_canonical\\_numeric\(\)](#), [is\\_equal\(\)](#), [is\\_less\(\)](#), [print\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::expairseq::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), and [swap\(\)](#).

The documentation for this class was generated from the following files:

- [expair.h](#)
- [expair.cpp](#)

## 6.54 GiNaC::expair\_is\_less Struct Reference

Function object for insertion into third argument of STL's [sort\(\)](#) etc.

```
#include <expair.h>
```

## Public Member Functions

- bool [operator\(\)](#) (const [expair](#) &lh, const [expair](#) &rh) const

### 6.54.1 Detailed Description

Function object for insertion into third argument of STL's sort() etc.

### 6.54.2 Member Function Documentation

#### 6.54.2.1 operator()

```
bool GiNaC::expair_is_less::operator() (
    const expair & lh,
    const expair & rh ) const [inline]
```

References [GiNaC::expair::is\\_less\(\)](#).

The documentation for this struct was generated from the following file:

- [expair.h](#)

## 6.55 GiNaC::expair\_rest\_is\_less Struct Reference

Function object not caring about the numerical coefficients for insertion into third argument of STL's sort().

```
#include <expair.h>
```

## Public Member Functions

- bool [operator\(\)](#) (const [expair](#) &lh, const [expair](#) &rh) const

### 6.55.1 Detailed Description

Function object not caring about the numerical coefficients for insertion into third argument of STL's sort().

Note that this does not define a strict weak ordering since for any symbol x we have neither  $3*x < 2*x$  or  $2*x < 3*x$ . Handle with care!

### 6.55.2 Member Function Documentation

### 6.55.2.1 operator()

```
bool GiNaC::expair_rest_is_less::operator() (
    const expair & lh,
    const expair & rh ) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::expair::rest](#).

The documentation for this struct was generated from the following file:

- [expair.h](#)

## 6.56 GiNaC::expair\_swap Struct Reference

```
#include <expair.h>
```

### Public Member Functions

- void [operator\(\)](#) ([expair](#) &lh, [expair](#) &rh) const

### 6.56.1 Member Function Documentation

#### 6.56.1.1 operator()

```
void GiNaC::expair_swap::operator() (
    expair & lh,
    expair & rh ) const [inline]
```

References [GiNaC::expair::swap\(\)](#).

The documentation for this struct was generated from the following file:

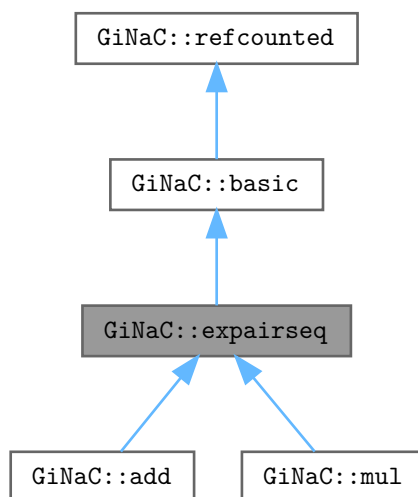
- [expair.h](#)

## 6.57 GiNaC::expairseq Class Reference

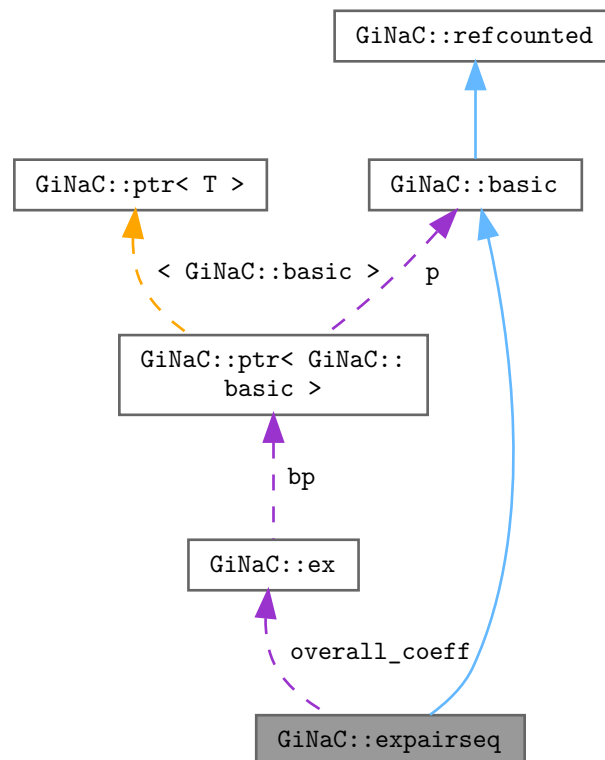
A sequence of class expair.

```
#include <expairseq.h>
```

Inheritance diagram for GiNaC::expairseq:



Collaboration diagram for `GiNaC::expairseq`:



## Public Member Functions

- `expairseq` (const `ex` &lh, const `ex` &rh)
- `expairseq` (const `exvector` &v)
- `expairseq` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false)
- `expairseq` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- size\_t `nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex map` (`map_function` &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex eval` () const override  
*Perform coefficient-wise automatic term rewriting rules in this class.*
- `ex to_rational` (`exmap` &repl) const override  
*Implementation of `ex::to_rational()` for expairseqs.*



- `ex to_polynomial` (`exmap` &`repl`) const override  
*Implementation of `ex::to_polynomial()` for `expairseqs`.*
- bool `match` (const `ex` &`pattern`, `exmap` &`repl_lst`) const override  
*Check whether the expression matches a given pattern.*
- `ex subs` (const `exmap` &`m`, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex conjugate` () const override
- void `archive` (`archive_node` &`n`) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
*Load (deserialize) the object from an archive node.*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &`i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &`c`, unsigned `level`=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthesizing output).*
- virtual bool `info` (unsigned `inf`) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t `i`) const  
*Return operand/member at position `i`.*
- virtual `ex operator[]` (const `ex` &`index`) const
- virtual `ex operator[]` (size\_t `i`) const
- virtual `ex & let_op` (size\_t `i`)  
*Return modifiable operand/member at position `i`.*
- virtual `ex & operator[]` (const `ex` &`index`)

- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_info` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

- Like `print()`, but dispatch to the specified class.
- virtual void `archive` (`archive_node` &n) const  
Save (serialize) the object into archive node.
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
Load (deserialize) the object from an archive node.
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
Helper function for `subs()`.
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
Default interface of `nth` derivative `ex::diff(s, n)`.
- int `compare` (const `basic` &other) const  
Compare objects syntactically to establish canonical ordering.
- bool `is_equal` (const `basic` &other) const  
Test for syntactic equality.
- const `basic` & `hold` () const  
Stop further evaluation.
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned `f`) const  
Set some `status_flags`.
- const `basic` & `clearflag` (unsigned `f`) const  
Clear some `status_flags`.

#### Public Member Functions inherited from GiNaC::refcounted

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

#### Protected Member Functions

- bool `is_equal_same_type` (const `basic` &other) const override  
Returns true if two objects of same type are equal.
- unsigned `return_type` () const override
- unsigned `calchash` () const override  
Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.
- `ex expand` (unsigned `options`=0) const override  
Expand expression, i.e.
- virtual `ex thisexpairseq` (const `epvector` &v, const `ex` &oc, bool `do_index_renaming`=false) const  
Create an object of this type.
- virtual `ex thisexpairseq` (`epvector` &&vp, const `ex` &oc, bool `do_index_renaming`=false) const
- virtual void `printseq` (const `print_context` &c, char `delim`, unsigned `this_precedence`, unsigned `upper_precedence`) const
- virtual void `printpair` (const `print_context` &c, const `expair` &p, unsigned `upper_precedence`) const
- virtual `expair split_ex_to_pair` (const `ex` &e) const  
Form an `expair` from an `ex`, using the corresponding semantics.
- virtual `expair combine_ex_with_coeff_to_pair` (const `ex` &e, const `ex` &c) const
- virtual `expair combine_pair_with_coeff_to_pair` (const `expair` &p, const `ex` &c) const
- virtual `ex recombine_pair_to_ex` (const `expair` &p) const  
Form an `ex` out of an `expair`, using the corresponding semantics.

- virtual bool `expire_needs_further_processing` (epp it)
- virtual `ex default_overall_coeff` () const
- virtual void `combine_overall_coeff` (const `ex` &c)
- virtual void `combine_overall_coeff` (const `ex` &c1, const `ex` &c2)
- virtual bool `can_make_flat` (const `expire` &p) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `construct_from_2_ex` (const `ex` &lh, const `ex` &rh)
- void `construct_from_2_expireseq` (const `expireseq` &s1, const `expireseq` &s2)
- void `construct_from_expireseq_ex` (const `expireseq` &s, const `ex` &e)
- void `construct_from_exvector` (const `exvector` &v)
- void `construct_from_epvector` (const `epvector` &v, bool do\_index\_renaming=false)
- void `construct_from_epvector` (`epvector` &&v, bool do\_index\_renaming=false)
- void `make_flat` (const `exvector` &v)  
Combine this `expireseq` with argument `exvector`.
- void `make_flat` (const `epvector` &v, bool do\_index\_renaming=false)  
Combine this `expireseq` with argument `epvector`.
- void `canonicalize` ()  
Brings this `expireseq` into a sorted (canonical) form.
- void `combine_same_terms_sorted_seq` ()  
Compact a presorted `expireseq` by combining all matching `expires` to one each.
- bool `is_canonical` () const  
Check if this `expireseq` is in sorted (canonical) form.
- `epvector expandchildren` (unsigned `options`) const  
Member-wise expand the `expires` in this sequence.
- `epvector evalchildren` () const  
Member-wise evaluate the `expires` in this sequence.
- `epvector subschildren` (const `exmap` &m, unsigned `options`=0) const  
Member-wise substitute in this sequence.

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
Returns true if the attributes of two objects are similar enough for a match.
- virtual `ex derivative` (const `symbol` &s) const  
Default implementation of `ex::diff()`.
- virtual int `compare_same_type` (const `basic` &other) const  
Returns order relation between two objects of same type.
- virtual bool `is_equal_same_type` (const `basic` &other) const  
Returns true if two objects of same type are equal.
- virtual unsigned `calchash` () const  
Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.
- void `ensure_if_modifiable` () const  
Ensure the object may be modified without hurting others, throws if this is not the case.
- void `do_print` (const `print_context` &c, unsigned level) const  
Default output to stream.
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
Tree output to stream.
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
Python parsable output to stream.

## Protected Attributes

- [epvector seq](#)
- [ex overall\\_coeff](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.57.1 Detailed Description

A sequence of class `expair`.

This is used for time-critical classes like sums and products of terms since handling a list of `coeff` and `rest` is much faster than handling a list of products or powers, respectively. (Not incidentally, Maple does it the same way, maybe others too.) The semantics is (at least) twofold: one for addition and one for multiplication and several methods have to be overridden by derived classes to reflect the change in semantics. However, most functionality turns out to be shared between addition and multiplication, which is the reason why there is this base class.

## 6.57.2 Constructor & Destructor Documentation

### 6.57.2.1 `expairseq()` [1/4]

```
GiNaC::expairseq::expairseq (
    const ex & lh,
    const ex & rh )
```

References [construct\\_from\\_2\\_ex\(\)](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

### 6.57.2.2 `expairseq()` [2/4]

```
GiNaC::expairseq::expairseq (
    const exvector & v )
```

References [construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

### 6.57.2.3 `expairseq()` [3/4]

```
GiNaC::expairseq::expairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false )
```

References [construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

### 6.57.2.4 `expairseq()` [4/4]

```
GiNaC::expairseq::expairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false )
```

References [construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

## 6.57.3 Member Function Documentation

### 6.57.3.1 `precedence()`

```
unsigned GiNaC::expairseq::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

Referenced by [do\\_print\(\)](#), and [printpair\(\)](#).

### 6.57.3.2 `info()`

```
bool GiNaC::expairseq::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::info\\_flags::expanded](#), [GiNaC::basic::flags](#), [GiNaC::status\\_flags::has\\_indices](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::status\\_flags::has\\_no\\_indices](#), [seq](#), and [GiNaC::basic::setflag\(\)](#).

### 6.57.3.3 nops()

```
size_t GiNaC::expairseq::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), and [seq](#).

Referenced by [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [GiNaC::mul::has\(\)](#), and [match\(\)](#).

### 6.57.3.4 op()

```
ex GiNaC::expairseq::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [seq](#).

Referenced by [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [match\(\)](#), [GiNaC::add::series\(\)](#), and [GiNaC::mul::series\(\)](#).

### 6.57.3.5 map()

```
ex GiNaC::expairseq::map (
    map_function & f ) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [thisexpairseq\(\)](#).

### 6.57.3.6 eval()

```
ex GiNaC::expairseq::eval ( ) const [override], [virtual]
```

Perform coefficient-wise automatic term rewriting rules in this class.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [evalchildren\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::basic::hold\(\)](#), and [overall\\_coeff](#).

### 6.57.3.7 to\_rational()

```
ex GiNaC::expairseq::to_rational (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for expairseqs.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), [GiNaC::ex::to\\_rational\(\)](#), and [to\\_rational\(\)](#).

Referenced by [to\\_rational\(\)](#).

### 6.57.3.8 to\_polynomial()

```
ex GiNaC::expairseq::to_polynomial (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for expairseqs.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), [GiNaC::ex::to\\_polynomial\(\)](#), and [to\\_polynomial\(\)](#).

Referenced by [to\\_polynomial\(\)](#).



### 6.57.3.9 match()

```
bool GiNaC::expairseq::match (
    const ex & pattern,
    exmap & repl_lst ) const [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to *repl\_lst*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::begin\(\)](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::nops\(\)](#), [nops\(\)](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), [split\\_ex\\_to\\_pair\(\)](#), and [thisexpairseq\(\)](#).

### 6.57.3.10 subs()

```
ex GiNaC::expairseq::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The *ex* returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::subs\\_options::algebraic](#), [m](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [options](#), [overall\\_coeff](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [subschildren\(\)](#), and [thisexpairseq\(\)](#).

### 6.57.3.11 conjugate()

```
ex GiNaC::expairseq::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::conjugateepvector\(\)](#), [overall\\_coeff](#), [seq](#), [thisexpairseq\(\)](#), and [x](#).

### 6.57.3.12 archive()

```
void GiNaC::expairseq::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [n](#), [overall\\_coeff](#), and [seq](#).

### 6.57.3.13 read\_archive()

```
void GiNaC::expairseq::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [canonicalize\(\)](#), [GiNaC::basic::coeff\(\)](#), [GINAC\\_ASSERT](#), [is\\_canonical\(\)](#), [n](#), [overall\\_coeff](#), and [seq](#).

### 6.57.3.14 is\_equal\_same\_type()

```
bool GiNaC::expairseq::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), and [seq](#).

### 6.57.3.15 return\_type()

```
unsigned GiNaC::expairseq::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

### 6.57.3.16 calchash()

```
unsigned GiNaC::expairseq::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [overall\\_coeff](#), [GiNaC::rotate\\_left\(\)](#), [seq](#), and [GiNaC::basic::setflag\(\)](#).

### 6.57.3.17 expand()

```
ex GiNaC::expairseq::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [options](#), [overall\\_coeff](#), [GiNaC::basic::setflag\(\)](#), and [thisexpairseq\(\)](#).

**6.57.3.18 thisexpairseq() [1/2]**

```
ex GiNaC::expairseq::thisexpairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false ) const [protected], [virtual]
```

Create an object of this type.

This method works similar to a constructor. It is useful because `expairseq` has (at least) two possible different semantics but we want to inherit methods thus avoiding code duplication. Sometimes a method in `expairseq` has to create a new one of the same semantics, which cannot be done by a ctor because the name (`add`, `mul`,...) is unknown on the `expairseq` level. In order for this trick to work a derived class must of course override this definition.

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

Referenced by [conjugate\(\)](#), [expand\(\)](#), [map\(\)](#), [match\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.57.3.19 thisexpairseq() [2/2]**

```
ex GiNaC::expairseq::thisexpairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false ) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

**6.57.3.20 printseq()**

```
void GiNaC::expairseq::printseq (
    const print\_context & c,
    char delim,
    unsigned this_precedence,
    unsigned upper_precedence ) const [protected], [virtual]
```

References [c](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), [GiNaC::ex::print\(\)](#), [printpair\(\)](#), and [seq](#).

Referenced by [do\\_print\(\)](#).

**6.57.3.21 printpair()**

```
void GiNaC::expairseq::printpair (
    const print\_context & c,
    const expair & p,
    unsigned upper_precedence ) const [protected], [virtual]
```

References [c](#), [GiNaC::expair::coeff](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::expair::rest](#).

Referenced by [is\\_canonical\(\)](#), and [printseq\(\)](#).

**6.57.3.22 split\_ex\_to\_pair()**

```
expair GiNaC::expairseq::split_ex_to_pair (
    const ex & e ) const [protected], [virtual]
```

Form an expair from an ex, using the corresponding semantics.

See also

[expairseq::recombine\\_pair\\_to\\_ex\(\)](#)

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::\\_ex1](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [make\\_flat\(\)](#), [map\(\)](#), [match\(\)](#), [subschildren\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.57.3.23 combine\_ex\_with\_coeff\_to\_pair()**

```
expair GiNaC::expairseq::combine_ex_with_coeff_to_pair (
    const ex & e,
    const ex & c ) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [c](#), and [GINAC\\_ASSERT](#).

Referenced by [evalchildren\(\)](#), and [subschildren\(\)](#).

**6.57.3.24 combine\_pair\_with\_coeff\_to\_pair()**

```
expair GiNaC::expairseq::combine_pair_with_coeff_to_pair (
    const expair & p,
    const ex & c ) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [c](#), [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), and [GiNaC::expair::rest](#).

**6.57.3.25 recombine\_pair\_to\_ex()**

```
ex GiNaC::expairseq::recombine_pair_to_ex (
    const expair & p ) const [protected], [virtual]
```

Form an ex out of an expair, using the corresponding semantics.

See also

[expairseq::split\\_ex\\_to\\_pair\(\)](#)

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::expair::coeff](#), and [GiNaC::expair::rest](#).

Referenced by [map\(\)](#), [op\(\)](#), [subschildren\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.57.3.26 expair\_needs\_further\_processing()**

```
bool GiNaC::expairseq::expair_needs_further_processing (
    exp it ) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GiNaC::\\_ex1](#).

Referenced by [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), and [construct\\_from\\_expairseq\\_ex\(\)](#).

**6.57.3.27 default\_overall\_coeff()**

```
ex GiNaC::expairseq::default_overall_coeff ( ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GiNaC::\\_ex0](#).

Referenced by [do\\_print\\_tree\(\)](#), [map\(\)](#), [match\(\)](#), [nops\(\)](#), [op\(\)](#), [printseq\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.57.3.28 combine\_overall\_coeff() [1/2]**

```
void GiNaC::expairseq::combine_overall_coeff (
    const ex & c ) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [c](#), [GINAC\\_ASSERT](#), and [overall\\_coeff](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), and [make\\_flat\(\)](#).

**6.57.3.29 combine\_overall\_coeff()** [2/2]

```
void GiNaC::expairseq::combine_overall_coeff (
    const ex & c1,
    const ex & c2 ) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GINAC\\_ASSERT](#), and [overall\\_coeff](#).

**6.57.3.30 can\_make\_flat()**

```
bool GiNaC::expairseq::can_make_flat (
    const expair & p ) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

**6.57.3.31 do\_print()**

```
void GiNaC::expairseq::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), [precedence\(\)](#), and [printseq\(\)](#).

**6.57.3.32 do\_print\_tree()**

```
void GiNaC::expairseq::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::is\\_equal\(\)](#), [nops\(\)](#), [overall\\_coeff](#), [GiNaC::ex::print\(\)](#), and [seq](#).

**6.57.3.33 construct\_from\_2\_ex()**

```
void GiNaC::expairseq::construct_from_2_ex (
    const ex & lh,
    const ex & rh ) [protected]
```

References [GiNaC::expair::coeff](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::ex::compare\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::expair::rest](#), [seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [GiNaC::add::add\(\)](#), [expairseq\(\)](#), and [GiNaC::mul::mul\(\)](#).

**6.57.3.34 construct\_from\_2\_expairseq()**

```
void GiNaC::expairseq::construct_from_2_expairseq (
    const expairseq & s1,
    const expairseq & s2 ) [protected]
```

References [combine\\_overall\\_coeff\(\)](#), [construct\\_from\\_epvector\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [overall\\_coeff](#), and [seq](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#).

**6.57.3.35 construct\_from\_expairseq\_ex()**

```
void GiNaC::expairseq::construct_from_expairseq_ex (
    const expairseq & s,
    const ex & e ) [protected]
```

References [GiNaC::expair::coeff](#), [combine\\_overall\\_coeff\(\)](#), [construct\\_from\\_epvector\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [last](#), [overall\\_coeff](#), [GiNaC::expair::rest](#), [seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#).

**6.57.3.36 construct\_from\_exvector()**

```
void GiNaC::expairseq::construct_from_exvector (
    const exvector & v ) [protected]
```

References [canonicalize\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), and [make\\_flat\(\)](#).

Referenced by [GiNaC::add::add\(\)](#), [expairseq\(\)](#), and [GiNaC::mul::mul\(\)](#).

**6.57.3.37 construct\_from\_epvector() [1/2]**

```
void GiNaC::expairseq::construct_from_epvector (
    const epvector & v,
    bool do_index_renaming = false ) [protected]
```

References [canonicalize\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), and [make\\_flat\(\)](#).

Referenced by [GiNaC::add::add\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [expairseq\(\)](#), and [GiNaC::mul::mul\(\)](#).



**6.57.3.38 construct\_from\_epvector()** [2/2]

```
void GiNaC::expairseq::construct_from_epvector (
    epvector && v,
    bool do_index_renaming = false ) [protected]
```

References [canonicalize\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), and [make\\_flat\(\)](#).

**6.57.3.39 make\_flat()** [1/2]

```
void GiNaC::expairseq::make_flat (
    const exvector & v ) [protected]
```

Combine this expairseq with argument exvector.

It cares for associativity as well as for special handling of numerics.

References [GiNaC::ex1](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [overall\\_coeff](#), [seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [construct\\_from\\_epvector\(\)](#), and [construct\\_from\\_exvector\(\)](#).

**6.57.3.40 make\_flat()** [2/2]

```
void GiNaC::expairseq::make_flat (
    const epvector & v,
    bool do_index_renaming = false ) [protected]
```

Combine this expairseq with argument epvector.

It cares for associativity as well as for special handling of numerics.

References [GiNaC::ex1](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::make\\_flat\\_inserter::h](#), [GiNaC::info\\_flags::has\\_indices](#), [overall\\_coeff](#), and [seq](#).

**6.57.3.41 canonicalize()**

```
void GiNaC::expairseq::canonicalize ( ) [protected]
```

Brings this expairseq into a sorted (canonical) form.

References [seq](#).

Referenced by [construct\\_from\\_epvector\(\)](#), [construct\\_from\\_exvector\(\)](#), and [read\\_archive\(\)](#).

**6.57.3.42 combine\_same\_terms\_sorted\_seq()**

```
void GiNaC::expairseq::combine_same_terms_sorted_seq ( ) [protected]
```

Compact a presorted expairseq by combining all matching expairs to one each.

On an add object, this is responsible for  $2*x+3*x+y \rightarrow 5*x+y$ , for instance.

References [construct\\_from\\_epvector\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [last](#), and [seq](#).

Referenced by [construct\\_from\\_epvector\(\)](#), and [construct\\_from\\_exvector\(\)](#).

**6.57.3.43 is\_canonical()**

```
bool GiNaC::expairseq::is_canonical ( ) const [protected]
```

Check if this expairseq is in sorted (canonical) form.

Useful mainly for debugging or in assertions since being sorted is an invariance.

References [printpair\(\)](#), and [seq](#).

Referenced by [GiNaC::add::add\(\)](#), [expairseq\(\)](#), [GiNaC::mul::mul\(\)](#), and [read\\_archive\(\)](#).

**6.57.3.44 expandchildren()**

```
epvector GiNaC::expairseq::expandchildren (
    unsigned options ) const [protected]
```

Member-wise expand the expairs in this sequence.

See also

[expairseq::expand\(\)](#)

Returns

epvector containing expanded pairs, empty if no members had to be changed.

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [last](#), [options](#), and [seq](#).

Referenced by [GiNaC::add::expand\(\)](#), and [expand\(\)](#).

#### 6.57.3.45 evalchildren()

```
epvector GiNaC::expairseq::evalchildren ( ) const [protected]
```

Member-wise evaluate the expairs in this sequence.

See also

[expairseq::eval\(\)](#)

Returns

epvector containing evaluated pairs, empty if no members had to be changed.

References [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expair::is\\_equal\(\)](#), [last](#), [seq](#), and [unlikely](#).

Referenced by [GiNaC::add::eval\(\)](#), [eval\(\)](#), and [GiNaC::mul::eval\(\)](#).

#### 6.57.3.46 subschildren()

```
epvector GiNaC::expairseq::subschildren (
    const exmap & m,
    unsigned options = 0 ) const [protected]
```

Member-wise substitute in this sequence.

See also

[expairseq::subs\(\)](#)

Returns

epvector containing expanded pairs, empty if no members had to be changed.

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expair::is\\_equal\(\)](#), [last](#), [m](#), [options](#), [GiNaC::subs\\_options::pattern\\_is\\_not\\_product](#), [GiNaC::subs\\_options::pattern\\_is\\_product](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [subs\(\)](#).

### 6.57.4 Member Data Documentation

### 6.57.4.1 seq

`epvector GiNaC::expairseq::seq [protected]`

Referenced by [archive\(\)](#), [calchash\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [canonicalize\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [construct\\_from\\_2\\_ex\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::add::degree\(\)](#), [GiNaC::mul::degree\(\)](#), [GiNaC::add::derivative\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::add::eval\\_ncmul\(\)](#), [GiNaC::mul::eval\\_ncmul\(\)](#), [evalchildren\(\)](#), [GiNaC::mul::evalf\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::add::info\(\)](#), [info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [is\\_canonical\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::add::is\\_polynomial\(\)](#), [GiNaC::mul::is\\_polynomial\(\)](#), [GiNaC::add::lddegree\(\)](#), [GiNaC::mul::lddegree\(\)](#), [make\\_flat\(\)](#), [map\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [nops\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [op\(\)](#), [GiNaC::add::print\\_add\(\)](#), [printseq\(\)](#), [read\\_archive\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::add::return\\_type\(\)](#), [GiNaC::mul::return\\_type\(\)](#), [GiNaC::add::return\\_type\\_tinfo\(\)](#), [GiNaC::mul::return\\_type\\_tinfo\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [subschildren\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 6.57.4.2 overall\_coeff

`ex GiNaC::expairseq::overall_coeff [protected]`

Referenced by [GiNaC::add::add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\(\)](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::add::degree\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::add::eval\(\)](#), [eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::mul::evalf\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::add::expand\(\)](#), [expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::add::info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::add::lddegree\(\)](#), [make\\_flat\(\)](#), [map\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [GiNaC::mul::mul\(\)](#), [nops\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [op\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [printseq\(\)](#), [read\\_archive\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

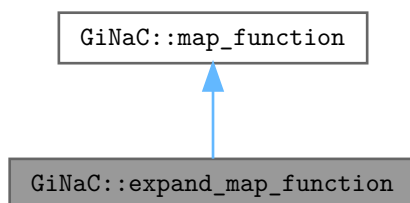
The documentation for this class was generated from the following files:

- [expairseq.h](#)
- [expairseq.cpp](#)
- [normal.cpp](#)

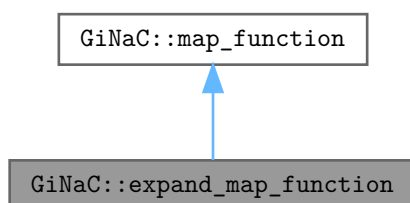
## 6.58 GiNaC::expand\_map\_function Struct Reference

Function object to be applied by [basic::expand\(\)](#).

Inheritance diagram for GiNaC::expand\_map\_function:



Collaboration diagram for GiNaC::expand\_map\_function:



## Public Member Functions

- [expand\\_map\\_function](#) (unsigned o)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()
- virtual [ex operator\(\)](#) (const [ex](#) &e)=0

## Public Attributes

- unsigned [options](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 6.58.1 Detailed Description

Function object to be applied by [basic::expand\(\)](#).

### 6.58.2 Constructor & Destructor Documentation

#### 6.58.2.1 `expand_map_function()`

```
GiNaC::expand_map_function::expand_map_function (
    unsigned o ) [inline]
```

### 6.58.3 Member Function Documentation

#### 6.58.3.1 `operator()()`

```
ex GiNaC::expand_map_function::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::ex::expand\(\)](#), and [options](#).

### 6.58.4 Member Data Documentation

#### 6.58.4.1 `options`

```
unsigned GiNaC::expand_map_function::options
```

The documentation for this struct was generated from the following file:

- [basic.cpp](#)

## 6.59 GiNaC::expand\_options Class Reference

Flags to control the behavior of [expand\(\)](#).

```
#include <flags.h>
```

## Public Types

- enum { [expand\\_indexed](#) = 0x0001 , [expand\\_function\\_args](#) = 0x0002 , [expand\\_rename\\_idx](#) = 0x0004 , [expand\\_transcendental](#) = 0x0008 }

### 6.59.1 Detailed Description

Flags to control the behavior of [expand\(\)](#).

### 6.59.2 Member Enumeration Documentation

#### 6.59.2.1 anonymous enum

anonymous enum

##### Enumerator

<a href="#">expand_indexed</a>	expands (a+b).i to a.i+b.i
<a href="#">expand_function_args</a>	expands the arguments of functions
<a href="#">expand_rename_idx</a>	used internally by <a href="#">mul::expand()</a>
<a href="#">expand_transcendental</a>	expands transcendental functions like log and exp

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.60 GiNaC::factor\_options Class Reference

Flags to control the polynomial factorization.

```
#include <flags.h>
```

## Public Types

- enum { [polynomial](#) = 0x0000 , [all](#) = 0x0001 }

### 6.60.1 Detailed Description

Flags to control the polynomial factorization.

## 6.60.2 Member Enumeration Documentation

### 6.60.2.1 anonymous enum

anonymous enum

#### Enumerator

polynomial	factor only expressions that are polynomials
all	factor all polynomial subexpressions

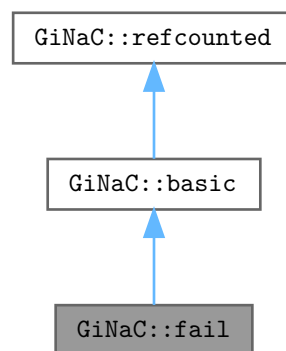
The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.61 GiNaC::fail Class Reference

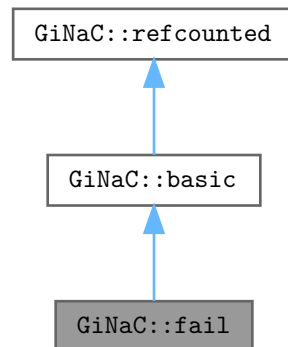
```
#include <fail.h>
```

Inheritance diagram for GiNaC::fail:





Collaboration diagram for GiNaC::fail:



## Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

- Check whether this is a polynomial in the given variables.*

  - virtual int [degree](#) (const [ex](#) &s) const

*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const

*Return degree of lowest power in object s.*
- virtual [ex](#) [coeff](#) (const [ex](#) &s, int [n](#)=1) const

*Return coefficient of degree n in object s.*
- virtual [ex](#) [expand](#) (unsigned [options](#)=0) const

*Expand expression, i.e.*
- virtual [ex](#) [collect](#) (const [ex](#) &s, bool [distributed](#)=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex](#) [series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const

*Default implementation of [ex::series\(\)](#).*
- virtual [ex](#) [normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const

*Default implementation of [ex::normal\(\)](#).*
- virtual [ex](#) [to\\_rational](#) ([exmap](#) &repl) const

*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex](#) [to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric](#) [integer\\_content](#) () const
- virtual [ex](#) [smod](#) (const [numeric](#) &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric](#) [max\\_coefficient](#) () const

*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector](#) [get\\_free\\_indices](#) () const

*Return a vector containing the free indices of an expression.*
- virtual [ex](#) [add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const

*Add two indexed expressions.*
- virtual [ex](#) [scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const

*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) ([exvector::iterator](#) self, [exvector::iterator](#) other, [exvector](#) &v) const

*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex](#) [conjugate](#) () const
- virtual [ex](#) [real\\_part](#) () const
- virtual [ex](#) [imag\\_part](#) () const
- template<class T >
  - void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const

*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const

*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const

*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)

*Load (deserialize) the object from an archive node.*
- [ex](#) [subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const

*Helper function for [subs\(\)](#).*
- [ex](#) [diff](#) (const [symbol](#) &s, unsigned [nth](#)=1) const

*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const

*Compare objects syntactically to establish canonical ordering.*

- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.61.1 Member Function Documentation

### 6.61.1.1 [return\\_type\(\)](#)

`unsigned GiNaC::fail::return_type ( ) const [inline], [override], [protected], [virtual]`

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

### 6.61.1.2 [do\\_print\(\)](#)

```
void GiNaC::fail::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following file:

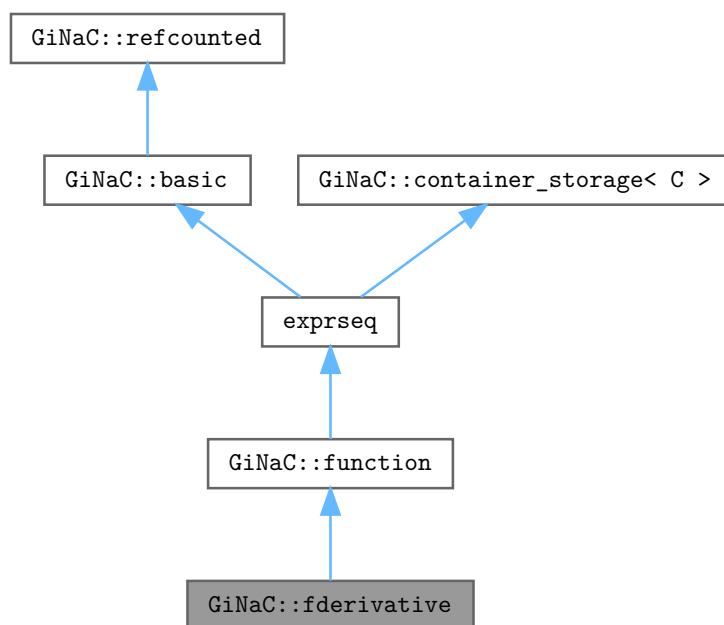
- [fail.h](#)

## 6.62 GiNaC::fderivative Class Reference

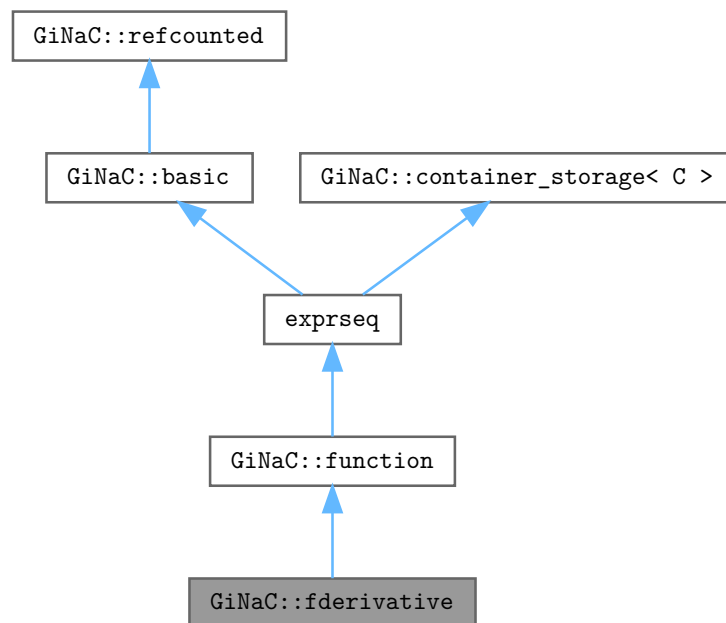
This class represents the (abstract) derivative of a symbolic function.

```
#include <fderivative.h>
```

Inheritance diagram for GiNaC::fderivative:



Collaboration diagram for `GiNaC::fderivative`:



## Public Member Functions

- `fderivative` (unsigned ser, unsigned param, const `exvector` &args)  
*Construct derivative with respect to one parameter.*
- `fderivative` (unsigned ser, const `paramset` &params, const `exvector` &args)  
*Construct derivative with respect to multiple parameters.*
- `fderivative` (unsigned ser, const `paramset` &params, `exvector` &&v)
- void `print` (const `print_context` &c, unsigned level=0) const override  
*Output to stream.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex series` (const `relational` &r, int order, unsigned options=0) const override  
*The series expansion of derivatives falls back to Taylor expansion.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- void `archive` (`archive_node` &n) const override  
*Archive the object.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Load (deserialize) the object from an archive node.*
- const `paramset` & `derivatives` () const  
*Expose this object's derivative structure.*

Public Member Functions inherited from [GiNaC::function](#)

- [function](#) (unsigned ser)
- [function](#) (unsigned ser, const [ex](#) &param1)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13, const [ex](#) &param14)
- [function](#) (unsigned ser, const [exprseq](#) &es)
- [function](#) (unsigned ser, const [exvector](#) &v)
- [function](#) (unsigned ser, [exvector](#) &&v)
- void [print](#) (const [print\\_context](#) &c, unsigned level=0) const override  
*Output to stream.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex eval\\_ncmul](#) (const [exvector](#) &v) const override  
*This method is defined to be in line with behavior of [function::return\\_type\(\)](#)*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series](#) for functions.*
- [ex thiscontainer](#) (const [exvector](#) &v) const override
- [ex thiscontainer](#) ([exvector](#) &&v) const override
- [ex conjugate](#) () const override  
*Implementation of [ex::conjugate](#) for functions.*

- `ex real_part ()` const override  
*Implementation of `ex::real_part` for functions.*
- `ex imag_part ()` const override  
*Implementation of `ex::imag_part` for functions.*
- `void archive (archive_node &n)` const override  
*Archive the object.*
- `void read_archive (const archive_node &n, lst &symbols)` override  
*Construct object from `archive_node`.*
- `bool info (unsigned inf)` const override  
*Implementation of `ex::info` for functions.*
- `ex power (const ex &exp)` const
- `unsigned get_serial ()` const
- `std::string get_name ()` const  
*Return the print name of the function.*

### Public Member Functions inherited from `GiNaC::container< C >`

- `container (STLT const &s)`
- `container (STLT &&v)`
- `container (exvector::const_iterator b, exvector::const_iterator e)`
- `container (std::initializer_list< ex > il)`
- `bool info (unsigned inf)` const override  
*Information about the object.*
- `unsigned precedence ()` const override  
*Return relative operator precedence (for parenthezing output).*
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position *i*.*
- `ex &let_op (size_t i)` override  
*Return modifiable operand/member at position *i*.*
- `ex subs (const exmap &m, unsigned options=0)` const override  
*Substitute a set of objects by arbitrary expressions.*
- `void read_archive (const archive_node &n, lst &sym_lst)` override  
*Load (deserialize) the object from an archive node.*
- `void archive (archive_node &n)` const override  
*Archive the object.*
- `container &prepend (const ex &b)`  
*Add element at front.*
- `container &append (const ex &b)`  
*Add element at back.*
- `container &remove_first ()`  
*Remove first element.*
- `container &remove_last ()`  
*Remove last element.*
- `container &remove_all ()`  
*Remove all elements.*
- `container &sort ()`  
*Sort elements.*
- `container &unique ()`  
*Remove adjacent duplicate elements.*
- `const_iterator begin ()` const
- `const_iterator end ()` const
- `const_reverse_iterator rbegin ()` const
- `const_reverse_iterator rend ()` const



Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for derivatives.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::function](#)

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for functions.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- [ex\\_pderivative](#) (unsigned diff\_param) const
- [ex\\_expl\\_derivative](#) (const [symbol](#) &s) const
- bool [lookup\\_remember\\_table](#) ([ex](#) &result) const
- void [store\\_remember\\_table](#) ([ex](#) const &result) const

### Protected Member Functions inherited from [GiNaC::container< C >](#)

- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- virtual [ex thiscontainer](#) (const [STLT](#) &v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- virtual [ex thiscontainer](#) ([STLT](#) &&v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this↔\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned [options](#)=0) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

## Protected Attributes

- [paramset parameter\\_set](#)  
*Set of parameter numbers with respect to which to take the derivative.*

## Protected Attributes inherited from [GiNaC::function](#)

- unsigned [serial](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)

## Additional Inherited Members

### Public Types inherited from [GiNaC::container< C >](#)

- typedef [STLT::const\\_iterator](#) [const\\_iterator](#)
- typedef [STLT::const\\_reverse\\_iterator](#) [const\\_reverse\\_iterator](#)

### Static Public Member Functions inherited from [GiNaC::function](#)

- static unsigned [register\\_new](#) ([function\\_options](#) const &opt)
- static unsigned [find\\_function](#) (const std::string &name, unsigned nparams)  
*Find serial number of function by name and number of parameters.*
- static std::vector< [function\\_options](#) > [get\\_registered\\_functions](#) ()

### Static Public Attributes inherited from [GiNaC::function](#)

- static unsigned [current\\_serial](#) = 0  
*This can be used as a hook for external applications.*

### Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

### Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef C< [ex](#) > [STLT](#)

### Static Protected Member Functions inherited from [GiNaC::function](#)

- static `std::vector< function\_options > & registered_functions ()`

### Static Protected Member Functions inherited from [GiNaC::container< C >](#)

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for `lst`.*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for `lst`.*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for `lst`.*

### Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) ([STLT](#) &, `size_t`)

## 6.62.1 Detailed Description

This class represents the (abstract) derivative of a symbolic function.

It is used to represent the derivatives of functions that do not have a derivative or series expansion procedure defined.

## 6.62.2 Constructor & Destructor Documentation

### 6.62.2.1 `fderivative()` [1/3]

```
GiNaC::fderivative::fderivative (
    unsigned ser,
    unsigned param,
    const exvector & args )
```

Construct derivative with respect to one parameter.

#### Parameters

<i>ser</i>	Serial number of function
<i>param</i>	Number of parameter with respect to which to take the derivative
<i>args</i>	Arguments of derivative function

References [parameter\\_set](#).

### 6.62.2.2 fderivative() [2/3]

```
GiNaC::fderivative::fderivative (
    unsigned ser,
    const paramset & params,
    const exvector & args )
```

Construct derivative with respect to multiple parameters.

#### Parameters

<i>ser</i>	Serial number of function
<i>params</i>	Set of numbers of parameters with respect to which to take the derivative
<i>args</i>	Arguments of derivative function

### 6.62.2.3 fderivative() [3/3]

```
GiNaC::fderivative::fderivative (
    unsigned ser,
    const paramset & params,
    exvector && v )
```

## 6.62.3 Member Function Documentation

### 6.62.3.1 print()

```
void GiNaC::fderivative::print (
    const print\_context & c,
    unsigned level = 0 ) const [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

#### Parameters

<i>c</i>	print context object that describes the output formatting
<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented from [GiNaC::basic](#).

References [c](#), and [GiNaC::basic::print\(\)](#).

**6.62.3.2 eval()**

```
ex GiNaC::fderivative::eval ( ) const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#), [parameter\\_set](#), [GiNaC::function::pderivative\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

**6.62.3.3 series()**

```
ex GiNaC::fderivative::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

The series expansion of derivatives falls back to Taylor expansion.

See also

[basic::series](#)

Reimplemented from [GiNaC::basic](#).

References [options](#), [order](#), [r](#), and [GiNaC::basic::series\(\)](#).

**6.62.3.4 thiscontainer() [1/2]**

```
ex GiNaC::fderivative::thiscontainer (
    const exvector & v ) const [override]
```

References [parameter\\_set](#), and [GiNaC::function::serial](#).

**6.62.3.5 thiscontainer() [2/2]**

```
ex GiNaC::fderivative::thiscontainer (
    exvector && v ) const [override]
```

References [parameter\\_set](#), and [GiNaC::function::serial](#).



### 6.62.3.6 archive()

```
void GiNaC::fderivative::archive (
    archive_node & n ) const [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::container< C >::end\(\)](#), [n](#), and [parameter\\_set](#).

### 6.62.3.7 read\_archive()

```
void GiNaC::fderivative::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::container< C >](#).

References [n](#), and [parameter\\_set](#).

### 6.62.3.8 derivative()

```
ex GiNaC::fderivative::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for derivatives.

It applies the chain rule.

#### See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_zero\(\)](#), [parameter\\_set](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

### 6.62.3.9 `is_equal_same_type()`

```
bool GiNaC::fderivative::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), and [parameter\\_set](#).

### 6.62.3.10 `match_same_type()`

```
bool GiNaC::fderivative::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [parameter\\_set](#).

### 6.62.3.11 `derivatives()`

```
const paramset & GiNaC::fderivative::derivatives ( ) const
```

Expose this object's derivative structure.

Parameter numbers occurring more than once stand for repeated differentiation with respect to that parameter. If a symbolic function  $f(x,y)$  is differentiated with respect to  $x$ , this method will return  $\{0\}$ . If  $f(x,y)$  is differentiated twice with respect to  $y$ , it will return  $\{1,1\}$ . (This corresponds to the way this object is printed.)

Returns

multiset of function's parameter numbers that are abstractly differentiated.

References [parameter\\_set](#).

### 6.62.3.12 do\_print()

```
void GiNaC::fderivative::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::container< C >::end\(\)](#), [parameter\\_set](#), [GiNaC::container< C >::precedence\(\)](#), [GiNaC::function::precedence\(\)](#), [GiNaC::container< C >::printseq\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), and [GiNaC::function::serial](#).

### 6.62.3.13 do\_print\_latex()

```
void GiNaC::fderivative::do_print_latex (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::container< C >::end\(\)](#), [order](#), [parameter\\_set](#), [GiNaC::container< C >::precedence\(\)](#), [GiNaC::function::precedence\(\)](#), [GiNaC::container< C >::printseq\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), and [GiNaC::function::serial](#).

### 6.62.3.14 do\_print\_csrc()

```
void GiNaC::fderivative::do_print_csrc (
    const print\_csrc & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::container< C >::end\(\)](#), [parameter\\_set](#), [GiNaC::container< C >::precedence\(\)](#), [GiNaC::function::precedence\(\)](#), [GiNaC::container< C >::printseq\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), and [GiNaC::function::serial](#).

### 6.62.3.15 do\_print\_tree()

```
void GiNaC::fderivative::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::container< C >::nops\(\)](#), [parameter\\_set](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

## 6.62.4 Member Data Documentation

#### 6.62.4.1 parameter\_set

```
paramset GiNaC::fderivative::parameter_set [protected]
```

Set of parameter numbers with respect to which to take the derivative.

Referenced by [archive\(\)](#), [derivative\(\)](#), [derivatives\(\)](#), [do\\_print\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [eval\(\)](#), [fderivative\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), and [thiscontainer\(\)](#).

The documentation for this class was generated from the following files:

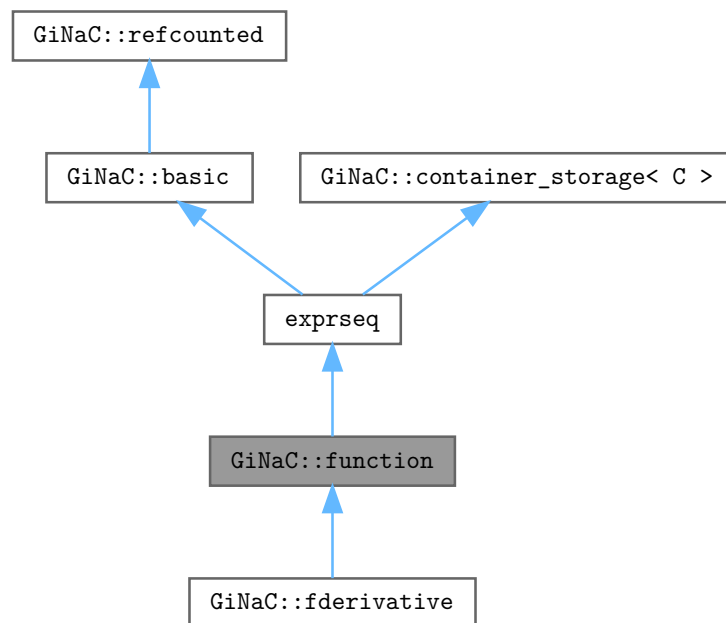
- [fderivative.h](#)
- [fderivative.cpp](#)

### 6.63 GiNaC::function Class Reference

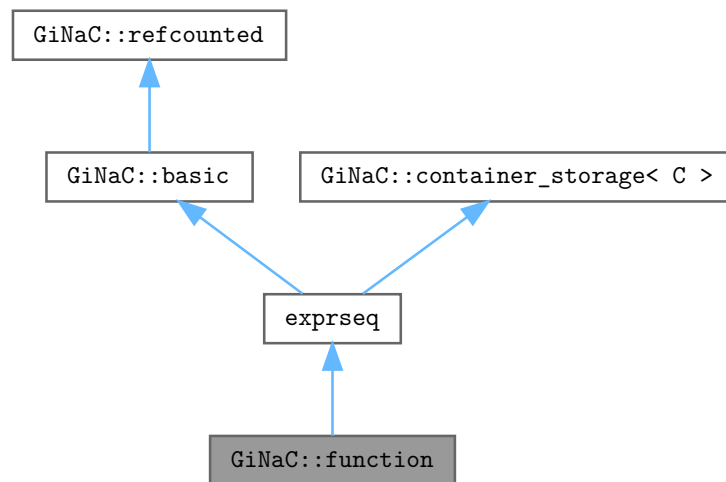
The class function is used to implement builtin functions like sin, cos... and user defined functions.

```
#include <function.h>
```

Inheritance diagram for GiNaC::function:



Collaboration diagram for GiNaC::function:



## Public Member Functions

- [function](#) (unsigned ser)
- [function](#) (unsigned ser, const [ex](#) &param1)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13, const [ex](#) &param14)

- [function](#) (unsigned ser, const [exprseq](#) &es)
- [function](#) (unsigned ser, const [exvector](#) &v)
- [function](#) (unsigned ser, [exvector](#) &&v)
- void [print](#) (const [print\\_context](#) &c, unsigned level=0) const override  
*Output to stream.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex eval\\_ncmul](#) (const [exvector](#) &v) const override  
*This method is defined to be in line with behavior of [function::return\\_type\(\)](#)*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series](#) for functions.*
- [ex thiscontainer](#) (const [exvector](#) &v) const override
- [ex thiscontainer](#) ([exvector](#) &&v) const override
- [ex conjugate](#) () const override  
*Implementation of [ex::conjugate](#) for functions.*
- [ex real\\_part](#) () const override  
*Implementation of [ex::real\\_part](#) for functions.*
- [ex imag\\_part](#) () const override  
*Implementation of [ex::imag\\_part](#) for functions.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Archive the object.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Construct object from [archive\\_node](#).*
- bool [info](#) (unsigned inf) const override  
*Implementation of [ex::info](#) for functions.*
- [ex power](#) (const [ex](#) &exp) const
- unsigned [get\\_serial](#) () const
- std::string [get\\_name](#) () const  
*Return the print name of the function.*

#### Public Member Functions inherited from [GiNaC::container](#)< [C](#) >

- [container](#) (STLT const &s)
- [container](#) (STLT &&v)
- [container](#) (exvector::const\_iterator b, exvector::const\_iterator e)
- [container](#) (std::initializer\_list< [ex](#) > il)
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- size\_t [nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override

- *Return operand/member at position i.*
- `ex & let_op (size_t i)` override
- *Return modifiable operand/member at position i.*
- `ex subs (const exmap &m, unsigned options=0)` const override
- *Substitute a set of objects by arbitrary expressions.*
- `void read_archive (const archive_node &n, lst &sym_lst)` override
- *Load (deserialize) the object from an archive node.*
- `void archive (archive_node &n)` const override
- *Archive the object.*
- `container & prepend (const ex &b)`
- *Add element at front.*
- `container & append (const ex &b)`
- *Add element at back.*
- `container & remove_first ()`
- *Remove first element.*
- `container & remove_last ()`
- *Remove last element.*
- `container & remove_all ()`
- *Remove all elements.*
- `container & sort ()`
- *Sort elements.*
- `container & unique ()`
- *Remove adjacent duplicate elements.*
- `const_iterator begin ()` const
- `const_iterator end ()` const
- `const_reverse_iterator rbegin ()` const
- `const_reverse_iterator rend ()` const

#### Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic ()`
- *basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`
- *basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate ()` const
- *Create a clone of this object on the heap.*
- `virtual ex eval ()` const
- *Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf ()` const
- *Evaluate object numerically.*
- `virtual ex evalm ()` const
- *Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ ()` const
- *Evaluate integrals, if result is known.*
- `virtual ex eval_indexed (const basic &i)` const
- *Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print (const print_context &c, unsigned level=0)` const
- *Output to stream.*
- `virtual void dbgprint ()` const

- Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const
- Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const
- Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const
- Information about the object.*
- virtual size\_t `nops` () const
- Number of operands/members.*
- virtual `ex op` (size\_t i) const
- Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)
- Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const
- Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
- Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const



- *Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- *Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- *Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- *Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
- void `print_dispatch` (const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const
- *Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- *Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- *Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- *Test for syntactic equality.*
- const `basic` & `hold` () const
- *Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- *Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const
- *Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Static Public Member Functions

- static unsigned `register_new` (`function_options` const &opt)
- static unsigned `find_function` (const std::string &name, unsigned nparams)
- *Find serial number of function by name and number of parameters.*
- static std::vector< `function_options` > `get_registered_functions` ()

## Static Public Attributes

- static unsigned `current_serial` = 0

*This can be used as a hook for external applications.*

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for functions.*
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- `ex pderivative` (unsigned `diff_param`) const
- `ex expl_derivative` (const `symbol` &s) const
- bool `lookup_remember_table` (ex &result) const
- void `store_remember_table` (ex const &result) const

## Protected Member Functions inherited from `GiNaC::container< C >`

- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- virtual `ex thiscontainer` (const `STLT` &v) const  
*Similar to `duplicate()`, but with a preset sequence.*
- virtual `ex thiscontainer` (`STLT` &&v) const  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- virtual void `printseq` (const `print_context` &c, char openbracket, char delim, char closebracket, unsigned `this_precedence`, unsigned `upper_precedence`=0) const  
*Print sequence of contained elements.*
- void `do_print` (const `print_context` &c, unsigned `level`) const
- void `do_print_tree` (const `print_tree` &c, unsigned `level`) const
- void `do_print_python` (const `print_python` &c, unsigned `level`) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned `level`) const
- `STLT` `subschildren` (const `exmap` &m, unsigned `options`=0) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*

- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

#### Protected Member Functions inherited from `GiNaC::container_storage< C >`

- `container_storage` ()
- `container_storage` (size\_t n, const `ex` &e)
- `container_storage` (std::initializer\_list< `ex` > il)
- template<class In >  
  `container_storage` (In b, In e)
- void `reserve` (size\_t)
- `~container_storage` ()
- void `reserve` (size\_t n)
- void `reserve` (std::vector< `ex` > &v, size\_t n)

#### Static Protected Member Functions

- static std::vector< `function_options` > &`registered_functions` ()

#### Static Protected Member Functions inherited from `GiNaC::container< C >`

- static unsigned `get_default_flags` ()  
*Specialization of `container::get_default_flags()` for `lst`.*
- static char `get_open_delim` ()  
*Specialization of `container::get_open_delim()` for `lst`.*
- static char `get_close_delim` ()  
*Specialization of `container::get_close_delim()` for `lst`.*

#### Static Protected Member Functions inherited from `GiNaC::container_storage< C >`

- static void `reserve` (STLT &, size\_t)

#### Protected Attributes

- unsigned `serial`

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

**Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)**

- [STLT seq](#)

**Friends**

- class [remember\\_table\\_entry](#)

**Additional Inherited Members****Public Types inherited from [GiNaC::container< C >](#)**

- typedef [STLT::const\\_iterator](#) [const\\_iterator](#)
- typedef [STLT::const\\_reverse\\_iterator](#) [const\\_reverse\\_iterator](#)

**Protected Types inherited from [GiNaC::container< C >](#)**

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

**Protected Types inherited from [GiNaC::container\\_storage< C >](#)**

- typedef [C< ex >](#) [STLT](#)

**6.63.1 Detailed Description**

The class function is used to implement builtin functions like sin, cos... and user defined functions.

**6.63.2 Constructor & Destructor Documentation****6.63.2.1 [function\(\)](#) [1/18]**

```
GiNaC::function::function (
    unsigned ser )
```

**6.63.2.2 function() [2/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1 )
```

**6.63.2.3 function() [3/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2 )
```

**6.63.2.4 function() [4/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3 )
```

**6.63.2.5 function() [5/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4 )
```

**6.63.2.6 function() [6/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5 )
```

### 6.63.2.7 function() [7/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6 )
```

### 6.63.2.8 function() [8/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7 )
```

### 6.63.2.9 function() [9/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8 )
```

### 6.63.2.10 function() [10/18]

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9 )
```

**6.63.2.11 function() [11/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10 )
```

**6.63.2.12 function() [12/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11 )
```

**6.63.2.13 function() [13/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11,
    const ex & param12 )
```

**6.63.2.14 function() [14/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11,
    const ex & param12,
    const ex & param13 )
```

**6.63.2.15 function() [15/18]**

```
GiNaC::function::function (
    unsigned ser,
    const ex & param1,
    const ex & param2,
    const ex & param3,
    const ex & param4,
    const ex & param5,
    const ex & param6,
    const ex & param7,
    const ex & param8,
    const ex & param9,
    const ex & param10,
    const ex & param11,
    const ex & param12,
    const ex & param13,
    const ex & param14 )
```

**6.63.2.16 function() [16/18]**

```
GiNaC::function::function (
    unsigned ser,
    const exprseq & es )
```

References [GiNaC::basic::clearflag\(\)](#), and [GiNaC::status\\_flags::evaluated](#).



**6.63.2.17 function()** [17/18]

```
GiNaC::function::function (
    unsigned ser,
    const exvector & v )
```

**6.63.2.18 function()** [18/18]

```
GiNaC::function::function (
    unsigned ser,
    exvector && v )
```

**6.63.3 Member Function Documentation****6.63.3.1 print()**

```
void GiNaC::function::print (
    const print\_context & c,
    unsigned level = 0 ) const [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

**Parameters**

<i>c</i>	print context object that describes the output formatting
<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented from [GiNaC::basic](#).

References [c](#), [current\\_serial](#), [GiNaC::basic::flags](#), [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hashvalue](#), [GiNaC::function\\_options::name](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::function\\_options::nparams](#), [GiNaC::class\\_info< OPT >::options](#), [GiNaC::container< C >::precedence\(\)](#), [precedence\(\)](#), [print\(\)](#), [GiNaC::function\\_options::print\\_di](#), [GiNaC::function\\_options::print\\_use\\_exvector\\_args](#), [GiNaC::container< C >::printseq\(\)](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::function\\_options::TeX\\_name](#).

Referenced by [print\(\)](#).

**6.63.3.2 precedence()**

```
unsigned GiNaC::function::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< C >](#).

Referenced by [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), and [print\(\)](#).

### 6.63.3.3 expand()

```
ex GiNaC::function::expand (
    unsigned options = 0 ) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GiNaC::function\\_options::expand\\_f](#), [GiNaC::expand\\_options::expand\\_function\\_args](#), [GiNaC::function\\_options::expand\\_use\\_exvector\\_args](#), [GiNaC::status\\_flags::expanded](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

### 6.63.3.4 eval()

```
ex GiNaC::function::eval ( ) const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::canonicalize\(\)](#), [current\\_serial](#), [GiNaC::function\\_options::eval\\_f](#), [GiNaC::function\\_options::eval\\_use\\_exvector\\_args](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lookup\\_remember\\_table\(\)](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), [store\\_remember\\_table\(\)](#), [GiNaC::function\\_options::symtree](#), [thiscontainer\(\)](#), and [GiNaC::function\\_options::use\\_remember](#).

### 6.63.3.5 evalf()

```
ex GiNaC::function::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GiNaC::function\\_options::evalf\\_f](#), [GiNaC::function\\_options::evalf\\_params\\_first](#), [GiNaC::function\\_options::evalf\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

Referenced by [GiNaC::iterated\\_integral2\\_eval\(\)](#), and [GiNaC::iterated\\_integral3\\_eval\(\)](#).

### 6.63.3.6 eval\_ncmul()

```
ex GiNaC::function::eval_ncmul (
    const exvector & v ) const [override], [virtual]
```

This method is defined to be in line with behavior of [function::return\\_type\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container\\_storage< C >::seq](#).

### 6.63.3.7 calchash()

```
unsigned GiNaC::function::calchash ( ) const [override], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::rotate\\_left\(\)](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

### 6.63.3.8 series()

```
ex GiNaC::function::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series](#) for functions.

@see [ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [options](#), [order](#), [r](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), [GiNaC::basic::series\(\)](#), [GiNaC::function\\_options::series\\_f](#), and [GiNaC::function\\_options::series\\_use\\_exvector\\_args](#).

**6.63.3.9 thiscontainer() [1/2]**

```
ex GiNaC::function::thiscontainer (
    const exvector & v ) const [override]
```

References [serial](#).

Referenced by [eval\(\)](#).

**6.63.3.10 thiscontainer() [2/2]**

```
ex GiNaC::function::thiscontainer (
    exvector && v ) const [override]
```

References [serial](#).

**6.63.3.11 conjugate()**

```
ex GiNaC::function::conjugate ( ) const [override], [virtual]
```

Implementation of [ex::conjugate](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::function\\_options::conjugate\\_f](#), [GiNaC::function\\_options::conjugate\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**6.63.3.12 real\_part()**

```
ex GiNaC::function::real_part ( ) const [override], [virtual]
```

Implementation of [ex::real\\_part](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [GiNaC::basic::real\\_part\(\)](#), [GiNaC::function\\_options::real\\_part\\_f](#), [GiNaC::function\\_options::real\\_part\\_use\\_exvector\\_args](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**6.63.3.13 imag\_part()**

```
ex GiNaC::function::imag_part ( ) const [override], [virtual]
```

Implementation of [ex::imag\\_part](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [GiNaC::basic::imag\\_part\(\)](#), [GiNaC::function\\_options::imag\\_part\\_f](#), [GiNaC::function\\_options::imag\\_part](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**6.63.3.14 archive()**

```
void GiNaC::function::archive (
    archive_node & n ) const [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [n](#), [registered\\_functions\(\)](#), and [serial](#).

**6.63.3.15 read\_archive()**

```
void GiNaC::function::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Construct object from [archive\\_node](#).

Reimplemented from [GiNaC::container< C >](#).

References [n](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**6.63.3.16 info()**

```
bool GiNaC::function::info (
    unsigned inf ) const [override], [virtual]
```

Implementation of [ex::info](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [GiNaC::basic::info\(\)](#), [GiNaC::function\\_options::info\\_f](#), [GiNaC::function\\_options::info\\_use\\_exvector\\_arg](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

### 6.63.3.17 derivative()

```
ex GiNaC::function::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for functions.

It applies the chain rule, except for the Order term function. @see [ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [expl\\_derivative\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pderivative\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

### 6.63.3.18 is\_equal\_same\_type()

```
bool GiNaC::function::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [GiNaC::container< C >::is\\_equal\\_same\\_type\(\)](#), and [serial](#).

### 6.63.3.19 match\_same\_type()

```
bool GiNaC::function::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [serial](#).

**6.63.3.20 return\_type()**

```
unsigned GiNaC::function::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GINAC\\_ASSERT](#), [registered\\_functions\(\)](#), [GiNaC::function\\_options::return\\_type](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::function\\_options::use\\_return\\_type](#).

**6.63.3.21 return\_type\_tinfo()**

```
return_type_t GiNaC::function::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [registered\\_functions\(\)](#), [GiNaC::function\\_options::return\\_type\\_tinfo](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::function\\_options::use\\_return\\_type](#).

**6.63.3.22 pderivative()**

```
ex GiNaC::function::pderivative (
    unsigned diff_param ) const [protected]
```

References [GiNaC::function\\_options::derivative\\_f](#), [GiNaC::function\\_options::derivative\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), and [GiNaC::function\\_options::nparams](#).

Referenced by [derivative\(\)](#), and [GiNaC::fderivative::eval\(\)](#).

**6.63.3.23 expl\_derivative()**

```
ex GiNaC::function::expl_derivative (
    const symbol & s ) const [protected]
```

References [GiNaC::function\\_options::expl\\_derivative\\_f](#), [GiNaC::function\\_options::expl\\_derivative\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), and [GiNaC::function\\_options::nparams](#).

Referenced by [derivative\(\)](#).

### 6.63.3.24 registered\_functions()

```
std::vector< function\_options > & GiNaC::function::registered_functions ( ) [static], [protected]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::eval\(\)](#), [eval\(\)](#), [evalf\(\)](#), [expand\(\)](#), [find\\_function\(\)](#), [get\\_name\(\)](#), [get\\_registered\\_functions\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [print\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [register\\_new\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), and [series\(\)](#).

### 6.63.3.25 lookup\_remember\_table()

```
bool GiNaC::function::lookup_remember_table (
    ex & result ) const [protected]
```

References [GiNaC::remember\\_table::remember\\_tables\(\)](#), and [serial](#).

Referenced by [eval\(\)](#).

### 6.63.3.26 store\_remember\_table()

```
void GiNaC::function::store_remember_table (
    ex const & result ) const [protected]
```

References [GiNaC::remember\\_table::remember\\_tables\(\)](#), and [serial](#).

Referenced by [eval\(\)](#).

### 6.63.3.27 power()

```
ex GiNaC::function::power (
    const ex & exp ) const
```

References [GiNaC::status\\_flags::evaluated](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [GiNaC::function\\_options::power\\_f](#), and [GiNaC::function\\_options::power\\_use\\_exvector\\_args](#).

### 6.63.3.28 register\_new()

```
unsigned GiNaC::function::register_new (
    function\_options const & opt ) [static]
```

References [GiNaC::function\\_options::functions\\_with\\_same\\_name](#), [GiNaC::function\\_options::name](#), [registered\\_functions\(\)](#), [GiNaC::function\\_options::remember\\_assoc\\_size](#), [GiNaC::function\\_options::remember\\_size](#), [GiNaC::function\\_options::remember\\_str](#), [GiNaC::remember\\_table::remember\\_tables\(\)](#), and [GiNaC::function\\_options::use\\_remember](#).



### 6.63.3.29 find\_function()

```
unsigned GiNaC::function::find_function (
    const std::string & name,
    unsigned nparams ) [static]
```

Find serial number of function by name and number of parameters.

Throws exception if function was not found.

References [registered\\_functions\(\)](#), and [serial](#).

### 6.63.3.30 get\_registered\_functions()

```
static std::vector< function_options > GiNaC::function::get_registered_functions ( ) [inline],
[static]
```

References [registered\\_functions\(\)](#).

### 6.63.3.31 get\_serial()

```
unsigned GiNaC::function::get_serial ( ) const [inline]
```

References [serial](#).

### 6.63.3.32 get\_name()

```
std::string GiNaC::function::get_name ( ) const
```

Return the print name of the function.

References [GINAC\\_ASSERT](#), [registered\\_functions\(\)](#), and [serial](#).

## 6.63.4 Friends And Related Function Documentation

### 6.63.4.1 remember\_table\_entry

```
friend class remember_table_entry [friend]
```

## 6.63.5 Member Data Documentation

### 6.63.5.1 `current_serial`

```
unsigned GiNaC::function::current_serial = 0 [static]
```

This can be used as a hook for external applications.

Referenced by [eval\(\)](#), [evalf\(\)](#), [expand\(\)](#), [print\(\)](#), and [series\(\)](#).

### 6.63.5.2 `serial`

```
unsigned GiNaC::function::serial [protected]
```

Referenced by [archive\(\)](#), [calchash\(\)](#), [conjugate\(\)](#), [GiNaC::fderivative::derivative\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::eval\(\)](#), [eval\(\)](#), [evalf\(\)](#), [expand\(\)](#), [find\\_function\(\)](#), [get\\_name\(\)](#), [get\\_serial\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [lookup\\_remember\\_table\(\)](#), [match\\_same\\_type\(\)](#), [print\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [series\(\)](#), [store\\_remember\\_table\(\)](#), [GiNaC::fderivative::thiscontainer\(\)](#), and [thiscontainer\(\)](#).

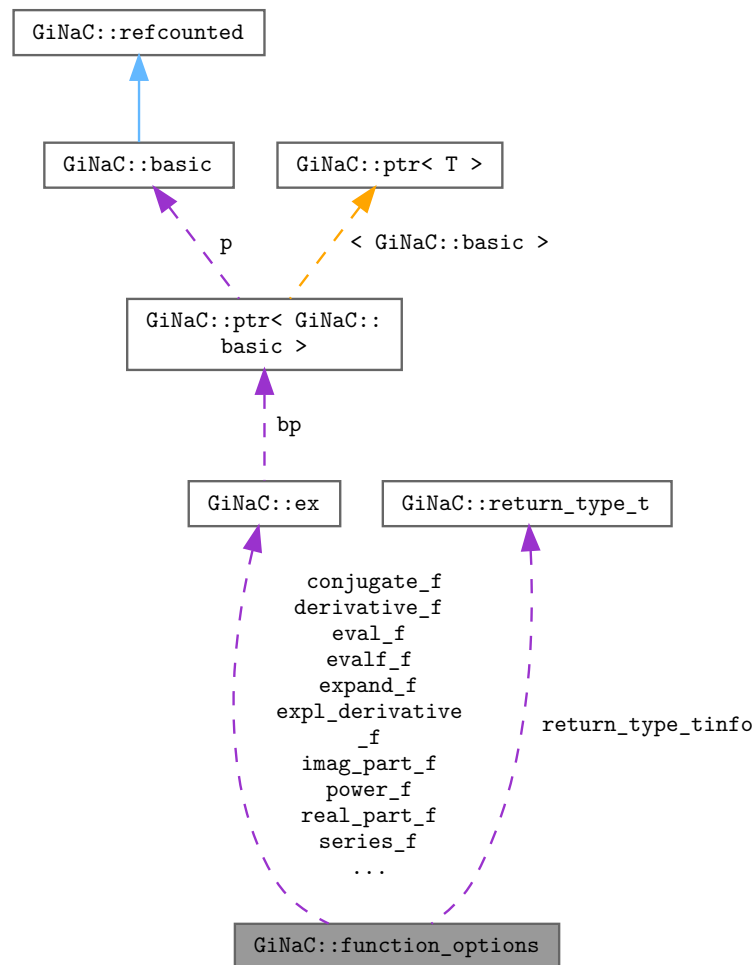
The documentation for this class was generated from the following files:

- [function.h](#)
- [function.cpp](#)

## 6.64 `GiNaC::function_options` Class Reference

```
#include <function.h>
```

Collaboration diagram for GiNaC::function\_options:



## Public Member Functions

- [function\\_options](#) ()
- [function\\_options](#) (std::string const &n, std::string const &tn=std::string())
- [function\\_options](#) (std::string const &n, unsigned np)
- [~function\\_options](#) ()
- void [initialize](#) ()
- [function\\_options](#) & [dummy](#) ()
- [function\\_options](#) & [set\\_name](#) (std::string const &n, std::string const &tn=std::string())
- [function\\_options](#) & [latex\\_name](#) (std::string const &tn)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_1 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_2 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_3 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_4 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_5 e)
- [function\\_options](#) & [eval\\_func](#) (eval\_funcp\_6 e)



- Generated by Doxygen

- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_5](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_6](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_7](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_8](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_9](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_10](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_11](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_12](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_13](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_14](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_1](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_2](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_3](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_4](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_5](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_6](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_7](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_8](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_9](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_10](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_11](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_12](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_13](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_14](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_1](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_2](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_3](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_4](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_5](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_6](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_7](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_8](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_9](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_10](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_11](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_12](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_13](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_14](#) e)
- [function\\_options](#) & [eval\\_func](#) ([eval\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [evalf\\_func](#) ([evalf\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [conjugate\\_func](#) ([conjugate\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [real\\_part\\_func](#) ([real\\_part\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [imag\\_part\\_func](#) ([imag\\_part\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [expand\\_func](#) ([expand\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [derivative\\_func](#) ([derivative\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [expl\\_derivative\\_func](#) ([expl\\_derivative\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [power\\_func](#) ([power\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [series\\_func](#) ([series\\_funcp\\_exvector](#) e)
- [function\\_options](#) & [info\\_func](#) ([info\\_funcp\\_exvector](#) e)
- [template<class Ctx >](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_1](#) p)
- [template<class Ctx >](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_2](#) p)
- [template<class Ctx >](#)  
[function\\_options](#) & [print\\_func](#) ([print\\_funcp\\_3](#) p)

- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_4` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_5` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_6` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_7` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_8` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_9` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_10` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_11` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_12` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_13` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_14` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_exvector` p)
- `function_options` & `set_return_type` (unsigned rt, const `return_type_t` \*rtt=nullptr)
- `function_options` & `do_not_evalf_params` ()
- `function_options` & `remember` (unsigned size, unsigned assoc\_size=0, unsigned strategy=`remember_strategies::delete_never`)
- `function_options` & `overloaded` (unsigned o)
- `function_options` & `set_symmetry` (const `symmetry` &s)
- `std::string` `get_name` () const
- unsigned `get_nparams` () const

## Protected Member Functions

- bool `has_derivative` () const
- bool `has_power` () const
- void `test_and_set_nparams` (unsigned n)
- void `set_print_func` (unsigned id, `print_funcp` f)

## Protected Attributes

- `std::string` `name`
- `std::string` `TeX_name`
- unsigned `nparams`
- `eval_funcp` `eval_f`
- `evalf_funcp` `evalf_f`
- `conjugate_funcp` `conjugate_f`
- `real_part_funcp` `real_part_f`
- `imag_part_funcp` `imag_part_f`
- `expand_funcp` `expand_f`
- `derivative_funcp` `derivative_f`
- `expl_derivative_funcp` `expl_derivative_f`
- `power_funcp` `power_f`

- [series\\_funcp series\\_f](#)
- [std::vector< \[print\\\_funcp\]\(#\) > \[print\\\_dispatch\\\_table\]\(#\)](#)
- [info\\_funcp info\\_f](#)
- [bool evalf\\_params\\_first](#)
- [bool use\\_return\\_type](#)
- [unsigned return\\_type](#)
- [return\\_type\\_t return\\_type\\_tinfo](#)
- [bool use\\_remember](#)
- [unsigned remember\\_size](#)
- [unsigned remember\\_assoc\\_size](#)
- [unsigned remember\\_strategy](#)
- [bool eval\\_use\\_exvector\\_args](#)
- [bool evalf\\_use\\_exvector\\_args](#)
- [bool conjugate\\_use\\_exvector\\_args](#)
- [bool real\\_part\\_use\\_exvector\\_args](#)
- [bool imag\\_part\\_use\\_exvector\\_args](#)
- [bool expand\\_use\\_exvector\\_args](#)
- [bool derivative\\_use\\_exvector\\_args](#)
- [bool expl\\_derivative\\_use\\_exvector\\_args](#)
- [bool power\\_use\\_exvector\\_args](#)
- [bool series\\_use\\_exvector\\_args](#)
- [bool print\\_use\\_exvector\\_args](#)
- [bool info\\_use\\_exvector\\_args](#)
- [unsigned functions\\_with\\_same\\_name](#)
- [ex symtree](#)

## Friends

- [class function](#)
- [class fderivative](#)

## 6.64.1 Constructor & Destructor Documentation

### 6.64.1.1 [function\\_options\(\)](#) [1/3]

`GiNaC::function_options::function_options ( )`

References [initialize\(\)](#).

### 6.64.1.2 [function\\_options\(\)](#) [2/3]

`GiNaC::function_options::function_options (`  
     `std::string const & n,`  
     `std::string const & tn = std::string() )`

References [initialize\(\)](#), [n](#), and [set\\_name\(\)](#).



**6.64.1.3 function\_options() [3/3]**

```
GiNaC::function_options::function_options (
    std::string const & n,
    unsigned np )
```

References [initialize\(\)](#), [n](#), [nparams](#), and [set\\_name\(\)](#).

**6.64.1.4 ~function\_options()**

```
GiNaC::function_options::~~function_options ( )
```

**6.64.2 Member Function Documentation****6.64.2.1 initialize()**

```
void GiNaC::function_options::initialize ( )
```

References [conjugate\\_f](#), [conjugate\\_use\\_exvector\\_args](#), [derivative\\_f](#), [derivative\\_use\\_exvector\\_args](#), [eval\\_f](#), [eval\\_use\\_exvector\\_args](#), [evalf\\_f](#), [evalf\\_params\\_first](#), [evalf\\_use\\_exvector\\_args](#), [expand\\_f](#), [expand\\_use\\_exvector\\_args](#), [expl\\_derivative\\_f](#), [expl\\_derivative\\_use\\_exvector\\_args](#), [functions\\_with\\_same\\_name](#), [imag\\_part\\_f](#), [imag\\_part\\_use\\_exvector\\_args](#), [info\\_f](#), [info\\_use\\_exvector\\_args](#), [nparams](#), [power\\_f](#), [power\\_use\\_exvector\\_args](#), [print\\_use\\_exvector\\_args](#), [real\\_part\\_f](#), [real\\_part\\_use\\_exvector\\_args](#), [series\\_f](#), [series\\_use\\_exvector\\_args](#), [set\\_name\(\)](#), [symtree](#), [use\\_remember](#), and [use\\_return\\_type](#).

Referenced by [function\\_options\(\)](#).

**6.64.2.2 dummy()**

```
function\_options & GiNaC::function_options::dummy ( ) [inline]
```

**6.64.2.3 set\_name()**

```
function\_options & GiNaC::function_options::set_name (
    std::string const & n,
    std::string const & tn = std::string() )
```

References [n](#), [name](#), and [TeX\\_name](#).

Referenced by [function\\_options\(\)](#), and [initialize\(\)](#).

#### 6.64.2.4 latex\_name()

```
function_options & GiNaC::function_options::latex_name (
    std::string const & tn )
```

References [TeX\\_name](#).

#### 6.64.2.5 eval\_func() [1/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_1 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

#### 6.64.2.6 eval\_func() [2/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_2 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

#### 6.64.2.7 eval\_func() [3/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_3 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

#### 6.64.2.8 eval\_func() [4/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_4 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

#### 6.64.2.9 eval\_func() [5/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_5 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.10 eval\_func()** [6/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_6 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.11 eval\_func()** [7/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_7 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.12 eval\_func()** [8/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_8 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.13 eval\_func()** [9/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_9 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.14 eval\_func()** [10/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_10 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.15 eval\_func()** [11/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_11 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.16 eval\_func()** [12/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_12 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.17 eval\_func()** [13/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_13 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.18 eval\_func()** [14/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_14 e )
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.19 evalf\_func()** [1/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_1 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.20 evalf\_func()** [2/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_2 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.21 evalf\_func()** [3/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_3 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.22 evalf\_func()** [4/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_4 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.23 evalf\_func()** [5/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_5 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.24 evalf\_func()** [6/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_6 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.25 evalf\_func()** [7/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_7 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.26 evalf\_func()** [8/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_8 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.27 evalf\_func()** [9/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_9 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.28 evalf\_func()** [10/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_10 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.29 evalf\_func()** [11/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_11 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.30 evalf\_func()** [12/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_12 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.31 evalf\_func()** [13/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_13 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.32 evalf\_func()** [14/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_14 e )
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.33 conjugate\_func()** [1/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_1 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.34 conjugate\_func()** [2/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_2 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.35 conjugate\_func()** [3/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_3 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.36 conjugate\_func()** [4/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_4 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.37 conjugate\_func()** [5/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_5 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.38 conjugate\_func()** [6/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_6 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.39 conjugate\_func()** [7/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_7 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.40 conjugate\_func()** [8/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_8 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.41 conjugate\_func()** [9/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_9 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.42 conjugate\_func()** [10/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_10 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.43 conjugate\_func()** [11/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_11 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.44 conjugate\_func()** [12/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_12 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.45 conjugate\_func()** [13/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_13 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).



**6.64.2.46 conjugate\_func()** [14/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_14 e )
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.47 real\_part\_func()** [1/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_1 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.48 real\_part\_func()** [2/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_2 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.49 real\_part\_func()** [3/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_3 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.50 real\_part\_func()** [4/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_4 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.51 real\_part\_func()** [5/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_5 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.52 real\_part\_func()** [6/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_6 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.53 real\_part\_func()** [7/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_7 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.54 real\_part\_func()** [8/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_8 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.55 real\_part\_func()** [9/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_9 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.56 real\_part\_func()** [10/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_10 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.57 real\_part\_func()** [11/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_11 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.58 real\_part\_func()** [12/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_12 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.59 real\_part\_func()** [13/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_13 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.60 real\_part\_func()** [14/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_14 e )
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.61 imag\_part\_func()** [1/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_1 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.62 imag\_part\_func()** [2/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_2 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.63 imag\_part\_func()** [3/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_3 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.64 imag\_part\_func() [4/15]**

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_4 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.65 imag\_part\_func() [5/15]**

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_5 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.66 imag\_part\_func() [6/15]**

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_6 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.67 imag\_part\_func() [7/15]**

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_7 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.68 imag\_part\_func() [8/15]**

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_8 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.69 imag\_part\_func() [9/15]**

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_9 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.70 imag\_part\_func()** [10/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_10 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.71 imag\_part\_func()** [11/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_11 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.72 imag\_part\_func()** [12/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_12 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.73 imag\_part\_func()** [13/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_13 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.74 imag\_part\_func()** [14/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_14 e )
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.75 expand\_func()** [1/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_1 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.76 expand\_func()** [2/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_2 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.77 expand\_func()** [3/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_3 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.78 expand\_func()** [4/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_4 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.79 expand\_func()** [5/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_5 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.80 expand\_func()** [6/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_6 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.81 expand\_func()** [7/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_7 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.82 expand\_func()** [8/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_8 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.83 expand\_func()** [9/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_9 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.84 expand\_func()** [10/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_10 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.85 expand\_func()** [11/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_11 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.86 expand\_func()** [12/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_12 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.87 expand\_func()** [13/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_13 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.88 expand\_func()** [14/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_14 e )
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.89 derivative\_func()** [1/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_1 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.90 derivative\_func()** [2/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_2 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.91 derivative\_func()** [3/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_3 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.92 derivative\_func()** [4/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_4 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.93 derivative\_func()** [5/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_5 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).



**6.64.2.94 derivative\_func()** [6/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_6 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.95 derivative\_func()** [7/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_7 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.96 derivative\_func()** [8/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_8 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.97 derivative\_func()** [9/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_9 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.98 derivative\_func()** [10/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_10 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.99 derivative\_func()** [11/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_11 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.100 derivative\_func()** [12/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_12 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.101 derivative\_func()** [13/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_13 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.102 derivative\_func()** [14/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_14 e )
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.103 expl\_derivative\_func()** [1/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_1 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.104 expl\_derivative\_func()** [2/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_2 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.105 expl\_derivative\_func()** [3/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_3 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.106 expl\_derivative\_func() [4/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_4 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.107 expl\_derivative\_func() [5/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_5 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.108 expl\_derivative\_func() [6/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_6 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.109 expl\_derivative\_func() [7/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_7 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.110 expl\_derivative\_func() [8/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_8 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.111 expl\_derivative\_func() [9/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_9 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.112 expl\_derivative\_func() [10/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_10 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.113 expl\_derivative\_func() [11/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_11 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.114 expl\_derivative\_func() [12/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_12 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.115 expl\_derivative\_func() [13/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_13 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.116 expl\_derivative\_func() [14/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_14 e )
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.117 power\_func() [1/15]**

```
function_options & GiNaC::function_options::power_func (
    power_funcp_1 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.118 power\_func() [2/15]**

```
function_options & GiNaC::function_options::power_func (
    power_funcp_2 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.119 power\_func() [3/15]**

```
function_options & GiNaC::function_options::power_func (
    power_funcp_3 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.120 power\_func() [4/15]**

```
function_options & GiNaC::function_options::power_func (
    power_funcp_4 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.121 power\_func() [5/15]**

```
function_options & GiNaC::function_options::power_func (
    power_funcp_5 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.122 power\_func() [6/15]**

```
function_options & GiNaC::function_options::power_func (
    power_funcp_6 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.123 power\_func() [7/15]**

```
function_options & GiNaC::function_options::power_func (
    power_funcp_7 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.124 power\_func()** [8/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_8 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.125 power\_func()** [9/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_9 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.126 power\_func()** [10/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_10 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.127 power\_func()** [11/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_11 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.128 power\_func()** [12/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_12 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.129 power\_func()** [13/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_13 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.130 power\_func()** [14/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_14 e )
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.131 series\_func()** [1/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_1 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.132 series\_func()** [2/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_2 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.133 series\_func()** [3/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_3 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.134 series\_func()** [4/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_4 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.135 series\_func()** [5/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_5 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.136 series\_func()** [6/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_6 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.137 series\_func()** [7/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_7 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.138 series\_func()** [8/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_8 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.139 series\_func()** [9/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_9 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.140 series\_func()** [10/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_10 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.141 series\_func()** [11/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_11 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).



**6.64.2.142 series\_func()** [12/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_12 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.143 series\_func()** [13/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_13 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.144 series\_func()** [14/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_14 e )
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.145 info\_func()** [1/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_1 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.146 info\_func()** [2/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_2 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.147 info\_func()** [3/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_3 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.148 info\_func()** [4/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_4 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.149 info\_func()** [5/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_5 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.150 info\_func()** [6/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_6 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.151 info\_func()** [7/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_7 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.152 info\_func()** [8/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_8 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.153 info\_func()** [9/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_9 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.154 info\_func()** [10/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_10 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.155 info\_func()** [11/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_11 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.156 info\_func()** [12/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_12 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.157 info\_func()** [13/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_13 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.158 info\_func()** [14/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_14 e )
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.159 eval\_func()** [15/15]

```
function_options & GiNaC::function_options::eval_func (
    eval_funcp_exvector e )
```

References [eval\\_f](#), and [eval\\_use\\_exvector\\_args](#).

**6.64.2.160 evalf\_func()** [15/15]

```
function_options & GiNaC::function_options::evalf_func (
    evalf_funcp_exvector e )
```

References [evalf\\_f](#), and [evalf\\_use\\_exvector\\_args](#).

**6.64.2.161 conjugate\_func()** [15/15]

```
function_options & GiNaC::function_options::conjugate_func (
    conjugate_funcp_exvector e )
```

References [conjugate\\_f](#), and [conjugate\\_use\\_exvector\\_args](#).

**6.64.2.162 real\_part\_func()** [15/15]

```
function_options & GiNaC::function_options::real_part_func (
    real_part_funcp_exvector e )
```

References [real\\_part\\_f](#), and [real\\_part\\_use\\_exvector\\_args](#).

**6.64.2.163 imag\_part\_func()** [15/15]

```
function_options & GiNaC::function_options::imag_part_func (
    imag_part_funcp_exvector e )
```

References [imag\\_part\\_f](#), and [imag\\_part\\_use\\_exvector\\_args](#).

**6.64.2.164 expand\_func()** [15/15]

```
function_options & GiNaC::function_options::expand_func (
    expand_funcp_exvector e )
```

References [expand\\_f](#), and [expand\\_use\\_exvector\\_args](#).

**6.64.2.165 derivative\_func()** [15/15]

```
function_options & GiNaC::function_options::derivative_func (
    derivative_funcp_exvector e )
```

References [derivative\\_f](#), and [derivative\\_use\\_exvector\\_args](#).

**6.64.2.166 expl\_derivative\_func()** [15/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
    expl_derivative_funcp_exvector e )
```

References [expl\\_derivative\\_f](#), and [expl\\_derivative\\_use\\_exvector\\_args](#).

**6.64.2.167 power\_func()** [15/15]

```
function_options & GiNaC::function_options::power_func (
    power_funcp_exvector e )
```

References [power\\_f](#), and [power\\_use\\_exvector\\_args](#).

**6.64.2.168 series\_func()** [15/15]

```
function_options & GiNaC::function_options::series_func (
    series_funcp_exvector e )
```

References [series\\_f](#), and [series\\_use\\_exvector\\_args](#).

**6.64.2.169 info\_func()** [15/15]

```
function_options & GiNaC::function_options::info_func (
    info_funcp_exvector e )
```

References [info\\_f](#), and [info\\_use\\_exvector\\_args](#).

**6.64.2.170 print\_func()** [1/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_1 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.171 print\_func() [2/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_2 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.172 print\_func() [3/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_3 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.173 print\_func() [4/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_4 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.174 print\_func() [5/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_5 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.175 print\_func() [6/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_6 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.176 print\_func() [7/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_7 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.177 print\_func() [8/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_8 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.178 print\_func() [9/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_9 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.179 print\_func() [10/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_10 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.180 print\_func() [11/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_11 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.181 print\_func() [12/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_12 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.182 print\_func() [13/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_13 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.183 print\_func() [14/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_14 p ) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.184 print\_func() [15/15]**

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
    print_funcp_exvector p ) [inline]
```

References [options](#), [print\\_use\\_exvector\\_args](#), and [set\\_print\\_func\(\)](#).

**6.64.2.185 set\_return\_type()**

```
function_options & GiNaC::function_options::set_return_type (
    unsigned rt,
    const return_type_t * rtt = nullptr )
```

References [return\\_type](#), [return\\_type\\_tinfo](#), and [use\\_return\\_type](#).



**6.64.2.186 do\_not\_evalf\_params()**

`function_options` & GiNaC::function\_options::do\_not\_evalf\_params ( )

References [evalf\\_params\\_first](#).

**6.64.2.187 remember()**

```
function_options & GiNaC::function_options::remember (
    unsigned size,
    unsigned assoc_size = 0,
    unsigned strategy = remember_strategies::delete_never )
```

References [remember\\_assoc\\_size](#), [remember\\_size](#), [remember\\_strategy](#), and [use\\_remember](#).

**6.64.2.188 overloaded()**

```
function_options & GiNaC::function_options::overloaded (
    unsigned o )
```

References [functions\\_with\\_same\\_name](#).

**6.64.2.189 set\_symmetry()**

```
function_options & GiNaC::function_options::set_symmetry (
    const symmetry & s )
```

References [symtree](#).

**6.64.2.190 get\_name()**

```
std::string GiNaC::function_options::get_name ( ) const [inline]
```

References [name](#).

**6.64.2.191 get\_nparams()**

```
unsigned GiNaC::function_options::get_nparams ( ) const [inline]
```

References [nparams](#).

#### 6.64.2.192 has\_derivative()

```
bool GiNaC::function_options::has_derivative ( ) const [inline], [protected]
```

References [derivative\\_f](#).

#### 6.64.2.193 has\_power()

```
bool GiNaC::function_options::has_power ( ) const [inline], [protected]
```

References [power\\_f](#).

#### 6.64.2.194 test\_and\_set\_nparams()

```
void GiNaC::function_options::test_and_set_nparams (
    unsigned n ) [protected]
```

References [n](#), [name](#), and [nparams](#).

Referenced by [conjugate\\_func\(\)](#), [derivative\\_func\(\)](#), [eval\\_func\(\)](#), [evalf\\_func\(\)](#), [expand\\_func\(\)](#), [expl\\_derivative\\_func\(\)](#), [imag\\_part\\_func\(\)](#), [info\\_func\(\)](#), [power\\_func\(\)](#), [print\\_func\(\)](#), [real\\_part\\_func\(\)](#), and [series\\_func\(\)](#).

#### 6.64.2.195 set\_print\_func()

```
void GiNaC::function_options::set_print_func (
    unsigned id,
    print_funcp f ) [protected]
```

References [print\\_dispatch\\_table](#).

Referenced by [print\\_func\(\)](#).

### 6.64.3 Friends And Related Function Documentation

#### 6.64.3.1 function

```
friend class function [friend]
```

### 6.64.3.2 fderivative

```
friend class fderivative [friend]
```

## 6.64.4 Member Data Documentation

### 6.64.4.1 name

```
std::string GiNaC::function_options::name [protected]
```

Referenced by [get\\_name\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::function::register\\_new\(\)](#), [set\\_name\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

### 6.64.4.2 TeX\_name

```
std::string GiNaC::function_options::TeX_name [protected]
```

Referenced by [latex\\_name\(\)](#), [GiNaC::function::print\(\)](#), and [set\\_name\(\)](#).

### 6.64.4.3 nparams

```
unsigned GiNaC::function_options::nparams [protected]
```

Referenced by [GiNaC::function::conjugate\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::function::evalf\(\)](#), [GiNaC::function::expand\(\)](#), [GiNaC::function::expl\\_derivative\(\)](#), [function\\_options\(\)](#), [get\\_nparams\(\)](#), [GiNaC::function::imag\\_part\(\)](#), [GiNaC::function::info\(\)](#), [initialize\(\)](#), [GiNaC::function::pderivative\(\)](#), [GiNaC::function::power\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::function::real\\_part\(\)](#), [GiNaC::function::series\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

### 6.64.4.4 eval\_f

```
eval_funcp GiNaC::function_options::eval_f [protected]
```

Referenced by [GiNaC::function::eval\(\)](#), [eval\\_func\(\)](#), and [initialize\(\)](#).

### 6.64.4.5 evalf\_f

```
evalf_funcp GiNaC::function_options::evalf_f [protected]
```

Referenced by [GiNaC::function::evalf\(\)](#), [evalf\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.6 conjugate\_f

`conjugate_funcp` `GiNaC::function_options::conjugate_f` [protected]

Referenced by [GiNaC::function::conjugate\(\)](#), [conjugate\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.7 real\_part\_f

`real_part_funcp` `GiNaC::function_options::real_part_f` [protected]

Referenced by [initialize\(\)](#), [GiNaC::function::real\\_part\(\)](#), and [real\\_part\\_func\(\)](#).

#### 6.64.4.8 imag\_part\_f

`imag_part_funcp` `GiNaC::function_options::imag_part_f` [protected]

Referenced by [GiNaC::function::imag\\_part\(\)](#), [imag\\_part\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.9 expand\_f

`expand_funcp` `GiNaC::function_options::expand_f` [protected]

Referenced by [GiNaC::function::expand\(\)](#), [expand\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.10 derivative\_f

`derivative_funcp` `GiNaC::function_options::derivative_f` [protected]

Referenced by [derivative\\_func\(\)](#), [has\\_derivative\(\)](#), [initialize\(\)](#), and [GiNaC::function::pderivative\(\)](#).

#### 6.64.4.11 expl\_derivative\_f

`expl_derivative_funcp` `GiNaC::function_options::expl_derivative_f` [protected]

Referenced by [GiNaC::function::expl\\_derivative\(\)](#), [expl\\_derivative\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.12 power\_f

`power_funcp` GiNaC::function\_options::power\_f [protected]

Referenced by [has\\_power\(\)](#), [initialize\(\)](#), [GiNaC::function::power\(\)](#), and [power\\_func\(\)](#).

#### 6.64.4.13 series\_f

`series_funcp` GiNaC::function\_options::series\_f [protected]

Referenced by [initialize\(\)](#), [GiNaC::function::series\(\)](#), and [series\\_func\(\)](#).

#### 6.64.4.14 print\_dispatch\_table

`std::vector<print_funcp>` GiNaC::function\_options::print\_dispatch\_table [protected]

Referenced by [GiNaC::function::print\(\)](#), and [set\\_print\\_func\(\)](#).

#### 6.64.4.15 info\_f

`info_funcp` GiNaC::function\_options::info\_f [protected]

Referenced by [GiNaC::function::info\(\)](#), [info\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.16 evalf\_params\_first

`bool` GiNaC::function\_options::evalf\_params\_first [protected]

Referenced by [do\\_not\\_evalf\\_params\(\)](#), [GiNaC::function::evalf\(\)](#), and [initialize\(\)](#).

#### 6.64.4.17 use\_return\_type

`bool` GiNaC::function\_options::use\_return\_type [protected]

Referenced by [initialize\(\)](#), [GiNaC::function::return\\_type\(\)](#), [GiNaC::function::return\\_type\\_tinfo\(\)](#), and [set\\_return\\_type\(\)](#).

#### 6.64.4.18 return\_type

`unsigned GiNaC::function_options::return_type [protected]`

Referenced by [GiNaC::function::return\\_type\(\)](#), and [set\\_return\\_type\(\)](#).

#### 6.64.4.19 return\_type\_tinfo

`return_type_t GiNaC::function_options::return_type_tinfo [protected]`

Referenced by [GiNaC::function::return\\_type\\_tinfo\(\)](#), and [set\\_return\\_type\(\)](#).

#### 6.64.4.20 use\_remember

`bool GiNaC::function_options::use_remember [protected]`

Referenced by [GiNaC::function::eval\(\)](#), [initialize\(\)](#), [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 6.64.4.21 remember\_size

`unsigned GiNaC::function_options::remember_size [protected]`

Referenced by [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 6.64.4.22 remember\_assoc\_size

`unsigned GiNaC::function_options::remember_assoc_size [protected]`

Referenced by [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 6.64.4.23 remember\_strategy

`unsigned GiNaC::function_options::remember_strategy [protected]`

Referenced by [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 6.64.4.24 eval\_use\_exvector\_args

```
bool GiNaC::function_options::eval_use_exvector_args [protected]
```

Referenced by [GiNaC::function::eval\(\)](#), [eval\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.25 evalf\_use\_exvector\_args

```
bool GiNaC::function_options::evalf_use_exvector_args [protected]
```

Referenced by [GiNaC::function::evalf\(\)](#), [evalf\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.26 conjugate\_use\_exvector\_args

```
bool GiNaC::function_options::conjugate_use_exvector_args [protected]
```

Referenced by [GiNaC::function::conjugate\(\)](#), [conjugate\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.27 real\_part\_use\_exvector\_args

```
bool GiNaC::function_options::real_part_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::real\\_part\(\)](#), and [real\\_part\\_func\(\)](#).

#### 6.64.4.28 imag\_part\_use\_exvector\_args

```
bool GiNaC::function_options::imag_part_use_exvector_args [protected]
```

Referenced by [GiNaC::function::imag\\_part\(\)](#), [imag\\_part\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.29 expand\_use\_exvector\_args

```
bool GiNaC::function_options::expand_use_exvector_args [protected]
```

Referenced by [GiNaC::function::expand\(\)](#), [expand\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.30 derivative\_use\_exvector\_args

```
bool GiNaC::function_options::derivative_use_exvector_args [protected]
```

Referenced by [derivative\\_func\(\)](#), [initialize\(\)](#), and [GiNaC::function::pderivative\(\)](#).

#### 6.64.4.31 expl\_derivative\_use\_exvector\_args

```
bool GiNaC::function_options::expl_derivative_use_exvector_args [protected]
```

Referenced by [GiNaC::function::expl\\_derivative\(\)](#), [expl\\_derivative\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.32 power\_use\_exvector\_args

```
bool GiNaC::function_options::power_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::power\(\)](#), and [power\\_func\(\)](#).

#### 6.64.4.33 series\_use\_exvector\_args

```
bool GiNaC::function_options::series_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::series\(\)](#), and [series\\_func\(\)](#).

#### 6.64.4.34 print\_use\_exvector\_args

```
bool GiNaC::function_options::print_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::print\(\)](#), and [print\\_func\(\)](#).

#### 6.64.4.35 info\_use\_exvector\_args

```
bool GiNaC::function_options::info_use_exvector_args [protected]
```

Referenced by [GiNaC::function::info\(\)](#), [info\\_func\(\)](#), and [initialize\(\)](#).



#### 6.64.4.36 functions\_with\_same\_name

```
unsigned GiNaC::function_options::functions_with_same_name [protected]
```

Referenced by [initialize\(\)](#), [overloaded\(\)](#), and [GiNaC::function::register\\_new\(\)](#).

#### 6.64.4.37 symtree

```
ex GiNaC::function_options::symtree [protected]
```

Referenced by [GiNaC::function::eval\(\)](#), [initialize\(\)](#), and [set\\_symmetry\(\)](#).

The documentation for this class was generated from the following files:

- [function.h](#)
- [function.cpp](#)

## 6.65 GiNaC::G2\_SERIAL Class Reference

Generalized multiple polylogarithm.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.65.1 Detailed Description

Generalized multiple polylogarithm.

### 6.65.2 Member Data Documentation

#### 6.65.2.1 serial

```
unsigned GiNaC::G2_SERIAL::serial [static]
```

##### Initial value:

```
= function::register_new(function_options("G", 2).  
    evalf_func(G2_evalf).  
    eval_func(G2_eval).  
    overloaded(2))
```

Referenced by [GiNaC::G\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 6.66 GiNaC::G3\_SERIAL Class Reference

Generalized multiple polylogarithm with explicit imaginary parts.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

#### 6.66.1 Detailed Description

Generalized multiple polylogarithm with explicit imaginary parts.

#### 6.66.2 Member Data Documentation

##### 6.66.2.1 serial

```
unsigned GiNaC::G3_SERIAL::serial [static]
```

**Initial value:**

```
= function::register_new(function_options("G", 3).
    evalf_func(G3_evalf).
    eval_func(G3_eval).
    overloaded(2))
```

Referenced by [GiNaC::G\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 6.67 GiNaC::gcd\_options Struct Reference

Flags to control the behavior of [gcd\(\)](#) and friends.

```
#include <normal.h>
```

### Public Types

- enum { [no\\_heur\\_gcd](#) = 2 , [no\\_part\\_factored](#) = 4 , [use\\_sr\\_gcd](#) = 8 }

#### 6.67.1 Detailed Description

Flags to control the behavior of [gcd\(\)](#) and friends.

#### 6.67.2 Member Enumeration Documentation

##### 6.67.2.1 anonymous enum

```
anonymous enum
```

## Enumerator

no_heur_gcd	Usually <a href="#">GiNaC</a> tries heuristic GCD first, because typically it's much faster than anything else. Even if heuristic algorithm fails, the overhead is negligible w.r.t. cost of computing the GCD by some other method. However, some people dislike it, so here's a flag which tells <a href="#">GiNaC</a> to NOT use the heuristic algorithm.
no_part_factored	<a href="#">GiNaC</a> tries to avoid expanding expressions when computing GCDs. This is a good idea, but some people dislike it. Hence the flag to disable special handling of partially factored polynomials. DON'T SET THIS unless you <i>really</i> know what are you doing!
use_sr_gcd	By default <a href="#">GiNaC</a> uses modular GCD algorithm. Typically it's much faster than PRS (pseudo remainder sequence) algorithm. This flag forces <a href="#">GiNaC</a> to use PRS algorithm

The documentation for this struct was generated from the following file:

- [normal.h](#)

## 6.68 GiNaC::gcdheu\_failed Class Reference

Exception thrown by [heur\\_gcd\(\)](#) to signal failure.

### 6.68.1 Detailed Description

Exception thrown by [heur\\_gcd\(\)](#) to signal failure.

The documentation for this class was generated from the following file:

- [normal.cpp](#)

## 6.69 GiNaC::has\_distance< T > Class Template Reference

SFINAE test for distance.

```
#include <utils_multi_iterator.h>
```

### Public Types

- enum { [value](#) = sizeof(test<T>(0)) == sizeof(yes\_type) }

### Private Types

- typedef char [yes\\_type](#)[1]
- typedef char [no\\_type](#)[2]

## Static Private Member Functions

- `template<typename C >`  
static `yes_type` & `test` (`decltype(std::distance< C >())`)
- `template<typename C >`  
static `no_type` & `test` (...)

### 6.69.1 Detailed Description

```
template<typename T>  
class GiNaC::has_distance< T >
```

SFINAE test for distance.

### 6.69.2 Member Typedef Documentation

#### 6.69.2.1 `yes_type`

```
template<typename T >  
typedef char GiNaC::has_distance< T >::yes_type[1] [private]
```

#### 6.69.2.2 `no_type`

```
template<typename T >  
typedef char GiNaC::has_distance< T >::no_type[2] [private]
```

### 6.69.3 Member Enumeration Documentation

#### 6.69.3.1 `anonymous enum`

```
template<typename T >  
anonymous enum
```

##### Enumerator

value	
-------	--

## 6.69.4 Member Function Documentation

### 6.69.4.1 test() [1/2]

```
template<typename T >
template<typename C >
static yes\_type & GiNaC::has\_distance< T >::test (
    decltype(std::distance< C >) ) [static], [private]
```

### 6.69.4.2 test() [2/2]

```
template<typename T >
template<typename C >
static no\_type & GiNaC::has\_distance< T >::test (
    ... ) [static], [private]
```

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

## 6.70 GiNaC::has\_options Class Reference

Flags to control the behavior of [has\(\)](#).

```
#include <flags.h>
```

### Public Types

- enum { [algebraic](#) = 0x0001 }

### 6.70.1 Detailed Description

Flags to control the behavior of [has\(\)](#).

## 6.70.2 Member Enumeration Documentation

### 6.70.2.1 anonymous enum

```
anonymous enum
```

## Enumerator

algebraic	enable algebraic matching
-----------	---------------------------

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.71 `std::hash< GiNaC::ex >` Struct Reference

Specialization of `std::hash()` for `ex` objects.

```
#include <ex.h>
```

### Public Member Functions

- `std::size_t operator() (const GiNaC::ex &e) const` noexcept

#### 6.71.1 Detailed Description

Specialization of `std::hash()` for `ex` objects.

#### 6.71.2 Member Function Documentation

##### 6.71.2.1 `operator()()`

```
std::size_t std::hash< GiNaC::ex >::operator() (
    const GiNaC::ex & e ) const    [inline], [noexcept]
```

The documentation for this struct was generated from the following file:

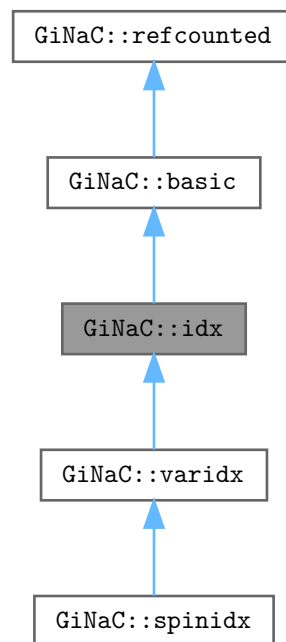
- [ex.h](#)

## 6.72 GiNaC::idx Class Reference

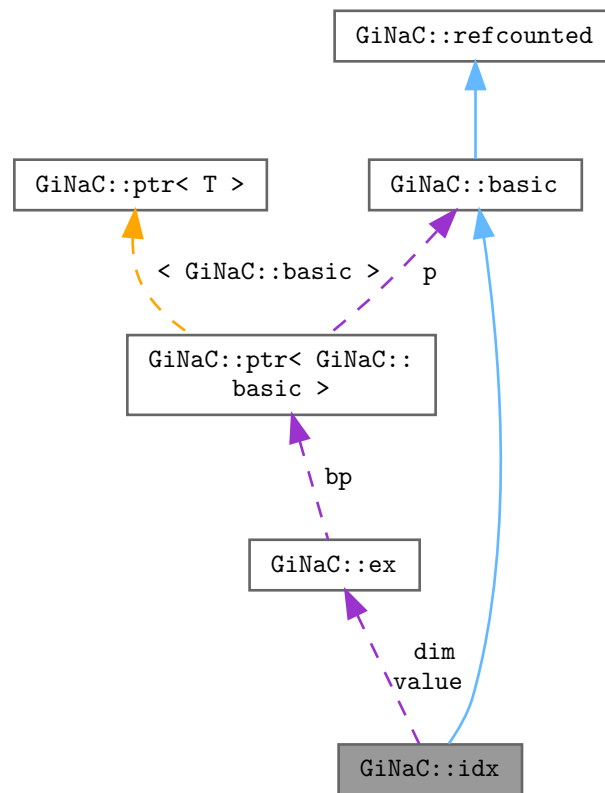
This class holds one index of an indexed object.

```
#include <idx.h>
```

Inheritance diagram for GiNaC::idx:



Collaboration diagram for `GiNaC::idx`:



## Public Member Functions

- `idx` (const `ex` &`v`, const `ex` &`dim`)  
Construct index with given value and dimension.
- bool `info` (unsigned `inf`) const override  
Information about the object.
- `size_t nops` () const override  
Number of operands/members.
- `ex op` (size\_t `i`) const override  
Return operand/member at position `i`.
- `ex map` (map\_function &`f`) const override  
Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).
- `ex evalf` () const override  
By default, `basic::evalf` would evaluate the index value but we don't want `a.1` to become `a`.
- `ex subs` (const `exmap` &`m`, unsigned `options`=0) const override  
Substitute a set of objects by arbitrary expressions.
- void `archive` (archive\_node &`n`) const override  
Save (serialize) the object into archive node.
- void `read_archive` (const archive\_node &`n`, lst &`syms`) override



- *Load (deserialize) the object from an archive node.*
- virtual bool `is_dummy_pair_same_type` (const `basic` &other) const  
*Check whether the index forms a dummy index pair with another index of the same type.*
- `ex get_value` () const  
*Get value of index.*
- bool `is_numeric` () const  
*Check whether the index is numeric.*
- bool `is_symbolic` () const  
*Check whether the index is symbolic.*
- `ex get_dim` () const  
*Get dimension of index space.*
- bool `is_dim_numeric` () const  
*Check whether the dimension is numeric.*
- bool `is_dim_symbolic` () const  
*Check whether the dimension is symbolic.*
- `ex replace_dim` (const `ex` &new\_dim) const  
*Make a new index with the same value but a different dimension.*
- `ex minimal_dim` (const `idx` &other) const  
*Return the minimum of the dimensions of this and another index.*

#### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthesizing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const

- Number of operands/members.*

  - virtual `ex op` (size\_t i) const

*Return operand/member at position i.*

  - virtual `ex operator[]` (const `ex` &index) const
  - virtual `ex operator[]` (size\_t i) const
  - virtual `ex & let_op` (size\_t i)

*Return modifiable operand/member at position i.*

  - virtual `ex & operator[]` (const `ex` &index)
  - virtual `ex & operator[]` (size\_t i)
  - virtual bool `has` (const `ex` &other, unsigned `options`=0) const

*Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const

*Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const

*Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const

*Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const

*Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const

*Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const

- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for an index always returns 0.*
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [print\\_index](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::basic`

- `basic ()`
- virtual `ex eval_ncmul (const exvector &v) const`
- virtual `bool match_same_type (const basic &other) const`  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative (const symbol &s) const`  
*Default implementation of `ex::diff()`.*
- virtual `int compare_same_type (const basic &other) const`  
*Returns order relation between two objects of same type.*
- virtual `bool is_equal_same_type (const basic &other) const`  
*Returns true if two objects of same type are equal.*
- virtual `unsigned calchash () const`  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable () const`  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print (const print_context &c, unsigned level) const`  
*Default output to stream.*
- void `do_print_tree (const print_tree &c, unsigned level) const`  
*Tree output to stream.*
- void `do_print_python_repr (const print_python_repr &c, unsigned level) const`  
*Python parsable output to stream.*

### Protected Attributes

- `ex value`  
*Expression that constitutes the index (numeric or symbolic name)*
- `ex dim`  
*Dimension of space (can be symbolic or numeric)*

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 6.72.1 Detailed Description

This class holds one index of an indexed object.

Indices can theoretically consist of any symbolic expression but they are usually only just a symbol (e.g. "mu", "i") or numeric (integer). Indices belong to a space with a certain numeric or symbolic dimension.

## 6.72.2 Constructor & Destructor Documentation

### 6.72.2.1 `idx()`

```
GiNaC::idx::idx (
    const ex & v,
    const ex & dim ) [explicit]
```

Construct index with given value and dimension.

## Parameters

<i>v</i>	Value of index (numeric or symbolic)
<i>dim</i>	Dimension of index space (numeric or symbolic)

## Returns

newly constructed index

References [dim](#), [GiNaC::ex::info\(\)](#), [is\\_dim\\_numeric\(\)](#), and [GiNaC::info\\_flags::posint](#).

## 6.72.3 Member Function Documentation

### 6.72.3.1 info()

```
bool GiNaC::idx::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

## See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::has\\_indices](#), and [GiNaC::info\\_flags::idx](#).

### 6.72.3.2 nops()

```
size_t GiNaC::idx::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.72.3.3 op()

```
ex GiNaC::idx::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position *i*.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [value](#).

### 6.72.3.4 map()

```
ex GiNaC::idx::map (
    map_function & f ) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::hash\\_calculate](#) and [value](#).

### 6.72.3.5 evalf()

```
ex GiNaC::idx::evalf ( ) const [override], [virtual]
```

By default, [basic::evalf](#) would evaluate the index value but we don't want a.1 to become a.(1.0).

Reimplemented from [GiNaC::basic](#).

### 6.72.3.6 subs()

```
ex GiNaC::idx::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::hash\\_calculate](#), [m](#), [options](#), [GiNaC::subs\\_options::really\\_subs\\_idx](#), [GiNaC::ex::subs\(\)](#), and [value](#).

### 6.72.3.7 archive()

```
void GiNaC::idx::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Loosely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::varidx](#), and [GiNaC::spinidx](#).

References [dim](#), [n](#), and [value](#).

### 6.72.3.8 read\_archive()

```
void GiNaC::idx::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::varidx](#), and [GiNaC::spinidx](#).

References [dim](#), [n](#), and [value](#).

### 6.72.3.9 derivative()

```
ex GiNaC::idx::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for an index always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#).

### 6.72.3.10 match\_same\_type()

```
bool GiNaC::idx::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::varidx](#), and [GiNaC::spinidx](#).

References [dim](#), [GINAC\\_ASSERT](#), and [GiNaC::ex::is\\_equal\(\)](#).

### 6.72.3.11 calchash()

```
unsigned GiNaC::idx::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 6.72.3.12 is\_dummy\_pair\_same\_type()

```
bool GiNaC::idx::is_dummy_pair_same_type (
    const basic & other ) const [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented in [GiNaC::varidx](#), and [GiNaC::spinidx](#).

References [dim](#), [GiNaC::ex::is\\_equal\(\)](#), and [value](#).

Referenced by [GiNaC::is\\_dummy\\_pair\(\)](#).

### 6.72.3.13 get\_value()

```
ex GiNaC::idx::get_value ( ) const [inline]
```

Get value of index.

References [value](#).

Referenced by [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), and [GiNaC::spinmetric::eval\\_indexed\(\)](#).

### 6.72.3.14 is\_numeric()

```
bool GiNaC::idx::is_numeric ( ) const [inline]
```

Check whether the index is numeric.

References [value](#).



### 6.72.3.15 is\_symbolic()

```
bool GiNaC::idx::is_symbolic ( ) const [inline]
```

Check whether the index is symbolic.

References [value](#).

Referenced by [GiNaC::spinmetric::contract\\_with\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

### 6.72.3.16 get\_dim()

```
ex GiNaC::idx::get_dim ( ) const [inline]
```

Get dimension of index space.

References [dim](#).

Referenced by [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), and [GiNaC::tensmetric::eval\\_indexed\(\)](#).

### 6.72.3.17 is\_dim\_numeric()

```
bool GiNaC::idx::is_dim_numeric ( ) const [inline]
```

Check whether the dimension is numeric.

References [dim](#).

Referenced by [idx\(\)](#).

### 6.72.3.18 is\_dim\_symbolic()

```
bool GiNaC::idx::is_dim_symbolic ( ) const [inline]
```

Check whether the dimension is symbolic.

References [dim](#).

### 6.72.3.19 `replace_dim()`

```
ex GiNaC::idx::replace_dim (
    const ex & new_dim ) const
```

Make a new index with the same value but a different dimension.

References [GiNaC::basic::clearflag\(\)](#), [dim](#), [GiNaC::basic::duplicate\(\)](#), and [GiNaC::status\\_flags::hash\\_calculated](#).

Referenced by [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

### 6.72.3.20 `minimal_dim()`

```
ex GiNaC::idx::minimal_dim (
    const idx & other ) const
```

Return the minimum of the dimensions of this and another index.

If this is undecidable, throw an exception.

References [dim](#), and [GiNaC::minimal\\_dim\(\)](#).

Referenced by [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

### 6.72.3.21 `print_index()`

```
void GiNaC::idx::print_index (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [dim](#), [GiNaC::ex::print\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [value](#).

Referenced by [do\\_print\(\)](#), [GiNaC::varidx::do\\_print\(\)](#), [GiNaC::spinidx::do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [GiNaC::spinidx::do\\_print\\_latex\(\)](#).

### 6.72.3.22 `do_print()`

```
void GiNaC::idx::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_index\(\)](#).

### 6.72.3.23 do\_print\_csrc()

```
void GiNaC::idx::do_print_csrc (
    const print\_csrc & c,
    unsigned level ) const    [protected]
```

References [c](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::print\(\)](#), and [value](#).

### 6.72.3.24 do\_print\_latex()

```
void GiNaC::idx::do_print_latex (
    const print\_latex & c,
    unsigned level ) const    [protected]
```

References [c](#), and [print\\_index\(\)](#).

### 6.72.3.25 do\_print\_tree()

```
void GiNaC::idx::do_print_tree (
    const print\_tree & c,
    unsigned level ) const    [protected]
```

References [c](#), [dim](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::print\(\)](#), and [value](#).

## 6.72.4 Member Data Documentation

### 6.72.4.1 value

```
ex GiNaC::idx::value    [protected]
```

Expression that constitutes the index (numeric or symbolic name)

Referenced by [archive\(\)](#), [calchash\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [get\\_value\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [is\\_numeric\(\)](#), [is\\_symbolic\(\)](#), [map\(\)](#), [op\(\)](#), [print\\_index\(\)](#), [read\\_archive\(\)](#), and [subs\(\)](#).

#### 6.72.4.2 dim

```
ex GiNaC::idx::dim [protected]
```

Dimension of space (can be symbolic or numeric)

Referenced by [archive\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [get\\_dim\(\)](#), [idx\(\)](#), [is\\_dim\\_numeric\(\)](#), [is\\_dim\\_symbolic\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [minimal\\_dim\(\)](#), [print\\_index\(\)](#), [read\\_archive\(\)](#), and [replace\\_dim\(\)](#).

The documentation for this class was generated from the following files:

- [idx.h](#)
- [idx.cpp](#)

### 6.73 GiNaC::idx\_is\_equal\_ignore\_dim Struct Reference

#### Public Member Functions

- [bool operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

#### 6.73.1 Member Function Documentation

##### 6.73.1.1 operator()

```
bool GiNaC::idx_is_equal_ignore_dim::operator() (
    const ex & lh,
    const ex & rh ) const [inline]
```

References [GiNaC::ex::is\\_equal\(\)](#).

The documentation for this struct was generated from the following file:

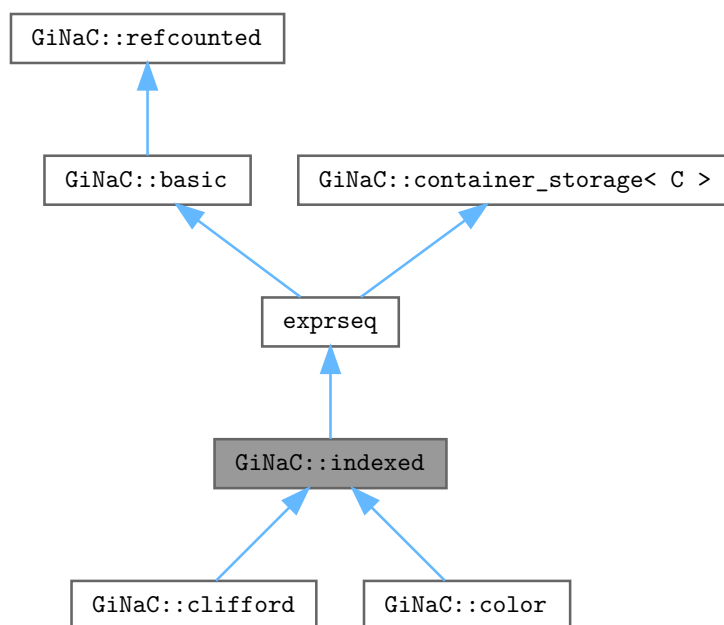
- [indexed.cpp](#)

## 6.74 GiNaC::indexed Class Reference

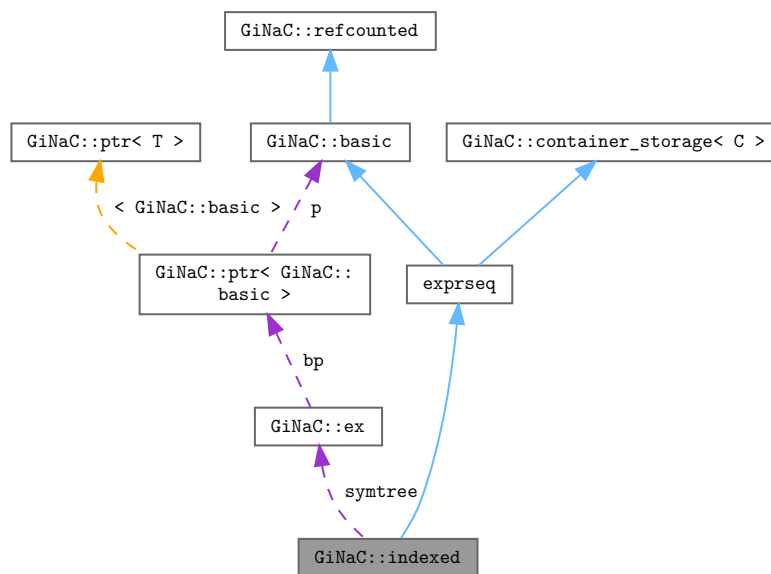
This class holds an indexed expression.

```
#include <indexed.h>
```

Inheritance diagram for GiNaC::indexed:



Collaboration diagram for `GiNaC::indexed`:



## Public Member Functions

- `indexed` (const `ex` &b)  
Construct indexed object with no index.
- `indexed` (const `ex` &b, const `ex` &i1)  
Construct indexed object with one index.
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2)  
Construct indexed object with two indices.
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3)  
Construct indexed object with three indices.
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)  
Construct indexed object with four indices.
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2)  
Construct indexed object with two indices and a specified symmetry.
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3)  
Construct indexed object with three indices and a specified symmetry.
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)  
Construct indexed object with four indices and a specified symmetry.
- `indexed` (const `ex` &b, const `exvector` &iv)  
Construct indexed object with a specified vector of indices.
- `indexed` (const `ex` &b, const `symmetry` &symm, const `exvector` &iv)  
Construct indexed object with a specified vector of indices and symmetry.
- `indexed` (const `symmetry` &symm, const `exprseq` &es)
- `indexed` (const `symmetry` &symm, const `exvector` &v)
- `indexed` (const `symmetry` &symm, `exvector` &&v)
- unsigned `precedence` () const override  
Return relative operator precedence (for parenthezing output).

- `bool info` (unsigned inf) const override  
*Information about the object.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex real_part` () const override
- `ex imag_part` () const override
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `void archive` (archive\_node &n) const override  
*Save (a.k.a.*
- `void read_archive` (const archive\_node &n, lst &syms) override  
*Read (a.k.a.*
- `bool all_index_values_are` (unsigned inf) const  
*Check whether all index values have a certain property.*
- `exvector get_indices` () const  
*Return a vector containing the object's indices.*
- `exvector get_dummy_indices` () const  
*Return a vector containing the dummy indices of the object, if any.*
- `exvector get_dummy_indices` (const indexed &other) const  
*Return a vector containing the dummy indices in the contraction with another indexed object.*
- `bool has_dummy_index_for` (const ex &i) const  
*Check whether the object has an index that forms a dummy index pair with a given index.*
- `ex get_symmetry` () const  
*Return symmetry properties.*

#### Public Member Functions inherited from `GiNaC::container< C >`

- `container` (STLT const &s)
- `container` (STLT &&v)
- `container` (exvector::const\_iterator b, exvector::const\_iterator e)
- `container` (std::initializer\_list< ex > il)
- `bool info` (unsigned inf) const override  
*Information about the object.*
- `unsigned precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex subs` (const exmap &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `void read_archive` (const archive\_node &n, lst &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- `void archive` (archive\_node &n) const override  
*Archive the object.*
- `container & prepend` (const ex &b)  
*Add element at front.*
- `container & append` (const ex &b)

- *Add element at back.*
- `container & remove_first ()`
- *Remove first element.*
- `container & remove_last ()`
- *Remove last element.*
- `container & remove_all ()`
- *Remove all elements.*
- `container & sort ()`
- *Sort elements.*
- `container & unique ()`
- *Remove adjacent duplicate elements.*
- `const_iterator begin () const`
- `const_iterator end () const`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rend () const`

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`
- *basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`
- *basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`
- *Create a clone of this object on the heap.*
- virtual `ex eval () const`
- *Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`
- *Evaluate object numerically.*
- virtual `ex evalm () const`
- *Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`
- *Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`
- *Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0) const`
- *Output to stream.*
- virtual void `dbgprint () const`
- *Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree () const`
- *Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence () const`
- *Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf) const`
- *Information about the object.*
- virtual size\_t `nops () const`
- *Number of operands/members.*
- virtual `ex op (size_t i) const`
- *Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`



- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0) const`  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0) const`  
*Default implementation of ex::series().*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of ex::normal().*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of ex::to\_rational().*
- virtual `ex to_polynomial (exmap &repl) const`
- virtual `numeric integer_content () const`
- virtual `ex smod (const numeric &xi) const`  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient () const`  
*Implementation ex::max\_coefficient().*
- virtual `exvector get_free_indices () const`  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other) const`  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other) const`  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const`  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `unsigned return_type () const`
- virtual `return_type_t return_type_tinfo () const`
- virtual `ex conjugate () const`
- virtual `ex real_part () const`
- virtual `ex imag_part () const`

- `template<class T >`  
`void print_dispatch (const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `virtual void archive (archive_node &n) const`  
*Save (serialize) the object into archive node.*
- `virtual void read_archive (const archive_node &n, lst &syms)`  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level (const exmap &m, unsigned options) const`  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1) const`  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare (const basic &other) const`  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal (const basic &other) const`  
*Test for syntactic equality.*
- `const basic & hold () const`  
*Stop further evaluation.*
- `unsigned gethash () const`
- `const basic & setflag (unsigned f) const`  
*Set some `status_flags`.*
- `const basic & clearflag (unsigned f) const`  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted () noexcept`
- `unsigned int add_reference () noexcept`
- `unsigned int remove_reference () noexcept`
- `unsigned int get_refcount () const noexcept`
- `void set_refcount (unsigned int r) noexcept`

### Protected Member Functions

- `ex derivative (const symbol &s) const override`  
*Implementation of `ex::diff()` for an indexed object always returns 0.*
- `ex thiscontainer (const exvector &v) const override`
- `ex thiscontainer (exvector &&v) const override`
- `unsigned return_type () const override`
- `return_type_t return_type_tinfo () const override`
- `ex expand (unsigned options=0) const override`  
*Expand expression, i.e.*
- `void printindices (const print_context &c, unsigned level) const`
- `void print_indexed (const print_context &c, const char *openbrace, const char *closebrace, unsigned level) const`
- `void do_print (const print_context &c, unsigned level) const`
- `void do_print_latex (const print_latex &c, unsigned level) const`
- `void do_print_tree (const print_tree &c, unsigned level) const`
- `void validate () const`  
*Check whether all indices are of class `idx` and validate the symmetry tree.*

**Protected Member Functions inherited from [GiNaC::container< C >](#)**

- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- virtual [ex thiscontainer](#) (const [STLT](#) &v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- virtual [ex thiscontainer](#) ([STLT](#) &&v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this↔\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned [options](#)=0) const

**Protected Member Functions inherited from [GiNaC::basic](#)**

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

**Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)**

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

## Protected Attributes

- [ex symtree](#)  
*Index symmetry (tree of symmetry objects)*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)

## Friends

- [ex simplify\\_indexed](#) (const [ex](#) &e, [exvector](#) &free\_indices, [exvector](#) &dummy\_indices, const [scalar\\_products](#) &sp)  
*Simplify indexed expression, return list of free indices.*
- [ex simplify\\_indexed\\_product](#) (const [ex](#) &e, [exvector](#) &free\_indices, [exvector](#) &dummy\_indices, const [scalar\\_products](#) &sp)  
*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.*
- bool [reposition\\_dummy\\_indices](#) ([ex](#) &e, [exvector](#) &variant\_dummy\_indices, [exvector](#) &moved\_indices)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*

## Additional Inherited Members

### Public Types inherited from [GiNaC::container< C >](#)

- typedef STLT::const\_iterator [const\\_iterator](#)
- typedef STLT::const\_reverse\_iterator [const\\_reverse\\_iterator](#)

### Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

### Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef C< [ex](#) > [STLT](#)

**Static Protected Member Functions inherited from [GiNaC::container< C >](#)**

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for *lst*.*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for *lst*.*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for *lst*.*

**Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)**

- static void [reserve](#) (STLT &, size\_t)

**6.74.1 Detailed Description**

This class holds an indexed expression.

It consists of a 'base' expression (the expression being indexed) which can be accessed as `op(0)`, and `n` (`n >= 0`) indices (all of class `idx`), accessible as `op(1)..op(n)`.

**6.74.2 Constructor & Destructor Documentation****6.74.2.1 `indexed()` [1/13]**

```
GiNaC::indexed::indexed (
    const ex & b )
```

Construct indexed object with no index.

**Parameters**

<i>b</i>	Base expression
----------	-----------------

**Returns**

newly constructed indexed object

References [GiNaC::not\\_symmetric\(\)](#), and [validate\(\)](#).

**6.74.2.2 `indexed()` [2/13]**

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & il )
```

Construct indexed object with one index.

The index must be of class `idx`.

#### Parameters

<i>b</i>	Base expression
<i>i1</i>	The index

#### Returns

newly constructed indexed object

References [validate\(\)](#).

### 6.74.2.3 indexed() [3/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1,
    const ex & i2 )
```

Construct indexed object with two indices.

The indices must be of class `idx`.

#### Parameters

<i>b</i>	Base expression
<i>i1</i>	First index
<i>i2</i>	Second index

#### Returns

newly constructed indexed object

References [validate\(\)](#).

### 6.74.2.4 indexed() [4/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1,
    const ex & i2,
    const ex & i3 )
```

Construct indexed object with three indices.

The indices must be of class `idx`.

## Parameters

<i>b</i>	Base expression
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index

## Returns

newly constructed indexed object

References [validate\(\)](#).

**6.74.2.5 indexed()** [5/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const ex & i1,
    const ex & i2,
    const ex & i3,
    const ex & i4 )
```

Construct indexed object with four indices.

The indices must be of class idx.

## Parameters

<i>b</i>	Base expression
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index
<i>i4</i>	Fourth index

## Returns

newly constructed indexed object

References [validate\(\)](#).

**6.74.2.6 indexed()** [6/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
```

```
const ex & i1,  
const ex & i2 )
```

Construct indexed object with two indices and a specified symmetry.

The indices must be of class idx.



## Parameters

<i>b</i>	Base expression
<i>symm</i>	Symmetry of indices
<i>i1</i>	First index
<i>i2</i>	Second index

## Returns

newly constructed indexed object

References [validate\(\)](#).

**6.74.2.7 indexed()** [7/13]

```
GiNaC::indexed::indexed (  
    const ex & b,  
    const symmetry & symm,  
    const ex & i1,  
    const ex & i2,  
    const ex & i3 )
```

Construct indexed object with three indices and a specified symmetry.

The indices must be of class idx.

## Parameters

<i>b</i>	Base expression
<i>symm</i>	Symmetry of indices
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index

## Returns

newly constructed indexed object

References [validate\(\)](#).

**6.74.2.8 indexed()** [8/13]

```
GiNaC::indexed::indexed (  
    const ex & b,  
    const symmetry & symm,
```

```

const ex & i1,
const ex & i2,
const ex & i3,
const ex & i4 )

```

Construct indexed object with four indices and a specified symmetry.

The indices must be of class `idx`.

#### Parameters

<i>b</i>	Base expression
<i>symm</i>	Symmetry of indices
<i>i1</i>	First index
<i>i2</i>	Second index
<i>i3</i>	Third index
<i>i4</i>	Fourth index

#### Returns

newly constructed indexed object

References [validate\(\)](#).

### 6.74.2.9 indexed() [9/13]

```

GiNaC::indexed::indexed (
    const ex & b,
    const exvector & iv )

```

Construct indexed object with a specified vector of indices.

The indices must be of class `idx`.

#### Parameters

<i>b</i>	Base expression
<i>iv</i>	Vector of indices

#### Returns

newly constructed indexed object

References [GiNaC::container\\_storage< C >::seq](#), and [validate\(\)](#).

**6.74.2.10 indexed()** [10/13]

```
GiNaC::indexed::indexed (
    const ex & b,
    const symmetry & symm,
    const exvector & iv )
```

Construct indexed object with a specified vector of indices and symmetry.

The indices must be of class `idx`.

**Parameters**

<i>b</i>	Base expression
<i>symm</i>	Symmetry of indices
<i>iv</i>	Vector of indices

**Returns**

newly constructed indexed object

References [GiNaC::container\\_storage< C >::seq](#), and [validate\(\)](#).

**6.74.2.11 indexed()** [11/13]

```
GiNaC::indexed::indexed (
    const symmetry & symm,
    const exprseq & es )
```

**6.74.2.12 indexed()** [12/13]

```
GiNaC::indexed::indexed (
    const symmetry & symm,
    const exvector & v )
```

**6.74.2.13 indexed()** [13/13]

```
GiNaC::indexed::indexed (
    const symmetry & symm,
    exvector && v )
```

**6.74.3 Member Function Documentation**

### 6.74.3.1 precedence()

```
unsigned GiNaC::indexed::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< C >](#).

Referenced by [print\\_indexed\(\)](#).

### 6.74.3.2 info()

```
bool GiNaC::indexed::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::info\\_flags::indexed](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [imag\\_part\(\)](#), and [real\\_part\(\)](#).

### 6.74.3.3 eval()

```
ex GiNaC::indexed::eval ( ) const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::canonicalize\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [symtree](#), and [thiscontainer\(\)](#).

### 6.74.3.4 real\_part()

```
ex GiNaC::indexed::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [info\(\)](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::info\\_flags::real](#).

### 6.74.3.5 imag\_part()

```
ex GiNaC::indexed::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [info\(\)](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::info\\_flags::real](#).

### 6.74.3.6 get\_free\_indices()

```
exvector GiNaC::indexed::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [get\\_dummy\\_indices\(\)](#).

### 6.74.3.7 archive()

```
void GiNaC::indexed::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) indexed object into archive.

Reimplemented from [GiNaC::container< C >](#).

References [n](#), and [symtree](#).

### 6.74.3.8 read\_archive()

```
void GiNaC::indexed::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) indexed object from archive.

Reimplemented from [GiNaC::container< C >](#).

References [n](#), [GiNaC::not\\_symmetric\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_symm\(\)](#), [GiNaC::syymm\(\)](#), [symtree](#), and [validate\(\)](#).

### 6.74.3.9 derivative()

```
ex GiNaC::indexed::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for an indexed object always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#).

### 6.74.3.10 thiscontainer() [1/2]

```
ex GiNaC::indexed::thiscontainer (
    const exvector & v ) const [override], [protected]
```

References [symtree](#).

Referenced by [eval\(\)](#), and [expand\(\)](#).

### 6.74.3.11 thiscontainer() [2/2]

```
ex GiNaC::indexed::thiscontainer (
    exvector && v ) const [override], [protected]
```

References [symtree](#).

### 6.74.3.12 return\_type()

```
unsigned GiNaC::indexed::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

**6.74.3.13 return\_type\_tinfo()**

```
return_type_t GiNaC::indexed::return_type_tinfo ( ) const [inline], [override], [protected],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container< C >::op\(\)](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

**6.74.3.14 expand()**

```
ex GiNaC::indexed::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_indexed](#), [GINAC\\_ASSERT](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [GiNaC::container\\_storage< C >::seq](#), and [thiscontainer\(\)](#).

**6.74.3.15 all\_index\_values\_are()**

```
bool GiNaC::indexed::all_index_values_are (
    unsigned inf ) const
```

Check whether all index values have a certain property.

See also

class [info\\_flags](#)

References [GiNaC::container\\_storage< C >::seq](#).

**6.74.3.16 get\_indices()**

```
exvector GiNaC::indexed::get_indices ( ) const
```

Return a vector containing the object's indices.

References [GINAC\\_ASSERT](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.74.3.17 get\_dummy\_indices() [1/2]**

```
exvector GiNaC::indexed::get_dummy_indices ( ) const
```

Return a vector containing the dummy indices of the object, if any.

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.74.3.18 get\_dummy\_indices() [2/2]**

```
exvector GiNaC::indexed::get_dummy_indices (
    const indexed & other ) const
```

Return a vector containing the dummy indices in the contraction with another indexed object.

This is symmetric: `a.get_dummy_indices(b) == b.get_dummy_indices(a)`

References [GiNaC::find\\_dummy\\_indices\(\)](#), and [get\\_free\\_indices\(\)](#).

**6.74.3.19 has\_dummy\_index\_for()**

```
bool GiNaC::indexed::has_dummy_index_for (
    const ex & i ) const
```

Check whether the object has an index that forms a dummy index pair with a given index.

References [GiNaC::is\\_dummy\\_pair\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.74.3.20 get\_symmetry()**

```
ex GiNaC::indexed::get_symmetry ( ) const [inline]
```

Return symmetry properties.

References [symtree](#).

Referenced by [GiNaC::clifford::get\\_metric\(\)](#).



### 6.74.3.21 printindices()

```
void GiNaC::indexed::printindices (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [do\\_print\\_tree\(\)](#), and [print\\_indexed\(\)](#).

### 6.74.3.22 print\_indexed()

```
void GiNaC::indexed::print_indexed (
    const print\_context & c,
    const char * openbrace,
    const char * closebrace,
    unsigned level ) const [protected]
```

References [c](#), [precedence\(\)](#), [printindices\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

### 6.74.3.23 do\_print()

```
void GiNaC::indexed::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_indexed\(\)](#).

### 6.74.3.24 do\_print\_latex()

```
void GiNaC::indexed::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_indexed\(\)](#).

### 6.74.3.25 do\_print\_tree()

```
void GiNaC::indexed::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [printindices\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [symtree](#).

### 6.74.3.26 validate()

```
void GiNaC::indexed::validate ( ) const [protected]
```

Check whether all indices are of class `idx` and validate the symmetry tree.

This function is used internally to make sure that all constructed indexed objects really carry indices and not some other classes.

References [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [symtree](#), and [validate\(\)](#).

Referenced by [indexed\(\)](#), [read\\_archive\(\)](#), and [validate\(\)](#).

## 6.74.4 Friends And Related Function Documentation

### 6.74.4.1 simplify\_indexed

```
ex simplify_indexed (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp ) [friend]
```

Simplify indexed expression, return list of free indices.

Referenced by [GiNaC::clifford::get\\_metric\(\)](#), and [GiNaC::clifford::same\\_metric\(\)](#).

### 6.74.4.2 simplify\_indexed\_product

```
ex simplify_indexed_product (
    const ex & e,
    exvector & free_indices,
    exvector & dummy_indices,
    const scalar_products & sp ) [friend]
```

Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.

### 6.74.4.3 reposition\_dummy\_indices

```
bool reposition_dummy_indices (
    ex & e,
    exvector & variant_dummy_indices,
    exvector & moved_indices ) [friend]
```

Raise/lower dummy indices in a single indexed objects to canonicalize their variance.

## Parameters

<i>e</i>	Object to work on
<i>variant_dummy_indices</i>	The set of indices that might need repositioning (will be changed by this function)
<i>moved_indices</i>	The set of indices that have been repositioned (will be changed by this function)

## Returns

true if 'e' was changed

## 6.74.5 Member Data Documentation

### 6.74.5.1 symtree

```
ex GiNaC::indexed::symtree    [protected]
```

Index symmetry (tree of symmetry objects)

Referenced by [archive\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [do\\_print\\_tree\(\)](#), [eval\(\)](#), [get\\_symmetry\(\)](#), [read\\_archive\(\)](#), [thiscontainer\(\)](#), and [validate\(\)](#).

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 6.75 GiNaC::info\_flags Class Reference

Possible attributes an object can have.

```
#include <flags.h>
```

### Public Types

- enum {  
[numeric](#) , [real](#) , [rational](#) , [integer](#) ,  
[crational](#) , [cinteger](#) , [positive](#) , [negative](#) ,  
[nonnegative](#) , [posint](#) , [negint](#) , [nonnegint](#) ,  
[even](#) , [odd](#) , [prime](#) , [relation](#) ,  
[relation\\_equal](#) , [relation\\_not\\_equal](#) , [relation\\_less](#) , [relation\\_less\\_or\\_equal](#) ,  
[relation\\_greater](#) , [relation\\_greater\\_or\\_equal](#) , [symbol](#) , [list](#) ,  
[exprseq](#) , [polynomial](#) , [integer\\_polynomial](#) , [cinteger\\_polynomial](#) ,  
[rational\\_polynomial](#) , [crational\\_polynomial](#) , [rational\\_function](#) , [indexed](#) ,  
[has\\_indices](#) , [idx](#) , [expanded](#) , [indefinite](#) }

### 6.75.1 Detailed Description

Possible attributes an object can have.

### 6.75.2 Member Enumeration Documentation

#### 6.75.2.1 anonymous enum

anonymous enum

Enumerator

numeric	
real	
rational	
integer	
crational	
cinteger	
positive	
negative	
nonnegative	
posint	
negint	
nonnegint	
even	
odd	
prime	
relation	
relation_equal	
relation_not_equal	
relation_less	
relation_less_or_equal	
relation_greater	
relation_greater_or_equal	
symbol	
list	
exprseq	
polynomial	
integer_polynomial	
cinteger_polynomial	
rational_polynomial	
crational_polynomial	
rational_function	
indexed	
has_indices	
idx	
expanded	
indefinite	

The documentation for this class was generated from the following file:

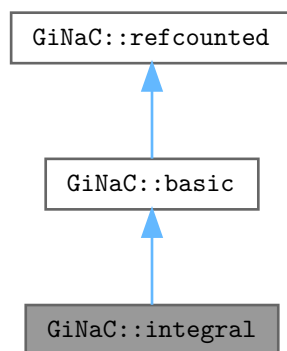
- [flags.h](#)

## 6.76 GiNaC::integral Class Reference

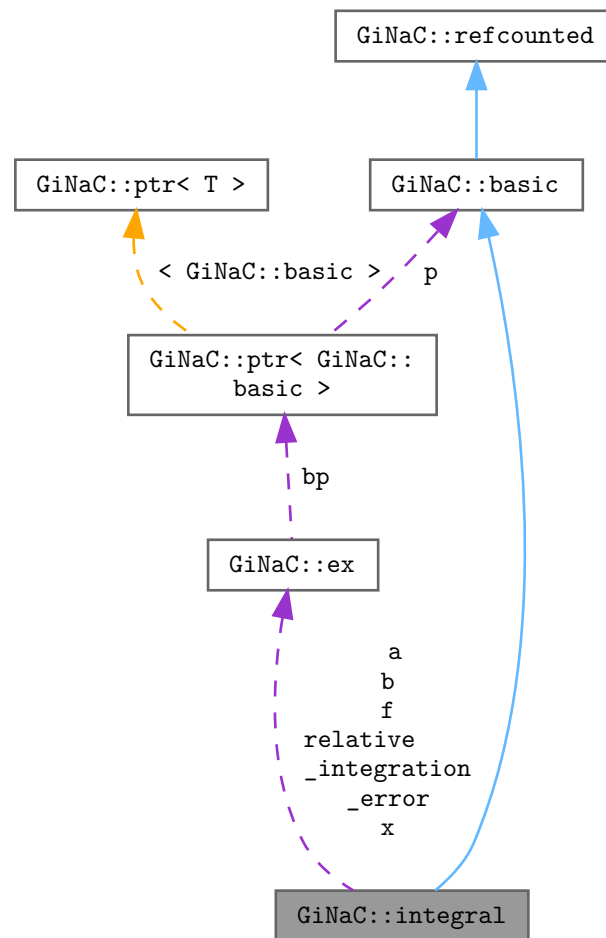
Symbolic integral.

```
#include <integral.h>
```

Inheritance diagram for GiNaC::integral:



Collaboration diagram for `GiNaC::integral`:



## Public Member Functions

- `integral` (const `ex` &`x_`, const `ex` &`a_`, const `ex` &`b_`, const `ex` &`f_`)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- int `degree` (const `ex` &`s`) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &`s`) const override  
*Return degree of lowest power in object s.*
- `ex eval_ncmul` (const `exvector` &`v`) const override
- size\_t `nops` () const override

- *Number of operands/members.*
- `ex op` (`size_t i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (`size_t i`) override  
*Return modifiable operand/member at position i.*
- `ex expand` (`unsigned options=0`) const override  
*Expand expression, i.e.*
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `unsigned return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex conjugate` () const override
- `ex eval_integ` () const override  
*Evaluate integrals, if result is known.*
- `void archive` (`archive_node &n`) const override  
*Save (a.k.a.*
- `void read_archive` (`const archive_node &n, lst &syms`) override  
*Read (a.k.a.*

#### Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (`const basic &other`)
- `const basic & operator=` (`const basic &other`)  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate` () const  
*Create a clone of this object on the heap.*
- `virtual ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf` () const  
*Evaluate object numerically.*
- `virtual ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- `virtual ex eval_indexed` (`const basic &i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print` (`const print_context &c, unsigned level=0`) const  
*Output to stream.*
- `virtual void dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- `virtual void dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- `virtual unsigned precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- `virtual bool info` (`unsigned inf`) const  
*Information about the object.*
- `virtual size_t nops` () const  
*Number of operands/members.*

- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const



- virtual `ex conjugate ()` const
- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Static Public Attributes

- static int `max_integration_level` = 15
- static `ex relative_integration_error` = 1e-8

### Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Default implementation of `ex::diff()`.*
- `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const override  
*Default implementation of `ex::series()`.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Private Attributes

- [ex](#) x
- [ex](#) a
- [ex](#) b
- [ex](#) f

### Additional Inherited Members

#### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

#### 6.76.1 Detailed Description

Symbolic integral.

#### 6.76.2 Constructor & Destructor Documentation

### 6.76.2.1 integral()

```
GiNaC::integral::integral (
    const ex & x_,
    const ex & a_,
    const ex & b_,
    const ex & f_ )
```

References [x](#).

## 6.76.3 Member Function Documentation

### 6.76.3.1 precedence()

```
unsigned GiNaC::integral::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\\_latex\(\)](#).

### 6.76.3.2 eval()

```
ex GiNaC::integral::eval ( ) const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [a](#), [b](#), [GiNaC::status\\_flags::evaluated](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::ex::has\(\)](#), [GiNaC::haswild\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

### 6.76.3.3 evalf()

```
ex GiNaC::integral::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::adaptivesimpson\(\)](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::ex::evalf\(\)](#), [f](#), [GiNaC::ex::subs\(\)](#), and [x](#).

#### 6.76.3.4 degree()

```
int GiNaC::integral::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), and [f](#).

#### 6.76.3.5 ldegree()

```
int GiNaC::integral::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), and [f](#).

#### 6.76.3.6 eval\_ncmul()

```
ex GiNaC::integral::eval_ncmul (
    const exvector & v ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::eval\\_ncmul\(\)](#), and [f](#).

#### 6.76.3.7 nops()

```
size_t GiNaC::integral::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.76.3.8 op()

```
ex GiNaC::integral::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), [GINAC\\_ASSERT](#), and [x](#).

### 6.76.3.9 let\_op()

```
ex & GiNaC::integral::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [f](#), and [x](#).

### 6.76.3.10 expand()

```
ex GiNaC::integral::expand (
    unsigned options = 0 ) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::basic::ex](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::ex::has\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [GiNaC::basic::setflag\(\)](#), and [x](#).

Referenced by [eval\\_integ\(\)](#), and [expand\(\)](#).

### 6.76.3.11 get\_free\_indices()

```
exvector GiNaC::integral::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), and [GiNaC::ex::get\\_free\\_indices\(\)](#).

### 6.76.3.12 return\_type()

```
unsigned GiNaC::integral::return_type ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [f](#), and [GiNaC::ex::return\\_type\(\)](#).

### 6.76.3.13 return\_type\_tinfo()

```
return_type_t GiNaC::integral::return_type_tinfo ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [f](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

### 6.76.3.14 conjugate()

```
ex GiNaC::integral::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::ex::conjugate\(\)](#), [f](#), [GiNaC::ex::subs\(\)](#), and [x](#).

### 6.76.3.15 eval\_integ()

```
ex GiNaC::integral::eval_integ ( ) const [override], [virtual]
```

Evaluate integrals, if result is known.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::ex::eval\\_integ\(\)](#), [expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::ex::has\(\)](#), [GiNaC::log\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

### 6.76.3.16 archive()

```
void GiNaC::integral::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), [n](#), and [x](#).

### 6.76.3.17 read\_archive()

```
void GiNaC::integral::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), [n](#), and [x](#).

### 6.76.3.18 derivative()

```
ex GiNaC::integral::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::ex::diff\(\)](#), [f](#), [GiNaC::ex::subs\(\)](#), and [x](#).

### 6.76.3.19 series()

```
ex GiNaC::integral::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [a](#), [b](#), [f](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::nops\(\)](#), [options](#), [order](#), [r](#), [GiNaC::ex::series\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

Referenced by [series\(\)](#).

### 6.76.3.20 do\_print()

```
void GiNaC::integral::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [a](#), [b](#), [c](#), [f](#), [GiNaC::ex::print\(\)](#), and [x](#).

### 6.76.3.21 do\_print\_latex()

```
void GiNaC::integral::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [a](#), [b](#), [c](#), [f](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), and [x](#).

## 6.76.4 Member Data Documentation

### 6.76.4.1 max\_integration\_level

```
int GiNaC::integral::max_integration_level = 15 [static]
```

Referenced by [GiNaC::adaptivesimpson\(\)](#).

### 6.76.4.2 relative\_integration\_error

```
ex GiNaC::integral::relative_integration_error = 1e-8 [static]
```

### 6.76.4.3 x

```
ex GiNaC::integral::x [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [integral\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).



#### 6.76.4.4 a

`ex GiNaC::integral::a [private]`

Referenced by [archive\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).

#### 6.76.4.5 b

`ex GiNaC::integral::b [private]`

Referenced by [archive\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).

#### 6.76.4.6 f

`ex GiNaC::integral::f [private]`

Referenced by [archive\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [eval\\_ncmul\(\)](#), [evalf\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), and [series\(\)](#).

The documentation for this class was generated from the following files:

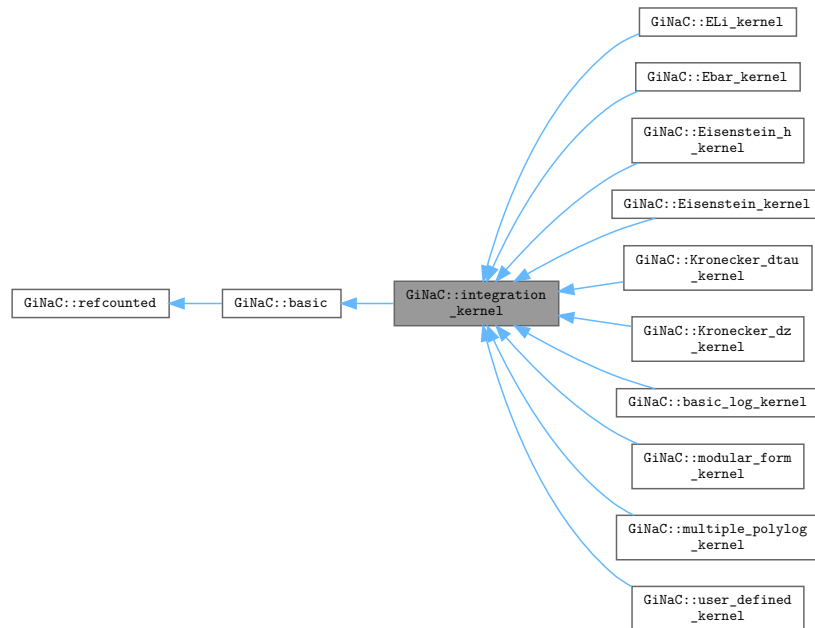
- [integral.h](#)
- [indexed.cpp](#)
- [integral.cpp](#)
- [pseries.cpp](#)

## 6.77 GiNaC::integration\_kernel Class Reference

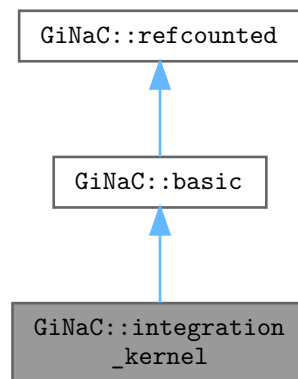
The base class for integration kernels for iterated integrals.

```
#include <integration_kernel.h>
```

Inheritance diagram for `GiNaC::integration_kernel`:



Collaboration diagram for `GiNaC::integration_kernel`:



## Public Member Functions

- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override  
Default implementation of `ex::series()`.
- virtual bool `has_trailing_zero` (void) const  
This routine returns true, if the integration kernel has a trailing zero.

- virtual bool `is_numeric` (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual `ex Laurent_series` (const `ex` &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex` &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N `series_coeff` (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const

- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0) const`  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0) const`  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl) const`
- virtual `numeric integer_content () const`
- virtual `ex smod (const numeric &xi) const`  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient () const`  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices () const`  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other) const`  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other) const`  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const`  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `unsigned return_type () const`
- virtual `return_type_t return_type_tinfo () const`
- virtual `ex conjugate () const`
- virtual `ex real_part () const`
- virtual `ex imag_part () const`

- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- virtual bool [uses\\_Laurent\\_series](#) () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method [Laurent\\_series](#) needs to be implemented).*
- virtual [cln::cl\\_N series\\_coeff\\_impl](#) (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- [ex get\\_numerical\\_value\\_impl](#) (const [ex](#) &lambda, const [ex](#) &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.77.1 Detailed Description

The base class for integration kernels for iterated integrals.

This class represents the differential one-form

$$\omega = d\lambda$$

The integration variable is a dummy variable and does not need to be specified.

## 6.77.2 Member Function Documentation

### 6.77.2.1 series()

```
ex GiNaC::integration_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), and [GiNaC::modular\\_form\\_kernel](#).

References [order](#), and [r](#).

### 6.77.2.2 has\_trailing\_zero()

```
bool GiNaC::integration_kernel::has_trailing_zero (
    void ) const [virtual]
```

This routine returns true, if the integration kernel has a trailing zero.

### 6.77.2.3 is\_numeric()

```
bool GiNaC::integration_kernel::is_numeric (
    void ) const [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented in [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), and [GiNaC::user\\_defined\\_kernel](#).

#### 6.77.2.4 Laurent\_series()

```
ex GiNaC::integration_kernel::Laurent_series (
    const ex & x,
    int order ) const [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented in [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), and [GiNaC::user\\_defined\\_kernel](#).

References [n](#), [order](#), [GiNaC::pow\(\)](#), [GiNaC::ex::series\(\)](#), and [x](#).

#### 6.77.2.5 get\_numerical\_value()

```
ex GiNaC::integration_kernel::get_numerical_value (
    const ex & lambda,
    int N_trunc = 0 ) const [virtual]
```

Evaluates the integrand at lambda.

Reimplemented in [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), and [GiNaC::Kronecker\\_dz\\_kernel](#).

#### 6.77.2.6 uses\_Laurent\_series()

```
bool GiNaC::integration_kernel::uses_Laurent_series ( ) const [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented in [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), and [GiNaC::user\\_defined\\_kernel](#).

#### 6.77.2.7 series\_coeff\_impl()

```
cln::cl_N GiNaC::integration_kernel::series_coeff_impl (
    int i ) const [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented in [GiNaC::basic\\_log\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), and [GiNaC::Kronecker\\_dz\\_kernel](#).



**6.77.2.8 get\_cache\_size()**

```
size_t GiNaC::integration_kernel::get_cache_size (
    void ) const
```

Returns the current size of the cache.

**6.77.2.9 set\_cache\_step()**

```
void GiNaC::integration_kernel::set_cache_step (
    int cache_steps ) const
```

Sets the step size by which the cache is increased.

**6.77.2.10 get\_series\_coeff()**

```
ex GiNaC::integration_kernel::get_series_coeff (
    int i ) const
```

Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.

**6.77.2.11 series\_coeff()**

```
cln::cl_N GiNaC::integration_kernel::series_coeff (
    int i ) const
```

Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_`↔`series`.

The method `series_coeff_impl` can be used, if a single coefficient can be computed independently of the others.

The method `Laurent_series` can be used, if it is more efficient to compute a Laurent series in one shot and to determine a range of coefficients from this Laurent series.

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::evalf\(\)](#), and `x`.

### 6.77.2.12 `get_numerical_value_impl()`

```
ex GiNaC::integration_kernel::get_numerical_value_impl (
    const ex & lambda,
    const ex & pre,
    int shift,
    int N_trunc ) const [protected]
```

The actual implementation for computing a numerical value for the integrand.

References [GiNaC::Digits](#), [GiNaC::ex::evalf\(\)](#), and [one](#).

Referenced by [GiNaC::ELi\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::modular\\_form\\_kernel::get\\_numerical\\_value\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::get\\_numerical\\_value\(\)](#).

### 6.77.2.13 `do_print()`

```
void GiNaC::integration_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#).

## 6.77.3 Member Data Documentation

### 6.77.3.1 `cache_step_size`

```
int GiNaC::integration_kernel::cache_step_size [mutable], [protected]
```

### 6.77.3.2 `series_vec`

```
std::vector<cln::cl_N> GiNaC::integration_kernel::series_vec [mutable], [protected]
```

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.78 `GiNaC::is_not_a_clifford` Struct Reference

Predicate for finding non-clifford objects.

## Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &e)

### 6.78.1 Detailed Description

Predicate for finding non-clifford objects.

### 6.78.2 Member Function Documentation

#### 6.78.2.1 [operator\(\)](#)

```
bool GiNaC::is_not_a_clifford::operator() (
    const ex & e ) [inline]
```

The documentation for this struct was generated from the following file:

- [clifford.cpp](#)

## 6.79 GiNaC::is\_summation\_idx Struct Reference

## Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &e)

### 6.79.1 Member Function Documentation

#### 6.79.1.1 [operator\(\)](#)

```
bool GiNaC::is_summation_idx::operator() (
    const ex & e ) [inline]
```

References [GiNaC::is\\_dummy\\_pair\(\)](#).

The documentation for this struct was generated from the following file:

- [indexed.cpp](#)

## 6.80 GiNaC::iterated\_integral2\_SERIAL Class Reference

Complete elliptic integral of the first kind.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.80.1 Detailed Description

Complete elliptic integral of the first kind.

Complete elliptic integral of the second kind. Iterated integral.

### 6.80.2 Member Data Documentation

#### 6.80.2.1 serial

```
unsigned GiNaC::iterated_integral2_SERIAL::serial [static]
```

**Initial value:**

```
=
    function::register_new(function_options("iterated_integral", 2).
        eval_func(iterated_integral2_eval).
        evalf_func(iterated_integral2_evalf).
        do_not_evalf_params().
        overloaded(2))
```

Referenced by [GiNaC::iterated\\_integral\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_elliptic.cpp](#)

## 6.81 GiNaC::iterated\_integral3\_SERIAL Class Reference

Iterated integral with explicit truncation.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.81.1 Detailed Description

Iterated integral with explicit truncation.

### 6.81.2 Member Data Documentation

#### 6.81.2.1 serial

```
unsigned GiNaC::iterated_integral3_SERIAL::serial [static]
```

**Initial value:**

```
=
    function::register_new(function_options("iterated_integral", 3).
        eval_func(iterated_integral3_eval).
        evalf_func(iterated_integral3_evalf).
        do_not_evalf_params().
        overloaded(2))
```

Referenced by [GiNaC::iterated\\_integral\(\)](#).

The documentation for this class was generated from the following files:

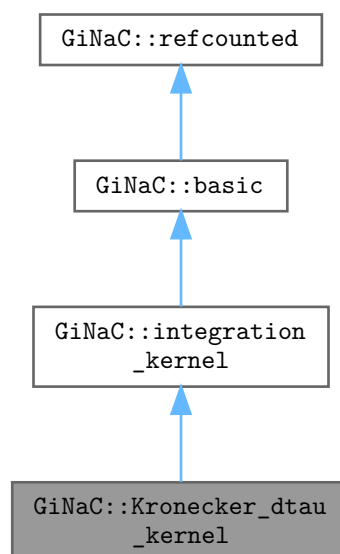
- [inifcns.h](#)
- [inifcns\\_elliptic.cpp](#)

## 6.82 GiNaC::Kronecker\_dtau\_kernel Class Reference

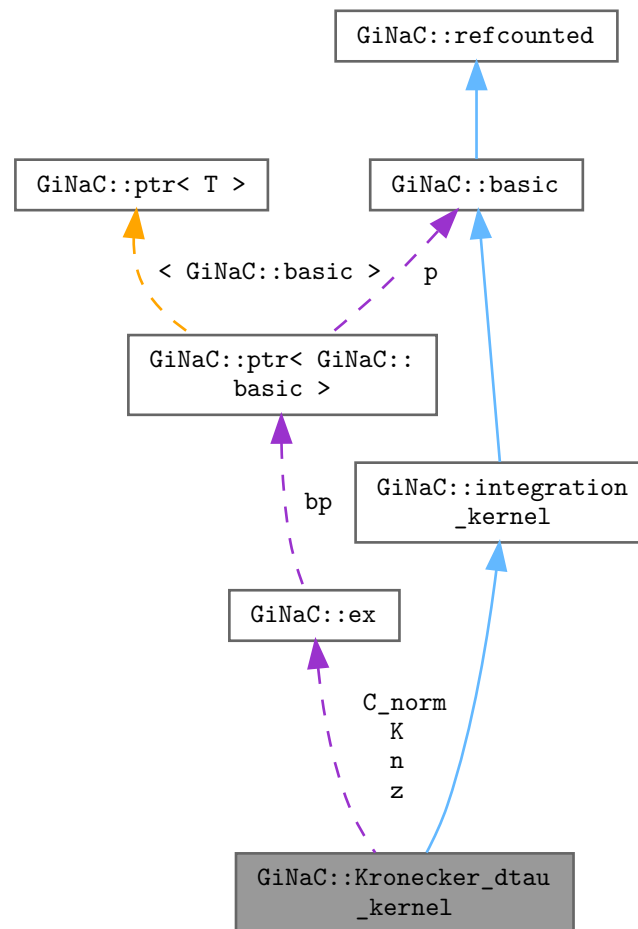
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Kronecker\_dtau\_kernel:



Collaboration diagram for `GiNaC::Kronecker_dtau_kernel`:



## Public Member Functions

- `Kronecker_dtau_kernel` (const `ex` &`n`, const `ex` &`z`, const `ex` &`K=numeric(1)`, const `ex` &`C_norm=numeric(1)`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex &let_op` (size\_t `i`) override  
*Return modifiable operand/member at position `i`.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc=0`) const override  
*Returns the value of the  $g^{\wedge}(n)(z, K*\tau)$ , where  $\tau$  is given by `qbar`.*

Public Member Functions inherited from [GiNaC::integration\\_kernel](#)

- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Default implementation of [ex::series\(\)](#).*
- virtual bool [has\\_trailing\\_zero](#) (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool [is\\_numeric](#) (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual [ex Laurent\\_series](#) (const [ex](#) &x, int [order](#)) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual [ex get\\_numerical\\_value](#) (const [ex](#) &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t [get\\_cache\\_size](#) (void) const  
*Returns the current size of the cache.*
- void [set\\_cache\\_step](#) (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- [ex get\\_series\\_coeff](#) (int i) const  
*Wrapper around [series\\_coeff\(i\)](#), converts [cl\\_N](#) to numeric.*
- [cln::cl\\_N series\\_coeff](#) (int i) const  
*Subclasses have either to implement [series\\_coeff\\_impl](#) or the two methods [Laurent\\_series](#) and [uses\\_Laurent\\_series](#).*

Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class [ex](#) will delete objects of derived classes via a [basic\\*](#).*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic \\* duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around [print](#) to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around [printtree](#) to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*

- virtual `size_t nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v)` const
- virtual `bool is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl)` const
- virtual `numeric integer_content ()` const
- virtual `ex smod (const numeric &xi)` const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*



- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- virtual `cln::cl_N series_coeff_impl` (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- [ex n](#)
- [ex z](#)
- [ex K](#)
- [ex C\\_norm](#)

### Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl\_N >` [series\\_vec](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.82.1 Detailed Description

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).

This class represents the differential one-form

$$\omega_{n,K}^{\text{Kronecker},\tau}(z_j) = \frac{C_n K(n-1)}{(2\pi i)^n} g^{(n)}(z_j, K\tau) \frac{d\bar{q}}{\bar{q}}$$

## 6.82.2 Constructor & Destructor Documentation

### 6.82.2.1 Kronecker\_dtau\_kernel()

```
GiNaC::Kronecker_dtau_kernel::Kronecker_dtau_kernel (
    const ex & n,
    const ex & z,
    const ex & K = numeric\(1\),
    const ex & C_norm = numeric\(1\) )
```

## 6.82.3 Member Function Documentation

### 6.82.3.1 nops()

```
size_t GiNaC::Kronecker_dtau_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.82.3.2 op()

```
ex GiNaC::Kronecker_dtau_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [K](#), [n](#), and [z](#).

### 6.82.3.3 let\_op()

```
ex & GiNaC::Kronecker_dtau_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [K](#), [n](#), and [z](#).

#### 6.82.3.4 is\_numeric()

```
bool GiNaC::Kronecker_dtau_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [K](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), and [z](#).

#### 6.82.3.5 get\_numerical\_value()

```
ex GiNaC::Kronecker_dtau_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the  $g^{(n)}(z, K \cdot \tau)$ , where  $\tau$  is given by  $qbar$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::l](#), [K](#), [n](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#),  $qbar$ , and [z](#).

#### 6.82.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::Kronecker_dtau_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::bernoulli\(\)](#), [C\\_norm](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::l](#), [K](#), [n](#), [GiNaC::Pi](#), [GiNaC::numeric::to\\_cl\\_N\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and [z](#).

#### 6.82.3.7 do\_print()

```
void GiNaC::Kronecker_dtau_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [C\\_norm](#), [K](#), [n](#), [GiNaC::ex::print\(\)](#), and [z](#).

## 6.82.4 Member Data Documentation

### 6.82.4.1 `n`

`ex` GiNaC::Kronecker\_dtau\_kernel::n [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.82.4.2 `z`

`ex` GiNaC::Kronecker\_dtau\_kernel::z [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.82.4.3 `K`

`ex` GiNaC::Kronecker\_dtau\_kernel::K [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.82.4.4 `C_norm`

`ex` GiNaC::Kronecker\_dtau\_kernel::C\_norm [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

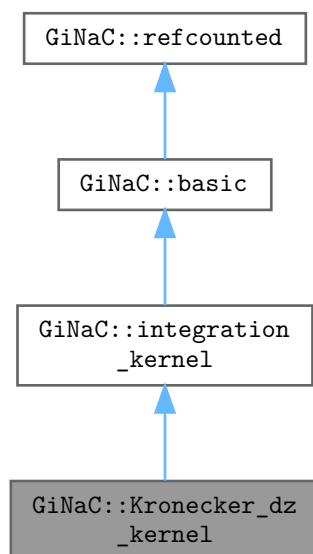
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.83 GiNaC::Kronecker\_dz\_kernel Class Reference

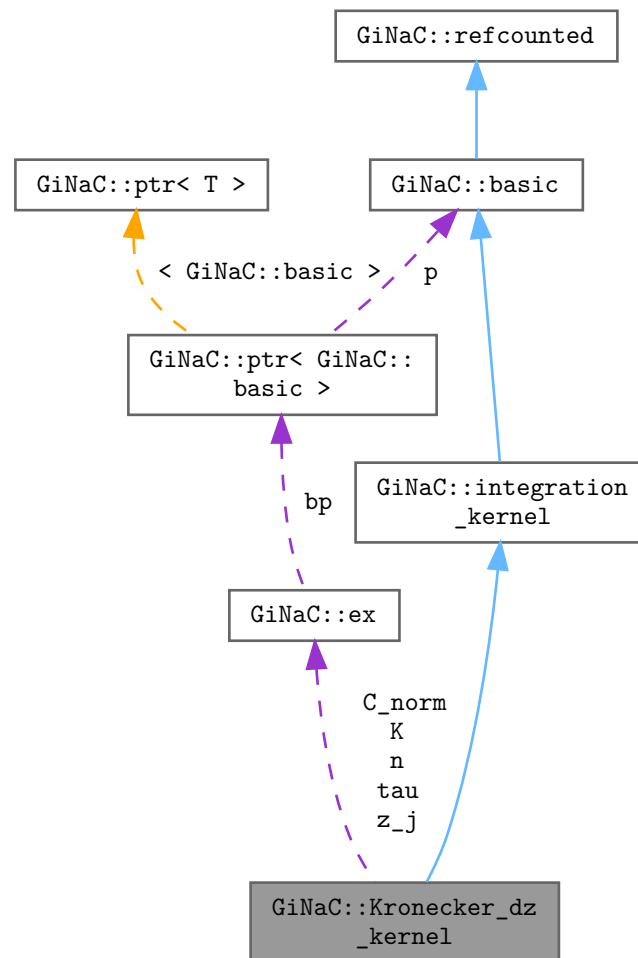
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Kronecker\_dz\_kernel:



Collaboration diagram for GiNaC::Kronecker\_dz\_kernel:



## Public Member Functions

- `Kronecker_dz_kernel` (const `ex` &`n`, const `ex` &`z_j`, const `ex` &`tau`, const `ex` &`K=numeric(1)`, const `ex` &`C_norm=numeric(1)`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position i.*
- `ex &let_op` (size\_t `i`) override  
*Return modifiable operand/member at position i.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`z`, int `N_trunc=0`) const override  
*Returns the value of the  $g^{(n-1)}(z-z_j, K*\tau)$ .*

## Public Member Functions inherited from [GiNaC::integration\\_kernel](#)

- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Default implementation of [ex::series\(\)](#).*
- virtual bool [has\\_trailing\\_zero](#) (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool [is\\_numeric](#) (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual [ex Laurent\\_series](#) (const [ex](#) &x, int [order](#)) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual [ex get\\_numerical\\_value](#) (const [ex](#) &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t [get\\_cache\\_size](#) (void) const  
*Returns the current size of the cache.*
- void [set\\_cache\\_step](#) (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- [ex get\\_series\\_coeff](#) (int i) const  
*Wrapper around [series\\_coeff\(i\)](#), converts [cl\\_N](#) to numeric.*
- [cln::cl\\_N series\\_coeff](#) (int i) const  
*Subclasses have either to implement [series\\_coeff\\_impl](#) or the two methods [Laurent\\_series](#) and [uses\\_Laurent\\_series](#).*

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class [ex](#) will delete objects of derived classes via a [basic\\*](#).*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic \\* duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around [print](#) to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around [printtree](#) to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*



- virtual `size_t nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v)` const
- virtual `bool is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl)` const
- virtual `numeric integer_content ()` const
- virtual `ex smod (const numeric &xi)` const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*

- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- virtual `cln::cl_N series_coeff_impl` (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex n](#)
- [ex z\\_j](#)
- [ex tau](#)
- [ex K](#)
- [ex C\\_norm](#)

Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl\_N >` [series\\_vec](#)

Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.83.1 Detailed Description

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .

This class represents the differential one-form

$$\omega_{n,K}^{\text{Kronecker},z}(z_j, \tau) = C_n (2\pi i)^{2-n} g^{(n-1)}(z - z_j, K\tau) dz$$

## 6.83.2 Constructor & Destructor Documentation

### 6.83.2.1 Kronecker\_dz\_kernel()

```
GiNaC::Kronecker_dz_kernel::Kronecker_dz_kernel (
    const ex & n,
    const ex & z_j,
    const ex & tau,
    const ex & K = numeric\(1\),
    const ex & C_norm = numeric\(1\) )
```

## 6.83.3 Member Function Documentation

### 6.83.3.1 nops()

```
size_t GiNaC::Kronecker_dz_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.83.3.2 op()

```
ex GiNaC::Kronecker_dz_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [K](#), [n](#), [tau](#), and [z\\_j](#).

### 6.83.3.3 let\_op()

```
ex & GiNaC::Kronecker_dz_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [K](#), [n](#), [tau](#), and [z\\_j](#).

#### 6.83.3.4 is\_numeric()

```
bool GiNaC::Kronecker_dz_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [K](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [tau](#), and [z\\_j](#).

#### 6.83.3.5 get\_numerical\_value()

```
ex GiNaC::Kronecker_dz_kernel::get_numerical_value (
    const ex & z,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the  $g^{(n-1)}(z-z_j, K \cdot \tau)$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::l](#), [n](#), [GiNaC::Pi](#), and [GiNaC::pow\(\)](#).

#### 6.83.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::Kronecker_dz_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::bernoulli\(\)](#), [C\\_norm](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::l](#), [GiNaC::is\\_even\(\)](#), [K](#), [n](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), [qbar](#), [tau](#), [GiNaC::numeric::to\\_int\(\)](#), and [z\\_j](#).

#### 6.83.3.7 do\_print()

```
void GiNaC::Kronecker_dz_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [C\\_norm](#), [K](#), [n](#), [GiNaC::ex::print\(\)](#), [tau](#), and [z\\_j](#).

## 6.83.4 Member Data Documentation

### 6.83.4.1 `n`

`ex` `GiNaC::Kronecker_dz_kernel::n` [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.83.4.2 `z_j`

`ex` `GiNaC::Kronecker_dz_kernel::z_j` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.83.4.3 `tau`

`ex` `GiNaC::Kronecker_dz_kernel::tau` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.83.4.4 `K`

`ex` `GiNaC::Kronecker_dz_kernel::K` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.83.4.5 `C_norm`

`ex` `GiNaC::Kronecker_dz_kernel::C_norm` [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.84 GiNaC::lanczos\_coeffs Class Reference

### Public Member Functions

- [lanczos\\_coeffs](#) ()
- bool [sufficiently\\_accurate](#) (int digits)
- int [get\\_order](#) () const
- cln::cl\_N [calc\\_lanczos\\_A](#) (const cln::cl\_N &) const

### Private Attributes

- std::vector< cln::cl\_N > \* [current\\_vector](#)

### Static Private Attributes

- static std::vector< cln::cl\_N > \* [coeffs](#) = nullptr

### 6.84.1 Constructor & Destructor Documentation

#### 6.84.1.1 lanczos\_coeffs()

GiNaC::lanczos\_coeffs::lanczos\_coeffs ( )

References [coeffs](#).

### 6.84.2 Member Function Documentation

#### 6.84.2.1 sufficiently\_accurate()

```
bool GiNaC::lanczos_coeffs::sufficiently_accurate (
    int digits )
```

References [coeffs](#), and [current\\_vector](#).

Referenced by [GiNaC::lgamma\(\)](#), and [GiNaC::tgamma\(\)](#).

### 6.84.2.2 `get_order()`

```
int GiNaC::lanczos_coeffs::get_order ( ) const [inline]
```

References [current\\_vector](#).

Referenced by [GiNaC::lgamma\(\)](#), and [GiNaC::tgamma\(\)](#).

### 6.84.2.3 `calc_lanczos_A()`

```
cln::cl_N GiNaC::lanczos_coeffs::calc_lanczos_A (
    const cln::cl_N & x ) const
```

References [current\\_vector](#), and [x](#).

Referenced by [GiNaC::lgamma\(\)](#), and [GiNaC::tgamma\(\)](#).

## 6.84.3 Member Data Documentation

### 6.84.3.1 `coeffs`

```
std::vector< cln::cl_N > * GiNaC::lanczos_coeffs::coeffs = nullptr [static], [private]
```

Referenced by [lanczos\\_coeffs\(\)](#), and [sufficiently\\_accurate\(\)](#).

### 6.84.3.2 `current_vector`

```
std::vector<cln::cl_N>* GiNaC::lanczos_coeffs::current_vector [private]
```

Referenced by [calc\\_lanczos\\_A\(\)](#), [get\\_order\(\)](#), and [sufficiently\\_accurate\(\)](#).

The documentation for this class was generated from the following file:

- [numeric.cpp](#)

## 6.85 `std::less< GiNaC::ptr< T > >` Struct Template Reference

Specialization of `std::less` for `ptr<T>` to enable ordering of `ptr<T>` objects (e.g.

```
#include <ptr.h>
```



## Public Member Functions

- `bool operator()` (const `GiNaC::ptr< T >` &lhs, const `GiNaC::ptr< T >` &rhs) const

### 6.85.1 Detailed Description

```
template<class T>
struct std::less< GiNaC::ptr< T > >
```

Specialization of `std::less` for `ptr<T>` to enable ordering of `ptr<T>` objects (e.g.

for the use as `std::map` keys).

### 6.85.2 Member Function Documentation

#### 6.85.2.1 operator()

```
template<class T >
bool std::less< GiNaC::ptr< T > >::operator() (
    const GiNaC::ptr< T > & lhs,
    const GiNaC::ptr< T > & rhs ) const [inline]
```

The documentation for this struct was generated from the following file:

- [ptr.h](#)

## 6.86 GiNaC::library\_init Class Reference

Helper class to initialize the library.

```
#include <ex.h>
```

## Public Member Functions

- `library_init()`  
*Ctor of static initialization helpers.*
- `~library_init()`  
*Dtor of static initialization helpers.*

## Static Private Member Functions

- static void `init_unarchivers()`

## Static Private Attributes

- static int [count](#) = 0

*How many static objects were created? Only the first one must create the static flyweights on the heap.*

### 6.86.1 Detailed Description

Helper class to initialize the library.

There must be one static object of this class in every object file that makes use of our flyweights in order to guarantee proper initialization. Hence we put it into this file which is included by every relevant file anyways. This is modeled after section [ios::Init] of the C++ standard, where cout and friends are set up.

See also

[utils.cpp](#)

### 6.86.2 Constructor & Destructor Documentation

#### 6.86.2.1 library\_init()

```
GiNaC::library_init::library_init ( )
```

Ctor of static initialization helpers.

The first call to this is going to initialize the library, the others do nothing.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex10](#), [GiNaC::\\_ex11](#), [GiNaC::\\_ex12](#), [GiNaC::\\_ex120](#), [GiNaC::\\_ex15](#), [GiNaC::\\_ex18](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex1\\_4](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex20](#), [GiNaC::\\_ex24](#), [GiNaC::\\_ex25](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex30](#), [GiNaC::\\_ex4](#), [GiNaC::\\_ex48](#), [GiNaC::\\_ex5](#), [GiNaC::\\_ex6](#), [GiNaC::\\_ex60](#), [GiNaC::\\_ex7](#), [GiNaC::\\_ex8](#), [GiNaC::\\_ex9](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_ex\\_10](#), [GiNaC::\\_ex\\_11](#), [GiNaC::\\_ex\\_12](#), [GiNaC::\\_ex\\_120](#), [GiNaC::\\_ex\\_15](#), [GiNaC::\\_ex\\_18](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_1\\_4](#), [GiNaC::\\_ex\\_2](#), [GiNaC::\\_ex\\_20](#), [GiNaC::\\_ex\\_24](#), [GiNaC::\\_ex\\_25](#), [GiNaC::\\_ex\\_3](#), [GiNaC::\\_ex\\_30](#), [GiNaC::\\_ex\\_4](#), [GiNaC::\\_ex\\_48](#), [GiNaC::\\_ex\\_5](#), [GiNaC::\\_ex\\_6](#), [GiNaC::\\_ex\\_60](#), [GiNaC::\\_ex\\_7](#), [GiNaC::\\_ex\\_8](#), [GiNaC::\\_ex\\_9](#), [GiNaC::\\_num0\\_bp](#), [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num120\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num15\\_p](#), [GiNaC::\\_num18\\_p](#), [GiNaC::\\_num1\\_2\\_p](#), [GiNaC::\\_num1\\_3\\_p](#), [GiNaC::\\_num1\\_4\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num20\\_p](#), [GiNaC::\\_num24\\_p](#), [GiNaC::\\_num25\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num30\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num48\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num60\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), [GiNaC::\\_num\\_10\\_p](#), [GiNaC::\\_num\\_11\\_p](#), [GiNaC::\\_num\\_120\\_p](#), [GiNaC::\\_num\\_12\\_p](#), [GiNaC::\\_num\\_15\\_p](#), [GiNaC::\\_num\\_18\\_p](#), [GiNaC::\\_num\\_1\\_2\\_p](#), [GiNaC::\\_num\\_1\\_3\\_p](#), [GiNaC::\\_num\\_1\\_4\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::\\_num\\_20\\_p](#), [GiNaC::\\_num\\_24\\_p](#), [GiNaC::\\_num\\_25\\_p](#), [GiNaC::\\_num\\_2\\_p](#), [GiNaC::\\_num\\_30\\_p](#), [GiNaC::\\_num\\_3\\_p](#), [GiNaC::\\_num\\_48\\_p](#), [GiNaC::\\_num\\_4\\_p](#), [GiNaC::\\_num\\_5\\_p](#), [GiNaC::\\_num\\_60\\_p](#), [GiNaC::\\_num\\_6\\_p](#), [GiNaC::\\_num\\_7\\_p](#), [GiNaC::\\_num\\_8\\_p](#), [GiNaC::\\_num\\_9\\_p](#), and [count](#).

### 6.86.2.2 ~library\_init()

```
GiNaC::library_init::~~library_init ( )
```

Dtor of static initialization helpers.

The last call to this is going to shut down the library, the others do nothing.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex10](#), [GiNaC::\\_ex11](#), [GiNaC::\\_ex12](#), [GiNaC::\\_ex120](#), [GiNaC::\\_ex15](#), [GiNaC::\\_ex18](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex1\\_4](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex20](#), [GiNaC::\\_ex24](#), [GiNaC::\\_ex25](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex30](#), [GiNaC::\\_ex4](#), [GiNaC::\\_ex48](#), [GiNaC::\\_ex5](#), [GiNaC::\\_ex6](#), [GiNaC::\\_ex60](#), [GiNaC::\\_ex7](#), [GiNaC::\\_ex8](#), [GiNaC::\\_ex9](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_ex\\_10](#), [GiNaC::\\_ex\\_11](#), [GiNaC::\\_ex\\_12](#), [GiNaC::\\_ex\\_120](#), [GiNaC::\\_ex\\_15](#), [GiNaC::\\_ex\\_18](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_1\\_4](#), [GiNaC::\\_ex\\_2](#), [GiNaC::\\_ex\\_20](#), [GiNaC::\\_ex\\_24](#), [GiNaC::\\_ex\\_25](#), [GiNaC::\\_ex\\_3](#), [GiNaC::\\_ex\\_30](#), [GiNaC::\\_ex\\_4](#), [GiNaC::\\_ex\\_48](#), [GiNaC::\\_ex\\_5](#), [GiNaC::\\_ex\\_6](#), [GiNaC::\\_ex\\_60](#), [GiNaC::\\_ex\\_7](#), [GiNaC::\\_ex\\_8](#), [GiNaC::\\_ex\\_9](#), and [count](#).

## 6.86.3 Member Function Documentation

### 6.86.3.1 init\_unarchivers()

```
void GiNaC::library_init::init_unarchivers ( ) [static], [private]
```

## 6.86.4 Member Data Documentation

### 6.86.4.1 count

```
int GiNaC::library_init::count = 0 [static], [private]
```

How many static objects were created? Only the first one must create the static flyweights on the heap.

Referenced by [library\\_init\(\)](#), and [~library\\_init\(\)](#).

The documentation for this class was generated from the following files:

- [ex.h](#)
- [utils.cpp](#)

## 6.87 GiNaC::make\_flat\_inserter Class Reference

Class to handle the renaming of dummy indices.

```
#include <expairseq.h>
```

## Public Member Functions

- [make\\_flat\\_inserter](#) (const [epvector](#) &epv, bool b)
- [make\\_flat\\_inserter](#) (const [exvector](#) &v, bool b)
- [ex\\_handle\\_factor](#) (const [ex](#) &x, const [ex](#) &coeff)

## Private Member Functions

- void [combine\\_indices](#) (const [exvector](#) &dummies\_of\_factor)

## Private Attributes

- bool [do\\_renaming](#)
- [exvector](#) [used\\_indices](#)

### 6.87.1 Detailed Description

Class to handle the renaming of dummy indices.

It holds a vector of indices that are being used in the expression so far. If the same index occurs again as a dummy index in a factor, it is to be renamed. Unless dummy index renaming was switched off, of course ;-)

### 6.87.2 Constructor & Destructor Documentation

#### 6.87.2.1 [make\\_flat\\_inserter\(\)](#) [1/2]

```
GiNaC::make_flat_inserter::make_flat_inserter (
    const epvector & epv,
    bool b ) [inline]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [combine\\_indices\(\)](#), and [do\\_renaming](#).

#### 6.87.2.2 [make\\_flat\\_inserter\(\)](#) [2/2]

```
GiNaC::make_flat_inserter::make_flat_inserter (
    const exvector & v,
    bool b ) [inline]
```

References [combine\\_indices\(\)](#), and [do\\_renaming](#).

### 6.87.3 Member Function Documentation

### 6.87.3.1 handle\_factor()

```
ex GiNaC::make_flat_inserter::handle_factor (
    const ex & x,
    const ex & coeff ) [inline]
```

References [GiNaC::coeff\(\)](#), [combine\\_indices\(\)](#), [do\\_renaming](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [used\\_indices](#), and [x](#).

Referenced by [GiNaC::ncmul::eval\(\)](#), and [GiNaC::expairseq::make\\_flat\(\)](#).

### 6.87.3.2 combine\_indices()

```
void GiNaC::make_flat_inserter::combine_indices (
    const exvector & dummies_of_factor ) [inline], [private]
```

References [used\\_indices](#).

Referenced by [handle\\_factor\(\)](#), and [make\\_flat\\_inserter\(\)](#).

## 6.87.4 Member Data Documentation

### 6.87.4.1 do\_renaming

```
bool GiNaC::make_flat_inserter::do_renaming [private]
```

Referenced by [handle\\_factor\(\)](#), and [make\\_flat\\_inserter\(\)](#).

### 6.87.4.2 used\_indices

```
exvector GiNaC::make_flat_inserter::used_indices [private]
```

Referenced by [combine\\_indices\(\)](#), and [handle\\_factor\(\)](#).

The documentation for this class was generated from the following file:

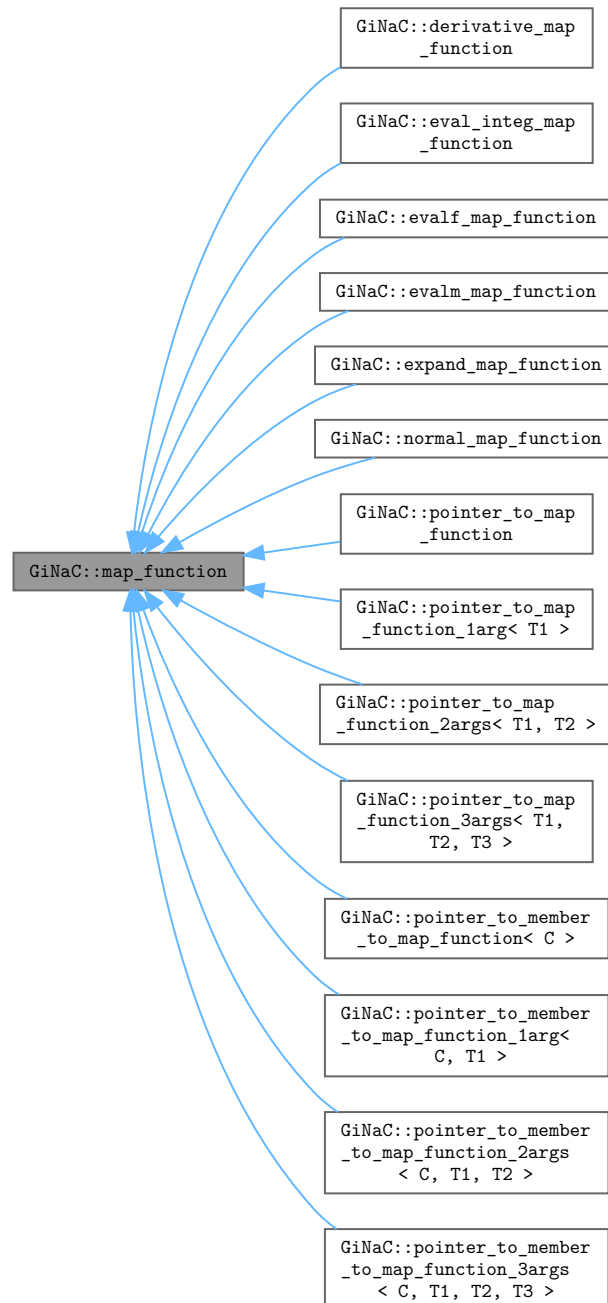
- [expairseq.h](#)

## 6.88 GiNaC::map\_function Struct Reference

Function object for map().

```
#include <basic.h>
```

Inheritance diagram for GiNaC::map\_function:



## Public Types

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## Public Member Functions

- virtual [~map\\_function](#) ()
- virtual [ex operator\(\)](#) (const [ex](#) &*e*)=0

### 6.88.1 Detailed Description

Function object for `map()`.

### 6.88.2 Member Typedef Documentation

#### 6.88.2.1 `argument_type`

```
typedef const ex& GiNaC::map\_function::argument\_type
```

#### 6.88.2.2 `result_type`

```
typedef ex GiNaC::map\_function::result\_type
```

### 6.88.3 Constructor & Destructor Documentation

#### 6.88.3.1 `~map_function()`

```
virtual GiNaC::map\_function::~~map\_function ( ) [inline], [virtual]
```

### 6.88.4 Member Function Documentation

#### 6.88.4.1 operator()

```
virtual ex GiNaC::map_function::operator() (
    const ex & e ) [pure virtual]
```

Implemented in [GiNaC::evalf\\_map\\_function](#), [GiNaC::evalm\\_map\\_function](#), [GiNaC::eval\\_integ\\_map\\_function](#), [GiNaC::derivative\\_map\\_function](#), [GiNaC::expand\\_map\\_function](#), [GiNaC::pointer\\_to\\_map\\_function](#), [GiNaC::pointer\\_to\\_map\\_function](#), [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >](#), [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2 >](#) and [GiNaC::normal\\_map\\_function](#).

The documentation for this struct was generated from the following file:

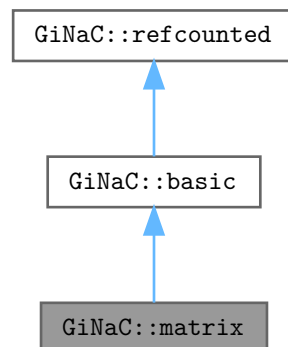
- [basic.h](#)

## 6.89 GiNaC::matrix Class Reference

Symbolic matrices.

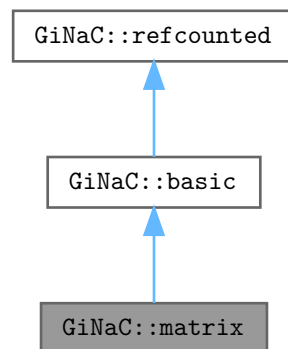
```
#include <matrix.h>
```

Inheritance diagram for GiNaC::matrix:





Collaboration diagram for GiNaC::matrix:



## Public Member Functions

- [matrix](#) (unsigned [r](#), unsigned [c](#))  
*Very common ctor.*
- [matrix](#) (unsigned [r](#), unsigned [c](#), const [lst](#) &l)  
*Construct matrix from (flat) list of elements.*
- [matrix](#) (std::initializer\_list< std::initializer\_list< [ex](#) > > l)  
*Construct a matrix from an 2 dimensional initializer list.*
- [size\\_t nops](#) () const override  
*nops is defined to be rows x columns.*
- [ex op](#) (size\_t i) const override  
*returns matrix entry at position (i/col, icol).*
- [ex & let\\_op](#) (size\_t i) override  
*returns writable matrix entry at position (i/col, icol).*
- [ex evalm](#) () const override  
*Evaluate sums, products and integer powers of matrices.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex eval\\_indexed](#) (const [basic](#) &i) const override  
*Automatic symbolic evaluation of an indexed matrix.*
- [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const override  
*Sum of two indexed matrices.*
- [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const override  
*Product of an indexed matrix with a number.*
- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of an indexed matrix with something else.*
- [ex conjugate](#) () const override  
*Complex conjugate every matrix entry.*
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- void [archive](#) ([archive\\_node](#) &n) const override

- *Save (a.k.a.*
  - void `read_archive` (const `archive_node` &n, `lst` &`syms`) override
- *Read (a.k.a.*
  - unsigned `rows` () const
- *Get number of rows.*
  - unsigned `cols` () const
- *Get number of columns.*
  - `matrix add` (const `matrix` &other) const
- *Sum of matrices.*
  - `matrix sub` (const `matrix` &other) const
- *Difference of matrices.*
  - `matrix mul` (const `matrix` &other) const
- *Product of matrices.*
  - `matrix mul` (const `numeric` &other) const
- *Product of matrix and scalar.*
  - `matrix mul_scalar` (const `ex` &other) const
- *Product of matrix and scalar expression.*
  - `matrix pow` (const `ex` &expn) const
- *Power of a matrix.*
  - const `ex` & `operator()` (unsigned ro, unsigned co) const
- *operator() to access elements for reading.*
  - `ex` & `operator()` (unsigned ro, unsigned co)
- *operator() to access elements for writing.*
  - `matrix` & `set` (unsigned ro, unsigned co, const `ex` &value)
- `matrix transpose` () const
- *Transposed of an  $m \times n$  matrix, producing a new  $n \times m$  matrix object that represents the transposed.*
  - `ex determinant` (unsigned algo=`determinant_algo::automatic`) const
- *Determinant of square matrix.*
  - `ex trace` () const
- *Trace of a matrix.*
  - `ex charpoly` (const `ex` &lambda) const
- *Characteristic Polynomial.*
  - `matrix inverse` () const
- *Inverse of this matrix, with automatic algorithm selection.*
  - `matrix inverse` (unsigned algo) const
- *Inverse of this matrix.*
  - `matrix solve` (const `matrix` &vars, const `matrix` &rhs, unsigned algo=`solve_algo::automatic`) const
- *Solve a linear system consisting of a  $m \times n$  matrix and a  $m \times p$  right hand side by applying an elimination scheme to the augmented matrix.*
  - unsigned `rank` () const
- *Compute the rank of this matrix.*
  - unsigned `rank` (unsigned `solve_algo`) const
- *Compute the rank of this matrix using the given algorithm, which should be a member of enum `solve_algo`.*
  - bool `is_zero_matrix` () const
- *Function to check that all elements of the matrix are zero.*

Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const*

  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- [matrix](#) (unsigned r, unsigned c, const [exvector](#) &m2)  
*Ctor from representation, for internal use only.*
- [matrix](#) (unsigned r, unsigned c, [exvector](#) &&m2)
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned [return\\_type](#) () const override
- [ex\\_determinant\\_minor](#) () const  
*Recursive determinant for small matrices having at least one symbolic entry.*
- std::vector< unsigned > [echelon\\_form](#) (unsigned algo, int n)
- int [gauss\\_elimination](#) (const bool det=false)  
*Perform the steps of an ordinary Gaussian elimination to bring the m x n matrix into an upper echelon form.*
- int [division\\_free\\_elimination](#) (const bool det=false)  
*Perform the steps of division free elimination to bring the m x n matrix into an upper echelon form.*
- int [fraction\\_free\\_elimination](#) (const bool det=false)  
*Perform the steps of Bareiss' one-step fraction free elimination to bring the matrix into an upper echelon form.*
- std::vector< unsigned > [markowitz\\_elimination](#) (unsigned n)
- int [pivot](#) (unsigned ro, unsigned co, bool symbolic=true)  
*Partial pivoting method for matrix elimination schemes.*
- void [print\\_elements](#) (const [print\\_context](#) &c, const char \*row\_start, const char \*row\_end, const char \*row←  
\_sep, const char \*col\_sep) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- unsigned [row](#)  
*number of rows*
- unsigned [col](#)  
*number of columns*
- [exvector](#) m  
*representation (cols indexed first)*

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.89.1 Detailed Description

Symbolic matrices.

## 6.89.2 Constructor & Destructor Documentation

### 6.89.2.1 matrix() [1/5]

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c )
```

Very common ctor.

Initializes to r x c-dimensional zero-matrix.

## Parameters

<i>r</i>	number of rows
<i>c</i>	number of cols

References [GiNaC::\\_ex0](#), [GiNaC::status\\_flags::not\\_shareable](#), and [GiNaC::basic::setflag\(\)](#).

**6.89.2.2 matrix()** [2/5]

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c,
    const lst & l )
```

Construct matrix from (flat) list of elements.

If the list has fewer elements than the matrix, the remaining matrix elements are set to zero. If the list has more elements than the matrix, the excessive elements are thrown away.

References [c](#), [m](#), [GiNaC::status\\_flags::not\\_shareable](#), [r](#), [GiNaC::basic::setflag\(\)](#), and [x](#).

**6.89.2.3 matrix()** [3/5]

```
GiNaC::matrix::matrix (
    std::initializer_list< std::initializer_list< ex > > l )
```

Construct a matrix from an 2 dimensional initializer list.

Throws an exception if some row has a different length than all the others.

References [c](#), [col](#), [m](#), [GiNaC::status\\_flags::not\\_shareable](#), [r](#), [row](#), and [GiNaC::basic::setflag\(\)](#).

**6.89.2.4 matrix()** [4/5]

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c,
    const exvector & m2 ) [protected]
```

Ctor from representation, for internal use only.

References [GiNaC::status\\_flags::not\\_shareable](#), and [GiNaC::basic::setflag\(\)](#).



### 6.89.2.5 matrix() [5/5]

```
GiNaC::matrix::matrix (
    unsigned r,
    unsigned c,
    exvector && m2 ) [protected]
```

References [GiNaC::status\\_flags::not\\_shareable](#), and [GiNaC::basic::setflag\(\)](#).

## 6.89.3 Member Function Documentation

### 6.89.3.1 nops()

```
size_t GiNaC::matrix::nops ( ) const [override], [virtual]
```

nops is defined to be rows x columns.

Reimplemented from [GiNaC::basic](#).

References [col](#), and [row](#).

Referenced by [let\\_op\(\)](#), and [op\(\)](#).

### 6.89.3.2 op()

```
ex GiNaC::matrix::op (
    size_t i ) const [override], [virtual]
```

returns matrix entry at position (i/col, icol).

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [m](#), and [nops\(\)](#).

Referenced by [contract\\_with\(\)](#).

### 6.89.3.3 let\_op()

```
ex & GiNaC::matrix::let_op (
    size_t i ) [override], [virtual]
```

returns writable matrix entry at position (i/col, icol).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [m](#), and [nops\(\)](#).

#### 6.89.3.4 evalm()

```
ex GiNaC::matrix::evalm ( ) const [inline], [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

#### 6.89.3.5 subs()

```
ex GiNaC::matrix::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [c](#), [col](#), [m](#), [options](#), [r](#), [row](#), [subs\(\)](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

Referenced by [fraction\\_free\\_elimination\(\)](#), and [subs\(\)](#).

#### 6.89.3.6 eval\_indexed()

```
ex GiNaC::matrix::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed matrix.

Reimplemented from [GiNaC::basic](#).

References [col](#), [GiNaC::idx::get\\_dim\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [row](#), and [trace\(\)](#).

#### 6.89.3.7 add\_indexed()

```
ex GiNaC::matrix::add_indexed (
    const ex & self,
    const ex & other ) const [override], [virtual]
```

Sum of two indexed matrices.

Reimplemented from [GiNaC::basic](#).

References [add\(\)](#), [col](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [row](#), and [transpose\(\)](#).

#### 6.89.3.8 scalar\_mul\_indexed()

```
ex GiNaC::matrix::scalar_mul_indexed (
    const ex & self,
    const numeric & other ) const [override], [virtual]
```

Product of an indexed matrix with a number.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [mul\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

#### 6.89.3.9 contract\_with()

```
bool GiNaC::matrix::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed matrix with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [col](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [mul\(\)](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), [row](#), and [transpose\(\)](#).

#### 6.89.3.10 conjugate()

```
ex GiNaC::matrix::conjugate ( ) const [override], [virtual]
```

Complex conjugate every matrix entry.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [col](#), [m](#), [row](#), and [x](#).

#### 6.89.3.11 real\_part()

```
ex GiNaC::matrix::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), and [row](#).

**6.89.3.12 imag\_part()**

```
ex GiNaC::matrix::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), and [row](#).

**6.89.3.13 archive()**

```
void GiNaC::matrix::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), [n](#), and [row](#).

**6.89.3.14 read\_archive()**

```
void GiNaC::matrix::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), [n](#), and [row](#).

**6.89.3.15 match\_same\_type()**

```
bool GiNaC::matrix::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [col](#), [cols\(\)](#), [GINAC\\_ASSERT](#), [row](#), and [rows\(\)](#).

**6.89.3.16 return\_type()**

```
unsigned GiNaC::matrix::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

**6.89.3.17 rows()**

```
unsigned GiNaC::matrix::rows ( ) const [inline]
```

Get number of rows.

References [row](#).

Referenced by [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [gauss\\_elimination\(\)](#), [GiNaC::lsolve\(\)](#), [match\\_same\\_type\(\)](#), [mul\(\)](#), [solve\(\)](#), and [transpose\(\)](#).

**6.89.3.18 cols()**

```
unsigned GiNaC::matrix::cols ( ) const [inline]
```

Get number of columns.

References [col](#).

Referenced by [determinant\\_minor\(\)](#), [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [gauss\\_elimination\(\)](#), [GiNaC::lsolve\(\)](#), [match\\_same\\_type\(\)](#), [mul\(\)](#), [solve\(\)](#), and [transpose\(\)](#).

**6.89.3.19 add()**

```
matrix GiNaC::matrix::add (
    const matrix & other ) const
```

Sum of matrices.

**Exceptions**

<i>logic_error</i>	(incompatible matrices)
--------------------	-------------------------

References [col](#), [m](#), and [row](#).

Referenced by [add\\_indexed\(\)](#), and [GiNaC::add::evalm\(\)](#).

**6.89.3.20 sub()**

```
matrix GiNaC::matrix::sub (
    const matrix & other ) const
```

Difference of matrices.

**Exceptions**

<i>logic_error</i>	(incompatible matrices)
--------------------	-------------------------

References [col](#), [m](#), and [row](#).

**6.89.3.21 mul() [1/2]**

```
matrix GiNaC::matrix::mul (
    const matrix & other ) const
```

Product of matrices.

**Exceptions**

<i>logic_error</i>	(incompatible matrices)
--------------------	-------------------------

References [c](#), [col](#), [cols\(\)](#), [GiNaC::is\\_zero\(\)](#), [m](#), [row](#), and [rows\(\)](#).

Referenced by [charpoly\(\)](#), [contract\\_with\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [pow\(\)](#), and [scalar\\_mul\\_indexed\(\)](#).

**6.89.3.22 mul() [2/2]**

```
matrix GiNaC::matrix::mul (
    const numeric & other ) const
```

Product of matrix and scalar.

References [c](#), [col](#), [m](#), [r](#), and [row](#).

**6.89.3.23 mul\_scalar()**

```
matrix GiNaC::matrix::mul_scalar (
    const ex & other ) const
```

Product of matrix and scalar expression.

References [c](#), [col](#), [GiNaC::return\\_types::commutative](#), [m](#), [r](#), [GiNaC::ex::return\\_type\(\)](#), and [row](#).

**6.89.3.24 pow()**

```
matrix GiNaC::matrix::pow (
    const ex & expn ) const
```

Power of a matrix.

Currently handles integer exponents only.

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [col](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [inverse\(\)](#), [GiNaC::numeric::is\\_odd\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [mul\(\)](#), [GiNaC::info\\_flags::negative](#), [r](#), and [row](#).

**6.89.3.25 operator>() [1/2]**

```
const ex & GiNaC::matrix::operator() (
    unsigned ro,
    unsigned co ) const
```

`operator()` to access elements for reading.

**Parameters**

<i>ro</i>	row of element
<i>co</i>	column of element

**Exceptions**

<i>range_error</i>	(index out of range)
--------------------	----------------------

References [col](#), [m](#), and [row](#).

**6.89.3.26 operator>() [2/2]**

```
ex & GiNaC::matrix::operator() (
    unsigned ro,
    unsigned co )
```

`operator()` to access elements for writing.

**Parameters**

<i>ro</i>	row of element
<i>co</i>	column of element

## Exceptions

<code>range_error</code>	(index out of range)
--------------------------	----------------------

References [col](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [m](#), and [row](#).

**6.89.3.27 set()**

```
matrix & GiNaC::matrix::set (
    unsigned ro,
    unsigned co,
    const ex & value ) [inline]
```

References [value](#).

**6.89.3.28 transpose()**

```
matrix GiNaC::matrix::transpose ( ) const
```

Transposed of an m x n matrix, producing a new n x m matrix object that represents the transposed.

References [c](#), [cols\(\)](#), [m](#), [r](#), and [rows\(\)](#).

Referenced by [add\\_indexed\(\)](#), and [contract\\_with\(\)](#).

**6.89.3.29 determinant()**

```
ex GiNaC::matrix::determinant (
    unsigned algo = determinant_algo::automatic ) const
```

Determinant of square matrix.

This routine doesn't actually calculate the determinant, it only implements some heuristics about which algorithm to run. If all the elements of the matrix are elements of an integral domain the determinant is also in that integral domain and the result is expanded only. If one or more elements are from a quotient field the determinant is usually also in that quotient field and the result is normalized before it is returned. This implies that the determinant of the symbolic 2x2 matrix  $\begin{bmatrix} a/(a-b) & 1 \\ b/(a-b) & 1 \end{bmatrix}$  is returned as unity. (In this respect, it behaves like MapleV and unlike Mathematica.)

## Parameters

<code>algo</code>	allows to chose an algorithm
-------------------	------------------------------



**Returns**

the determinant as a new expression

**Exceptions**

<i>logic_error</i>	(matrix not square)
--------------------	---------------------

**See also**

[determinant\\_algo](#)

References [GiNaC::\\_ex0](#), [GiNaC::determinant\\_algo::automatic](#), [GiNaC::determinant\\_algo::bareiss](#), [c](#), [col](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [determinant\\_minor\(\)](#), [GiNaC::determinant\\_algo::divfree](#), [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [GiNaC::determinant\\_algo::gauss](#), [gauss\\_elimination\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [GiNaC::determinant\\_algo::laplace](#), [m](#), [GiNaC::ex::normal\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::permutation\\_sign\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_function](#), and [row](#).

Referenced by [charpoly\(\)](#), and [GiNaC::tensepsilon::contract\\_with\(\)](#).

**6.89.3.30 trace()**

```
ex GiNaC::matrix::trace ( ) const
```

Trace of a matrix.

The result is normalized if it is in some quotient field and expanded only otherwise. This implies that the trace of the symbolic 2x2 matrix  $\begin{bmatrix} a/(a-b), x \\ y, b/(b-a) \end{bmatrix}$  is recognized to be unity.

**Returns**

the sum of diagonal elements

**Exceptions**

<i>logic_error</i>	(matrix not square)
--------------------	---------------------

References [col](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [GiNaC::ex::normal\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_function](#), and [row](#).

Referenced by [charpoly\(\)](#), and [eval\\_indexed\(\)](#).

**6.89.3.31 charpoly()**

```
ex GiNaC::matrix::charpoly (
    const ex & lambda ) const
```

Characteristic Polynomial.

Following mathematica notation the characteristic polynomial of a matrix  $M$  is defined as the determinant of  $(M - \lambda * 1)$  where  $1$  stands for the unit matrix of the same dimension as  $M$ . Note that some CASs define it with a sign inside the determinant which gives rise to an overall sign if the dimension is odd. This method returns the characteristic polynomial collected in powers of  $\lambda$  as a new expression.

#### Returns

characteristic polynomial as new expression

#### Exceptions

<i>logic_error</i>	(matrix not square)
--------------------	---------------------

#### See also

[matrix::determinant\(\)](#)

References [c](#), [col](#), [GiNaC::ex::collect\(\)](#), [determinant\(\)](#), [GiNaC::basic::ex](#), [m](#), [mul\(\)](#), [GiNaC::info\\_flags::numeric](#), [poly](#), [r](#), [row](#), and [trace\(\)](#).

### 6.89.3.32 `inverse()` [1/2]

```
matrix GiNaC::matrix::inverse ( ) const
```

Inverse of this matrix, with automatic algorithm selection.

References [GiNaC::solve\\_algo::automatic](#), and [inverse\(\)](#).

Referenced by [inverse\(\)](#), and [pow\(\)](#).

### 6.89.3.33 `inverse()` [2/2]

```
matrix GiNaC::matrix::inverse (
    unsigned algo ) const
```

Inverse of this matrix.

#### Parameters

<i>algo</i>	selects the algorithm (one of <a href="#">solve_algo</a> )
-------------	--

**Returns**

the inverted matrix

**Exceptions**

<i>logic_error</i>	(matrix not square)
<i>runtime_error</i>	(singular matrix)

References [GiNaC::\\_ex1](#), [c](#), [col](#), [r](#), [row](#), and [solve\(\)](#).

**6.89.3.34 solve()**

```
matrix GiNaC::matrix::solve (
    const matrix & vars,
    const matrix & rhs,
    unsigned algo = solve_algo::automatic ) const
```

Solve a linear system consisting of a  $m \times n$  matrix and a  $m \times p$  right hand side by applying an elimination scheme to the augmented matrix.

**Parameters**

<i>vars</i>	$n \times p$ matrix, all elements must be symbols
<i>rhs</i>	$m \times p$ matrix
<i>algo</i>	selects the solving algorithm

**Returns**

$n \times p$  solution matrix

**Exceptions**

<i>logic_error</i>	(incompatible matrices)
<i>invalid_argument</i>	(1st argument must be matrix of symbols)
<i>runtime_error</i>	(inconsistent linear system)

**See also**

[solve\\_algo](#)

References [c](#), [cols\(\)](#), [echelon\\_form\(\)](#), [GiNaC::basic::info\(\)](#), [m](#), [n](#), [GiNaC::basic::normal\(\)](#), [r](#), [GiNaC::rhs\(\)](#), [rows\(\)](#), and [GiNaC::info\\_flags::symbol](#).

Referenced by [inverse\(\)](#), [GiNaC::lsolve\(\)](#), and [GiNaC::sqrfree\\_parfrac\(\)](#).

**6.89.3.35 rank()** [1/2]

```
unsigned GiNaC::matrix::rank ( ) const
```

Compute the rank of this matrix.

References [GiNaC::solve\\_algo::automatic](#), and [rank\(\)](#).

Referenced by [rank\(\)](#).

**6.89.3.36 rank()** [2/2]

```
unsigned GiNaC::matrix::rank (
    unsigned solve_algo ) const
```

Compute the rank of this matrix using the given algorithm, which should be a member of enum [solve\\_algo](#).

References [col](#), [echelon\\_form\(\)](#), [GINAC\\_ASSERT](#), [m](#), [r](#), and [row](#).

**6.89.3.37 is\_zero\_matrix()**

```
bool GiNaC::matrix::is_zero_matrix ( ) const
```

Function to check that all elements of the matrix are zero.

References [m](#).

**6.89.3.38 determinant\_minor()**

```
ex GiNaC::matrix::determinant_minor ( ) const [protected]
```

Recursive determinant for small matrices having at least one symbolic entry.

The basic algorithm, known as Laplace-expansion, is enhanced by some bookkeeping to avoid calculation of the same submatrices ("minors") more than once. According to W.M.Gentleman and S.C.Johnson this algorithm is better than elimination schemes for matrices of sparse multivariate polynomials and also for matrices of dense univariate polynomials if the matrix' dimension is larger than 7.

**Returns**

the determinant as a new expression (in expanded form)

**See also**

[matrix::determinant\(\)](#)

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [cols\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [m](#), [n](#), and [r](#).

Referenced by [determinant\(\)](#).

**6.89.3.39 echelon\_form()**

```
std::vector< unsigned > GiNaC::matrix::echelon_form (
    unsigned algo,
    int n ) [protected]
```

References [GiNaC::solve\\_algo::automatic](#), [GiNaC::solve\\_algo::bareiss](#), [c](#), [col](#), [GiNaC::solve\\_algo::divfree](#), [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [GiNaC::solve\\_algo::gauss](#), [gauss\\_elimination\(\)](#), [m](#), [GiNaC::solve\\_algo::markowitz](#), [markowitz\\_elimination\(\)](#), [n](#), [GiNaC::info\\_flags::numeric](#), [r](#), and [row](#).

Referenced by [rank\(\)](#), and [solve\(\)](#).

**6.89.3.40 gauss\_elimination()**

```
int GiNaC::matrix::gauss_elimination (
    const bool det = false ) [protected]
```

Perform the steps of an ordinary Gaussian elimination to bring the m x n matrix into an upper echelon form.

The algorithm is ok for matrices with numeric coefficients but quite unsuited for symbolic matrices.

**Parameters**

<i>det</i>	may be set to true to save a lot of space if one is only interested in the diagonal elements (i.e. for calculating determinants). The others are set to zero in this case.
------------	--

**Returns**

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References [GiNaC::ex0](#), [c](#), [cols\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::info\(\)](#), [GiNaC::is\\_zero\(\)](#), [m](#), [n](#), [GiNaC::ex::normal\(\)](#), [GiNaC::info\\_flags::numeric](#), [pivot\(\)](#), [r](#), and [rows\(\)](#).

Referenced by [determinant\(\)](#), and [echelon\\_form\(\)](#).

**6.89.3.41 division\_free\_elimination()**

```
int GiNaC::matrix::division_free_elimination (
    const bool det = false ) [protected]
```

Perform the steps of division free elimination to bring the m x n matrix into an upper echelon form.

**Parameters**

<i>det</i>	may be set to true to save a lot of space if one is only interested in the diagonal elements (i.e. for calculating determinants). The others are set to zero in this case.
------------	--

**Returns**

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References [GiNaC::\\_ex0](#), [c](#), [cols\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [m](#), [n](#), [pivot\(\)](#), [r](#), and [rows\(\)](#).

Referenced by [determinant\(\)](#), and [echelon\\_form\(\)](#).

**6.89.3.42 fraction\_free\_elimination()**

```
int GiNaC::matrix::fraction_free_elimination (
    const bool det = false ) [protected]
```

Perform the steps of Bareiss' one-step fraction free elimination to bring the matrix into an upper echelon form.

Fraction free elimination means that divide is used straightforwardly, without computing GCDs first. This is possible, since we know the divisor at each step.

**Parameters**

<i>det</i>	may be set to true to save a lot of space if one is only interested in the last element (i.e. for calculating determinants). The others are set to zero in this case.
------------	---

**Returns**

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [cols\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::expand\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_zero\(\)](#), [m](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::normal\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::ex::op\(\)](#), [r](#), [rows\(\)](#), [subs\(\)](#), and [GiNaC::ex::to\\_rational\(\)](#).

Referenced by [determinant\(\)](#), and [echelon\\_form\(\)](#).

**6.89.3.43 markowitz\_elimination()**

```
std::vector< unsigned > GiNaC::matrix::markowitz_elimination (
    unsigned n ) [protected]
```

References [GiNaC::\\_ex0](#), [c](#), [col](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [k](#), [m](#), [n](#), [GiNaC::basic::normal\(\)](#), [r](#), [row](#), and [std::swap\(\)](#).

Referenced by [echelon\\_form\(\)](#).

**6.89.3.44 pivot()**

```
int GiNaC::matrix::pivot (
    unsigned ro,
    unsigned co,
    bool symbolic = true ) [protected]
```

Partial pivoting method for matrix elimination schemes.

Usual pivoting (`symbolic==false`) returns the index to the element with the largest absolute value in column `ro` and swaps the current row with the one where the element was found. With (`symbolic==true`) it does the same thing with the first non-zero element.

**Parameters**

<i>ro</i>	is the row from where to begin
<i>co</i>	is the column to be inspected
<i>symbolic</i>	signal if we want the first non-zero element to be pivoted (true) or the one with the largest absolute value (false).

**Returns**

0 if no interchange occurred, -1 if all are zero (usually signaling a degeneracy) and positive integer `k` means that rows `ro` and `k` were swapped.

References [GiNaC::abs\(\)](#), [c](#), [col](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::expand\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [k](#), [m](#), [row](#), and [GiNaC::swap\(\)](#).

Referenced by [division\\_free\\_elimination\(\)](#), and [gauss\\_elimination\(\)](#).

**6.89.3.45 print\_elements()**

```
void GiNaC::matrix::print_elements (
    const print_context & c,
    const char * row_start,
    const char * row_end,
    const char * row_sep,
    const char * col_sep ) const [protected]
```

References [c](#), [col](#), [m](#), and [row](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

**6.89.3.46 do\_print()**

```
void GiNaC::matrix::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_elements\(\)](#).

### 6.89.3.47 do\_print\_latex()

```
void GiNaC::matrix::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), [col](#), and [print\\_elements\(\)](#).

### 6.89.3.48 do\_print\_python\_repr()

```
void GiNaC::matrix::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_elements\(\)](#).

## 6.89.4 Member Data Documentation

### 6.89.4.1 row

```
unsigned GiNaC::matrix::row [protected]
```

number of rows

Referenced by [add\(\)](#), [add\\_indexed\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [conjugate\(\)](#), [contract\\_with\(\)](#), [determinant\(\)](#), [echelon\\_form\(\)](#), [eval\\_indexed\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [markowitz\\_elimination\(\)](#), [match\\_same\\_type\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [nops\(\)](#), [operator\(\)\(\)](#), [pivot\(\)](#), [pow\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [rows\(\)](#), [sub\(\)](#), [subs\(\)](#), and [trace\(\)](#).

### 6.89.4.2 col

```
unsigned GiNaC::matrix::col [protected]
```

number of columns

Referenced by [add\(\)](#), [add\\_indexed\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [cols\(\)](#), [conjugate\(\)](#), [contract\\_with\(\)](#), [determinant\(\)](#), [do\\_print\\_latex\(\)](#), [echelon\\_form\(\)](#), [eval\\_indexed\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [markowitz\\_elimination\(\)](#), [match\\_same\\_type\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [nops\(\)](#), [operator\(\)\(\)](#), [pivot\(\)](#), [pow\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [sub\(\)](#), [subs\(\)](#), and [trace\(\)](#).



## 6.89.4.3 m

`exvector` GiNaC::matrix::m [protected]

representation (cols indexed first)

Referenced by `add()`, `archive()`, `charpoly()`, `conjugate()`, `determinant()`, `determinant_minor()`, `division_free_elimination()`, `echelon_form()`, `fraction_free_elimination()`, `gauss_elimination()`, `imag_part()`, `is_zero_matrix()`, `let_op()`, `markowitz_elimination()`, `matrix()`, `mul()`, `mul_scalar()`, `op()`, `operator()`, `pivot()`, `print_elements()`, `rank()`, `read_archive()`, `real_part()`, `solve()`, `sub()`, `subs()`, `trace()`, and `transpose()`.

The documentation for this class was generated from the following files:

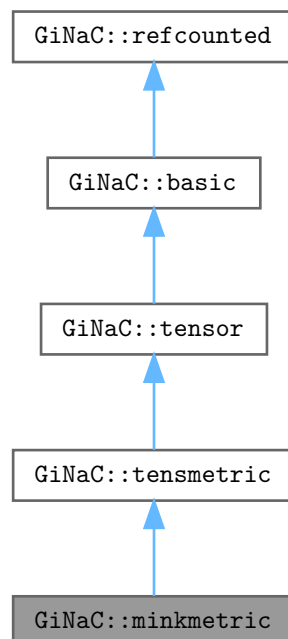
- [matrix.h](#)
- [matrix.cpp](#)

## 6.90 GiNaC::minkmetric Class Reference

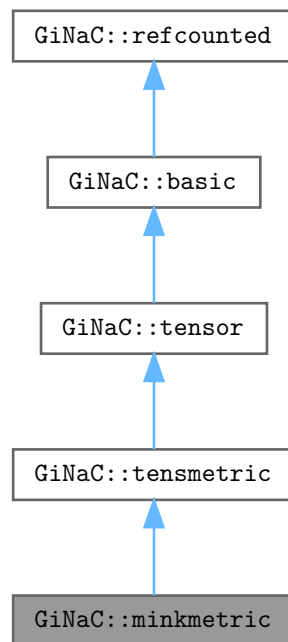
This class represents a Minkowski metric tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::minkmetric:



Collaboration diagram for `GiNaC::minkmetric`:



## Public Member Functions

- `minkmetric` (bool `pos_sig`)  
*Construct Lorentz metric tensor with given signature.*
- bool `info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed Lorentz metric tensor.*
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, list &syms) override  
*Read (a.k.a.*

## Public Member Functions inherited from `GiNaC::tensmetric`

- bool `info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed metric tensor with something else.*

Public Member Functions inherited from [GiNaC::tensor](#)

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*

- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*

- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

#### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

#### Protected Member Functions

- unsigned `return_type` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

#### Protected Member Functions inherited from `GiNaC::tensmetric`

- unsigned `return_type` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- unsigned `return_type` () const override

#### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- bool `pos_sig`  
*If true, the metric is  $\text{diag}(-1, 1, 1, \dots)$ .*

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 6.90.1 Detailed Description

This class represents a Minkowski metric tensor.

It has all the properties of a metric tensor and is (as a matrix) equal to  $\text{diag}(1, -1, -1, \dots)$  or  $\text{diag}(-1, 1, 1, \dots)$ .

## 6.90.2 Constructor & Destructor Documentation

### 6.90.2.1 `minkmetric()`

```
GiNaC::minkmetric::minkmetric (
    bool pos_sig )
```

Construct Lorentz metric tensor with given signature.

## 6.90.3 Member Function Documentation

### 6.90.3.1 info()

```
bool GiNaC::minkmetric::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::info\\_flags::real](#).

### 6.90.3.2 eval\_indexed()

```
ex GiNaC::minkmetric::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed Lorentz metric tensor.

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), and [pos\\_sig](#).

### 6.90.3.3 archive()

```
void GiNaC::minkmetric::archive (
    archive\_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), and [pos\\_sig](#).

#### 6.90.3.4 read\_archive()

```
void GiNaC::minkmetric::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), and [pos\\_sig](#).

#### 6.90.3.5 return\_type()

```
unsigned GiNaC::minkmetric::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::return\\_types::commutative](#).

#### 6.90.3.6 do\_print()

```
void GiNaC::minkmetric::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

#### 6.90.3.7 do\_print\_latex()

```
void GiNaC::minkmetric::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

### 6.90.4 Member Data Documentation



#### 6.90.4.1 pos\_sig

```
bool GiNaC::minkmetric::pos_sig [private]
```

If true, the metric is  $\text{diag}(-1,1,1,\dots)$ .

Otherwise it is  $\text{diag}(1,-1,-1,\dots)$ .

Referenced by [archive\(\)](#), [eval\\_indexed\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

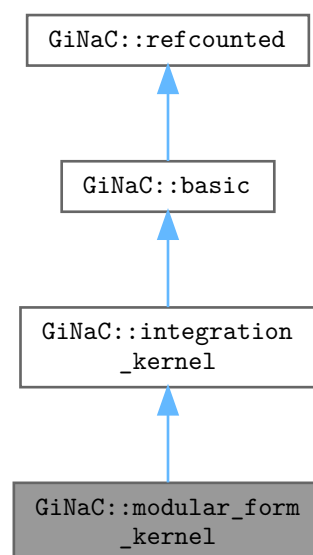
- [tensor.h](#)
- [tensor.cpp](#)

## 6.91 GiNaC::modular\_form\_kernel Class Reference

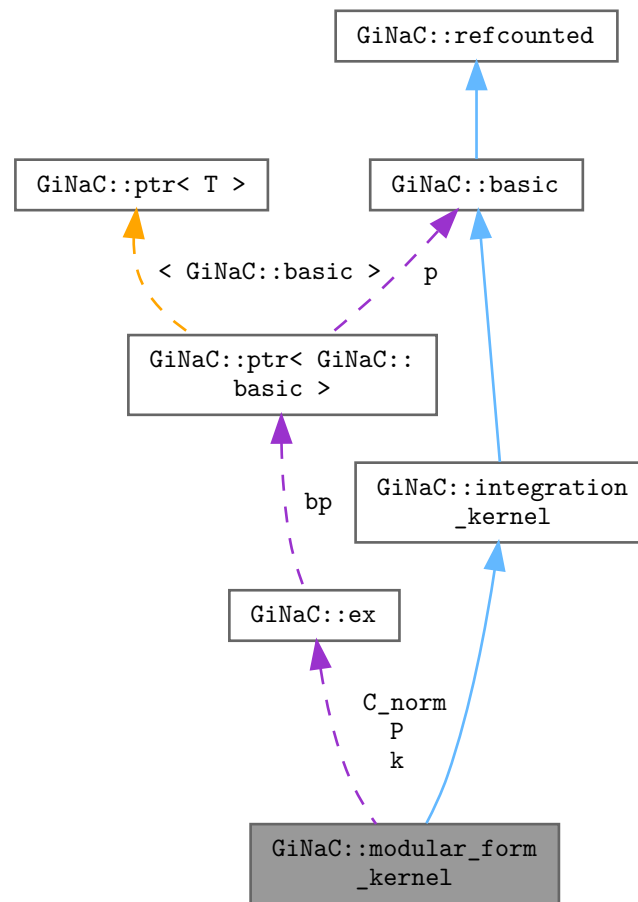
A kernel corresponding to a polynomial in Eisenstein series.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::modular\_form\_kernel:



Collaboration diagram for `GiNaC::modular_form_kernel`:



## Public Member Functions

- `modular_form_kernel` (const `ex` &`k`, const `ex` &`P`, const `ex` &`C_norm`=`numeric`(1))
- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position i.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series` (const `ex` &`qbar`, int `order`) const override  
*Returns the Laurent series, starting possibly with the pole term.*

- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc`=0) const override  
*Returns the value of the modular form.*
- `ex q_expansion_modular_form` (const `ex` &`q`, int `order`) const

#### Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool `is_numeric` (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual `ex Laurent_series` (const `ex` &`x`, int `order`) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex` &`lambda`, int `N_trunc`=0) const  
*Evaluates the integrand at `lambda`.*
- `size_t get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int `cache_steps`) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int `i`) const  
*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- `cln::cl_N series_coeff` (int `i`) const  
*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

#### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &`i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &`c`, unsigned `level`=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*

- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned [options](#)=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*

- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t` `return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- bool `uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::integration\\_kernel](#)

- virtual bool [uses\\_Laurent\\_series](#) () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method [Laurent\\_series](#) needs to be implemented).*
- virtual [cln::cl\\_N](#) [series\\_coeff\\_impl](#) (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- [ex](#) [get\\_numerical\\_value\\_impl](#) (const [ex](#) &lambda, const [ex](#) &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex](#) [eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex](#) [derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- [ex](#) k
- [ex](#) P
- [ex](#) C\_norm

### Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- [std::vector](#)< [cln::cl\\_N](#) > [series\\_vec](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.91.1 Detailed Description

A kernel corresponding to a polynomial in Eisenstein series.

This class represents the differential one-form

$$\omega^{\text{modular}}(P_k(\eta_{k_1}^{(1)}, \dots, \eta_{k_r}^{(r)})) = C_k P_k(\eta_{k_1}^{(1)}, \dots, \eta_{k_r}^{(r)}) \frac{d\bar{q}_N}{\bar{q}_N}.$$

### 6.91.2 Constructor & Destructor Documentation

#### 6.91.2.1 modular\_form\_kernel()

```
GiNaC::modular_form_kernel::modular_form_kernel (
    const ex & k,
    const ex & P,
    const ex & C_norm = numeric(1) )
```

### 6.91.3 Member Function Documentation

#### 6.91.3.1 series()

```
ex GiNaC::modular_form_kernel::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const \[override\], \[virtual\]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [order](#), [P](#), [GiNaC::pow\(\)](#), [qbar](#), [r](#), and [GiNaC::ex::series\(\)](#).

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

#### 6.91.3.2 nops()

```
size_t GiNaC::modular_form_kernel::nops ( ) const \[override\], \[virtual\]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.91.3.3 op()

```
ex GiNaC::modular_form_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [k](#), and [P](#).

### 6.91.3.4 let\_op()

```
ex & GiNaC::modular_form_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [k](#), and [P](#).

### 6.91.3.5 is\_numeric()

```
bool GiNaC::modular_form_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [k](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [GiNaC::series\\_to\\_poly\(\)](#), and [GiNaC::ex::subs\(\)](#).

### 6.91.3.6 Laurent\_series()

```
ex GiNaC::modular_form_kernel::Laurent_series (
    const ex & x,
    int order ) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [GiNaC::ex::series\(\)](#), and [GiNaC::series\\_to\\_poly\(\)](#).



### 6.91.3.7 get\_numerical\_value()

```
ex GiNaC::modular_form_kernel::get_numerical_value (
    const ex & qbar,
    int N_trunc = 0 ) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

### 6.91.3.8 uses\_Laurent\_series()

```
bool GiNaC::modular_form_kernel::uses_Laurent_series ( ) const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.91.3.9 q\_expansion\_modular\_form()

```
ex GiNaC::modular_form_kernel::q_expansion_modular_form (
    const ex & q,
    int order ) const
```

References [series\(\)](#).

Referenced by [is\\_numeric\(\)](#), and [Laurent\\_series\(\)](#).

### 6.91.3.10 do\_print()

```
void GiNaC::modular_form_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [C\\_norm](#), [k](#), [P](#), and [GiNaC::ex::print\(\)](#).

## 6.91.4 Member Data Documentation

#### 6.91.4.1 **k**

`ex GiNaC::modular_form_kernel::k` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

#### 6.91.4.2 **P**

`ex GiNaC::modular_form_kernel::P` [protected]

Referenced by [do\\_print\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\(\)](#).

#### 6.91.4.3 **C\_norm**

`ex GiNaC::modular_form_kernel::C_norm` [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.92 GiNaC::basic\_partition\_generator::mpartition2 Struct Reference

```
#include <utils.h>
```

### Public Member Functions

- [mpartition2](#) (unsigned n\_, unsigned m\_)
- bool [next\\_partition](#) ()

### Public Attributes

- `std::vector< unsigned >` [x](#)
- unsigned [n](#)
- unsigned [m](#)

#### 6.92.1 Constructor & Destructor Documentation

### 6.92.1.1 mpartition2()

```
GiNaC::basic_partition_generator::mpartition2::mpartition2 (
    unsigned n_,
    unsigned m_ ) [inline]
```

References [k](#), [m](#), [n](#), and [x](#).

## 6.92.2 Member Function Documentation

### 6.92.2.1 next\_partition()

```
bool GiNaC::basic_partition_generator::mpartition2::next_partition ( ) [inline]
```

References [k](#), [m](#), and [x](#).

Referenced by [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#), and [GiNaC::partition\\_generator::next\(\)](#).

## 6.92.3 Member Data Documentation

### 6.92.3.1 x

```
std::vector<unsigned> GiNaC::basic_partition_generator::mpartition2::x
```

Referenced by [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::partition\\_generator::get\(\)](#), [mpartition2\(\)](#), and [next\\_partition\(\)](#).

### 6.92.3.2 n

```
unsigned GiNaC::basic_partition_generator::mpartition2::n
```

Referenced by [mpartition2\(\)](#), and [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#).

### 6.92.3.3 m

```
unsigned GiNaC::basic_partition_generator::mpartition2::m
```

Referenced by [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::partition\\_generator::get\(\)](#), [mpartition2\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#), and [next\\_partition\(\)](#).

The documentation for this struct was generated from the following file:

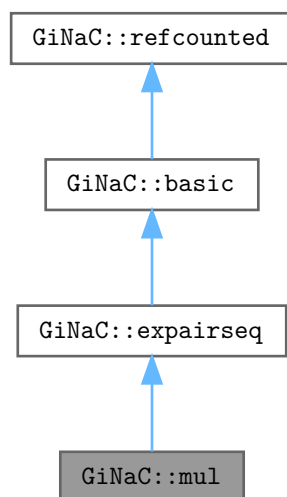
- [utils.h](#)

## 6.93 GiNaC::mul Class Reference

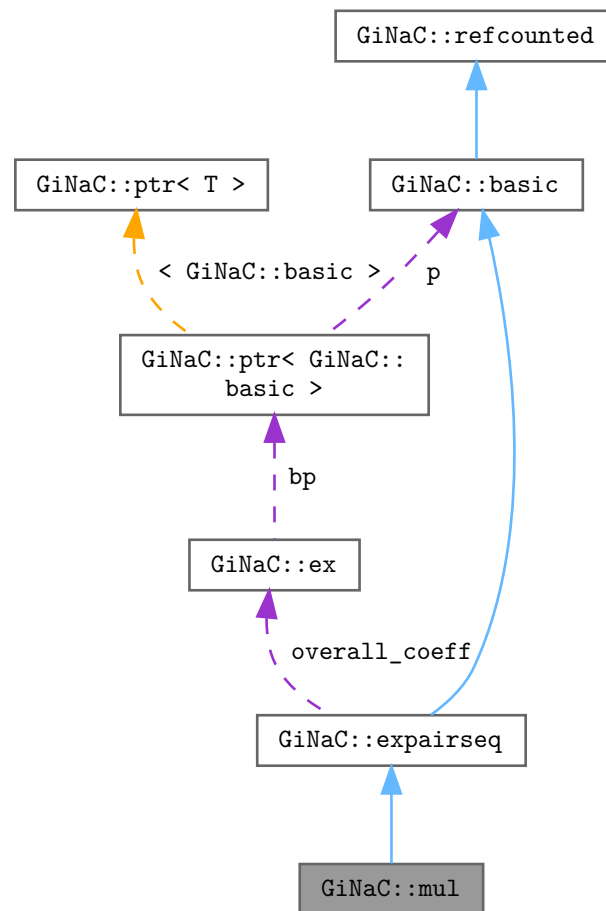
Product of expressions.

```
#include <mul.h>
```

Inheritance diagram for GiNaC::mul:



Collaboration diagram for GiNaC::mul:



## Public Member Functions

- `mul` (const `ex` &lh, const `ex` &rh)
- `mul` (const `exvector` &v)
- `mul` (const `epvector` &v)
- `mul` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false)
- `mul` (`epvector` &&vp)
- `mul` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false)
- `mul` (const `ex` &lh, const `ex` &mh, const `ex` &rh)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthesizing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- int `degree` (const `ex` &s) const override

- *Return degree of highest power in object s.*
- `int ldegree (const ex &s) const` override
- *Return degree of lowest power in object s.*
- `ex coeff (const ex &s, int n=1) const` override
- *Return coefficient of degree n in object s.*
- `bool has (const ex &other, unsigned options=0) const` override
- *Test for occurrence of a pattern.*
- `ex eval ()` const override
- *Perform automatic term rewriting rules in this class.*
- `ex evalf ()` const override
- *Evaluate object numerically.*
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `ex evalm ()` const override
- *Evaluate sums, products and integer powers of matrices.*
- `ex series (const relational &s, int order, unsigned options=0) const` override
- *Implementation of `ex::series()` for product.*
- `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const` override
- *Implementation of `ex::normal()` for a product.*
- `numeric integer_content ()` const override
- `ex smod (const numeric &xi) const` override
- *Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient ()` const override
- *Implementation `ex::max_coefficient()`.*
- `exvector get_free_indices ()` const override
- *Return a vector containing the free indices of an expression.*
- `ex conjugate ()` const override
- `ex algebraic_subs_mul (const exmap &m, unsigned options) const`

### Public Member Functions inherited from `GiNaC::expairseq`

- `expairseq (const ex &lh, const ex &rh)`
- `expairseq (const exvector &v)`
- `expairseq (const epvector &v, const ex &oc, bool do_index_renaming=false)`
- `expairseq (epvector &&vp, const ex &oc, bool do_index_renaming=false)`
- `unsigned precedence ()` const override
- *Return relative operator precedence (for parenthezing output).*
- `bool info (unsigned inf) const` override
- *Information about the object.*
- `size_t nops ()` const override
- *Number of operands/members.*
- `ex op (size_t i) const` override
- *Return operand/member at position i.*
- `ex map (map_function &f) const` override
- *Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex eval ()` const override
- *Perform coefficient-wise automatic term rewriting rules in this class.*
- `ex to_rational (exmap &repl) const` override
- *Implementation of `ex::to_rational()` for expairseqs.*
- `ex to_polynomial (exmap &repl) const` override
- *Implementation of `ex::to_polynomial()` for expairseqs.*

- bool **match** (const **ex** &pattern, **exmap** &repl\_lst) const override  
*Check whether the expression matches a given pattern.*
- **ex subs** (const **exmap** &m, unsigned **options**=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- **ex conjugate** () const override
- void **archive** (**archive\_node** &n) const override  
*Save (serialize) the object into archive node.*
- void **read\_archive** (const **archive\_node** &n, **lst** &syms) override  
*Load (deserialize) the object from an archive node.*

### Public Member Functions inherited from **GiNaC::basic**

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const **basic** &other)
- const **basic** & **operator=** (const **basic** &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic** \* **duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval** () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual **ex evalf** () const  
*Evaluate object numerically.*
- virtual **ex evalm** () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const **basic** &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const **print\_context** &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned **precedence** () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool **info** (unsigned inf) const  
*Information about the object.*
- virtual size\_t **nops** () const  
*Number of operands/members.*
- virtual **ex op** (size\_t i) const  
*Return operand/member at position i.*
- virtual **ex operator[]** (const **ex** &index) const
- virtual **ex operator[]** (size\_t i) const
- virtual **ex** & **let\_op** (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual **ex** & **operator[]** (const **ex** &index)
- virtual **ex** & **operator[]** (size\_t i)
- virtual bool **has** (const **ex** &other, unsigned **options**=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const



- *Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &symbols)
- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- *Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- *Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- *Test for syntactic equality.*
- const `basic` & `hold` () const
- *Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned `f`) const
- *Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned `f`) const
- *Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

### Protected Member Functions

- `ex derivative` (const `symbol` &s) const override
- *Implementation of `ex::diff()` for a product.*
- `ex eval_ncmul` (const `exvector` &v) const override
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- `ex thisexpairseq` (const `epvector` &v, const `ex` &oc, bool `do_index_renaming`=false) const override
- *Create an object of this type.*
- `ex thisexpairseq` (`epvector` &&vp, const `ex` &oc, bool `do_index_renaming`=false) const override
- `expair split_ex_to_pair` (const `ex` &e) const override
- *Form an `expair` from an `ex`, using the corresponding semantics.*
- `expair combine_ex_with_coeff_to_pair` (const `ex` &e, const `ex` &c) const override
- `expair combine_pair_with_coeff_to_pair` (const `expair` &p, const `ex` &c) const override
- `ex recombine_pair_to_ex` (const `expair` &p) const override
- *Form an `ex` out of an `expair`, using the corresponding semantics.*
- bool `expair_needs_further_processing` (`epp` it) override
- `ex default_overall_coeff` () const override
- void `combine_overall_coeff` (const `ex` &c) override
- void `combine_overall_coeff` (const `ex` &c1, const `ex` &c2) override
- bool `can_make_flat` (const `expair` &p) const override
- `ex expand` (unsigned `options`=0) const override
- *Expand expression, i.e.*
- void `find_real_imag` (`ex` &, `ex` &) const

- void `print_overall_coeff` (const `print_context` &c, const char \*mul\_sym) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_csrc` (const `print_csrc` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- `epvector` `expandchildren` (unsigned `options`) const

*Member-wise expand the expires representing this sequence.*

### Protected Member Functions inherited from `GiNaC::expairseq`

- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `return_type` () const override
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `ex` `expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- virtual `ex` `thisexpairseq` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false) const  
*Create an object of this type.*
- virtual `ex` `thisexpairseq` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false) const
- virtual void `printseq` (const `print_context` &c, char delim, unsigned this\_precedence, unsigned upper\_precedence) const
- virtual void `printpair` (const `print_context` &c, const `expair` &p, unsigned upper\_precedence) const
- virtual `expair` `split_ex_to_pair` (const `ex` &e) const  
*Form an expair from an ex, using the corresponding semantics.*
- virtual `expair` `combine_ex_with_coeff_to_pair` (const `ex` &e, const `ex` &c) const
- virtual `expair` `combine_pair_with_coeff_to_pair` (const `expair` &p, const `ex` &c) const
- virtual `ex` `recombine_pair_to_ex` (const `expair` &p) const  
*Form an ex out of an expair, using the corresponding semantics.*
- virtual bool `expair_needs_further_processing` (`epp` it)
- virtual `ex` `default_overall_coeff` () const
- virtual void `combine_overall_coeff` (const `ex` &c)
- virtual void `combine_overall_coeff` (const `ex` &c1, const `ex` &c2)
- virtual bool `can_make_flat` (const `expair` &p) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `construct_from_2_ex` (const `ex` &lh, const `ex` &rh)
- void `construct_from_2_expairseq` (const `expairseq` &s1, const `expairseq` &s2)
- void `construct_from_expairseq_ex` (const `expairseq` &s, const `ex` &e)
- void `construct_from_exvector` (const `exvector` &v)
- void `construct_from_epvector` (const `epvector` &v, bool do\_index\_renaming=false)
- void `construct_from_epvector` (`epvector` &&v, bool do\_index\_renaming=false)
- void `make_flat` (const `exvector` &v)  
*Combine this expairseq with argument exvector.*
- void `make_flat` (const `epvector` &v, bool do\_index\_renaming=false)  
*Combine this expairseq with argument epvector.*
- void `canonicalize` ()  
*Brings this expairseq into a sorted (canonical) form.*
- void `combine_same_terms_sorted_seq` ()  
*Compact a presorted expairseq by combining all matching expires to one each.*
- bool `is_canonical` () const  
*Check if this expairseq is in sorted (canonical) form.*

- [epvector expandchildren](#) (unsigned [options](#)) const  
*Member-wise expand the expires in this sequence.*
- [epvector evalchildren](#) () const  
*Member-wise evaluate the expires in this sequence.*
- [epvector subschildren](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Member-wise substitute in this sequence.*

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Static Protected Member Functions

- static bool [can\\_be\\_further\\_expanded](#) (const [ex](#) &e)

### Friends

- class [add](#)
- class [ncmul](#)
- class [power](#)

### Additional Inherited Members

#### Protected Attributes inherited from [GiNaC::expairseq](#)

- [epvector seq](#)
- [ex overall\\_coeff](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.93.1 Detailed Description

Product of expressions.

## 6.93.2 Constructor & Destructor Documentation

### 6.93.2.1 `mul()` [1/7]

```
GiNaC::mul::mul (
    const ex & lh,
    const ex & rh )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

### 6.93.2.2 `mul()` [2/7]

```
GiNaC::mul::mul (
    const exvector & v )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

### 6.93.2.3 `mul()` [3/7]

```
GiNaC::mul::mul (
    const epvector & v )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

#### 6.93.2.4 mul() [4/7]

```
GiNaC::mul::mul (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false )
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

#### 6.93.2.5 mul() [5/7]

```
GiNaC::mul::mul (
    epvector && vp )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

#### 6.93.2.6 mul() [6/7]

```
GiNaC::mul::mul (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false )
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

#### 6.93.2.7 mul() [7/7]

```
GiNaC::mul::mul (
    const ex & lh,
    const ex & mh,
    const ex & rh )
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [factors](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

### 6.93.3 Member Function Documentation

### 6.93.3.1 precedence()

```
unsigned GiNaC::mul::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::expairseq](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_latex\(\)](#), and [print\\_overall\\_coeff\(\)](#).

### 6.93.3.2 info()

```
bool GiNaC::mul::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_num1\\_p](#), [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::factor\(\)](#), [GiNaC::basic::flags](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::status\\_flags::is\\_negative](#), [GiNaC::status\\_flags::is\\_positive](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::status\\_flags::purely\\_indefinite](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [expand\(\)](#), and [info\(\)](#).

### 6.93.3.3 is\_polynomial()

```
bool GiNaC::mul::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::nonnegint](#), and [GiNaC::expairseq::seq](#).

#### 6.93.3.4 degree()

```
int GiNaC::mul::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::degree\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

#### 6.93.3.5 ldegree()

```
int GiNaC::mul::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::ldegree\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

#### 6.93.3.6 coeff()

```
ex GiNaC::mul::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [c](#), [GiNaC::ex::coeff\(\)](#), [coeff\(\)](#), [n](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [coeff\(\)](#), and [print\\_overall\\_coeff\(\)](#).

#### 6.93.3.7 has()

```
bool GiNaC::mul::has (
    const ex & pattern,
    unsigned options = 0 ) const [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::has\\_options::algebraic](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::basic::has\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [options](#).

### 6.93.3.8 eval()

```
ex GiNaC::mul::eval ( ) const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following  $x$ ,  $x_1$ ,  $x_2$ ,... stand for a symbolic variables of type `ex` and  $c$ ,  $c_1$ ,  $c_2$ ... stand for such expressions that contain a plain number.

- $*(...,x;0) \rightarrow 0$
- $*(+(x_1,x_2,...);c) \rightarrow *(*(x_1,c),*(x_2,c),...)$
- $*(x;1) \rightarrow x$
- $*(;c) \rightarrow c$

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_c](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [last](#), [likely](#), [GiNaC::numeric::mul\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [unlikely](#).

Referenced by [do\\_print\\_latex\(\)](#).

### 6.93.3.9 evalf()

```
ex GiNaC::mul::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.10 real\_part()

```
ex GiNaC::mul::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [find\\_real\\_imag\(\)](#).



### 6.93.3.11 imag\_part()

```
ex GiNaC::mul::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [find\\_real\\_imag\(\)](#).

### 6.93.3.12 evalm()

```
ex GiNaC::mul::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::end\(\)](#), [GiNaC::ex::evalm\(\)](#), [m](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.93.3.13 series()

```
ex GiNaC::mul::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for product.

This performs series multiplication when multiplying series.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::lddegree\(\)](#), [GiNaC::pseries::mul\\_const\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::expairseq::op\(\)](#), [options](#), [order](#), [GiNaC::expairseq::overall\\_coeff](#), [r](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::ex::series\(\)](#).

### 6.93.3.14 normal()

```
ex GiNaC::mul::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a product.

It cancels common factors from fractions.

See also

[ex::normal\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::frac\\_cancel\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.15 integer\_content()

```
numeric GiNaC::mul::integer_content ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.16 smod()

```
ex GiNaC::mul::smod (
    const numeric & xi ) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

Parameters

$xi$	modulus
------	---------

Returns

mapped polynomial

See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::smod\(\)](#).

### 6.93.3.17 max\_coefficient()

```
numeric GiNaC::mul::max_coefficient ( ) const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.18 get\_free\_indices()

```
exvector GiNaC::mul::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

### 6.93.3.19 conjugate()

```
ex GiNaC::mul::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [c](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), and [x](#).

**6.93.3.20 derivative()**

```
ex GiNaC::mul::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a product.

It applies the product rule.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::basic::diff\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::pow\(\)](#), [GiNaC::expairseq::seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [GiNaC::expair::swap\(\)](#).

**6.93.3.21 eval\_ncmul()**

```
ex GiNaC::mul::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#), and [GiNaC::expairseq::seq](#).

**6.93.3.22 return\_type()**

```
unsigned GiNaC::mul::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_comp](#) and [GiNaC::expairseq::seq](#).

**6.93.3.23 return\_type\_tinfo()**

```
return\_type\_t GiNaC::mul::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#), and [GiNaC::expairseq::seq](#).

**6.93.3.24 thisexpairseq() [1/2]**

```
ex GiNaC::mul::thisexpairseq (
    const epvector & v,
    const ex & oc,
    bool do_index_renaming = false ) const [override], [protected], [virtual]
```

Create an object of this type.

This method works similar to a constructor. It is useful because expairseq has (at least) two possible different semantics but we want to inherit methods thus avoiding code duplication. Sometimes a method in expairseq has to create a new one of the same semantics, which cannot be done by a ctor because the name (add, mul,...) is unknown on the expairseq level. In order for this trick to work a derived class must of course override this definition.

Reimplemented from [GiNaC::expairseq](#).

Referenced by [conjugate\(\)](#).

**6.93.3.25 thisexpairseq() [2/2]**

```
ex GiNaC::mul::thisexpairseq (
    epvector && vp,
    const ex & oc,
    bool do_index_renaming = false ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

**6.93.3.26 split\_ex\_to\_pair()**

```
expair GiNaC::mul::split_ex_to_pair (
    const ex & e ) const [override], [protected], [virtual]
```

Form an expair from an ex, using the corresponding semantics.

See also

[expairseq::recombine\\_pair\\_to\\_ex\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::power::basis](#), and [GiNaC::power::exponent](#).

Referenced by [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [conjugate\(\)](#), [derivative\(\)](#), [evalm\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [expand\(\)](#), and [expandchildren\(\)](#).

**6.93.3.27 combine\_ex\_with\_coeff\_to\_pair()**

```

expair GiNaC::mul::combine_ex_with_coeff_to_pair (
    const ex & e,
    const ex & c ) const [override], [protected], [virtual]

```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [c](#), [GINAC\\_ASSERT](#), [GiNaC::pow\(\)](#), and [split\\_ex\\_to\\_pair\(\)](#).

**6.93.3.28 combine\_pair\_with\_coeff\_to\_pair()**

```

expair GiNaC::mul::combine_pair_with_coeff_to_pair (
    const expair & p,
    const ex & c ) const [override], [protected], [virtual]

```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [c](#), [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::pow\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expair::rest](#), and [split\\_ex\\_to\\_pair\(\)](#).

**6.93.3.29 recombine\_pair\_to\_ex()**

```

ex GiNaC::mul::recombine_pair_to_ex (
    const expair & p ) const [override], [protected], [virtual]

```

Form an ex out of an expair, using the corresponding semantics.

See also

[expairseq::split\\_ex\\_to\\_pair\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::expair::coeff](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::expair::rest](#).

Referenced by [coeff\(\)](#), [combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [evalm\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [expandchildren\(\)](#), [find\\_real\\_imag\(\)](#), [info\(\)](#), [integer\\_content\(\)](#), [ldegree\(\)](#), [max\\_coefficient\(\)](#), [normal\(\)](#), [series\(\)](#), and [smod\(\)](#).

**6.93.3.30 expair\_needs\_further\_processing()**

```

bool GiNaC::mul::expair_needs_further_processing (
    epp it ) [override], [protected], [virtual]

```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::expair::is\\_equal\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.93.3.31 default\_overall\_coeff()

```
ex GiNaC::mul::default_overall_coeff ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#).

### 6.93.3.32 combine\_overall\_coeff() [1/2]

```
void GiNaC::mul::combine_overall_coeff (
    const ex & c ) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [c](#), [GINAC\\_ASSERT](#), and [GiNaC::expairseq::overall\\_coeff](#).

### 6.93.3.33 combine\_overall\_coeff() [2/2]

```
void GiNaC::mul::combine_overall_coeff (
    const ex & c1,
    const ex & c2 ) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), and [power](#).

### 6.93.3.34 can\_make\_flat()

```
bool GiNaC::mul::can_make_flat (
    const expair & p ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), and [GiNaC::info\\_flags::integer](#).

### 6.93.3.35 expand()

```
ex GiNaC::mul::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_num0\\_p](#), [can\\_be\\_further\\_expanded\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_ren](#), [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [factors](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numeric::mul\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.93.3.36 algebraic\_subs\_mul()

```
ex GiNaC::mul::algebraic_subs_mul (
    const exmap & m,
    unsigned options ) const
```

References [GiNaC::subs\\_options::algebraic](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), [options](#), [GiNaC::pow\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

### 6.93.3.37 find\_real\_imag()

```
void GiNaC::mul::find_real_imag (
    ex & rp,
    ex & ip ) const [protected]
```

References [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::ex::real\\_part\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [imag\\_part\(\)](#), and [real\\_part\(\)](#).

### 6.93.3.38 print\_overall\_coeff()

```
void GiNaC::mul::print_overall_coeff (
    const print_context & c,
    const char * mul_sym ) const [protected]
```

References [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [precedence\(\)](#), [GiNaC::basic::print\(\)](#), and [GiNaC::ex::print\(\)](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).



### 6.93.3.39 do\_print()

```
void GiNaC::mul::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [print\\_overall\\_coeff\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.40 do\_print\_latex()

```
void GiNaC::mul::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), [eval\(\)](#), [GINAC\\_ASSERT](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [print\\_overall\\_coeff\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.41 do\_print\_csrc()

```
void GiNaC::mul::do_print_csrc (
    const print\_csrc & c,
    unsigned level ) const [protected]
```

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [c](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::expairseq::overall\\_coeff](#), [power](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.42 do\_print\_python\_repr()

```
void GiNaC::mul::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), and [GiNaC::ex::print\(\)](#).

### 6.93.3.43 can\_be\_further\_expanded()

```
bool GiNaC::mul::can_be_further_expanded (
    const ex & e ) [static], [protected]
```

References [GiNaC::ex::info\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::posint](#), and [GiNaC::expairseq::seq](#).

Referenced by [expand\(\)](#).

### 6.93.3.44 `expandchildren()`

```
epvector GiNaC::mul::expandchildren (
    unsigned options ) const [protected]
```

Member-wise expand the expairs representing this sequence.

This must be overridden from `expairseq::expandchildren()` and done iteratively in order to allow for early cancellations and thus save memory.

See also

[mul::expand\(\)](#)

#### Returns

epvector containing expanded pairs, empty if no members had to be changed.

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [last](#), [options](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [expand\(\)](#).

## 6.93.4 Friends And Related Function Documentation

### 6.93.4.1 `add`

```
friend class add [friend]
```

### 6.93.4.2 `ncmul`

```
friend class ncmul [friend]
```

### 6.93.4.3 `power`

```
friend class power [friend]
```

Referenced by [combine\\_overall\\_coeff\(\)](#), and [do\\_print\\_csrc\(\)](#).

The documentation for this class was generated from the following files:

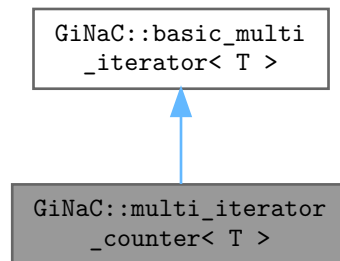
- [mul.h](#)
- [indexed.cpp](#)
- [mul.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.94 GiNaC::multi\_iterator\_counter< T > Class Template Reference

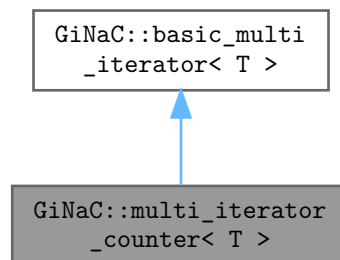
The class `multi_iterator_counter` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for `GiNaC::multi_iterator_counter< T >`:



Collaboration diagram for `GiNaC::multi_iterator_counter< T >`:



### Public Member Functions

- `multi_iterator_counter` (void)  
*Default constructor.*
- `multi_iterator_counter` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- `multi_iterator_counter` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- `basic_multi_iterator< T > &init` (void)  
*Initialize the multi-index to.*
- `basic_multi_iterator< T > &operator++` (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

### Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), size\_t [k](#))  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*
- virtual [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- virtual [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*No effect for basic\_multi\_iterator.*

### Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_counter](#)< TT > &v)

### Additional Inherited Members

#### Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- T [N](#)
- T [B](#)
- std::vector< T > [v](#)
- bool [flag\\_overflow](#)

### 6.94.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_counter< T >
```

The class [multi\\_iterator\\_counter](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N$$

## 6.94.2 Constructor & Destructor Documentation

### 6.94.2.1 multi\_iterator\_counter() [1/3]

```
template<class T >
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
    void ) [inline]
```

Default constructor.

### 6.94.2.2 multi\_iterator\_counter() [2/3]

```
template<class T >
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

### 6.94.2.3 multi\_iterator\_counter() [3/3]

```
template<class T >
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

## 6.94.3 Member Function Documentation

### 6.94.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

### 6.94.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

## 6.94.4 Friends And Related Function Documentation

### 6.94.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_counter< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

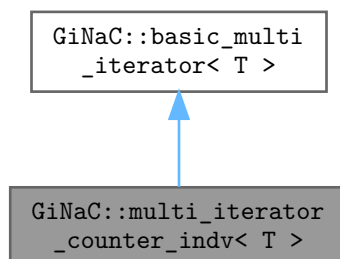
- [utils\\_multi\\_iterator.h](#)

## 6.95 GiNaC::multi\_iterator\_counter\_indv< T > Class Template Reference

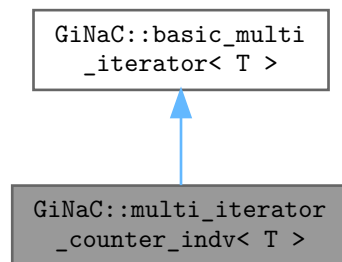
The class [multi\\_iterator\\_counter\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#):



Collaboration diagram for GiNaC::multi\_iterator\_counter\_indv< T >:



## Public Member Functions

- `multi_iterator_counter_indv` (void)  
*Default constructor.*
- `multi_iterator_counter_indv` (T B, const std::vector< T > &Nv, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- `multi_iterator_counter_indv` (T B, const std::vector< T > &Nv, const std::vector< T > &vv)  
*Construct from a vector.*
- `basic_multi_iterator< T > &init` (void)  
*Initialize the multi-index to.*
- `basic_multi_iterator< T > &operator++` (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Public Member Functions inherited from `GiNaC::basic_multi_iterator< T >`

- `basic_multi_iterator` (void)  
*Default constructor.*
- `basic_multi_iterator` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- `basic_multi_iterator` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual `~basic_multi_iterator` ()  
*Destructor.*
- `size_t size` (void) const  
*Returns the size of a multi\_iterator.*
- `bool overflow` (void) const  
*Return the overflow flag.*
- const std::vector< T > &`get_vector` (void) const  
*Returns a reference to the vector v.*
- T `operator[]` (size\_t i) const  
*Subscription via [].*
- T &`operator[]` (size\_t i)  
*Subscription via [].*

- `T operator() (size_t i) const`  
*Subscription via ()*
- `T & operator() (size_t i)`  
*Subscription via ()*
- virtual `basic_multi_iterator< T > & init (void)`  
*Initialize the multi-index to.*
- virtual `basic_multi_iterator< T > & operator++ (int)`  
*No effect for `basic_multi_iterator`.*

## Protected Attributes

- `std::vector< T > Nv`

## Protected Attributes inherited from `GiNaC::basic_multi_iterator< T >`

- `T N`
- `T B`
- `std::vector< T > v`
- `bool flag_overflow`

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_counter_indv< TT > &v)`

## 6.95.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_counter_indv< T >
```

The class `multi_iterator_counter_indv` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N_j$$

## 6.95.2 Constructor & Destructor Documentation

### 6.95.2.1 `multi_iterator_counter_indv()` [1/3]

```
template<class T >
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
    void ) [inline]
```

Default constructor.



**6.95.2.2 multi\_iterator\_counter\_indv()** [2/3]

```
template<class T >
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
    T B,
    const std::vector< T > & Nv,
    size_t k ) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

**6.95.2.3 multi\_iterator\_counter\_indv()** [3/3]

```
template<class T >
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
    T B,
    const std::vector< T > & Nv,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

**6.95.3 Member Function Documentation****6.95.3.1 init()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter_indv< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

**6.95.3.2 operator++()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter_indv< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.

If n is in the last configuration and the increment operator ++ is applied to n, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

## 6.95.4 Friends And Related Function Documentation

### 6.95.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_counter_indv< TT > & v ) [friend]
```

## 6.95.5 Member Data Documentation

### 6.95.5.1 Nv

```
template<class T >
std::vector<T> GiNaC::multi_iterator_counter_indv< T >::Nv [protected]
```

The documentation for this class was generated from the following file:

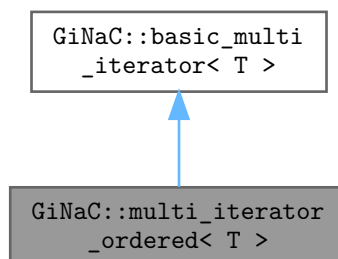
- [utils\\_multi\\_iterator.h](#)

## 6.96 GiNaC::multi\_iterator\_ordered< T > Class Template Reference

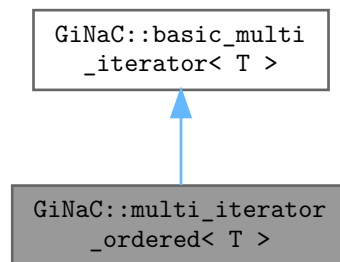
The class [multi\\_iterator\\_ordered](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_ordered< T >:



Collaboration diagram for GiNaC::multi\_iterator\_ordered< T >:



## Public Member Functions

- [multi\\_iterator\\_ordered](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- [multi\\_iterator\\_ordered](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator< T > & init](#) (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator< T > & operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k.*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*

- `T operator() (size_t i) const`  
*Subscription via ()*
- `T & operator() (size_t i)`  
*Subscription via ()*
- virtual `basic_multi_iterator< T > & init (void)`  
*Initialize the multi-index to.*
- virtual `basic_multi_iterator< T > & operator++ (int)`  
*No effect for `basic_multi_iterator`.*

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_ordered< TT > &v)`

## Additional Inherited Members

Protected Attributes inherited from `GiNaC::basic_multi_iterator< T >`

- `T N`
- `T B`
- `std::vector< T > v`
- `bool flag_overflow`

### 6.96.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_ordered< T >
```

The class `multi_iterator_ordered` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N$$

and

$$i_j < i_{j+1}.$$

It is assumed that  $k > 0$  and  $N - B \geq k$ .

### 6.96.2 Constructor & Destructor Documentation

#### 6.96.2.1 `multi_iterator_ordered()` [1/3]

```
template<class T >
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
    void ) [inline]
```

Default constructor.

**6.96.2.2 multi\_iterator\_ordered()** [2/3]

```
template<class T >
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

**6.96.2.3 multi\_iterator\_ordered()** [3/3]

```
template<class T >
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

**6.96.3 Member Function Documentation****6.96.3.1 init()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

**6.96.3.2 operator++()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.

If n is in the last configuration and the increment operator ++ is applied to n, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

## 6.96.4 Friends And Related Function Documentation

### 6.96.4.1 `operator<<`

```
template<class T >
template<class TT >
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_ordered< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

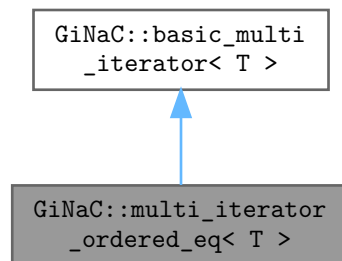
- [utils\\_multi\\_iterator.h](#)

## 6.97 GiNaC::multi\_iterator\_ordered\_eq< T > Class Template Reference

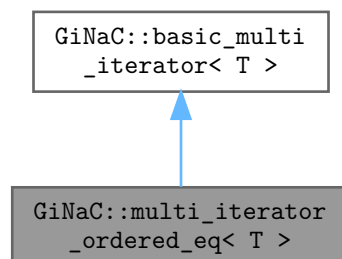
The class `multi_iterator_ordered_eq` defines a `multi_iterator` ( $i_1, i_2, \dots, i_k$ ), such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for `GiNaC::multi_iterator_ordered_eq< T >`:



Collaboration diagram for `GiNaC::multi_iterator_ordered_eq< T >`:



## Public Member Functions

- [multi\\_iterator\\_ordered\\_eq](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered\\_eq](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- [multi\\_iterator\\_ordered\\_eq](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator](#)< T >

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k.*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*
- virtual [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- virtual [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*No effect for [basic\\_multi\\_iterator](#).*

## Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered\\_eq](#)< TT > &v)

## Additional Inherited Members

Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [T N](#)
- [T B](#)
- `std::vector< T > v`
- `bool flag\_overflow`

### 6.97.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_ordered_eq< T >
```

The class [multi\\_iterator\\_ordered\\_eq](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N$$

and

$$i_j \leq i_{j+1}.$$

It is assumed that  $k > 0$ .

### 6.97.2 Constructor & Destructor Documentation

#### 6.97.2.1 `multi_iterator_ordered_eq()` [1/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
    void ) [inline]
```

Default constructor.

#### 6.97.2.2 `multi_iterator_ordered_eq()` [2/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a `multi_iterator` with upper limit N and size k .



**6.97.2.3 multi\_iterator\_ordered\_eq() [3/3]**

```
template<class T >
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

**6.97.3 Member Function Documentation****6.97.3.1 init()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

**6.97.3.2 operator++()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

**6.97.4 Friends And Related Function Documentation**

### 6.97.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

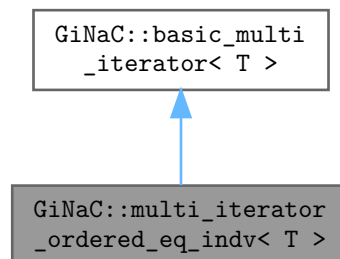
- [utils\\_multi\\_iterator.h](#)

## 6.98 GiNaC::multi\_iterator\_ordered\_eq\_indv< T > Class Template Reference

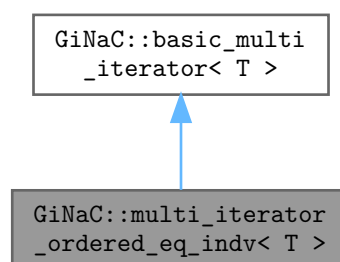
The class [multi\\_iterator\\_ordered\\_eq\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_ordered\_eq\_indv< T >:



Collaboration diagram for GiNaC::multi\_iterator\_ordered\_eq\_indv< T >:



## Public Member Functions

- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (T B, const std::vector< T > &Nv, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (T B, const std::vector< T > &Nv, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > &init (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator](#)< T > &operator++ (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > &[get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T &[operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T &[operator\(\)](#) (size\_t i)  
*Subscription via ().*
- virtual [basic\\_multi\\_iterator](#)< T > &init (void)  
*Initialize the multi-index to.*
- virtual [basic\\_multi\\_iterator](#)< T > &operator++ (int)  
*No effect for basic\_multi\_iterator.*

## Protected Attributes

- std::vector< T > Nv

## Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- T N
- T B
- std::vector< T > v
- bool [flag\\_overflow](#)

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_ordered_eq_indv< TT > &v)`

### 6.98.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_ordered_eq_indv< T >
```

The class `multi_iterator_ordered_eq_indv` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N_j$$

and

$$i_j \leq i_{j+1}.$$

### 6.98.2 Constructor & Destructor Documentation

#### 6.98.2.1 multi\_iterator\_ordered\_eq\_indv() [1/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
    void ) [inline]
```

Default constructor.

#### 6.98.2.2 multi\_iterator\_ordered\_eq\_indv() [2/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
    T B,
    const std::vector< T > & Nv,
    size_t k ) [inline], [explicit]
```

Construct a `multi_iterator` with upper limit N and size k .

#### 6.98.2.3 multi\_iterator\_ordered\_eq\_indv() [3/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
    T B,
    const std::vector< T > & Nv,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

## 6.98.3 Member Function Documentation

### 6.98.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq_indv< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

### 6.98.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq_indv< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

## 6.98.4 Friends And Related Function Documentation

### 6.98.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_ordered_eq_indv< TT > & v ) [friend]
```

## 6.98.5 Member Data Documentation

### 6.98.5.1 Nv

```
template<class T >
std::vector<T> GiNaC::multi_iterator_ordered_eq_indv< T >::Nv [protected]
```

The documentation for this class was generated from the following file:

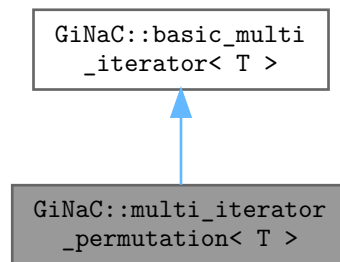
- [utils\\_multi\\_iterator.h](#)

## 6.99 GiNaC::multi\_iterator\_permutation< T > Class Template Reference

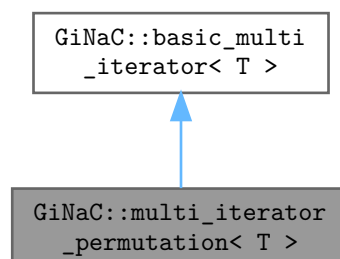
The class [multi\\_iterator\\_permutation](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , for which.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_permutation< T >:



Collaboration diagram for GiNaC::multi\_iterator\_permutation< T >:



## Public Member Functions

- [multi\\_iterator\\_permutation](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_permutation](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_permutation](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*
- int [get\\_sign](#) (void) const  
*Returns the sign of the permutation, defined by.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*
- virtual [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- virtual [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*No effect for [basic\\_multi\\_iterator](#).*

## Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_permutation](#)< TT > &v)

## Additional Inherited Members

Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [T N](#)
- [T B](#)
- `std::vector< T > v`
- `bool flag\_overflow`

### 6.99.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_permutation< T >
```

The class [multi\\_iterator\\_permutation](#) defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , for which.

$$B \leq i_j < N$$

and

$$i_i \neq i_j$$

In particular, if  $N - B = k$ , [multi\\_iterator\\_permutation](#) loops over all permutations of  $k$  elements.

### 6.99.2 Constructor & Destructor Documentation

#### 6.99.2.1 `multi_iterator_permutation()` [1/3]

```
template<class T >
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
    void ) [inline]
```

Default constructor.

#### 6.99.2.2 `multi_iterator_permutation()` [2/3]

```
template<class T >
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
    T B,
    T N,
    size_t k ) [inline], [explicit]
```

Construct a `multi_iterator` with upper limit  $N$  and size  $k$ .



**6.99.2.3 multi\_iterator\_permutation()** [3/3]

```
template<class T >
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
    T B,
    T N,
    const std::vector< T > & vv ) [inline], [explicit]
```

Construct from a vector.

**6.99.3 Member Function Documentation****6.99.3.1 init()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_permutation< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

**6.99.3.2 operator++()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_permutation< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.

If `n` is in the last configuration and the increment operator `++` is applied to `n`, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

### 6.99.3.3 get\_sign()

```
template<class T >
int GiNaC::multi_iterator_permutation< T >::get_sign (
    void ) const [inline]
```

Returns the sign of the permutation, defined by.

$$(-1)^{n_{inv}},$$

where  $n_{inv}$  is the number of inversions, e.g. the number of pairs  $i < j$  for which

$$n_i > n_j.$$

References [k](#).

## 6.99.4 Friends And Related Function Documentation

### 6.99.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_permutation< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

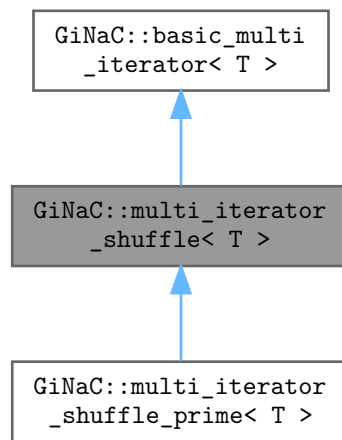
- [utils\\_multi\\_iterator.h](#)

## 6.100 GiNaC::multi\_iterator\_shuffle< T > Class Template Reference

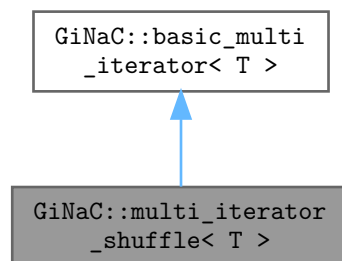
The class [multi\\_iterator\\_shuffle](#) defines a multi\_iterator, which runs over all shuffles of a and b.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_shuffle< T >:



Collaboration diagram for GiNaC::multi\_iterator\_shuffle< T >:



## Public Member Functions

- [multi\\_iterator\\_shuffle](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_shuffle](#) (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to the first shuffle.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), size\_t [k](#))  
*Construct a multi\_iterator with upper limit N, lower limit B and size k.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > &[get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*
- virtual [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- virtual [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*No effect for basic\_multi\_iterator.*

## Protected Attributes

- size\_t [N\\_internal](#)
- std::vector< size\_t > [v\\_internal](#)
- std::vector< T > [v\\_orig](#)

## Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- T [N](#)
- T [B](#)
- std::vector< T > [v](#)
- bool [flag\\_overflow](#)

## Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_shuffle](#)< TT > &v)

## 6.100.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_shuffle< T >
```

The class [multi\\_iterator\\_shuffle](#) defines a multi\_iterator, which runs over all shuffles of a and b.

## 6.100.2 Constructor & Destructor Documentation

### 6.100.2.1 multi\_iterator\_shuffle() [1/2]

```
template<class T >
GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle (
    void ) [inline]
```

Default constructor.

### 6.100.2.2 multi\_iterator\_shuffle() [2/2]

```
template<class T >
GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle (
    const std::vector< T > & a,
    const std::vector< T > & b ) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::multi\\_iterator\\_shuffle< T >::N\\_internal](#), [GiNaC::basic\\_multi\\_iterator< T >::v\\_internal](#), and [GiNaC::multi\\_iterator\\_shuffle< T >::v\\_orig](#).

## 6.100.3 Member Function Documentation

### 6.100.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to the first shuffle.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

Reimplemented in [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#).

### 6.100.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle< T >::operator++ (
    int ) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.

If `n` is in the last configuration and the increment operator `++` is applied to `n`, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

## 6.100.4 Friends And Related Function Documentation

### 6.100.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_shuffle< TT > & v ) [friend]
```

## 6.100.5 Member Data Documentation

### 6.100.5.1 N\_internal

```
template<class T >
size_t GiNaC::multi_iterator_shuffle< T >::N_internal [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

### 6.100.5.2 v\_internal

```
template<class T >
std::vector<size_t> GiNaC::multi_iterator_shuffle< T >::v_internal [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

### 6.100.5.3 v\_orig

```
template<class T >
std::vector<T> GiNaC::multi_iterator_shuffle< T >::v_orig [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

The documentation for this class was generated from the following file:

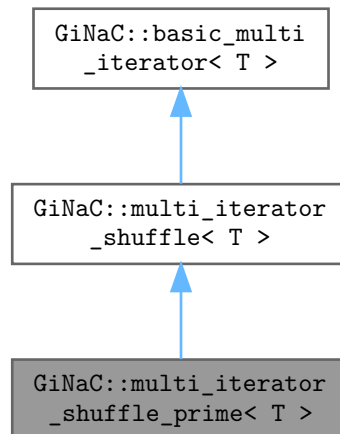
- [utils\\_multi\\_iterator.h](#)

## 6.101 GiNaC::multi\_iterator\_shuffle\_prime< T > Class Template Reference

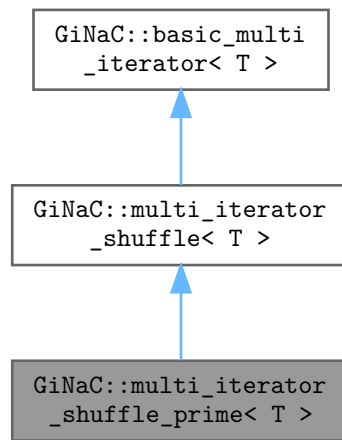
The class [multi\\_iterator\\_shuffle\\_prime](#) defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_shuffle\_prime< T >:



Collaboration diagram for `GiNaC::multi_iterator_shuffle< T >`:



## Public Member Functions

- `multi_iterator_shuffle_prime` (void)  
*Default constructor.*
- `multi_iterator_shuffle_prime` (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- `basic_multi_iterator< T > &init` (void)  
*Initialize the multi-index to the first shuffle.*

## Public Member Functions inherited from `GiNaC::multi_iterator_shuffle< T >`

- `multi_iterator_shuffle` (void)  
*Default constructor.*
- `multi_iterator_shuffle` (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- `basic_multi_iterator< T > &init` (void)  
*Initialize the multi-index to the first shuffle.*
- `basic_multi_iterator< T > &operator++` (int)  
*The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.*

## Public Member Functions inherited from `GiNaC::basic_multi_iterator< T >`

- `basic_multi_iterator` (void)  
*Default constructor.*
- `basic_multi_iterator` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k.*
- `basic_multi_iterator` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*



- virtual `~basic_multi_iterator()`  
*Destructor.*
- `size_t size()` const  
*Returns the size of a multi\_iterator.*
- `bool overflow()` const  
*Return the overflow flag.*
- `const std::vector< T > & get_vector()` const  
*Returns a reference to the vector v.*
- `T operator[] (size_t i)` const  
*Subscription via [].*
- `T & operator[] (size_t i)`  
*Subscription via [].*
- `T operator() (size_t i)` const  
*Subscription via ().*
- `T & operator() (size_t i)`  
*Subscription via ().*
- virtual `basic_multi_iterator< T > & init()` (void)  
*Initialize the multi-index to.*
- virtual `basic_multi_iterator< T > & operator++` (int)  
*No effect for basic\_multi\_iterator.*

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_shuffle_prime< TT > &v)`

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::multi_iterator_shuffle< T >`

- `size_t N_internal`
- `std::vector< size_t > v_internal`
- `std::vector< T > v_orig`

### Protected Attributes inherited from `GiNaC::basic_multi_iterator< T >`

- `T N`
- `T B`
- `std::vector< T > v`
- `bool flag_overflow`

## 6.101.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_shuffle_prime< T >
```

The class `multi_iterator_shuffle_prime` defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).

## 6.101.2 Constructor & Destructor Documentation

### 6.101.2.1 multi\_iterator\_shuffle\_prime() [1/2]

```
template<class T >
GiNaC::multi_iterator_shuffle_prime< T >::multi_iterator_shuffle_prime (
    void ) [inline]
```

Default constructor.

### 6.101.2.2 multi\_iterator\_shuffle\_prime() [2/2]

```
template<class T >
GiNaC::multi_iterator_shuffle_prime< T >::multi_iterator_shuffle_prime (
    const std::vector< T > & a,
    const std::vector< T > & b ) [inline], [explicit]
```

Construct from a vector.

## 6.101.3 Member Function Documentation

### 6.101.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle_prime< T >::init (
    void ) [inline], [virtual]
```

Initialize the multi-index to the first shuffle.

Reimplemented from [GiNaC::multi\\_iterator\\_shuffle< T >](#).

## 6.101.4 Friends And Related Function Documentation

### 6.101.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
    std::ostream & os,
    const multi_iterator_shuffle_prime< TT > & v ) [friend]
```

The documentation for this class was generated from the following file:

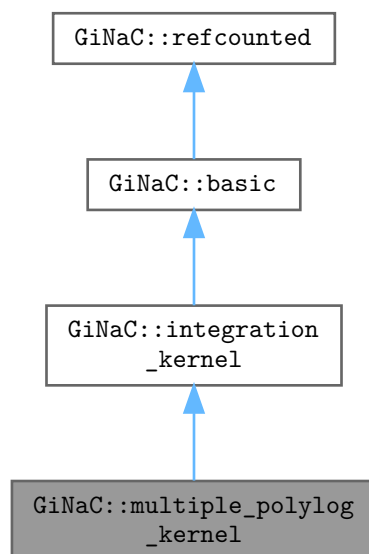
- [utils\\_multi\\_iterator.h](#)

## 6.102 GiNaC::multiple\_polylog\_kernel Class Reference

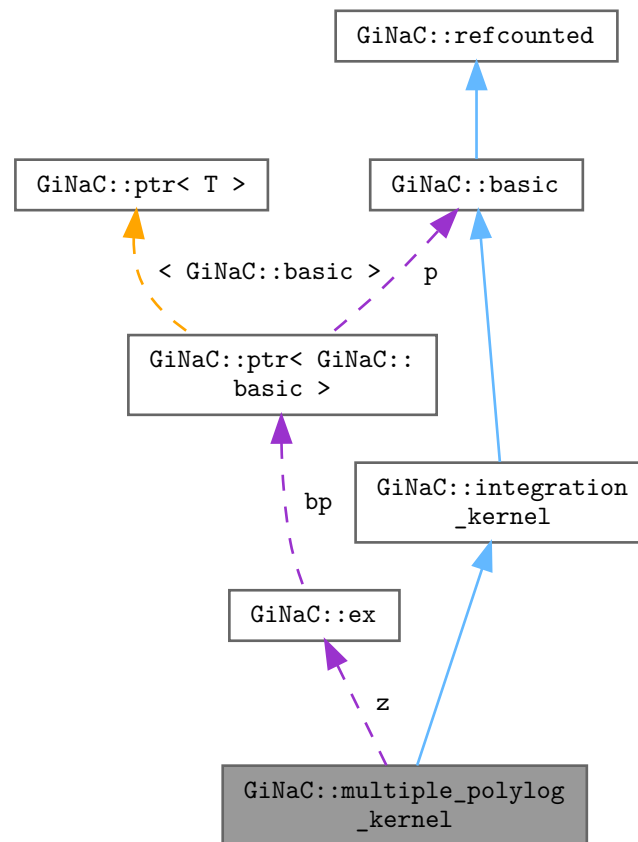
The integration kernel for multiple polylogarithms.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::multiple\_polylog\_kernel:



Collaboration diagram for `GiNaC::multiple_polylog_kernel`:



## Public Member Functions

- `multiple_polylog_kernel` (const `ex` &`z`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position `i`.*
- bool `is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*

## Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const

- *This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool `is_numeric` (void) const
- *This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual `ex Laurent_series` (const `ex &x`, int `order`) const
- *Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex &lambda`, int `N_trunc=0`) const
- *Evaluates the integrand at lambda.*
- size\_t `get_cache_size` (void) const
- *Returns the current size of the cache.*
- void `set_cache_step` (int `cache_steps`) const
- *Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int `i`) const
- *Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- `cln::cl_N series_coeff` (int `i`) const
- *Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()
- *basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic &other`)
- const `basic & operator=` (const `basic &other`)
- *basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const
- *Create a clone of this object on the heap.*
- virtual `ex eval` () const
- *Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const
- *Evaluate object numerically.*
- virtual `ex evalm` () const
- *Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const
- *Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic &i`) const
- *Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context &c`, unsigned `level=0`) const
- *Output to stream.*
- virtual void `dbgprint` () const
- *Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const
- *Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const
- *Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned `inf`) const
- *Information about the object.*
- virtual size\_t `nops` () const
- *Number of operands/members.*
- virtual `ex op` (size\_t `i`) const
- *Return operand/member at position i.*
- virtual `ex operator[]` (const `ex &index`) const

- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const

- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- [cln::cl\\_N\\_series\\_coeff\\_impl](#) (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::integration\\_kernel](#)

- virtual bool [uses\\_Laurent\\_series](#) () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method [Laurent\\_series](#) needs to be implemented).*
- virtual [cln::cl\\_N\\_series\\_coeff\\_impl](#) (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- [ex get\\_numerical\\_value\\_impl](#) (const [ex](#) &lambda, const [ex](#) &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- [ex](#) z

### Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.102.1 Detailed Description

The integration kernel for multiple polylogarithms.

This class represents the differential one-form

$$\omega^{\text{mpl}}(z) = \frac{d\lambda}{\lambda - z}$$

For the case  $z = 0$  the class [basic\\_log\\_kernel](#) should be used.



## 6.102.2 Constructor & Destructor Documentation

### 6.102.2.1 multiple\_polylog\_kernel()

```
GiNaC::multiple_polylog_kernel::multiple_polylog_kernel (
    const ex & z )
```

## 6.102.3 Member Function Documentation

### 6.102.3.1 nops()

```
size_t GiNaC::multiple_polylog_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.102.3.2 op()

```
ex GiNaC::multiple_polylog_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position *i*.

Reimplemented from [GiNaC::basic](#).

References [z](#).

### 6.102.3.3 let\_op()

```
ex & GiNaC::multiple_polylog_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position *i*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [z](#).

#### 6.102.3.4 `is_numeric()`

```
bool GiNaC::multiple_polylog_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), and [z](#).

#### 6.102.3.5 `series_coeff_impl()`

```
cln::cl_N GiNaC::multiple_polylog_kernel::series_coeff_impl (
    int i ) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), and [z](#).

#### 6.102.3.6 `do_print()`

```
void GiNaC::multiple_polylog_kernel::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::ex::print\(\)](#), and [z](#).

### 6.102.4 Member Data Documentation

#### 6.102.4.1 `z`

```
ex GiNaC::multiple_polylog_kernel::z [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

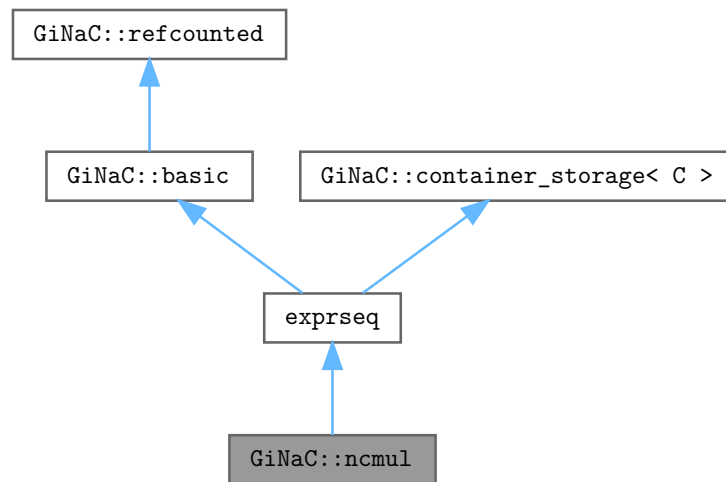
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.103 GiNaC::ncmul Class Reference

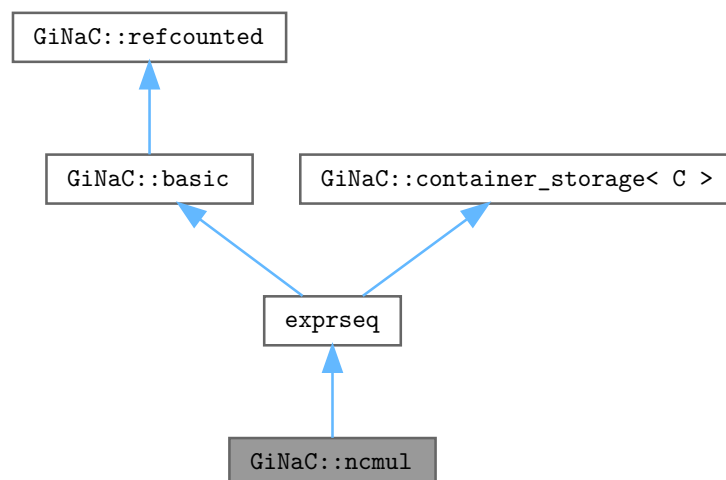
Non-commutative product of expressions.

```
#include <ncmul.h>
```

Inheritance diagram for GiNaC::ncmul:



Collaboration diagram for GiNaC::ncmul:



## Public Member Functions

- `ncmul` (const `ex` &lh, const `ex` &rh)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4, const `ex` &f5)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4, const `ex` &f5, const `ex` &f6)
- `ncmul` (const `exvector` &v)
- `ncmul` (`exvector` &&v)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- `ex coeff` (const `ex` &s, int `n`=1) const override  
*Return coefficient of degree n in object s.*
- `ex eval` () const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalm` () const override  
*Evaluate sums, products and integer powers of matrices.*
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- const `exvector` & `get_factors` () const

## Public Member Functions inherited from `GiNaC::container< C >`

- `container` (STLT const &s)
- `container` (STLT &&v)
- `container` (`exvector::const_iterator` b, `exvector::const_iterator` e)
- `container` (std::initializer\_list< `ex` > il)
- bool `info` (unsigned inf) const override  
*Information about the object.*
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- size\_t `nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex subs` (const `exmap` &m, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*

- void `read_archive` (const `archive_node` &n, `lst` &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- void `archive` (`archive_node` &n) const override  
*Archive the object.*
- `container` & `prepend` (const `ex` &b)  
*Add element at front.*
- `container` & `append` (const `ex` &b)  
*Add element at back.*
- `container` & `remove_first` ()  
*Remove first element.*
- `container` & `remove_last` ()  
*Remove last element.*
- `container` & `remove_all` ()  
*Remove all elements.*
- `container` & `sort` ()  
*Sort elements.*
- `container` & `unique` ()  
*Remove adjacent duplicate elements.*
- `const_iterator` `begin` () const
- `const_iterator` `end` () const
- `const_reverse_iterator` `rbegin` () const
- `const_reverse_iterator` `rend` () const

#### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex` `eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex` `evalf` () const  
*Evaluate object numerically.*
- virtual `ex` `evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex` `eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex` `eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*

- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex & let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex & operator\[\]](#) (const [ex](#) &index)
- virtual [ex & operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned [options](#)=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*

- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a non-commutative product.*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_csrc` (const `print_context` &c, unsigned level) const
- size\_t `count_factors` (const `ex` &e) const
- void `append_factors` (`exvector` &v, const `ex` &e) const
- `exvector expandchildren` (unsigned `options`) const

### Protected Member Functions inherited from [GiNaC::container< C >](#)

- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- virtual [ex thiscontainer](#) (const [STLT](#) &v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- virtual [ex thiscontainer](#) ([STLT](#) &&v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this↔\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned [options](#)=0) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)



## Friends

- class [power](#)
- [ex reeval\\_ncmul](#) (const [exvector](#) &v)
- [ex hold\\_ncmul](#) (const [exvector](#) &v)

## Additional Inherited Members

### Public Types inherited from [GiNaC::container< C >](#)

- typedef [STLT::const\\_iterator](#) [const\\_iterator](#)
- typedef [STLT::const\\_reverse\\_iterator](#) [const\\_reverse\\_iterator](#)

### Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

### Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef [C< ex >](#) [STLT](#)

### Static Protected Member Functions inherited from [GiNaC::container< C >](#)

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for [lst](#).*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for [lst](#).*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for [lst](#).*

### Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) ([STLT](#) &, [size\\_t](#))

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)

### 6.103.1 Detailed Description

Non-commutative product of expressions.

### 6.103.2 Constructor & Destructor Documentation

#### 6.103.2.1 `ncmul()` [1/7]

```
GiNaC::ncmul::ncmul (
    const ex & lh,
    const ex & rh )
```

#### 6.103.2.2 `ncmul()` [2/7]

```
GiNaC::ncmul::ncmul (
    const ex & f1,
    const ex & f2,
    const ex & f3 )
```

#### 6.103.2.3 `ncmul()` [3/7]

```
GiNaC::ncmul::ncmul (
    const ex & f1,
    const ex & f2,
    const ex & f3,
    const ex & f4 )
```

#### 6.103.2.4 `ncmul()` [4/7]

```
GiNaC::ncmul::ncmul (
    const ex & f1,
    const ex & f2,
    const ex & f3,
    const ex & f4,
    const ex & f5 )
```

**6.103.2.5 ncmul()** [5/7]

```
GiNaC::ncmul::ncmul (
    const ex & f1,
    const ex & f2,
    const ex & f3,
    const ex & f4,
    const ex & f5,
    const ex & f6 )
```

**6.103.2.6 ncmul()** [6/7]

```
GiNaC::ncmul::ncmul (
    const exvector & v )
```

**6.103.2.7 ncmul()** [7/7]

```
GiNaC::ncmul::ncmul (
    exvector && v )
```

**6.103.3 Member Function Documentation****6.103.3.1 precedence()**

```
unsigned GiNaC::ncmul::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< C >](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_csrc\(\)](#).

**6.103.3.2 info()**

```
bool GiNaC::ncmul::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::container< C >](#).

**6.103.3.3 degree()**

```
int GiNaC::ncmul::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::is\\_equal\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.103.3.4 ldegree()**

```
int GiNaC::ncmul::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::is\\_equal\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.103.3.5 expand()**

```
ex GiNaC::ncmul::expand (
    unsigned options = 0 ) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [k](#), [GiNaC::container< C >::op\(\)](#), [options](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

**6.103.3.6 coeff()**

```
ex GiNaC::ncmul::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [GiNaC::basic::is\\_equal\(\)](#), [n](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::add::coeff\(\)](#).

### 6.103.3.7 eval()

```
ex GiNaC::ncmul::eval ( ) const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following  $x, x_1, x_2, \dots$  stand for a symbolic variables of type `ex` and  $c, c_1, c_2, \dots$  stand for such expressions that contain a plain number.

- $\text{ncmul}(\dots, *(x_1, x_2), \dots, \text{ncmul}(x_3, x_4), \dots) \rightarrow \text{ncmul}(\dots, x_1, x_2, \dots, x_3, x_4, \dots)$  (associativity)
- $\text{ncmul}(x) \rightarrow x$
- $\text{ncmul}() \rightarrow 1$
- $\text{ncmul}(\dots, c_1, \dots, c_2, \dots) \rightarrow *(c_1, c_2, \text{ncmul}(\dots))$  (pull out commutative elements)
- $\text{ncmul}(x_1, y_1, x_2, y_2) \rightarrow *(\text{ncmul}(x_1, x_2), \text{ncmul}(y_1, y_2))$  (collect elements of same type)
- $\text{ncmul}(x_1, x_2, x_3, \dots) \rightarrow x::\text{eval\_ncmul}(x_1, x_2, x_3, \dots)$

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [append\\_factors\(\)](#), [GiNaC::return\\_types::commutative](#), [count\\_factors\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::factor\(\)](#), [factors](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_composite](#), [GiNaC::container\\_storage< C >::reserve](#) and [GiNaC::container\\_storage< C >::seq](#).

### 6.103.3.8 evalm()

```
ex GiNaC::ncmul::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::begin\(\)](#), [GiNaC::matrix::mul\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

### 6.103.3.9 get\_free\_indices()

```
exvector GiNaC::ncmul::get_free_indices ( ) const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::container< C >::nops\(\)](#), and [GiNaC::container< C >::op\(\)](#).

#### 6.103.3.10 thiscontainer() [1/2]

```
ex GiNaC::ncmul::thiscontainer (
    const exvector & v ) const [override]
```

#### 6.103.3.11 thiscontainer() [2/2]

```
ex GiNaC::ncmul::thiscontainer (
    exvector && v ) const [override]
```

#### 6.103.3.12 conjugate()

```
ex GiNaC::ncmul::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::is\\_clifford\\_tinfo\(\)](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::container< C >::nops\(\)](#), [return\\_type\(\)](#), and [return\\_type\\_tinfo\(\)](#).

#### 6.103.3.13 real\_part()

```
ex GiNaC::ncmul::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::basic::real\\_part\(\)](#).

#### 6.103.3.14 imag\_part()

```
ex GiNaC::ncmul::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::basic::imag\\_part\(\)](#).

**6.103.3.15 derivative()**

```
ex GiNaC::ncmul::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a non-commutative product.

It applies the product rule.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::ex::swap\(\)](#).

**6.103.3.16 return\_type()**

```
unsigned GiNaC::ncmul::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::container< C >::end\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_composite](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [conjugate\(\)](#).

**6.103.3.17 return\_type\_tinfo()**

```
return\_type\_t GiNaC::ncmul::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [conjugate\(\)](#).

**6.103.3.18 do\_print()**

```
void GiNaC::ncmul::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), [precedence\(\)](#), and [GiNaC::container< C >::printseq\(\)](#).

**6.103.3.19 do\_print\_csrc()**

```
void GiNaC::ncmul::do_print_csrc (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), [precedence\(\)](#), and [GiNaC::container< C >::printseq\(\)](#).

**6.103.3.20 count\_factors()**

```
size_t GiNaC::ncmul::count_factors (
    const ex & e ) const [protected]
```

References [GiNaC::return\\_types::commutative](#), [count\\_factors\(\)](#), [factors](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [count\\_factors\(\)](#), and [eval\(\)](#).

**6.103.3.21 append\_factors()**

```
void GiNaC::ncmul::append_factors (
    exvector & v,
    const ex & e ) const [protected]
```

References [append\\_factors\(\)](#), [GiNaC::return\\_types::commutative](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [append\\_factors\(\)](#), and [eval\(\)](#).

**6.103.3.22 expandchildren()**

```
exvector GiNaC::ncmul::expandchildren (
    unsigned options ) const [protected]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::ex::expand\(\)](#), [options](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [expand\(\)](#).

**6.103.3.23 get\_factors()**

```
const exvector & GiNaC::ncmul::get_factors ( ) const
```

References [GiNaC::container\\_storage< C >::seq](#).



## 6.103.4 Friends And Related Function Documentation

### 6.103.4.1 power

```
friend class power [friend]
```

### 6.103.4.2 reeval\_ncmul

```
ex reeval_ncmul (  
    const exvector & v ) [friend]
```

### 6.103.4.3 hold\_ncmul

```
ex hold_ncmul (  
    const exvector & v ) [friend]
```

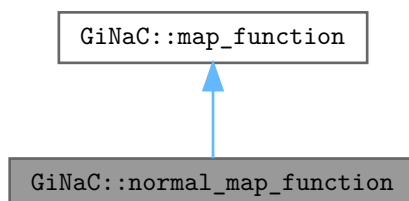
The documentation for this class was generated from the following files:

- [ncmul.h](#)
- [indexed.cpp](#)
- [ncmul.cpp](#)

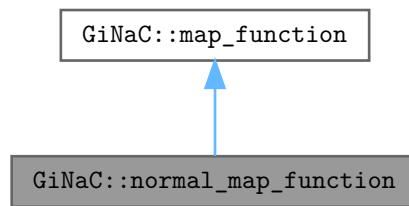
## 6.104 GiNaC::normal\_map\_function Struct Reference

Function object to be applied by [basic::normal\(\)](#).

Inheritance diagram for GiNaC::normal\_map\_function:



Collaboration diagram for `GiNaC::normal_map_function`:



## Public Member Functions

- `ex operator()` (const `ex` &`e`) override

## Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function()`
- virtual `ex operator()` (const `ex` &`e`)=0

## Additional Inherited Members

## Public Types inherited from `GiNaC::map_function`

- typedef const `ex` & `argument_type`
- typedef `ex` `result_type`

## 6.104.1 Detailed Description

Function object to be applied by `basic::normal()`.

## 6.104.2 Member Function Documentation

### 6.104.2.1 `operator()()`

```

ex GiNaC::normal_map_function::operator() (
    const ex & e ) [inline], [override], [virtual]
  
```

Implements `GiNaC::map_function`.

References `GiNaC::normal()`.

The documentation for this struct was generated from the following file:

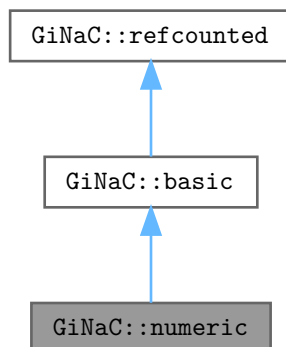
- `normal.cpp`

## 6.105 GiNaC::numeric Class Reference

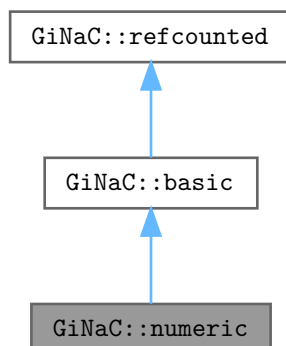
This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

```
#include <numeric.h>
```

Inheritance diagram for GiNaC::numeric:



Collaboration diagram for GiNaC::numeric:



### Public Member Functions

- [numeric](#) (int i)
- [numeric](#) (unsigned int i)
- [numeric](#) (long i)
- [numeric](#) (unsigned long i)

- `numeric` (long long i)
- `numeric` (unsigned long long i)
- `numeric` (long `numer`, long `denom`)  
*Constructor for rational numerics a/b.*
- `numeric` (double d)
- `numeric` (const char \*)  
*ctor from C-style string.*
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex` `coeff` (const `ex` &s, int `n`=1) const override  
*Return coefficient of degree n in object s.*
- bool `has` (const `ex` &other, unsigned `options`=0) const override  
*Disassemble real part and imaginary part to scan for the occurrence of a single number.*
- `ex` `eval` () const override  
*Evaluation of numbers doesn't do anything at all.*
- `ex` `evalf` () const override  
*Cast numeric into a floating-point object.*
- `ex` `subs` (const `exmap` &m, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex` `normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const override  
*Implementation of `ex::normal()` for a numeric.*
- `ex` `to_rational` (`exmap` &repl) const override  
*Implementation of `ex::to_rational()` for a numeric.*
- `ex` `to_polynomial` (`exmap` &repl) const override  
*Implementation of `ex::to_polynomial()` for a numeric.*
- `numeric` `integer_content` () const override
- `ex` `smod` (const `numeric` &xi) const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric` `max_coefficient` () const override  
*Implementation `ex::max_coefficient()`.*
- `ex` `conjugate` () const override
- `ex` `real_part` () const override
- `ex` `imag_part` () const override
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &`syms`) override  
*Read (a.k.a.*
- const `numeric` `add` (const `numeric` &other) const  
*Numerical addition method.*
- const `numeric` `sub` (const `numeric` &other) const  
*Numerical subtraction method.*
- const `numeric` `mul` (const `numeric` &other) const  
*Numerical multiplication method.*

- const [numeric div](#) (const [numeric](#) &other) const  
*Numerical division method.*
- const [numeric power](#) (const [numeric](#) &other) const  
*Numerical exponentiation.*
- const [numeric & add\\_dyn](#) (const [numeric](#) &other) const  
*Numerical addition method.*
- const [numeric & sub\\_dyn](#) (const [numeric](#) &other) const  
*Numerical subtraction method.*
- const [numeric & mul\\_dyn](#) (const [numeric](#) &other) const  
*Numerical multiplication method.*
- const [numeric & div\\_dyn](#) (const [numeric](#) &other) const  
*Numerical division method.*
- const [numeric & power\\_dyn](#) (const [numeric](#) &other) const  
*Numerical exponentiation.*
- const [numeric & operator=](#) (int i)
- const [numeric & operator=](#) (unsigned int i)
- const [numeric & operator=](#) (long i)
- const [numeric & operator=](#) (unsigned long i)
- const [numeric & operator=](#) (double d)
- const [numeric & operator=](#) (const char \*s)
- const [numeric inverse](#) () const  
*Inverse of a number.*
- [numeric step](#) () const  
*Return the step function of a numeric.*
- int [csgn](#) () const  
*Return the complex half-plane (left or right) in which the number lies.*
- int [compare](#) (const [numeric](#) &other) const  
*This method establishes a canonical order on all numbers.*
- bool [is\\_equal](#) (const [numeric](#) &other) const
- bool [is\\_zero](#) () const  
*True if object is zero.*
- bool [is\\_positive](#) () const  
*True if object is not complex and greater than zero.*
- bool [is\\_negative](#) () const  
*True if object is not complex and less than zero.*
- bool [is\\_integer](#) () const  
*True if object is a non-complex integer.*
- bool [is\\_pos\\_integer](#) () const  
*True if object is an exact integer greater than zero.*
- bool [is\\_nonneg\\_integer](#) () const  
*True if object is an exact integer greater or equal zero.*
- bool [is\\_even](#) () const  
*True if object is an exact even integer.*
- bool [is\\_odd](#) () const  
*True if object is an exact odd integer.*
- bool [is\\_prime](#) () const  
*Probabilistic primality test.*
- bool [is\\_rational](#) () const  
*True if object is an exact rational number, may even be complex (denominator may be unity).*
- bool [is\\_real](#) () const  
*True if object is a real integer, rational or float (but not complex).*

- bool `is_cinteger` () const  
*True if object is element of the domain of integers extended by I, i.e.*
- bool `is_crational` () const  
*True if object is an exact rational number, may even be complex (denominator may be unity).*
- bool `operator==` (const `numeric` &other) const
- bool `operator!=` (const `numeric` &other) const
- bool `operator<` (const `numeric` &other) const  
*Numerical comparison: less.*
- bool `operator<=` (const `numeric` &other) const  
*Numerical comparison: less or equal.*
- bool `operator>` (const `numeric` &other) const  
*Numerical comparison: greater.*
- bool `operator>=` (const `numeric` &other) const  
*Numerical comparison: greater or equal.*
- int `to_int` () const  
*Converts numeric types to machine's int.*
- long `to_long` () const  
*Converts numeric types to machine's long.*
- double `to_double` () const  
*Converts numeric types to machine's double.*
- `cln::cl_N to_cl_N` () const  
*Returns a new CLN object of type cl\_N, representing the value of \*this.*
- const `numeric real` () const  
*Real part of a number.*
- const `numeric imag` () const  
*Imaginary part of a number.*
- const `numeric numer` () const  
*Numerator.*
- const `numeric denom` () const  
*Denominator.*
- int `int_length` () const  
*Size in binary notation.*
- `numeric` (const `cln::cl_N` &z)  
*Ctor from CLN types.*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*

- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex &let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex &operator[] (const ex &index)`
- virtual `ex &operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept



## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff` for a numeric always returns 0.*
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `print_numeric` (const `print_context` &c, const char \*par\_open, const char \*par\_close, const char \*imag←\_sym, const char \*mul\_sym, unsigned level) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_csrc` (const `print_csrc` &c, unsigned level) const
- void `do_print_csrc_cl_N` (const `print_csrc_cl_N` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- `cln::cl_N` value

## Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.105.1 Detailed Description

This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

Objects of this type may directly be created by the user.

### 6.105.2 Constructor & Destructor Documentation

#### 6.105.2.1 `numeric()` [1/10]

```
GiNaC::numeric::numeric (  
    int i )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

#### 6.105.2.2 `numeric()` [2/10]

```
GiNaC::numeric::numeric (  
    unsigned int i )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

#### 6.105.2.3 `numeric()` [3/10]

```
GiNaC::numeric::numeric (  
    long i )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

#### 6.105.2.4 `numeric()` [4/10]

```
GiNaC::numeric::numeric (  
    unsigned long i )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.2.5 numeric()** [5/10]

```
GiNaC::numeric::numeric (
    long long i )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.2.6 numeric()** [6/10]

```
GiNaC::numeric::numeric (
    unsigned long long i )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.2.7 numeric()** [7/10]

```
GiNaC::numeric::numeric (
    long numer,
    long denom )
```

Constructor for rational numerics a/b.

**Exceptions**

<code>overflow_error</code>	(division by zero)
-----------------------------	--------------------

References [denom\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [numer\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.2.8 numeric()** [8/10]

```
GiNaC::numeric::numeric (
    double d )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 6.105.2.9 `numeric()` [9/10]

```
GiNaC::numeric::numeric (
    const char * s )
```

ctor from C-style string.

It also accepts complex numbers in [GiNaC](#) notation like "2+5\*I".

References [GiNaC::Digits](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 6.105.2.10 `numeric()` [10/10]

```
GiNaC::numeric::numeric (
    const cln::cl_N & z ) [explicit]
```

Ctor from CLN types.

This is for the initiated user or internal use only.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

## 6.105.3 Member Function Documentation

### 6.105.3.1 `precedence()`

```
unsigned GiNaC::numeric::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [print\\_numeric\(\)](#).

### 6.105.3.2 info()

```
bool GiNaC::numeric::info (
    unsigned int ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::expanded](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [is\\_even\(\)](#), [is\\_integer\(\)](#), [is\\_negative\(\)](#), [is\\_nonneg\\_integer\(\)](#), [is\\_odd\(\)](#), [is\\_pos\\_integer\(\)](#), [is\\_positive\(\)](#), [is\\_prime\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [GiNaC::info\\_flags::real](#).

Referenced by [GiNaC::abs\\_power\(\)](#), [GiNaC::csgn\\_power\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).

### 6.105.3.3 is\_polynomial()

```
bool GiNaC::numeric::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

### 6.105.3.4 degree()

```
int GiNaC::numeric::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

### 6.105.3.5 ldegree()

```
int GiNaC::numeric::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

### 6.105.3.6 coeff()

```
ex GiNaC::numeric::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), and [n](#).

Referenced by [GiNaC::pseries::mul\\_const\(\)](#).

### 6.105.3.7 has()

```
bool GiNaC::numeric::has (
    const ex & other,
    unsigned options = 0 ) const [override], [virtual]
```

Disassemble real part and imaginary part to scan for the occurrence of a single number.

Also handles the imaginary unit. It ignores the sign on both this and the argument, which may lead to what might appear as funny results:  $(2+i).has(-2) \rightarrow true$ . But this is consistent, since we also would like to have  $(-2+i).has(2) \rightarrow true$  and we want to think about the sign as a multiplicative factor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_num0\\_p](#), [GiNaC::i](#), [imag\(\)](#), [is\\_equal\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), and [real\(\)](#).

### 6.105.3.8 eval()

```
ex GiNaC::numeric::eval ( ) const [override], [virtual]
```

Evaluation of numbers doesn't do anything at all.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

### 6.105.3.9 evalf()

```
ex GiNaC::numeric::evalf ( ) const [override], [virtual]
```

Cast numeric into a floating-point object.

For example `exact.numeric(1)` is returned as a `1.000000000000000000000000` and so on according to how `Digits` is currently set. In case the object already was a floating point number the precision is trimmed to match the currently set default.

## Returns

an ex-handle to a numeric.

Reimplemented from [GiNaC::basic](#).

References [value](#).

### 6.105.3.10 subs()

```
ex GiNaC::numeric::subs (
    const exmap & m,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References `m`, `options`, and `GiNaC::basic::subs_one_level()`.

### 6.105.3.11 normal()

```
ex GiNaC::numeric::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of `ex::normal()` for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-rational real numbers with a temporary symbol.

## See also

ex::normal

Reimplemented from [GiNaC::basic](#).

References [denom\(\)](#), [GiNaC::l](#), [imag\(\)](#), [is\\_integer\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [numer\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.105.3.12 to\_rational()**

```
ex GiNaC::numeric::to_rational (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-rational real numbers with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::I](#), [imag\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.105.3.13 to\_polynomial()**

```
ex GiNaC::numeric::to_polynomial (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-integer real numbers with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::I](#), [imag\(\)](#), [is\\_integer\(\)](#), [is\\_real\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.105.3.14 integer\_content()**

```
numeric GiNaC::numeric::integer_content ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#).

**6.105.3.15 smod()**

```
ex GiNaC::numeric::smod (
    const numeric & xi ) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).



## Parameters

$xi$	modulus
------	---------

## Returns

mapped polynomial

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::smod\(\)](#).

**6.105.3.16 max\_coefficient()**

```
numeric GiNaC::numeric::max_coefficient ( ) const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#).

**6.105.3.17 conjugate()**

```
ex GiNaC::numeric::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [is\\_real\(\)](#), and [value](#).

**6.105.3.18 real\_part()**

```
ex GiNaC::numeric::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [value](#).

**6.105.3.19 imag\_part()**

`ex` `GiNaC::numeric::imag_part ( ) const [override], [virtual]`

Reimplemented from [GiNaC::basic](#).

References [value](#).

**6.105.3.20 archive()**

```
void GiNaC::numeric::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), [value](#), and [GiNaC::write\\_real\\_float\(\)](#).

**6.105.3.21 read\_archive()**

```
void GiNaC::numeric::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [c](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [n](#), [GiNaC::read\\_real\\_float\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.3.22 derivative()**

```
ex GiNaC::numeric::derivative (
    const symbol & s ) const [inline], [override], [protected], [virtual]
```

Implementation of [ex::diff](#) for a numeric always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

### 6.105.3.23 is\_equal\_same\_type()

```
bool GiNaC::numeric::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [is\\_equal\(\)](#).

### 6.105.3.24 calchash()

```
unsigned GiNaC::numeric::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 6.105.3.25 add()

```
const numeric GiNaC::numeric::add (
    const numeric & other ) const
```

Numerical addition method.

Adds argument to \*this and returns result as a numeric object.

References [value](#).

Referenced by [GiNaC::operator+\(\)](#), [GiNaC::operator++\(\)](#), [GiNaC::operator+=\(\)](#), and [GiNaC::operator--\(\)](#).

**6.105.3.26 sub()**

```
const numeric GiNaC::numeric::sub (
    const numeric & other ) const
```

Numerical subtraction method.

Subtracts argument from \*this and returns result as a numeric object.

References [value](#).

Referenced by [GiNaC::operator-\(\)](#), and [GiNaC::operator-=\(\)](#).

**6.105.3.27 mul()**

```
const numeric GiNaC::numeric::mul (
    const numeric & other ) const
```

Numerical multiplication method.

Multiplies \*this and argument and returns result as a numeric object.

References [value](#).

Referenced by [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::operator\\*\(\)](#), [GiNaC::operator\\*=\(\)](#), and [GiNaC::operator-\(\)](#).

**6.105.3.28 div()**

```
const numeric GiNaC::numeric::div (
    const numeric & other ) const
```

Numerical division method.

Divides \*this by argument and returns result as a numeric object.

**Exceptions**

<a href="#">overflow_error</a>	(division by zero)
--------------------------------	--------------------

References [value](#).

Referenced by [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::operator/\(\)](#), [GiNaC::operator/=\(\)](#), [GiNaC::basic::series\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

### 6.105.3.29 power()

```
const numeric GiNaC::numeric::power (
    const numeric & other ) const
```

Numerical exponentiation.

Raises \*this to the power given as argument and returns result as a numeric object.

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), and [value](#).

Referenced by [GiNaC::binomial\(\)](#), and [GiNaC::power::eval\(\)](#).

### 6.105.3.30 add\_dyn()

```
const numeric & GiNaC::numeric::add_dyn (
    const numeric & other ) const
```

Numerical addition method.

Adds argument to \*this and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num0\\_p](#), and [value](#).

### 6.105.3.31 sub\_dyn()

```
const numeric & GiNaC::numeric::sub_dyn (
    const numeric & other ) const
```

Numerical subtraction method.

Subtracts argument from \*this and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num0\\_p](#), and [value](#).

### 6.105.3.32 mul\_dyn()

```
const numeric & GiNaC::numeric::mul_dyn (
    const numeric & other ) const
```

Numerical multiplication method.

Multiplies \*this and argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num1\\_p](#), and [value](#).

Referenced by [GiNaC::power::expand\\_add\\_2\(\)](#).

### 6.105.3.33 div\_dyn()

```
const numeric & GiNaC::numeric::div_dyn (
    const numeric & other ) const
```

Numerical division method.

Divides \*this by argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

## Exceptions

<code>overflow_error</code>	(division by zero)
-----------------------------	--------------------

References [GiNaC::\\_num1\\_p](#), and [value](#).

**6.105.3.34 power\_dyn()**

```
const numeric & GiNaC::numeric::power_dyn (
    const numeric & other ) const
```

Numerical exponentiation.

Raises \*this to the power given as argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), and [value](#).

**6.105.3.35 operator=() [1/6]**

```
const numeric & GiNaC::numeric::operator= (
    int i )
```

References [operator=\(\)](#).

Referenced by [operator=\(\)](#).

**6.105.3.36 operator=() [2/6]**

```
const numeric & GiNaC::numeric::operator= (
    unsigned int i )
```

References [operator=\(\)](#).

**6.105.3.37 operator=() [3/6]**

```
const numeric & GiNaC::numeric::operator= (
    long i )
```

References [operator=\(\)](#).

**6.105.3.38 operator=()** [4/6]

```
const numeric & GiNaC::numeric::operator= (
    unsigned long i )
```

References [operator=\(\)](#).

**6.105.3.39 operator=()** [5/6]

```
const numeric & GiNaC::numeric::operator= (
    double d )
```

References [operator=\(\)](#).

**6.105.3.40 operator=()** [6/6]

```
const numeric & GiNaC::numeric::operator= (
    const char * s )
```

References [operator=\(\)](#).

**6.105.3.41 inverse()**

```
const numeric GiNaC::numeric::inverse ( ) const
```

Inverse of a number.

References [value](#).

Referenced by [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::interpolate\(\)](#), and [GiNaC::psi1\\_eval\(\)](#).

**6.105.3.42 step()**

```
numeric GiNaC::numeric::step ( ) const
```

Return the step function of a numeric.

The imaginary part of it is ignored because the step function is generally considered real but a numeric may develop a small imaginary part due to rounding errors.

References [r](#), and [value](#).

**6.105.3.43 csgn()**

```
int GiNaC::numeric::csgn ( ) const
```

Return the complex half-plane (left or right) in which the number lies.

$\text{csgn}(x) == 0$  for  $x == 0$ ,  $\text{csgn}(x) == 1$  for  $\text{Re}(x) > 0$  or  $\text{Re}(x) = 0$  and  $\text{Im}(x) > 0$ ,  $\text{csgn}(x) == -1$  for  $\text{Re}(x) < 0$  or  $\text{Re}(x) = 0$  and  $\text{Im}(x) < 0$ .

See also

`numeric::compare(const numeric &other)`

References [r](#), and [value](#).

**6.105.3.44 compare()**

```
int GiNaC::numeric::compare (
    const numeric & other ) const
```

This method establishes a canonical order on all numbers.

For complex numbers this is not possible in a mathematically consistent way but we need to establish some order and it ought to be fast. So we simply define it to be compatible with our method `csgn`.

Returns

`csgn(*this-other)`

See also

[numeric::csgn\(\)](#)

References [value](#).

Referenced by [GiNaC::power::eval\(\)](#).

**6.105.3.45 is\_equal()**

```
bool GiNaC::numeric::is_equal (
    const numeric & other ) const
```

References [value](#).

Referenced by [GiNaC::atan\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::exp\\_eval\(\)](#), [has\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::tan\\_eval\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).



### 6.105.3.46 is\_zero()

```
bool GiNaC::numeric::is_zero ( ) const
```

True if object is zero.

References [value](#).

Referenced by [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_expairseq\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::gcd\(\)](#), [has\(\)](#), [info\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::step\\_eval\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).

### 6.105.3.47 is\_positive()

```
bool GiNaC::numeric::is_positive ( ) const
```

True if object is not complex and greater than zero.

References [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [info\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi2\\_eval\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

### 6.105.3.48 is\_negative()

```
bool GiNaC::numeric::is_negative ( ) const
```

True if object is not complex and less than zero.

References [value](#).

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [info\(\)](#), and [GiNaC::pseries::power\\_const\(\)](#).

### 6.105.3.49 is\_integer()

```
bool GiNaC::numeric::is_integer ( ) const
```

True if object is a non-complex integer.

References [value](#).

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::gcd\(\)](#), [info\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::mod\(\)](#), [normal\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::smod\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [to\\_int\(\)](#), [to\\_long\(\)](#), [to\\_polynomial\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).

#### 6.105.3.50 `is_pos_integer()`

```
bool GiNaC::numeric::is_pos_integer ( ) const
```

True if object is an exact integer greater than zero.

References [value](#).

Referenced by [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), and [info\(\)](#).

#### 6.105.3.51 `is_nonneg_integer()`

```
bool GiNaC::numeric::is_nonneg_integer ( ) const
```

True if object is an exact integer greater or equal zero.

References [value](#).

Referenced by [GiNaC::binomial\\_sym\(\)](#), [info\(\)](#), and [print\\_numeric\(\)](#).

#### 6.105.3.52 `is_even()`

```
bool GiNaC::numeric::is_even ( ) const
```

True if object is an exact even integer.

References [value](#).

Referenced by [info\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

#### 6.105.3.53 `is_odd()`

```
bool GiNaC::numeric::is_odd ( ) const
```

True if object is an exact odd integer.

References [value](#).

Referenced by [info\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), and [GiNaC::matrix::pow\(\)](#).

#### 6.105.3.54 is\_prime()

```
bool GiNaC::numeric::is_prime ( ) const
```

Probabilistic primality test.

##### Returns

true if object is exact integer and prime.

References [value](#).

Referenced by [info\(\)](#).

#### 6.105.3.55 is\_rational()

```
bool GiNaC::numeric::is_rational ( ) const
```

True if object is an exact rational number, may even be complex (denominator may be unity).

References [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [info\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [normal\(\)](#), and [to\\_rational\(\)](#).

#### 6.105.3.56 is\_real()

```
bool GiNaC::numeric::is_real ( ) const
```

True if object is a real integer, rational or float (but not complex).

References [value](#).

Referenced by [GiNaC::atan\(\)](#), [GiNaC::beta\\_eval\(\)](#), [conjugate\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [denom\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::fsolve\(\)](#), [has\(\)](#), [info\(\)](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [normal\(\)](#), [numer\(\)](#), [operator<\(\)](#), [operator<=\(\)](#), [operator>\(\)](#), [operator>=\(\)](#), [GiNaC::step\\_eval\(\)](#), [to\\_double\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

#### 6.105.3.57 is\_cinteger()

```
bool GiNaC::numeric::is_cinteger ( ) const
```

True if object is element of the domain of integers extended by  $i$ , i.e.

is of the form  $a+bi$ , where  $a$  and  $b$  are integers.

References [is\\_real\(\)](#), and [value](#).

Referenced by [info\(\)](#).

**6.105.3.58 is\_crational()**

```
bool GiNaC::numeric::is_crational ( ) const
```

True if object is an exact rational number, may even be complex (denominator may be unity).

References [is\\_real\(\)](#), and [value](#).

Referenced by [GiNaC::power::eval\(\)](#), and [info\(\)](#).

**6.105.3.59 operator==()**

```
bool GiNaC::numeric::operator== (
    const numeric & other ) const
```

References [value](#).

**6.105.3.60 operator"!="()**

```
bool GiNaC::numeric::operator!= (
    const numeric & other ) const
```

References [value](#).

**6.105.3.61 operator<()**

```
bool GiNaC::numeric::operator< (
    const numeric & other ) const
```

Numerical comparison: less.

**Exceptions**

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References [is\\_real\(\)](#), and [value](#).

**6.105.3.62 operator<=()**

```
bool GiNaC::numeric::operator<= (
    const numeric & other ) const
```

Numerical comparison: less or equal.

## Exceptions

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References [is\\_real\(\)](#), and [value](#).

**6.105.3.63 operator>()**

```
bool GiNaC::numeric::operator> (
    const numeric & other ) const
```

Numerical comparison: greater.

## Exceptions

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References [is\\_real\(\)](#), and [value](#).

**6.105.3.64 operator>=()**

```
bool GiNaC::numeric::operator>= (
    const numeric & other ) const
```

Numerical comparison: greater or equal.

## Exceptions

<i>invalid_argument</i>	(complex inequality)
-------------------------	----------------------

References [is\\_real\(\)](#), and [value](#).

**6.105.3.65 to\_int()**

```
int GiNaC::numeric::to_int ( ) const
```

Converts numeric types to machine's int.

You should check with [is\\_integer\(\)](#) if the number is really an integer before calling this method. You may also consider checking the range first.

References [GINAC\\_ASSERT](#), [is\\_integer\(\)](#), and [value](#).

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::s](#)

**6.105.3.66 to\_long()**

```
long GiNaC::numeric::to_long ( ) const
```

Converts numeric types to machine's long.

You should check with [is\\_integer\(\)](#) if the number is really an integer before calling this method. You may also consider checking the range first.

References [GINAC\\_ASSERT](#), [is\\_integer\(\)](#), and [value](#).

Referenced by [GiNaC::power::expand\(\)](#), and [GiNaC::power::expand\\_add\(\)](#).

**6.105.3.67 to\_double()**

```
double GiNaC::numeric::to_double ( ) const
```

Converts numeric types to machine's double.

You should check with [is\\_real\(\)](#) if the number is really not complex before calling this method.

References [GINAC\\_ASSERT](#), [is\\_real\(\)](#), and [value](#).

**6.105.3.68 to\_cl\_N()**

```
cln::cl_N GiNaC::numeric::to_cl_N ( ) const
```

Returns a new CLN object of type `cl_N`, representing the value of `*this`.

This method may be used when mixing [GiNaC](#) and CLN in one project.

References [value](#).

Referenced by [GiNaC::atan\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::mod\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::smod\(\)](#).

**6.105.3.69 real()**

```
const numeric GiNaC::numeric::real ( ) const
```

Real part of a number.

References [value](#).

Referenced by [GiNaC::csgn\\_eval\(\)](#), [GiNaC::power::eval\(\)](#), [has\(\)](#), [normal\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::step\\_eval\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.105.3.70 imag()**

```
const numeric GiNaC::numeric::imag ( ) const
```

Imaginary part of a number.

References [value](#).

Referenced by [GiNaC::csgn\\_eval\(\)](#), [has\(\)](#), [normal\(\)](#), [GiNaC::step\\_eval\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.105.3.71 numer()**

```
const numeric GiNaC::numeric::numer ( ) const
```

Numerator.

Computes the numerator of rational numbers, rationalized numerator of complex if real and imaginary part are both rational numbers (i.e `numer(4/3+5/6*I) == 8+5*I`), the number carrying the sign in all other cases.

References [is\\_real\(\)](#), [r](#), and [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [normal\(\)](#), and [numeric\(\)](#).

**6.105.3.72 denom()**

```
const numeric GiNaC::numeric::denom ( ) const
```

Denominator.

Computes the denominator of rational numbers, common integer denominator of complex if real and imaginary part are both rational numbers (i.e `denom(4/3+5/6*I) == 6`), one in all other cases.

References [GiNaC::\\_num1\\_p](#), [is\\_real\(\)](#), [r](#), and [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [normal\(\)](#), and [numeric\(\)](#).

**6.105.3.73 int\_length()**

```
int GiNaC::numeric::int_length ( ) const
```

Size in binary notation.

For integers, this is the smallest  $n \geq 0$  such that  $-2^n \leq x < 2^n$ . If  $x > 0$ , this is the unique  $n > 0$  such that  $2^{n-1} \leq x < 2^n$ .

**Returns**

number of bits (excluding sign) needed to represent that number in two's complement if it is an integer, 0 otherwise.

References [value](#).

Referenced by [GiNaC::heur\\_gcd\\_z\(\)](#).



#### 6.105.3.74 print\_numeric()

```
void GiNaC::numeric::print_numeric (
    const print\_context & c,
    const char * par_open,
    const char * par_close,
    const char * imag_sym,
    const char * mul_sym,
    unsigned level ) const [protected]
```

References [c](#), [is\\_nonneg\\_integer\(\)](#), [precedence\(\)](#), [GiNaC::print\\_real\\_number\(\)](#), [r](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

#### 6.105.3.75 do\_print()

```
void GiNaC::numeric::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_numeric\(\)](#).

#### 6.105.3.76 do\_print\_latex()

```
void GiNaC::numeric::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_numeric\(\)](#).

#### 6.105.3.77 do\_print\_csrc()

```
void GiNaC::numeric::do_print_csrc (
    const print\_csrc & c,
    unsigned level ) const [protected]
```

References [c](#), [is\\_real\(\)](#), [GiNaC::print\\_real\\_csrc\(\)](#), and [value](#).

**6.105.3.78 do\_print\_csrc\_cl\_N()**

```
void GiNaC::numeric::do_print_csrc_cl_N (
    const print\_csrc\_cl\_N & c,
    unsigned level ) const [protected]
```

References [c](#), [is\\_real\(\)](#), [GiNaC::print\\_real\\_cl\\_N\(\)](#), and [value](#).

**6.105.3.79 do\_print\_tree()**

```
void GiNaC::numeric::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), and [value](#).

**6.105.3.80 do\_print\_python\_repr()**

```
void GiNaC::numeric::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_numeric\(\)](#).

**6.105.4 Member Data Documentation****6.105.4.1 value**

```
cln::cl_N GiNaC::numeric::value [protected]
```

Referenced by [add\(\)](#), [add\\_dyn\(\)](#), [archive\(\)](#), [calchash\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [csgn\(\)](#), [denom\(\)](#), [div\(\)](#), [div\\_dyn\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_tree\(\)](#), [evalf\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [imag\(\)](#), [imag\\_part\(\)](#), [int\\_length\(\)](#), [inverse\(\)](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [is\\_equal\(\)](#), [is\\_even\(\)](#), [is\\_integer\(\)](#), [is\\_negative\(\)](#), [is\\_nonneg\\_integer\(\)](#), [is\\_odd\(\)](#), [is\\_pos\\_integer\(\)](#), [is\\_positive\(\)](#), [is\\_prime\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), [mul\(\)](#), [mul\\_dyn\(\)](#), [numer\(\)](#), [numeric\(\)](#), [operator!=\(\)](#), [operator<\(\)](#), [operator<=\(\)](#), [operator==\(\)](#), [operator>\(\)](#), [operator>=\(\)](#), [power\(\)](#), [power\\_dyn\(\)](#), [print\\_numeric\(\)](#), [read\\_archive\(\)](#), [real\(\)](#), [real\\_part\(\)](#), [step\(\)](#), [sub\(\)](#), [sub\\_dyn\(\)](#), [to\\_cl\\_N\(\)](#), [to\\_double\(\)](#), [to\\_int\(\)](#), and [to\\_long\(\)](#).

The documentation for this class was generated from the following files:

- [numeric.h](#)
- [normal.cpp](#)
- [numeric.cpp](#)

## 6.106 GiNaC::op0\_is\_equal Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- `bool operator() (const ex &lh, const ex &rh) const`

### 6.106.1 Member Function Documentation

#### 6.106.1.1 operator()()

```
bool GiNaC::op0_is_equal::operator() (
    const ex & lh,
    const ex & rh ) const [inline]
```

References [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

The documentation for this struct was generated from the following file:

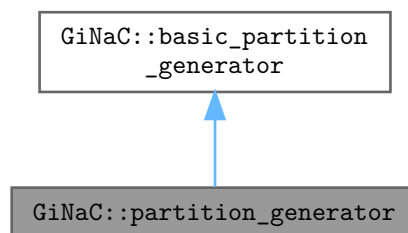
- [ex.h](#)

## 6.107 GiNaC::partition\_generator Class Reference

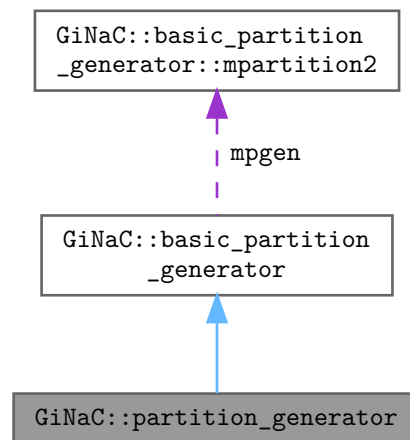
Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for `GiNaC::partition_generator`:



Collaboration diagram for `GiNaC::partition_generator`:



## Public Member Functions

- [partition\\_generator](#) (unsigned n\_, unsigned m\_)
- const std::vector< unsigned > & [get](#) () const
- bool [next](#) ()

## Private Attributes

- std::vector< unsigned > [partition](#)
- bool [current\\_updated](#)

## Additional Inherited Members

Protected Member Functions inherited from [GiNaC::basic\\_partition\\_generator](#)

- [basic\\_partition\\_generator](#) (unsigned n\_, unsigned m\_)

Protected Attributes inherited from [GiNaC::basic\\_partition\\_generator](#)

- [mpartition2](#) [mpgen](#)

### 6.107.1 Detailed Description

Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.

## 6.107.2 Constructor & Destructor Documentation

### 6.107.2.1 partition\_generator()

```
GiNaC::partition_generator::partition_generator (
    unsigned n_,
    unsigned m_ ) [inline]
```

## 6.107.3 Member Function Documentation

### 6.107.3.1 get()

```
const std::vector< unsigned > & GiNaC::partition_generator::get ( ) const [inline]
```

References [current\\_updated](#), [GiNaC::basic\\_partition\\_generator::mpartition2::m](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), [partition](#), and [GiNaC::basic\\_partition\\_generator::mpartition2::x](#).

### 6.107.3.2 next()

```
bool GiNaC::partition_generator::next ( ) [inline]
```

References [current\\_updated](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), and [GiNaC::basic\\_partition\\_generator::mpartition2::next\\_pa](#)

## 6.107.4 Member Data Documentation

### 6.107.4.1 partition

```
std::vector<unsigned> GiNaC::partition_generator::partition [mutable], [private]
```

Referenced by [get\(\)](#).

### 6.107.4.2 `current_updated`

```
bool GiNaC::partition_generator::current_updated [mutable], [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

The documentation for this class was generated from the following file:

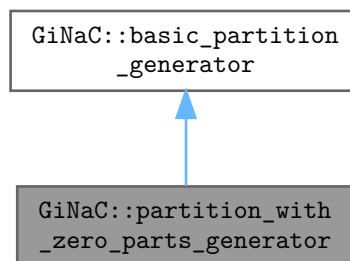
- [utils.h](#)

## 6.108 `GiNaC::partition_with_zero_parts_generator` Class Reference

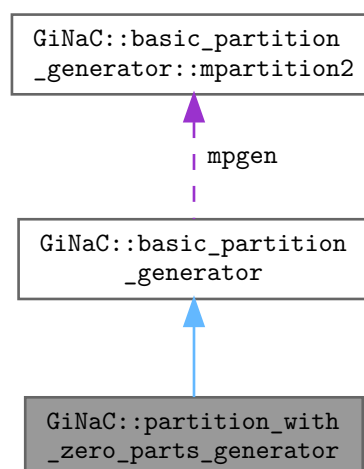
Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for `GiNaC::partition_with_zero_parts_generator`:



Collaboration diagram for `GiNaC::partition_with_zero_parts_generator`:



## Public Member Functions

- [partition\\_with\\_zero\\_parts\\_generator](#) (unsigned n\_, unsigned m\_)
- const std::vector< unsigned > & [get](#) () const
- bool [next](#) ()

## Private Attributes

- unsigned [m](#)
- std::vector< unsigned > [partition](#)
- bool [current\\_updated](#)

## Additional Inherited Members

### Protected Member Functions inherited from [GiNaC::basic\\_partition\\_generator](#)

- [basic\\_partition\\_generator](#) (unsigned n\_, unsigned m\_)

### Protected Attributes inherited from [GiNaC::basic\\_partition\\_generator](#)

- [mpartition2](#) [mpgen](#)

## 6.108.1 Detailed Description

Generate all bounded combinatorial partitions of an integer n with exactly m parts (including zero parts) in non-decreasing order.

## 6.108.2 Constructor & Destructor Documentation

### 6.108.2.1 [partition\\_with\\_zero\\_parts\\_generator](#)()

```
GiNaC::partition_with_zero_parts_generator::partition_with_zero_parts_generator (
    unsigned n_,
    unsigned m_ ) [inline]
```

## 6.108.3 Member Function Documentation

### 6.108.3.1 `get()`

```
const std::vector< unsigned > & GiNaC::partition_with_zero_parts_generator::get ( ) const  
[inline]
```

References [current\\_updated](#), [GiNaC::basic\\_partition\\_generator::mpartition2::m](#), [m](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), [partition](#), and [GiNaC::basic\\_partition\\_generator::mpartition2::x](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

### 6.108.3.2 `next()`

```
bool GiNaC::partition_with_zero_parts_generator::next ( ) [inline]
```

References [current\\_updated](#), [GiNaC::basic\\_partition\\_generator::mpartition2::m](#), [m](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), [GiNaC::basic\\_partition\\_generator::mpartition2::n](#), and [GiNaC::basic\\_partition\\_generator::mpartition2::next\\_partition\(\)](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

## 6.108.4 Member Data Documentation

### 6.108.4.1 `m`

```
unsigned GiNaC::partition_with_zero_parts_generator::m [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

### 6.108.4.2 `partition`

```
std::vector<unsigned> GiNaC::partition_with_zero_parts_generator::partition [mutable], [private]
```

Referenced by [get\(\)](#).

### 6.108.4.3 `current_updated`

```
bool GiNaC::partition_with_zero_parts_generator::current_updated [mutable], [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

The documentation for this class was generated from the following file:

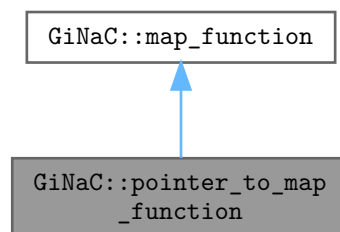
- [utils.h](#)



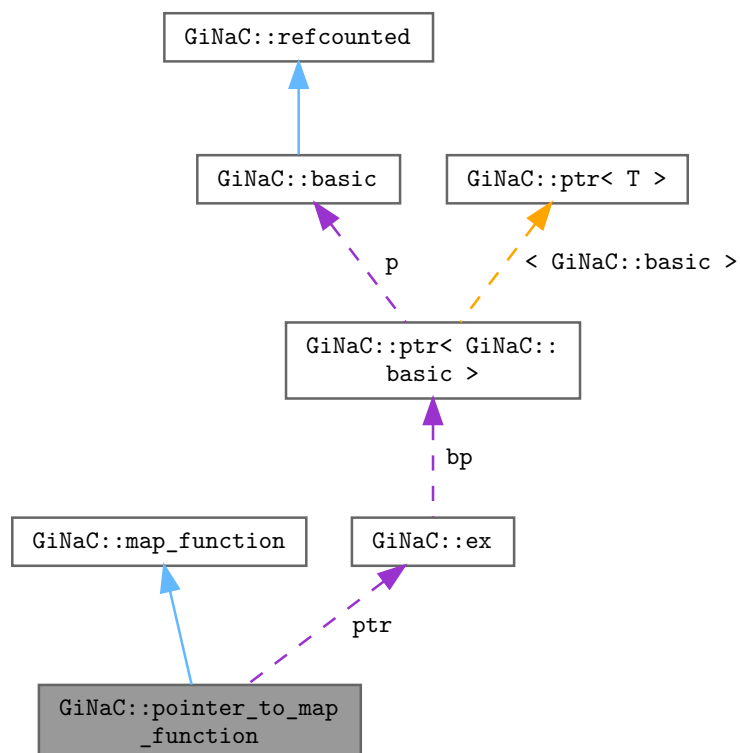
## 6.109 GiNaC::pointer\_to\_map\_function Class Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function:



Collaboration diagram for GiNaC::pointer\_to\_map\_function:



## Public Member Functions

- [pointer\\_to\\_map\\_function](#) ([ex](#) x(const [ex](#) &))
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()
- virtual [ex operator\(\)](#) (const [ex](#) &e)=0

## Protected Attributes

- [ex\(\\* ptr\)](#)(const [ex](#) &)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.109.1 Constructor & Destructor Documentation

### 6.109.1.1 [pointer\\_to\\_map\\_function\(\)](#)

```
GiNaC::pointer_to_map_function::pointer_to_map_function (
    ex xconst ex & ) [inline], [explicit]
```

## 6.109.2 Member Function Documentation

### 6.109.2.1 [operator>\(\)\(\)](#)

```
ex GiNaC::pointer_to_map_function::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [ptr](#).

### 6.109.3 Member Data Documentation

#### 6.109.3.1 ptr

```
ex(* GiNaC::pointer_to_map_function::ptr) (const ex &) [protected]
```

Referenced by [operator\(\)\(\)](#).

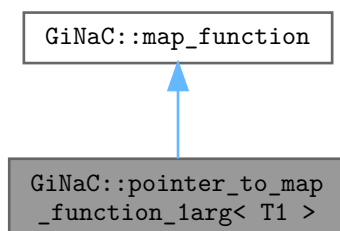
The documentation for this class was generated from the following file:

- [ex.h](#)

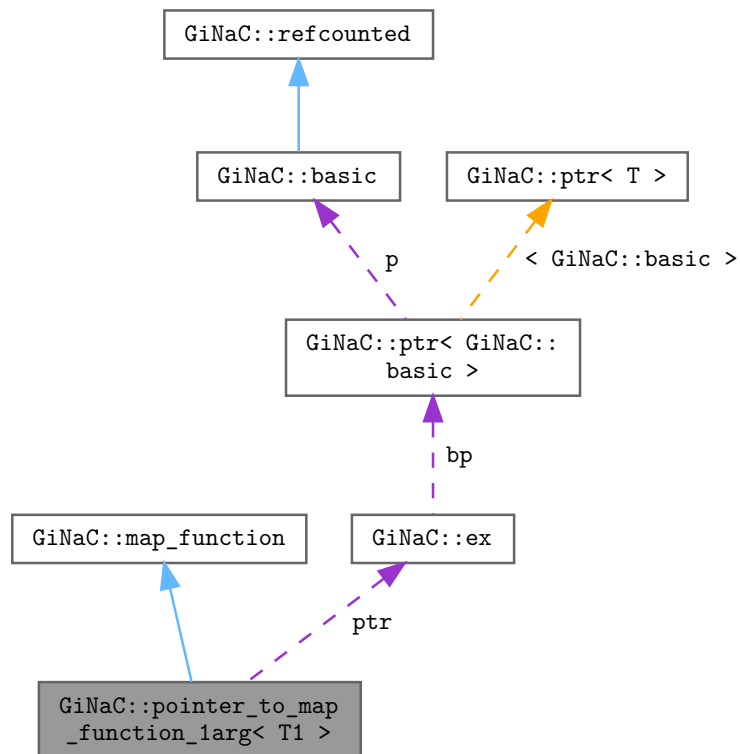
### 6.110 GiNaC::pointer\_to\_map\_function\_1arg< T1 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function\_1arg< T1 >:



Collaboration diagram for `GiNaC::pointer_to_map_function_1arg< T1 >`:



## Public Member Functions

- `pointer_to_map_function_1arg` (`ex` x(const `ex` &, T1), T1 a1)
- `ex operator()` (const `ex` &e) override

## Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function` ()
- virtual `ex operator()` (const `ex` &e)=0

## Protected Attributes

- `ex(* ptr)`(const `ex` &, T1)
- T1 `arg1`

## Additional Inherited Members

### Public Types inherited from `GiNaC::map_function`

- typedef const `ex` & `argument_type`
- typedef `ex` `result_type`

## 6.110.1 Constructor & Destructor Documentation

### 6.110.1.1 pointer\_to\_map\_function\_1arg()

```
template<class T1 >
GiNaC::pointer_to_map_function_1arg< T1 >::pointer_to_map_function_1arg (
    ex xconst ex &, T1,
    T1 a1 ) [inline], [explicit]
```

## 6.110.2 Member Function Documentation

### 6.110.2.1 operator>()

```
template<class T1 >
ex GiNaC::pointer_to_map_function_1arg< T1 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::arg1](#), and [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::ptr](#).

## 6.110.3 Member Data Documentation

### 6.110.3.1 ptr

```
template<class T1 >
ex(* GiNaC::pointer_to_map_function_1arg< T1 >::ptr) (const ex &, T1) [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::operator>\(\)](#).

### 6.110.3.2 arg1

```
template<class T1 >
T1 GiNaC::pointer_to_map_function_1arg< T1 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::operator>\(\)](#).

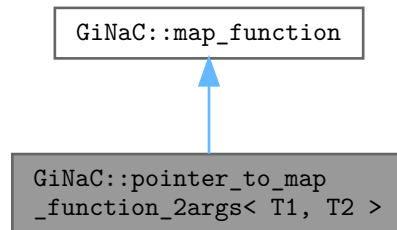
The documentation for this class was generated from the following file:

- [ex.h](#)

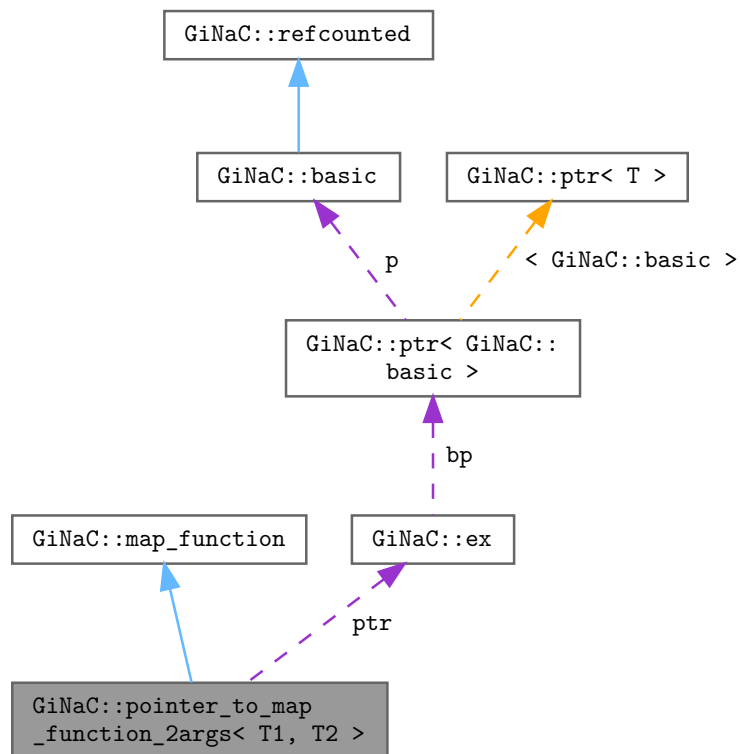
## 6.111 GiNaC::pointer\_to\_map\_function\_2args< T1, T2 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >:



Collaboration diagram for GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >:



## Public Member Functions

- [pointer\\_to\\_map\\_function\\_2args](#) ([ex](#) x(const [ex](#) &, T1, T2), T1 a1, T2 a2)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()
- virtual [ex operator\(\)](#) (const [ex](#) &e)=0

## Protected Attributes

- [ex](#)(\* [ptr](#) )(const [ex](#) &, T1, T2)
- T1 [arg1](#)
- T2 [arg2](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.111.1 Constructor & Destructor Documentation

### 6.111.1.1 [pointer\\_to\\_map\\_function\\_2args\(\)](#)

```
template<class T1 , class T2 >
GiNaC::pointer_to_map_function_2args< T1, T2 >::pointer_to_map_function_2args (
    ex xconst ex &, T1, T2,
    T1 a1,
    T2 a2 ) [inline], [explicit]
```

## 6.111.2 Member Function Documentation

### 6.111.2.1 [operator\(\)](#)()

```
template<class T1 , class T2 >
ex GiNaC::pointer_to_map_function_2args< T1, T2 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::arg1](#), [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::arg2](#), and [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::ptr](#).

### 6.111.3 Member Data Documentation

#### 6.111.3.1 ptr

```
template<class T1 , class T2 >
ex(* GiNaC::pointer_to_map_function_2args< T1, T2 >::ptr) (const ex &, T1, T2) [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::operator\(\)\(\)](#).

#### 6.111.3.2 arg1

```
template<class T1 , class T2 >
T1 GiNaC::pointer_to_map_function_2args< T1, T2 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::operator\(\)\(\)](#).

#### 6.111.3.3 arg2

```
template<class T1 , class T2 >
T2 GiNaC::pointer_to_map_function_2args< T1, T2 >::arg2 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::operator\(\)\(\)](#).

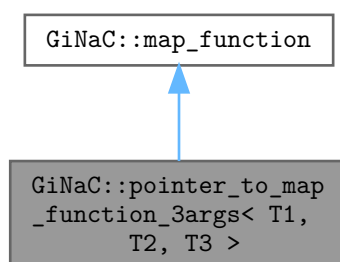
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.112 GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_map_function_3args< T1, T2, T3 >`:







## Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.112.1 Constructor & Destructor Documentation

### 6.112.1.1 [pointer\\_to\\_map\\_function\\_3args\(\)](#)

```
template<class T1 , class T2 , class T3 >
GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >::pointer_to_map_function_3args (
    ex xconst ex &, T1, T2, T3,
    T1 a1,
    T2 a2,
    T3 a3 ) [inline], [explicit]
```

## 6.112.2 Member Function Documentation

### 6.112.2.1 [operator>\(\)\(\)](#)

```
template<class T1 , class T2 , class T3 >
ex GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::arg1](#), [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::a](#), [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::arg3](#), and [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::ptr](#).

## 6.112.3 Member Data Documentation

### 6.112.3.1 [ptr](#)

```
template<class T1 , class T2 , class T3 >
ex(* GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >::ptr) (const ex &, T1, T2, T3) [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator>\(\)\(\)](#).

### 6.112.3.2 arg1

```
template<class T1 , class T2 , class T3 >
T1 GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator\(\)](#).

### 6.112.3.3 arg2

```
template<class T1 , class T2 , class T3 >
T2 GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::arg2 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator\(\)](#).

### 6.112.3.4 arg3

```
template<class T1 , class T2 , class T3 >
T3 GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::arg3 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator\(\)](#).

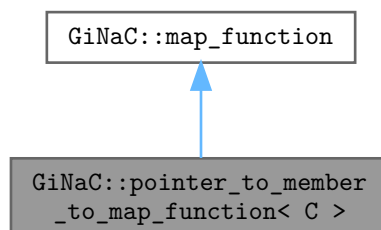
The documentation for this class was generated from the following file:

- [ex.h](#)

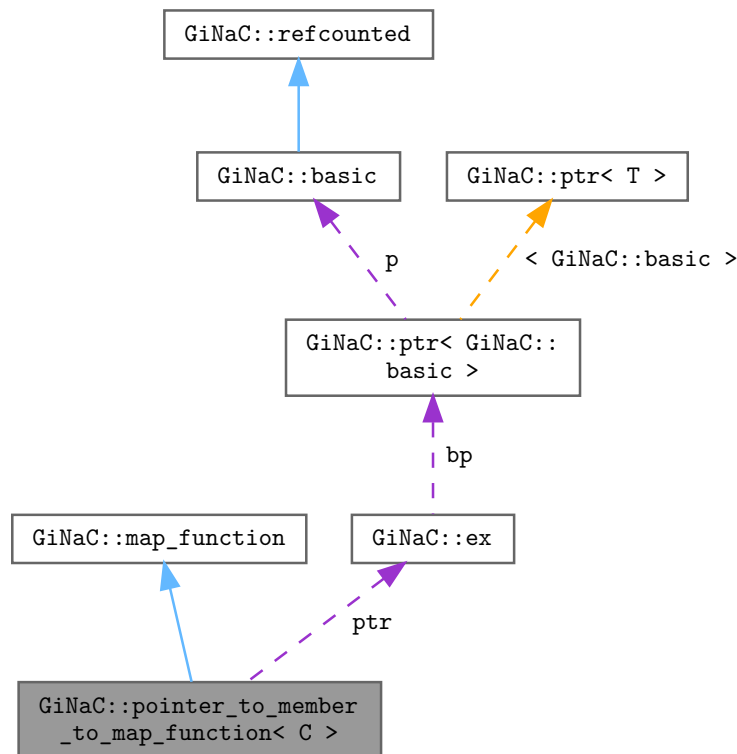
## 6.113 GiNaC::pointer\_to\_member\_to\_map\_function< C > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_member\_to\_map\_function< C >:



Collaboration diagram for `GiNaC::pointer_to_member_to_map_function< C >`:



## Public Member Functions

- `pointer_to_member_to_map_function` (`ex(C::*member)(const ex &), C &obj`)
- `ex operator()` (`const ex &e`) override

## Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function` ()
- virtual `ex operator()` (`const ex &e`)=0

## Protected Attributes

- `ex(C::* ptr)` (`const ex &`)
- `C &c`

## Additional Inherited Members

### Public Types inherited from `GiNaC::map_function`

- typedef const `ex & argument_type`
- typedef `ex result_type`

## 6.113.1 Constructor & Destructor Documentation

### 6.113.1.1 pointer\_to\_member\_to\_map\_function()

```
template<class C >
GiNaC::pointer_to_member_to_map_function< C >::pointer_to_member_to_map_function (
    ex(C::*)(const ex &) member,
    C & obj ) [inline], [explicit]
```

## 6.113.2 Member Function Documentation

### 6.113.2.1 operator>()()

```
template<class C >
ex GiNaC::pointer_to_member_to_map_function< C >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >::c](#).

## 6.113.3 Member Data Documentation

### 6.113.3.1 ptr

```
template<class C >
ex(C::* GiNaC::pointer_to_member_to_map_function< C >::ptr) (const ex &) [protected]
```

### 6.113.3.2 c

```
template<class C >
C& GiNaC::pointer_to_member_to_map_function< C >::c [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >::operator>\(\)\(\)](#).

The documentation for this class was generated from the following file:

- [ex.h](#)



## Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg](#) ([ex](#)(C::\*[member](#))(const [ex](#) &, T1), C &[obj](#), T1 [a1](#))
- [ex operator\(\)](#) (const [ex](#) &[e](#)) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()
- virtual [ex operator\(\)](#) (const [ex](#) &[e](#))=0

## Protected Attributes

- [ex](#)(C::\* [ptr](#) )(const [ex](#) &, T1)
- C & [c](#)
- T1 [arg1](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.114.1 Constructor & Destructor Documentation

### 6.114.1.1 [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg\(\)](#)

```
template<class C , class T1 >
GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::pointer_to_member_to_map_function_1arg
(
    ex(C::*)(const ex &, T1) member,
    C & obj,
    T1 a1 ) [inline], [explicit]
```

## 6.114.2 Member Function Documentation

### 6.114.2.1 [operator\(\)](#)()

```
template<class C , class T1 >
ex GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >::arg1](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg](#).

### 6.114.3 Member Data Documentation

#### 6.114.3.1 ptr

```
template<class C , class T1 >
ex(C::* GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::ptr) (const ex &, T1) [protected]
```

#### 6.114.3.2 c

```
template<class C , class T1 >
C& GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::c [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >::operator\(\)\(\)](#).

#### 6.114.3.3 arg1

```
template<class C , class T1 >
T1 GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >::operator\(\)\(\)](#).

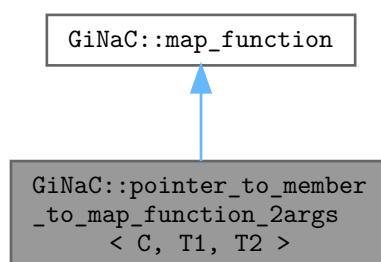
The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.115 GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 > Class Template Reference

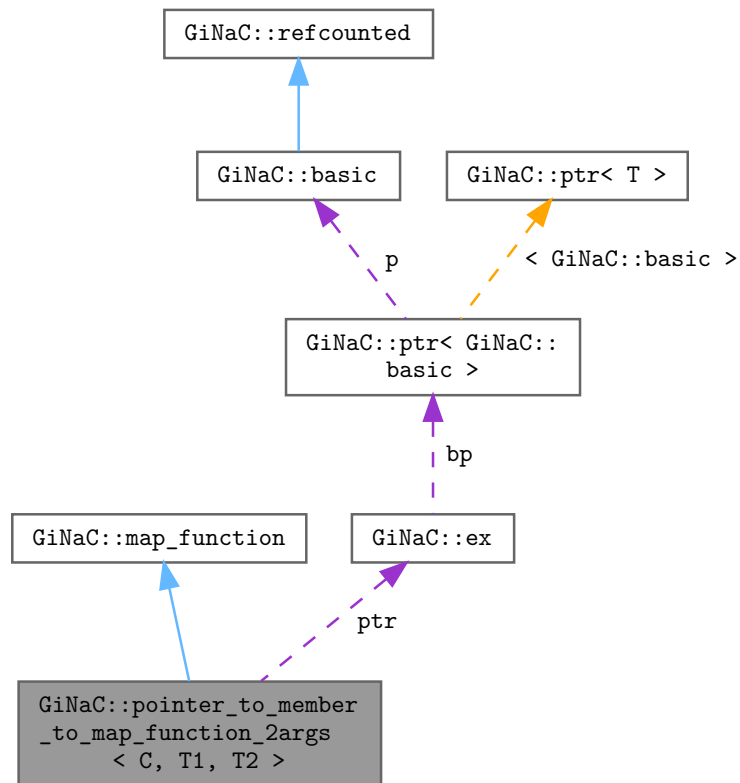
```
#include <ex.h>
```

Inheritance diagram for [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#):





Collaboration diagram for GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >:



## Public Member Functions

- `pointer_to_member_to_map_function_2args` (`ex(C::*member)(const ex &, T1, T2), C &obj, T1 a1, T2 a2`)
- `ex operator()` (`const ex &e`) override

## Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function` ()
- virtual `ex operator()` (`const ex &e`)=0

## Protected Attributes

- `ex(C::* ptr)` (`const ex &, T1, T2`)
- `C & c`
- `T1 arg1`
- `T2 arg2`

## Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.115.1 Constructor & Destructor Documentation

### 6.115.1.1 [pointer\\_to\\_member\\_to\\_map\\_function\\_2args\(\)](#)

```
template<class C , class T1 , class T2 >
GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::pointer_to_member_to_map_function_2args (
    ex(C::*)(const ex &, T1, T2) member,
    C & obj,
    T1 a1,
    T2 a2 ) [inline], [explicit]
```

## 6.115.2 Member Function Documentation

### 6.115.2.1 [operator>\(\)\(\)](#)

```
template<class C , class T1 , class T2 >
ex GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::arg1](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::c](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::c](#).

## 6.115.3 Member Data Documentation

### 6.115.3.1 [ptr](#)

```
template<class C , class T1 , class T2 >
ex(C::* GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >::ptr) (const ex &, T1, T2)
[protected]
```

## 6.115.3.2 c

```
template<class C , class T1 , class T2 >
C& GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::c [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::operator\(\)\(\)](#).

## 6.115.3.3 arg1

```
template<class C , class T1 , class T2 >
T1 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::operator\(\)\(\)](#).

## 6.115.3.4 arg2

```
template<class C , class T1 , class T2 >
T2 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::arg2 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::operator\(\)\(\)](#).

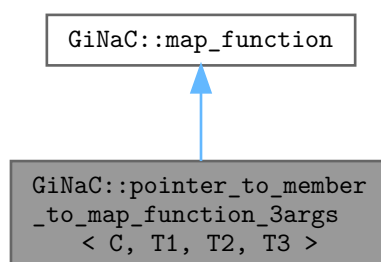
The documentation for this class was generated from the following file:

- [ex.h](#)

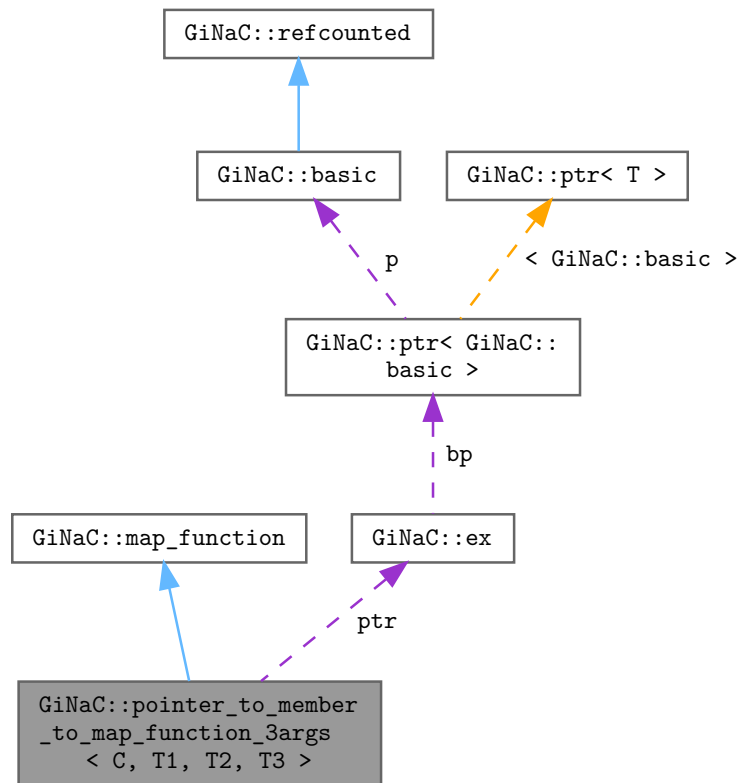
## 6.116 GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >`:



Collaboration diagram for `GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >`:



## Public Member Functions

- `pointer_to_member_to_map_function_3args` (`ex(C::*member)(const ex &, T1, T2, T3), C &obj, T1 a1, T2 a2, T3 a3`)
- `ex operator()` (`const ex &e`) override

## Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function()`
- virtual `ex operator()` (`const ex &e`)=0

## Protected Attributes

- `ex(C::* ptr)` (`const ex &, T1, T2, T3`)
- `C & c`
- `T1 arg1`
- `T2 arg2`
- `T3 arg3`

## Additional Inherited Members

Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.116.1 Constructor & Destructor Documentation

### 6.116.1.1 pointer\_to\_member\_to\_map\_function\_3args()

```
template<class C , class T1 , class T2 , class T3 >
GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::pointer_to_member_to_map_↵
function_3args (
    ex(C::*)(const ex &, T1, T2, T3) member,
    C & obj,
    T1 a1,
    T2 a2,
    T3 a3 ) [inline], [explicit]
```

## 6.116.2 Member Function Documentation

### 6.116.2.1 operator>()()

```
template<class C , class T1 , class T2 , class T3 >
ex GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::operator() (
    const ex & e ) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg1](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg2](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg3](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg4](#).

## 6.116.3 Member Data Documentation

### 6.116.3.1 ptr

```
template<class C , class T1 , class T2 , class T3 >
ex(C::* GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >::ptr) (const ex &, T1,
T2, T3) [protected]
```

### 6.116.3.2 c

```
template<class C , class T1 , class T2 , class T3 >  
C& GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::c [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::operator\(\)\(\)](#).

### 6.116.3.3 arg1

```
template<class C , class T1 , class T2 , class T3 >  
T1 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::operator\(\)\(\)](#).

### 6.116.3.4 arg2

```
template<class C , class T1 , class T2 , class T3 >  
T2 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg2 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::operator\(\)\(\)](#).

### 6.116.3.5 arg3

```
template<class C , class T1 , class T2 , class T3 >  
T3 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg3 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::operator\(\)\(\)](#).

The documentation for this class was generated from the following file:

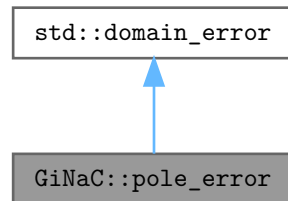
- [ex.h](#)

## 6.117 GiNaC::pole\_error Class Reference

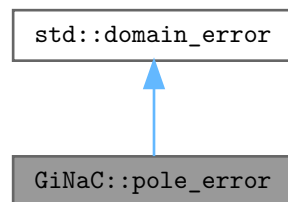
Exception class thrown when a singularity is encountered.

```
#include <numeric.h>
```

Inheritance diagram for GiNaC::pole\_error:



Collaboration diagram for GiNaC::pole\_error:



### Public Member Functions

- [pole\\_error](#) (const std::string &what\_arg, int [degree](#))  
*ctor for [pole\\_error](#) exception class.*
- int [degree](#) () const  
*Return the degree of the [pole\\_error](#) exception class.*

### Private Attributes

- int [deg](#)

#### 6.117.1 Detailed Description

Exception class thrown when a singularity is encountered.

## 6.117.2 Constructor & Destructor Documentation

### 6.117.2.1 `pole_error()`

```
GiNaC::pole_error::pole_error (
    const std::string & what_arg,
    int degree ) [explicit]
```

ctor for [pole\\_error](#) exception class.

## 6.117.3 Member Function Documentation

### 6.117.3.1 `degree()`

```
int GiNaC::pole_error::degree ( ) const
```

Return the degree of the [pole\\_error](#) exception class.

References [deg](#).

## 6.117.4 Member Data Documentation

### 6.117.4.1 `deg`

```
int GiNaC::pole_error::deg [private]
```

Referenced by [degree\(\)](#).

The documentation for this class was generated from the following files:

- [numeric.h](#)
- [utils.cpp](#)

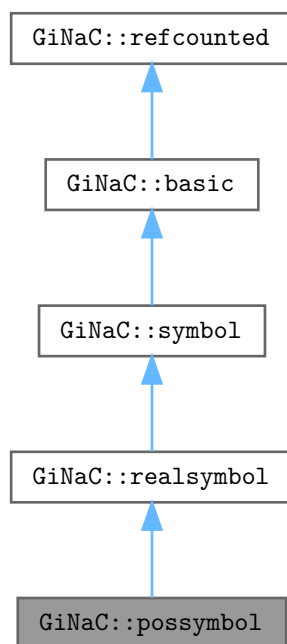


## 6.118 GiNaC::possymbol Class Reference

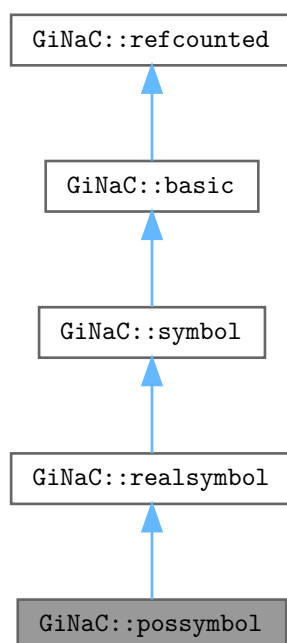
Specialization of symbol to real positive domain.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::possymbol:



Collaboration diagram for `GiNaC::possymbol`:



## Public Member Functions

- [possymbol](#) ()
- [possymbol](#) (const std::string &initname)
- [possymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [possymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

## Public Member Functions inherited from [GiNaC::realsymbol](#)

- [realsymbol](#) ()
- [realsymbol](#) (const std::string &initname)
- [realsymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [realsymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

Public Member Functions inherited from [GiNaC::symbol](#)

- [symbol](#) (const std::string &initname)
- [symbol](#) (const std::string &initname, const std::string &texname)
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex series](#) (const [relational](#) &s, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series\(\)](#) for symbols.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override  
*Implementation of [ex::normal\(\)](#) for symbols.*
- [ex to\\_rational](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_rational\(\)](#) for symbols.*
- [ex to\\_polynomial](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_polynomial\(\)](#) for symbols.*
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_polynomial](#) (const [ex](#) &var) const override  
*Check whether this is a polynomial in the given variables.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- virtual unsigned [get\\_domain](#) () const
- void [set\\_name](#) (const std::string &n)
- void [set\\_TeX\\_name](#) (const std::string &n)
- std::string [get\\_name](#) () const
- std::string [get\\_TeX\\_name](#) () const

Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const

- Evaluate integrals, if result is known.*

  - virtual `ex eval_indexed` (const `basic` &i) const

*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const

*Output to stream.*
- virtual void `dbgprint` () const

*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprintree` () const

*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const

*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const

*Information about the object.*
- virtual size\_t `nops` () const

*Number of operands/members.*
- virtual `ex op` (size\_t i) const

*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)

*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const

*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const

*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const

*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const

*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const

- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Additional Inherited Members

### Protected Member Functions inherited from [GiNaC::symbol](#)

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for single differentiation of a symbol.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex\\_derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes inherited from [GiNaC::symbol](#)

- unsigned [serial](#)  
*unique serial number for comparison*
- std::string [name](#)  
*printname of this symbol*
- std::string [TeX\\_name](#)  
*LaTeX name of this symbol.*

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
*hash value*

**6.118.1 Detailed Description**

Specialization of symbol to real positive domain.

**6.118.2 Constructor & Destructor Documentation****6.118.2.1 possymbol() [1/3]**

```
GiNaC::possymbol::possymbol ( )
```

Referenced by [duplicate\(\)](#).

**6.118.2.2 possymbol() [2/3]**

```
GiNaC::possymbol::possymbol (
    const std::string & initname ) [explicit]
```

**6.118.2.3 possymbol() [3/3]**

```
GiNaC::possymbol::possymbol (
    const std::string & initname,
    const std::string & texname )
```

**6.118.3 Member Function Documentation****6.118.3.1 get\_domain()**

```
unsigned GiNaC::possymbol::get_domain ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::realsymbol](#).

References [GiNaC::domain::positive](#).

### 6.118.3.2 duplicate()

```
possymbol * GiNaC::possymbol::duplicate ( ) const [inline], [override], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented from [GiNaC::realsymbol](#).

References [GiNaC::status\\_flags::dynallocated](#), [possymbol\(\)](#), and [GiNaC::basic::setflag\(\)](#).

The documentation for this class was generated from the following files:

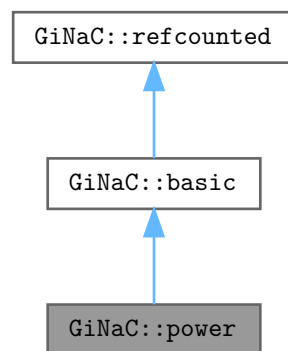
- [symbol.h](#)
- [symbol.cpp](#)

## 6.119 GiNaC::power Class Reference

This class holds a two-component object, a basis and an exponent representing exponentiation.

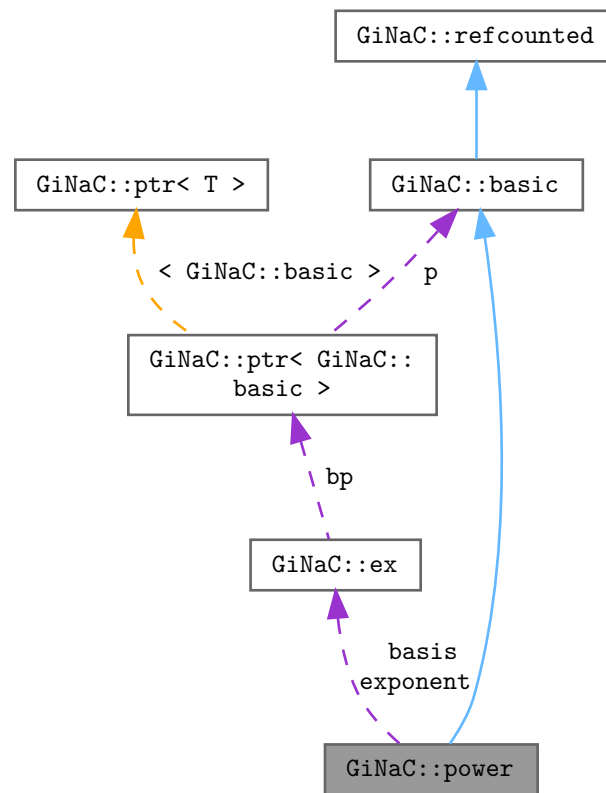
```
#include <power.h>
```

Inheritance diagram for GiNaC::power:





Collaboration diagram for GiNaC::power:



## Public Member Functions

- `power` (const `ex` &lh, const `ex` &rh)
- `template<typename T > power` (const `ex` &lh, const T &rh)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex map` (`map_function` &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*

- `int ldegree (const ex &s) const` override  
*Return degree of lowest power in object s.*
- `ex coeff (const ex &s, int n=1) const` override  
*Return coefficient of degree n in object s.*
- `ex eval ()` const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalf ()` const override  
*Evaluate object numerically.*
- `ex evalm ()` const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex series (const relational &s, int order, unsigned options=0) const` override  
*Implementation of `ex::series()` for powers.*
- `ex subs (const exmap &m, unsigned options=0) const` override  
*Substitute a set of objects by arbitrary expressions.*
- `bool has (const ex &other, unsigned options=0) const` override  
*Test for occurrence of a pattern.*
- `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const` override  
*Implementation of `ex::normal()` for powers.*
- `ex to_rational (exmap &repl) const` override  
*Implementation of `ex::to_rational()` for powers.*
- `ex to_polynomial (exmap &repl) const` override  
*Implementation of `ex::to_polynomial()` for powers.*
- `ex conjugate ()` const override
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `void archive (archive_node &n) const` override  
*Save (a.k.a.*
- `void read_archive (const archive_node &n, lst &syms) override`  
*Read (a.k.a.*

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate () const`  
*Create a clone of this object on the heap.*
- `virtual ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf () const`  
*Evaluate object numerically.*
- `virtual ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- `virtual ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print (const print_context &c, unsigned level=0) const`

- *Output to stream.*  
virtual void [dbgprint](#) () const
- *Little wrapper around print to be called within a debugger.*  
virtual void [dbgprinttree](#) () const
- *Little wrapper around printtree to be called within a debugger.*  
virtual unsigned [precedence](#) () const
- *Return relative operator precedence (for parenthezing output).*  
virtual bool [info](#) (unsigned inf) const
- *Information about the object.*  
virtual size\_t [nops](#) () const
- *Number of operands/members.*  
virtual [ex op](#) (size\_t i) const
- *Return operand/member at position i.*  
virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex & let\\_op](#) (size\_t i)
- *Return modifiable operand/member at position i.*  
virtual [ex & operator\[\]](#) (const [ex](#) &index)
- virtual [ex & operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const
- *Test for occurrence of a pattern.*  
virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const
- *Check whether the expression matches a given pattern.*  
virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const
- *Substitute a set of objects by arbitrary expressions.*  
virtual [ex map](#) ([map\\_function](#) &f) const
- *Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*  
virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const
- *Check whether this is a polynomial in the given variables.*  
virtual int [degree](#) (const [ex](#) &s) const
- *Return degree of highest power in object s.*  
virtual int [ldegree](#) (const [ex](#) &s) const
- *Return degree of lowest power in object s.*  
virtual [ex coeff](#) (const [ex](#) &s, int n=1) const
- *Return coefficient of degree n in object s.*  
virtual [ex expand](#) (unsigned [options](#)=0) const
- *Expand expression, i.e.*  
virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const
- *Sort expanded expression in terms of powers of some object(s).*  
virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const
- *Default implementation of [ex::series\(\)](#).*  
virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const
- *Default implementation of [ex::normal\(\)](#).*  
virtual [ex to\\_rational](#) ([exmap](#) &repl) const
- *Default implementation of [ex::to\\_rational\(\)](#).*  
virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const
- *Apply symmetric modular homomorphism to an expanded multivariate polynomial.*  
virtual [numeric max\\_coefficient](#) () const

- Implementation `ex::max_coefficient()`.
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type ()` const
- virtual `return_type_t return_type_info ()` const
- virtual `ex conjugate ()` const
- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T >  
 void `print_dispatch (const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive (archive_node &n)` const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive (const archive_node &n, lst &syms)`  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level (const exmap &m, unsigned options)` const  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1)` const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare (const basic &other)` const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal (const basic &other)` const  
*Test for syntactic equality.*
- const `basic & hold ()` const  
*Stop further evaluation.*
- unsigned `gethash ()` const
- const `basic & setflag (unsigned f)` const  
*Set some `status_flags`.*
- const `basic & clearflag (unsigned f)` const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted ()` noexcept
- unsigned int `add_reference ()` noexcept
- unsigned int `remove_reference ()` noexcept
- unsigned int `get_refcount ()` const noexcept
- void `set_refcount (unsigned int r)` noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a power.*
- `ex eval_ncmul` (const `exvector` &v) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- void `print_power` (const `print_context` &c, const char \*powersymbol, const char \*openbrace, const char \*closebrace, unsigned level) const
- void `do_print_dflt` (const `print_dflt` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_csrc` (const `print_csrc` &c, unsigned level) const
- void `do_print_python` (const `print_python` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- void `do_print_csrc_cl_N` (const `print_csrc_cl_N` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Static Protected Member Functions

- static `ex expand_add` (const `add` &a, long n, unsigned `options`)  
*expand  $a^n$  where  $a$  is an add and  $n$  is a positive integer.*
- static `ex expand_add_2` (const `add` &a, unsigned `options`)  
*Special case of `power::expand_add`.*
- static `ex expand_mul` (const `mul` &m, const `numeric` &n, unsigned `options`, bool from\_expand=false)  
*Expand factors of  $m$  in  $m^n$  where  $m$  is a mul and  $n$  is an integer.*

## Protected Attributes

- [ex basis](#)
- [ex exponent](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## Friends

- class [mul](#)

## 6.119.1 Detailed Description

This class holds a two-component object, a basis and an exponent representing exponentiation.

## 6.119.2 Constructor & Destructor Documentation

### 6.119.2.1 [power\(\)](#) [1/2]

```
GiNaC::power::power (
    const ex & lh,
    const ex & rh ) [inline]
```

### 6.119.2.2 [power\(\)](#) [2/2]

```
template<typename T >
GiNaC::power::power (
    const ex & lh,
    const T & rh ) [inline]
```

## 6.119.3 Member Function Documentation

### 6.119.3.1 precedence()

```
unsigned GiNaC::power::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [print\\_power\(\)](#).

### 6.119.3.2 info()

```
bool GiNaC::power::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::info\\_flags::expanded](#), [exponent](#), [GiNaC::basic::flags](#), [GiNaC::status\\_flags::has\\_indices](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::status\\_flags::has\\_no\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::info\\_flags::real](#), and [GiNaC::basic::setflag\(\)](#).

### 6.119.3.3 nops()

```
size_t GiNaC::power::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.119.3.4 op()

```
ex GiNaC::power::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position *i*.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), and [GINAC\\_ASSERT](#).

### 6.119.3.5 map()

```
ex GiNaC::power::map (
    map_function & f ) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), and [exponent](#).

### 6.119.3.6 is\_polynomial()

```
bool GiNaC::power::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), [GiNaC::ex::has\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_polynomial\(\)](#), and [GiNaC::info\\_flags::nonnegint](#).

### 6.119.3.7 degree()

```
int GiNaC::power::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::ex::degree\(\)](#), [exponent](#), [GiNaC::ex::has\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::is\\_integer\(\)](#).

### 6.119.3.8 ldegree()

```
int GiNaC::power::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), [GiNaC::ex::has\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), and [GiNaC::ex::ldegree\(\)](#).



6.119.3.9 `coeff()`

```
ex GiNaC::power::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [basis](#), [exponent](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), and [n](#).

Referenced by [expand\(\)](#), and [expand\\_add\(\)](#).

6.119.3.10 `eval()`

```
ex GiNaC::power::eval ( ) const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following x, x1, x2,... stand for a symbolic variables of type ex and c, c1, c2... stand for such expressions that contain a plain number.

- $x^0 \rightarrow 1$  (also handles  $0^0$ )
- $x^1 \rightarrow x$
- $0^c \rightarrow 0$  or exception (depending on the real part of c)
- $1^x \rightarrow 1$
- $c_1^{c_2} \rightarrow c_1^n c_1^{c_2-n}$  (so that  $0 < (c_2-n) < 1$ , try to evaluate roots, possibly in numerator and denominator of c1)
- $x^{(x,c_1),c_2} \rightarrow x^{x,c_1 c_2}$  if x is positive and c1 is real.
- $x^{(x,c_1),c_2} \rightarrow x^{x,c_1 c_2}$  (c2 integer or  $-1 < c_1 \leq 1$  or  $(c_1=-1 \text{ and } c_2 > 0)$ , case  $c_1=1$  should not happen, see below!)
- $x^{(x,y,z),c} \rightarrow x^x y^c z^c$  (if c integer)
- $x^{(x,c_1),c_2} \rightarrow x^{x,c_2} c_1^{c_2}$  ( $c_1 > 0$ )
- $x^{(x,c_1),c_2} \rightarrow x^{-x,c_2} c_1^{c_2}$  ( $c_1 < 0$ )

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex1](#), [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::abs\(\)](#), [basis](#), [c](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::return\\_types::commutative](#), [GiNaC::numeric::compare\(\)](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::status\\_flags::evaluated](#), [expand\\_mul\(\)](#), [exponent](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::integer\\_content\(\)](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::numeric::is\\_crational\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::is\\_negative\(\)](#), [GiNaC::numeric::is\\_pos\\_integ](#), [GiNaC::numeric::is\\_positive\(\)](#), [GiNaC::numeric::is\\_rational\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [likely](#), [m](#), [GiNaC::numeric::mul\(\)](#), [n](#), [GiNaC::numeric::numer\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::numeric::power\(\)](#), [r](#), [GiNaC::info\\_flags::real](#), [GiNaC::numeric::real\(\)](#), [GiNaC::ex::return\\_type\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [do\\_print\\_latex\(\)](#).

**6.119.3.11 evalf()**

```
ex GiNaC::power::evalf ( ) const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::ex::evalf\(\)](#), and [exponent](#).

**6.119.3.12 evalm()**

```
ex GiNaC::power::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::ex::evalm\(\)](#), [exponent](#), and [GiNaC::pow\(\)](#).

**6.119.3.13 series()**

```
ex GiNaC::power::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for powers.

This performs Laurent expansion of reciprocals of series at singularities.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [basis](#), [GiNaC::exp\(\)](#), [GiNaC::ex::expand\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::lddegree\(\)](#), [GiNaC::log\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [r](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::ex::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::to\\_int\(\)](#).

Referenced by [GiNaC::psi1\\_series\(\)](#).

**6.119.3.14 subs()**

```
ex GiNaC::power::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::subs\\_options::algebraic](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [exponent](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [GiNaC::pow\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

**6.119.3.15 has()**

```
bool GiNaC::power::has (
    const ex & pattern,
    unsigned options = 0 ) const [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::has\\_options::algebraic](#), [basis](#), [exponent](#), [GiNaC::basic::has\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::match\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::ex::op\(\)](#), [options](#), and [GiNaC::info\\_flags::posint](#).

**6.119.3.16 normal()**

```
ex GiNaC::power::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for powers.

It normalizes the basis, distributes integer exponents to numerator and denominator, and replaces non-integer powers by temporary symbols.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [basis](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::subs\(\)](#).

**6.119.3.17 to\_rational()**

```
ex GiNaC::power::to_rational (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for powers.

It replaces non-integer powers by temporary symbols.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::to\\_rational\(\)](#).

**6.119.3.18 to\_polynomial()**

```
ex GiNaC::power::to_polynomial (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for powers.

It replaces non-posint powers by temporary symbols.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex\\_1](#), [basis](#), [GiNaC::collect\\_common\\_factors\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::posint](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::to\\_polynomial\(\)](#).

**6.119.3.19 conjugate()**

```
ex GiNaC::power::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [GiNaC::ex::conjugate\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), and [GiNaC::info\\_flags::positive](#).

**6.119.3.20 real\_part()**

```
ex GiNaC::power::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::abs\(\)](#), [basis](#), [GiNaC::binomial\(\)](#), [c](#), [GiNaC::cos\(\)](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::log\(\)](#), [n](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::numer\(\)](#), [GiNaC::pow\(\)](#), and [GiNaC::ex::real\\_part\(\)](#).

**6.119.3.21 imag\_part()**

```
ex GiNaC::power::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::abs\(\)](#), [basis](#), [GiNaC::binomial\(\)](#), [c](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::log\(\)](#), [n](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::numer\(\)](#), [GiNaC::pow\(\)](#), [GiNaC::ex::real\\_part\(\)](#), and [GiNaC::sin\(\)](#).

**6.119.3.22 archive()**

```
void GiNaC::power::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), and [n](#).

**6.119.3.23 read\_archive()**

```
void GiNaC::power::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), and [n](#).

**6.119.3.24 derivative()**

```
ex GiNaC::power::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a power.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [basis](#), [GiNaC::ex::diff\(\)](#), [exponent](#), [GiNaC::log\(\)](#), and [GiNaC::pow\(\)](#).

**6.119.3.25 eval\_ncmul()**

```
ex GiNaC::power::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.119.3.26 return\_type()**

```
unsigned GiNaC::power::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [basis](#), and [GiNaC::ex::return\\_type\(\)](#).

**6.119.3.27 return\_type\_tinfo()**

```
return\_type\_t GiNaC::power::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [basis](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

**6.119.3.28 expand()**

```
ex GiNaC::power::expand (
    unsigned options = 0 ) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [expand\\_add\(\)](#), [expand\\_mul\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_integer\(\)](#), [m](#), [GiNaC::info\\_flags::negative](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::status\\_flags::purely\\_indefinite](#), [r](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and [GiNaC::numeric::to\\_long\(\)](#).

Referenced by [expand\(\)](#), [expand\\_add\(\)](#), and [expand\\_add\\_2\(\)](#).

### 6.119.3.29 print\_power()

```
void GiNaC::power::print_power (
    const print\_context & c,
    const char * powersymbol,
    const char * openbrace,
    const char * closebrace,
    unsigned level ) const [protected]
```

References [basis](#), [c](#), [exponent](#), [precedence\(\)](#), and [GiNaC::ex::print\(\)](#).

Referenced by [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\(\)](#).

### 6.119.3.30 do\_print\_dflt()

```
void GiNaC::power::do_print_dflt (
    const print\_dflt & c,
    unsigned level ) const [protected]
```

References [GiNaC::\\_ex1\\_2](#), [basis](#), [c](#), [exponent](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::print\(\)](#), and [print\\_power\(\)](#).

### 6.119.3.31 do\_print\_latex()

```
void GiNaC::power::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [GiNaC::\\_ex1\\_2](#), [basis](#), [c](#), [eval\(\)](#), [exponent](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_negative\(\)](#), [GiNaC::ex::print\(\)](#), and [print\\_power\(\)](#).

### 6.119.3.32 do\_print\_csrc()

```
void GiNaC::power::do_print_csrc (
    const print\_csrc & c,
    unsigned level ) const [protected]
```

References [GiNaC::\\_ex1\\_1](#), [basis](#), [c](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

**6.119.3.33 do\_print\_python()**

```
void GiNaC::power::do_print_python (
    const print\_python & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_power\(\)](#).

**6.119.3.34 do\_print\_python\_repr()**

```
void GiNaC::power::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [basis](#), [c](#), [exponent](#), and [GiNaC::ex::print\(\)](#).

**6.119.3.35 do\_print\_csrc\_cl\_N()**

```
void GiNaC::power::do_print_csrc_cl_N (
    const print\_csrc\_cl\_N & c,
    unsigned level ) const [protected]
```

References [GiNaC::\\_ex\\_1](#), [basis](#), [c](#), [exponent](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::print\(\)](#).

**6.119.3.36 expand\_add()**

```
ex GiNaC::power::expand_add (
    const add & a,
    long n,
    unsigned options ) [static], [protected]
```

expand  $a^n$  where  $a$  is an [add](#) and  $n$  is a positive integer.

See also

[power::expand](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [basis](#), [GiNaC::binomial\(\)](#), [c](#), [coeff\(\)](#), [expand\(\)](#), [expand\\_add\\_2\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GiNaC::factor\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [k](#), [mul](#), [GiNaC::multinomial\\_coefficient\(\)](#), [n](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#), [GiNaC::composition\\_generator::next\(\)](#), [GiNaC::expairseq::nops\(\)](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::pow\(\)](#), [r](#), [GiNaC::expairseq::seq](#), and [GiNaC::numeric::to\\_long\(\)](#).

Referenced by [expand\(\)](#).



### 6.119.3.37 expand\_add\_2()

```
ex GiNaC::power::expand_add_2 (
    const add & a,
    unsigned options ) [static], [protected]
```

Special case of [power::expand\\_add](#).

Expands  $a^2$  where  $a$  is an add.

See also

[power::expand\\_add](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex2](#), [GiNaC::\\_num2\\_p](#), [basis](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair](#), [expand\(\)](#), [expand\\_mul\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [last](#), [GiNaC::numeric::mul\(\)](#), [mul](#), [GiNaC::numeric::mul\\_dyn\(\)](#), [GiNaC::expairseq::nops\(\)](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [r](#), and [GiNaC::expairseq::seq](#).

Referenced by [expand\\_add\(\)](#).

### 6.119.3.38 expand\_mul()

```
ex GiNaC::power::expand_mul (
    const mul & m,
    const numeric & n,
    unsigned options,
    bool from_expand = false ) [static], [protected]
```

Expand factors of  $m$  in  $m^n$  where  $m$  is a mul and  $n$  is an integer.

See also

[power::expand](#)

References [GiNaC::\\_ex1](#), [GiNaC::expair::coeff](#), [GiNaC::basic::ex](#), [GiNaC::expand\\_options::expand\\_rename\\_idx](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::get\\_all\\_dummy\\_indices\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::has\\_indices](#), [m](#), [n](#), [options](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [eval\(\)](#), [expand\(\)](#), and [expand\\_add\\_2\(\)](#).

## 6.119.4 Friends And Related Function Documentation

### 6.119.4.1 mul

```
friend class mul [friend]
```

Referenced by [expand\\_add\(\)](#), and [expand\\_add\\_2\(\)](#).

## 6.119.5 Member Data Documentation

### 6.119.5.1 basis

`ex GiNaC::power::basis` [protected]

Referenced by [archive\(\)](#), [coeff\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [eval\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [expand\\_add\(\)](#), [expand\\_add\\_2\(\)](#), [has\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [map\(\)](#), [normal\(\)](#), [op\(\)](#), [print\\_power\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [series\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 6.119.5.2 exponent

`ex GiNaC::power::exponent` [protected]

Referenced by [archive\(\)](#), [coeff\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [eval\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [expand\\_add\(\)](#), [expand\\_add\\_2\(\)](#), [has\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [map\(\)](#), [normal\(\)](#), [op\(\)](#), [print\\_power\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

The documentation for this class was generated from the following files:

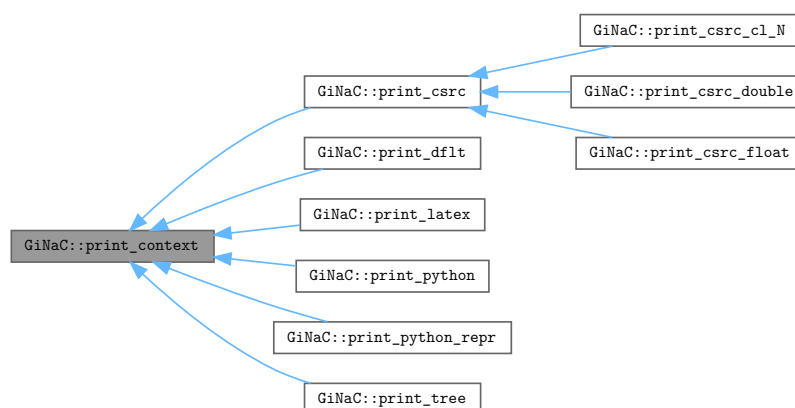
- [power.h](#)
- [normal.cpp](#)
- [power.cpp](#)
- [pseries.cpp](#)

## 6.120 GiNaC::print\_context Class Reference

Base class for print\_contexts.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_context:



## Public Member Functions

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Public Attributes

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.120.1 Detailed Description

Base class for print\_contexts.

### 6.120.2 Constructor & Destructor Documentation

#### 6.120.2.1 print\_context()

```
GiNaC::print_context::print_context (
    std::ostream & os,
    unsigned options = 0 )
```

#### 6.120.2.2 ~print\_context()

```
virtual GiNaC::print_context::~~print_context ( ) [inline], [virtual]
```

### 6.120.3 Member Data Documentation

#### 6.120.3.1 s

```
std::ostream& GiNaC::print_context::s
```

stream to output to

Referenced by [GiNaC::numeric::print\\_numeric\(\)](#).

### 6.120.3.2 options

`unsigned GiNaC::print_context::options`

option flags

Referenced by [GiNaC::get\\_print\\_options\(\)](#), [GiNaC::set\\_print\\_context\(\)](#), and [GiNaC::set\\_print\\_options\(\)](#).

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 6.121 GiNaC::print\_context\_options Class Reference

This class stores information about a registered [print\\_context](#) class.

```
#include <print.h>
```

### Public Member Functions

- [print\\_context\\_options](#) (const char \*n, const char \*p, unsigned i)
- const char \* [get\\_name](#) () const
- const char \* [get\\_parent\\_name](#) () const
- unsigned [get\\_id](#) () const

### Private Attributes

- const char \* [name](#)  
*Class name.*
- const char \* [parent\\_name](#)  
*Name of superclass.*
- unsigned [id](#)  
*ID number (assigned automatically).*

### 6.121.1 Detailed Description

This class stores information about a registered [print\\_context](#) class.

### 6.121.2 Constructor & Destructor Documentation

### 6.121.2.1 print\_context\_options()

```
GiNaC::print_context_options::print_context_options (
    const char * n,
    const char * p,
    unsigned i ) [inline]
```

## 6.121.3 Member Function Documentation

### 6.121.3.1 get\_name()

```
const char * GiNaC::print_context_options::get_name ( ) const [inline]
```

References [name](#).

### 6.121.3.2 get\_parent\_name()

```
const char * GiNaC::print_context_options::get_parent_name ( ) const [inline]
```

References [parent\\_name](#).

### 6.121.3.3 get\_id()

```
unsigned GiNaC::print_context_options::get_id ( ) const [inline]
```

References [id](#).

## 6.121.4 Member Data Documentation

### 6.121.4.1 name

```
const char* GiNaC::print_context_options::name [private]
```

Class name.

Referenced by [get\\_name\(\)](#).

#### 6.121.4.2 parent\_name

```
const char* GiNaC::print_context_options::parent_name [private]
```

Name of superclass.

Referenced by [get\\_parent\\_name\(\)](#).

#### 6.121.4.3 id

```
unsigned GiNaC::print_context_options::id [private]
```

ID number (assigned automatically).

Referenced by [get\\_id\(\)](#).

The documentation for this class was generated from the following file:

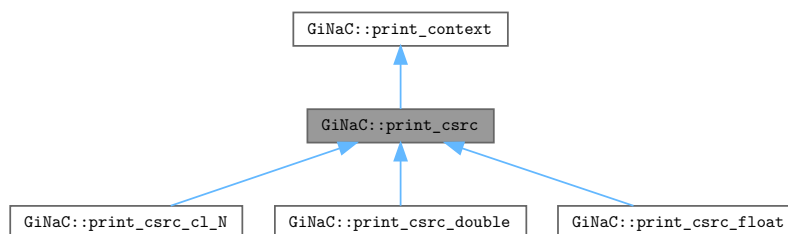
- [print.h](#)

## 6.122 GiNaC::print\_csrc Class Reference

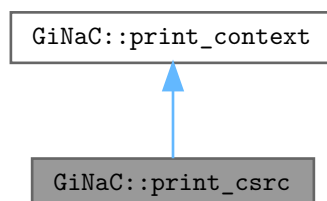
Base context for C source output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc:



Collaboration diagram for GiNaC::print\_csrc:



## Public Member Functions

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

## Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Additional Inherited Members

## Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.122.1 Detailed Description

Base context for C source output.

### 6.122.2 Constructor & Destructor Documentation

#### 6.122.2.1 [print\\_csrc\(\)](#)

```
GiNaC::print_csrc::print_csrc (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

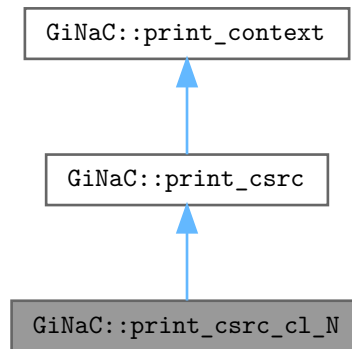
- [print.h](#)
- [print.cpp](#)

## 6.123 GiNaC::print\_csrc\_cl\_N Class Reference

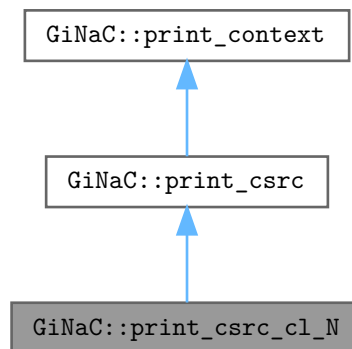
Context for C source output using CLN numbers.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_cl\_N:



Collaboration diagram for GiNaC::print\_csrc\_cl\_N:



### Public Member Functions

- [print\\_csrc\\_cl\\_N](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_csrc](#)

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)



**Public Member Functions inherited from [GiNaC::print\\_context](#)**

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

**Additional Inherited Members****Public Attributes inherited from [GiNaC::print\\_context](#)**

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

**6.123.1 Detailed Description**

Context for C source output using CLN numbers.

**6.123.2 Constructor & Destructor Documentation****6.123.2.1 [print\\_csrc\\_cl\\_N\(\)](#)**

```
GiNaC::print_csrc_cl_N::print_csrc_cl_N (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

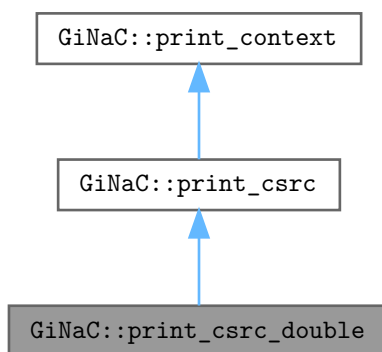
- [print.h](#)
- [print.cpp](#)

## 6.124 GiNaC::print\_csrc\_double Class Reference

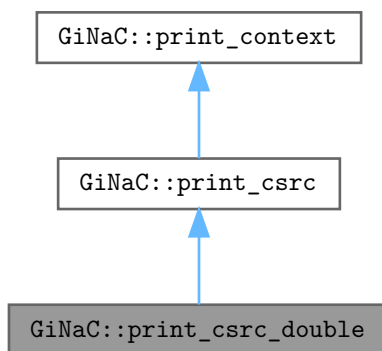
Context for C source output using double precision.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_double:



Collaboration diagram for GiNaC::print\_csrc\_double:



### Public Member Functions

- [print\\_csrc\\_double](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_csrc](#)

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

**Public Member Functions inherited from [GiNaC::print\\_context](#)**

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

**Additional Inherited Members****Public Attributes inherited from [GiNaC::print\\_context](#)**

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

**6.124.1 Detailed Description**

Context for C source output using double precision.

**6.124.2 Constructor & Destructor Documentation****6.124.2.1 [print\\_csrc\\_double\(\)](#)**

```
GiNaC::print_csrc_double::print_csrc_double (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

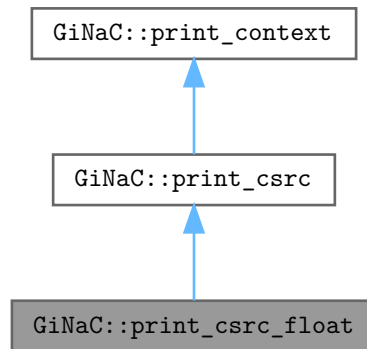
- [print.h](#)
- [print.cpp](#)

## 6.125 GiNaC::print\_csrc\_float Class Reference

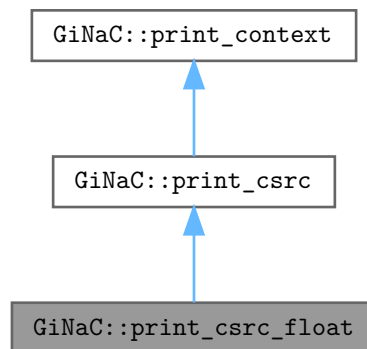
Context for C source output using float precision.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_float:



Collaboration diagram for GiNaC::print\_csrc\_float:



### Public Member Functions

- [print\\_csrc\\_float](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_csrc](#)

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

**Public Member Functions inherited from [GiNaC::print\\_context](#)**

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

**Additional Inherited Members****Public Attributes inherited from [GiNaC::print\\_context](#)**

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

**6.125.1 Detailed Description**

Context for C source output using float precision.

**6.125.2 Constructor & Destructor Documentation****6.125.2.1 [print\\_csrc\\_float\(\)](#)**

```
GiNaC::print_csrc_float::print_csrc_float (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

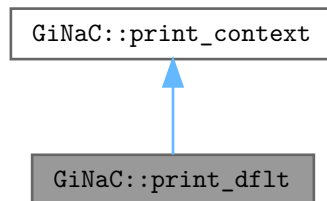
- [print.h](#)
- [print.cpp](#)

## 6.126 GiNaC::print\_dflt Class Reference

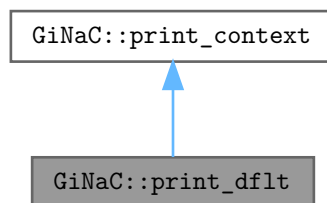
Context for default (ginsh-parsable) output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_dflt:



Collaboration diagram for GiNaC::print\_dflt:



### Public Member Functions

- [print\\_dflt](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Additional Inherited Members

#### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.126.1 Detailed Description

Context for default (ginsh-parsable) output.

### 6.126.2 Constructor & Destructor Documentation

#### 6.126.2.1 print\_dflt()

```
GiNaC::print_dflt::print_dflt (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 6.127 GiNaC::print\_funcor Class Reference

This class represents a print method for a certain algebraic class and [print\\_context](#) type.

```
#include <print.h>
```

### Public Member Functions

- [print\\_funcor](#) ()
- [print\\_funcor](#) (const [print\\_funcor](#) &other)
- [print\\_funcor](#) (std::unique\_ptr< [print\\_funcor\\_impl](#) > impl\_)
- template<class T, class C >  
[print\\_funcor](#) (void f(const T &, const C &, unsigned))
- template<class T, class C >  
[print\\_funcor](#) (void(T::\*f)(const C &, unsigned) const)
- [print\\_funcor](#) & [operator=](#) (const [print\\_funcor](#) &other)
- void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const
- bool [is\\_valid](#) () const

### Private Attributes

- std::unique\_ptr< [print\\_funcor\\_impl](#) > impl

### 6.127.1 Detailed Description

This class represents a print method for a certain algebraic class and [print\\_context](#) type.

Its main purpose is to hide the difference between member functions and nonmember functions behind one unified [operator\(\)](#) interface. [print\\_funcor](#) has value semantics and acts as a smart pointer (with deep copy) to a class derived from [print\\_funcor\\_impl](#) which implements the actual function call.

## 6.127.2 Constructor & Destructor Documentation

### 6.127.2.1 `print_functor()` [1/5]

```
GiNaC::print_functor::print_functor ( ) [inline]
```

### 6.127.2.2 `print_functor()` [2/5]

```
GiNaC::print_functor::print_functor (
    const print_functor & other ) [inline]
```

### 6.127.2.3 `print_functor()` [3/5]

```
GiNaC::print_functor::print_functor (
    std::unique_ptr< print_functor_impl > impl_ ) [inline]
```

### 6.127.2.4 `print_functor()` [4/5]

```
template<class T , class C >
GiNaC::print_functor::print_functor (
    void fconst T &, const C &, unsigned ) [inline]
```

### 6.127.2.5 `print_functor()` [5/5]

```
template<class T , class C >
GiNaC::print_functor::print_functor (
    void(T::*)(const C &, unsigned) const f ) [inline]
```

## 6.127.3 Member Function Documentation



### 6.127.3.1 operator=()

```
print_functor & GiNaC::print_functor::operator= (
    const print_functor & other ) [inline]
```

References [impl](#).

### 6.127.3.2 operator>()()

```
void GiNaC::print_functor::operator() (
    const basic & obj,
    const print_context & c,
    unsigned level ) const [inline]
```

References [c](#).

### 6.127.3.3 is\_valid()

```
bool GiNaC::print_functor::is_valid ( ) const [inline]
```

References [impl](#).

## 6.127.4 Member Data Documentation

### 6.127.4.1 impl

```
std::unique_ptr<print_functor_impl> GiNaC::print_functor::impl [private]
```

Referenced by [is\\_valid\(\)](#), and [operator=\(\)](#).

The documentation for this class was generated from the following file:

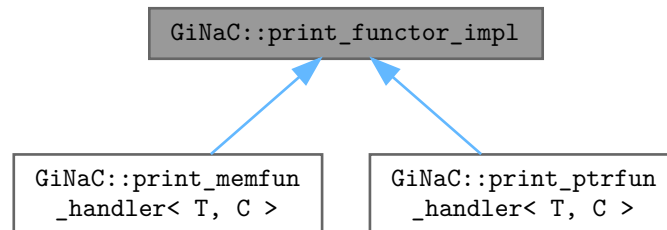
- [print.h](#)

## 6.128 GiNaC::print\_functor\_impl Class Reference

Base class for [print\\_functor](#) handlers.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_functor\_impl:



### Public Member Functions

- virtual [~print\\_functor\\_impl](#) ()
- virtual [print\\_functor\\_impl](#) \* [duplicate](#) () const =0
- virtual void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const =0

### 6.128.1 Detailed Description

Base class for [print\\_functor](#) handlers.

### 6.128.2 Constructor & Destructor Documentation

#### 6.128.2.1 ~print\_functor\_impl()

```
virtual GiNaC::print_functor_impl::~~print_functor_impl ( ) [inline], [virtual]
```

### 6.128.3 Member Function Documentation

**6.128.3.1 duplicate()**

```
virtual print\_functor\_impl * GiNaC::print_functor_impl::duplicate ( ) const [pure virtual]
```

Implemented in [GiNaC::print\\_ptrfun\\_handler< T, C >](#), and [GiNaC::print\\_memfun\\_handler< T, C >](#).

**6.128.3.2 operator()()**

```
virtual void GiNaC::print_functor_impl::operator() (
    const basic & obj,
    const print\_context & c,
    unsigned level ) const [pure virtual]
```

Implemented in [GiNaC::print\\_ptrfun\\_handler< T, C >](#), and [GiNaC::print\\_memfun\\_handler< T, C >](#).

The documentation for this class was generated from the following file:

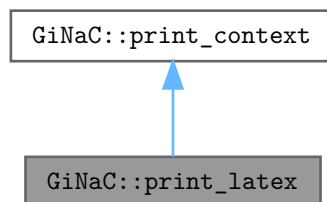
- [print.h](#)

**6.129 GiNaC::print\_latex Class Reference**

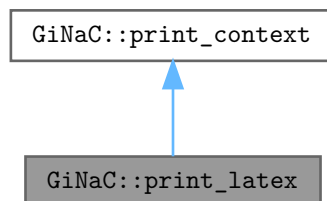
Context for latex-parsable output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_latex:



Collaboration diagram for GiNaC::print\_latex:



## Public Member Functions

- [print\\_latex](#) (std::ostream &, unsigned [options](#)=0)

## Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Additional Inherited Members

## Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.129.1 Detailed Description

Context for latex-parsable output.

### 6.129.2 Constructor & Destructor Documentation

#### 6.129.2.1 [print\\_latex\(\)](#)

```
GiNaC::print_latex::print_latex (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

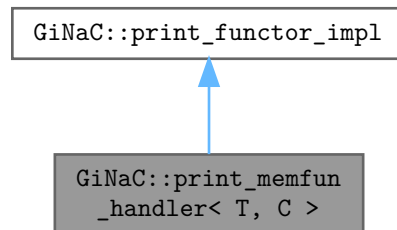
- [print.h](#)
- [print.cpp](#)

## 6.130 GiNaC::print\_memfun\_handler< T, C > Class Template Reference

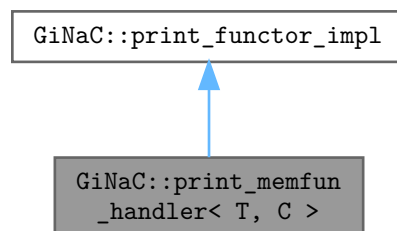
[print\\_functor](#) handler for member functions of class T, context type C

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_memfun\_handler< T, C >:



Collaboration diagram for GiNaC::print\_memfun\_handler< T, C >:



### Public Types

- typedef void(T::\* [F](#)) (const C &[c](#), unsigned level) const

### Public Member Functions

- [print\\_memfun\\_handler](#) ([F](#) f\_)
- [print\\_memfun\\_handler](#) \* [duplicate](#) () const override
- void [operator](#)() (const [basic](#) &obj, const [print\\_context](#) &[c](#), unsigned level) const override

### Public Member Functions inherited from [GiNaC::print\\_functor\\_impl](#)

- virtual [~print\\_functor\\_impl](#) ()
- virtual [print\\_functor\\_impl](#) \* [duplicate](#) () const =0
- virtual void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const =0

### Private Attributes

- [F f](#)

## 6.130.1 Detailed Description

```
template<class T, class C>
class GiNaC::print_memfun_handler< T, C >
```

[print\\_functor](#) handler for member functions of class T, context type C

## 6.130.2 Member Typedef Documentation

### 6.130.2.1 F

```
template<class T , class C >
typedef void(T::* GiNaC::print\_memfun\_handler< T, C >::F) (const C &c, unsigned level) const
```

## 6.130.3 Constructor & Destructor Documentation

### 6.130.3.1 [print\\_memfun\\_handler\(\)](#)

```
template<class T , class C >
GiNaC::print\_memfun\_handler< T, C >::print_memfun_handler (
    F f_ ) [inline]
```

## 6.130.4 Member Function Documentation

#### 6.130.4.1 duplicate()

```
template<class T , class C >
print_memfun_handler * GiNaC::print_memfun_handler< T, C >::duplicate ( ) const [inline],
[override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

#### 6.130.4.2 operator>()()

```
template<class T , class C >
void GiNaC::print_memfun_handler< T, C >::operator() (
    const basic & obj,
    const print_context & c,
    unsigned level ) const [inline], [override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

References [c](#), and [GiNaC::print\\_memfun\\_handler< T, C >::f](#).

### 6.130.5 Member Data Documentation

#### 6.130.5.1 f

```
template<class T , class C >
F GiNaC::print_memfun_handler< T, C >::f [private]
```

Referenced by [GiNaC::print\\_memfun\\_handler< T, C >::operator>\(\)](#).

The documentation for this class was generated from the following file:

- [print.h](#)

## 6.131 GiNaC::print\_options Class Reference

Flags to control the behavior of a [print\\_context](#).

```
#include <print.h>
```

### Public Types

- enum { [print\\_index\\_dimensions](#) = 0x0001 }

### 6.131.1 Detailed Description

Flags to control the behavior of a [print\\_context](#).

### 6.131.2 Member Enumeration Documentation

#### 6.131.2.1 anonymous enum

`anonymous enum`



## Enumerator

print_index_dimensions	print the dimensions of indices
------------------------	---------------------------------

The documentation for this class was generated from the following file:

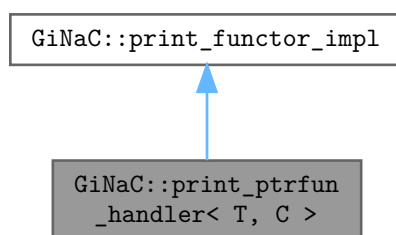
- [print.h](#)

## 6.132 GiNaC::print\_ptrfun\_handler< T, C > Class Template Reference

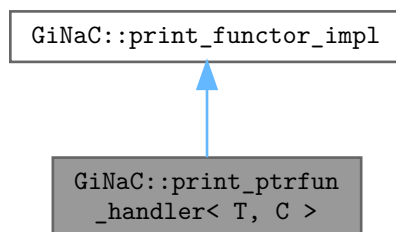
[print\\_functor](#) handler for pointer-to-functions of class T, context type C

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_ptrfun\_handler< T, C >:



Collaboration diagram for GiNaC::print\_ptrfun\_handler< T, C >:



### Public Types

- typedef void(\* [F](#)) (const T &, const C &, unsigned)

## Public Member Functions

- [print\\_ptrfun\\_handler](#) (F f\_)
- [print\\_ptrfun\\_handler](#) \* [duplicate](#) () const override
- void [operator](#)() (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const override

## Public Member Functions inherited from [GiNaC::print\\_functor\\_impl](#)

- virtual [~print\\_functor\\_impl](#) ()
- virtual [print\\_functor\\_impl](#) \* [duplicate](#) () const =0
- virtual void [operator](#)() (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const =0

## Private Attributes

- [F f](#)

### 6.132.1 Detailed Description

```
template<class T, class C>
class GiNaC::print_ptrfun_handler< T, C >
```

[print\\_functor](#) handler for pointer-to-functions of class T, context type C

### 6.132.2 Member Typedef Documentation

#### 6.132.2.1 F

```
template<class T , class C >
typedef void(* GiNaC::print\_ptrfun\_handler< T, C >::F) (const T &, const C &, unsigned)
```

### 6.132.3 Constructor & Destructor Documentation

#### 6.132.3.1 [print\\_ptrfun\\_handler](#)()

```
template<class T , class C >
GiNaC::print\_ptrfun\_handler< T, C >::print_ptrfun_handler (
    F f_ ) [inline]
```

### 6.132.4 Member Function Documentation

#### 6.132.4.1 duplicate()

```
template<class T , class C >
print\_ptrfun\_handler * GiNaC::print\_ptrfun\_handler< T, C >::duplicate ( ) const [inline],
[override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

#### 6.132.4.2 operator>()()

```
template<class T , class C >
void GiNaC::print\_ptrfun\_handler< T, C >::operator() (
    const basic & obj,
    const print\_context & c,
    unsigned level ) const [inline], [override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

References [c](#), and [GiNaC::print\\_ptrfun\\_handler< T, C >::f](#).

### 6.132.5 Member Data Documentation

#### 6.132.5.1 f

```
template<class T , class C >
F GiNaC::print\_ptrfun\_handler< T, C >::f [private]
```

Referenced by [GiNaC::print\\_ptrfun\\_handler< T, C >::operator>\(\)\(\)](#).

The documentation for this class was generated from the following file:

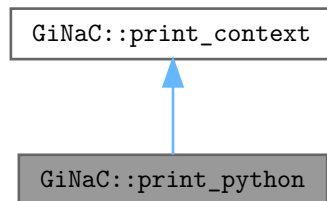
- [print.h](#)

## 6.133 GiNaC::print\_python Class Reference

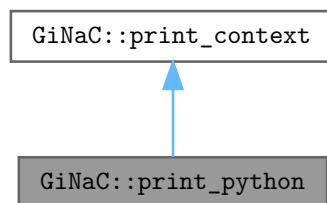
Context for python pretty-print output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_python:



Collaboration diagram for GiNaC::print\_python:



### Public Member Functions

- [print\\_python](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Additional Inherited Members

#### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.133.1 Detailed Description

Context for python pretty-print output.

### 6.133.2 Constructor & Destructor Documentation

#### 6.133.2.1 print\_python()

```
GiNaC::print_python::print_python (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

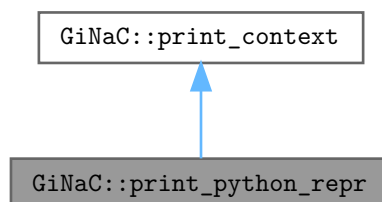
- [print.h](#)
- [print.cpp](#)

## 6.134 GiNaC::print\_python\_repr Class Reference

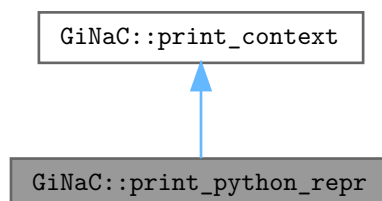
Context for python-parsable output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_python\_repr:



Collaboration diagram for GiNaC::print\_python\_repr:



## Public Member Functions

- [print\\_python\\_repr](#) (std::ostream &, unsigned [options](#)=0)

## Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Additional Inherited Members

## Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.134.1 Detailed Description

Context for python-parsable output.

### 6.134.2 Constructor & Destructor Documentation

#### 6.134.2.1 [print\\_python\\_repr\(\)](#)

```
GiNaC::print_python_repr::print_python_repr (
    std::ostream & os,
    unsigned options = 0 )
```

The documentation for this class was generated from the following files:

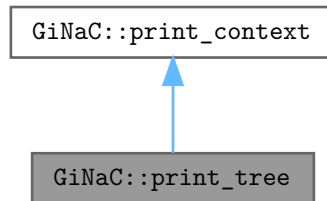
- [print.h](#)
- [print.cpp](#)

## 6.135 GiNaC::print\_tree Class Reference

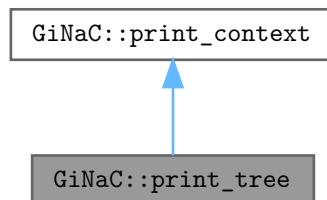
Context for tree-like output for debugging.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_tree:



Collaboration diagram for GiNaC::print\_tree:



### Public Member Functions

- [print\\_tree](#) (unsigned d)
- [print\\_tree](#) (std::ostream &, unsigned [options](#)=0, unsigned d=4)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Public Attributes

- const unsigned [delta\\_indent](#)  
*size of indentation step*

**Public Attributes inherited from [GiNaC::print\\_context](#)**

- `std::ostream & s`  
*stream to output to*
- unsigned `options`  
*option flags*

**6.135.1 Detailed Description**

Context for tree-like output for debugging.

**6.135.2 Constructor & Destructor Documentation****6.135.2.1 `print_tree()` [1/2]**

```
GiNaC::print_tree::print_tree (  
    unsigned d )
```

**6.135.2.2 `print_tree()` [2/2]**

```
GiNaC::print_tree::print_tree (  
    std::ostream & os,  
    unsigned options = 0,  
    unsigned d = 4 )
```

**6.135.3 Member Data Documentation****6.135.3.1 `delta_indent`**

```
const unsigned GiNaC::print_tree::delta_indent
```

size of indentation step

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)



## 6.136 GiNaC::archive\_node::property Struct Reference

Archived property (data type, name and associated data)

```
#include <archive.h>
```

### Public Member Functions

- [property](#) ()
- [property](#) ([archive\\_atom](#) n, [property\\_type](#) t, unsigned v)

### Public Attributes

- [property\\_type](#) type  
*Data type of property.*
- [archive\\_atom](#) name  
*Name of property.*
- unsigned [value](#)  
*Stored value.*

### 6.136.1 Detailed Description

Archived property (data type, name and associated data)

### 6.136.2 Constructor & Destructor Documentation

#### 6.136.2.1 [property\(\)](#) [1/2]

```
GiNaC::archive_node::property::property ( ) [inline]
```

#### 6.136.2.2 [property\(\)](#) [2/2]

```
GiNaC::archive_node::property::property (
    archive\_atom n,
    property\_type t,
    unsigned v ) [inline]
```

### 6.136.3 Member Data Documentation

### 6.136.3.1 type

`property_type` `GiNaC::archive_node::property::type`

Data type of property.

### 6.136.3.2 name

`archive_atom` `GiNaC::archive_node::property::name`

Name of property.

### 6.136.3.3 value

`unsigned` `GiNaC::archive_node::property::value`

Stored value.

The documentation for this struct was generated from the following file:

- [archive.h](#)

## 6.137 GiNaC::archive\_node::property\_info Struct Reference

Information about a stored property.

```
#include <archive.h>
```

### Public Member Functions

- `property_info()`
- `property_info(property_type t, const std::string &n, unsigned c=1)`

### Public Attributes

- `property_type type`  
*Data type of property.*
- `std::string name`  
*Name of property.*
- `unsigned count`  
*Number of occurrences.*

### 6.137.1 Detailed Description

Information about a stored property.

A vector of these structures is returned by [get\\_properties\(\)](#).

See also

[get\\_properties](#)

### 6.137.2 Constructor & Destructor Documentation

#### 6.137.2.1 property\_info() [1/2]

```
GiNaC::archive_node::property_info::property_info ( ) [inline]
```

#### 6.137.2.2 property\_info() [2/2]

```
GiNaC::archive_node::property_info::property_info (
    property\_type t,
    const std::string & n,
    unsigned c = 1 ) [inline]
```

### 6.137.3 Member Data Documentation

#### 6.137.3.1 type

[property\\_type](#) GiNaC::archive\_node::property\_info::type

Data type of property.

#### 6.137.3.2 name

std::string GiNaC::archive\_node::property\_info::name

Name of property.

### 6.137.3.3 count

```
unsigned GiNaC::archive_node::property_info::count
```

Number of occurrences.

The documentation for this struct was generated from the following file:

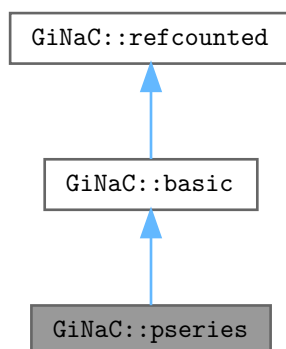
- [archive.h](#)

## 6.138 GiNaC::pseries Class Reference

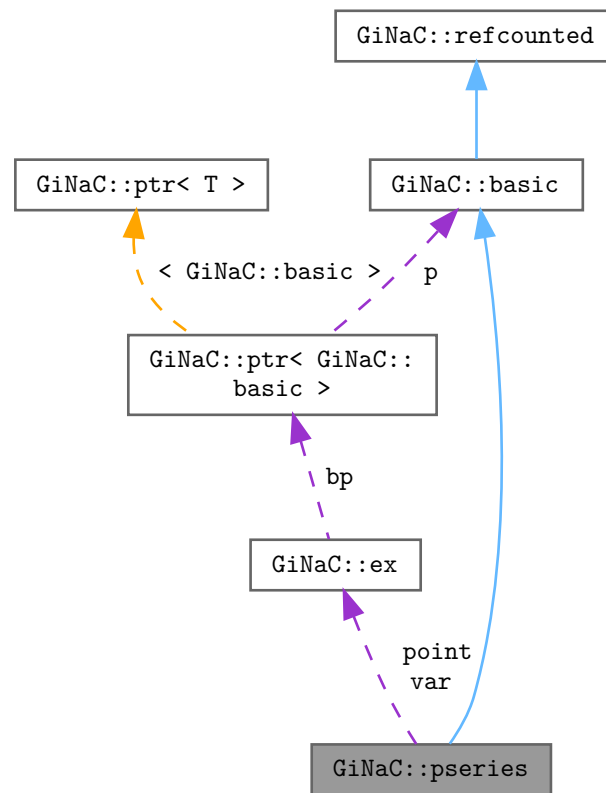
This class holds a extended truncated power series (positive and negative integer powers).

```
#include <pseries.h>
```

Inheritance diagram for GiNaC::pseries:



Collaboration diagram for GiNaC::pseries:



## Public Member Functions

- `pseries` (const `ex` &rel\_, const `epvector` &ops\_)  
Construct `pseries` from a vector of coefficients and powers.
- `pseries` (const `ex` &rel\_, `epvector` &&ops\_)
- unsigned `precedence` () const override  
Return relative operator precedence (for parenthezing output).
- `size_t nops` () const override  
Return the number of operands including a possible order term.
- `ex op` (size\_t i) const override  
Return the *i*th term in the series when represented as a sum.
- int `degree` (const `ex` &s) const override  
Return degree of highest power of the series.
- int `ldegree` (const `ex` &s) const override  
Return degree of lowest power of the series.
- `ex coeff` (const `ex` &s, int `n`=1) const override  
Return coefficient of degree *n* in power series if *s* is the expansion variable.
- `ex collect` (const `ex` &s, bool distributed=false) const override  
Does nothing.

- `ex eval ()` const override  
*Perform coefficient-wise automatic term rewriting rules in this class.*
- `ex evalf ()` const override  
*Evaluate coefficients numerically.*
- `ex series (const relational &r, int order, unsigned options=0)` const override  
*Re-expansion of a pseries object.*
- `ex subs (const exmap &m, unsigned options=0)` const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const override  
*Implementation of `ex::normal()` for pseries.*
- `ex expand (unsigned options=0)` const override  
*Implementation of `ex::expand()` for a power series.*
- `ex conjugate ()` const override
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `ex eval_integ ()` const override  
*Evaluate integrals, if result is known.*
- `ex evalm ()` const override  
*Evaluate sums, products and integer powers of matrices.*
- `void archive (archive_node &n)` const override  
*Save (a.k.a.*
- `void read_archive (const archive_node &n, lst &syms)` override  
*Read (a.k.a.*
- `ex get_var ()` const  
*Get the expansion variable.*
- `ex get_point ()` const  
*Get the expansion point.*
- `ex convert_to_poly (bool no_order=false)` const  
*Convert the pseries object to an ordinary polynomial.*
- `bool is_compatible_to (const pseries &other)` const  
*Check whether series is compatible to another series (expansion variable and point are the same).*
- `bool is_zero ()` const  
*Check whether series has the value zero.*
- `bool is_terminating ()` const  
*Returns true if there is no order term, i.e.*
- `ex coeffop (size_t i)` const  
*Get coefficients and exponents.*
- `ex exponop (size_t i)` const
- `ex add_series (const pseries &other)` const  
*Add one series object to another, producing a pseries object that represents the sum.*
- `ex mul_const (const numeric &other)` const  
*Multiply a pseries object with a numeric constant, producing a pseries object that represents the product.*
- `ex mul_series (const pseries &other)` const  
*Multiply one pseries object to another, producing a pseries object that represents the product.*
- `ex power_const (const numeric &p, int deg)` const  
*Compute the p-th power of a series.*
- `pseries shift_exponents (int deg)` const  
*Return a new pseries object with the powers shifted by deg.*

Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*



- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for a power series.*
- void [print\\_series](#) (const [print\\_context](#) &c, const char \*openbrace, const char \*closebrace, const char \*mul←  
\_sym, const char \*pow\_sym, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex\\_derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [epvector seq](#)  
*Vector of {coefficient, power} pairs.*
- [ex var](#)  
*Series variable (holds a symbol)*
- [ex point](#)  
*Expansion point.*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.138.1 Detailed Description

This class holds a extended truncated power series (positive and negative integer powers).

It consists of expression coefficients (only non-zero coefficients are stored), an expansion variable and an expansion point. Other classes must provide members to convert into this type.

### 6.138.2 Constructor & Destructor Documentation

#### 6.138.2.1 [pseries\(\)](#) [1/2]

```
GiNaC::pseries::pseries (
    const ex & rel_,
    const epvector & ops_ )
```

Construct pseries from a vector of coefficients and powers.

[expair.rest](#) holds the coefficient, [expair.coeff](#) holds the power. The powers must be integers (positive or negative) and in ascending order; the last coefficient can be `Order(_ex1)` to represent a truncated, non-terminating series.

#### Parameters

<a href="#">rel</a> ↔ —	expansion variable and point (must hold a relational)
<a href="#">ops</a> ↔ —	vector of {coefficient, power} pairs (coefficient must not be zero)

**Returns**

newly constructed pseries

References [GINAC\\_ASSERT](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::lhs\(\)](#), [point](#), [GiNaC::ex::rhs\(\)](#), [seq](#), and [var](#).

**6.138.2.2 pseries() [2/2]**

```
GiNaC::pseries::pseries (
    const ex & rel_,
    epvector && ops_ )
```

References [GINAC\\_ASSERT](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::lhs\(\)](#), [point](#), [GiNaC::ex::rhs\(\)](#), [seq](#), and [var](#).

**6.138.3 Member Function Documentation****6.138.3.1 precedence()**

```
unsigned GiNaC::pseries::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [print\\_series\(\)](#).

**6.138.3.2 nops()**

```
size_t GiNaC::pseries::nops ( ) const [override], [virtual]
```

Return the number of operands including a possible order term.

Reimplemented from [GiNaC::basic](#).

References [seq](#).

Referenced by [coeffop\(\)](#), [exponop\(\)](#), and [GiNaC::log\\_series\(\)](#).

### 6.138.3.3 op()

```
ex GiNaC::pseries::op (
    size_t i ) const [override], [virtual]
```

Return the ith term in the series when represented as a sum.

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [GiNaC::pow\(\)](#), [seq](#), and [var](#).

### 6.138.3.4 degree()

```
int GiNaC::pseries::degree (
    const ex & s ) const [override], [virtual]
```

Return degree of highest power of the series.

This is usually the exponent of the Order term. If s is not the expansion variable of the series, the series is examined termwise.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_equal\(\)](#), [seq](#), and [var](#).

Referenced by [mul\\_series\(\)](#), and [series\(\)](#).

### 6.138.3.5 ldegree()

```
int GiNaC::pseries::ldegree (
    const ex & s ) const [override], [virtual]
```

Return degree of lowest power of the series.

This is usually the exponent of the leading term. If s is not the expansion variable of the series, the series is examined termwise. If s is the expansion variable but the expansion point is not zero the series is not expanded to find the degree. I.e.:  $(1-x) + (1-x)^2 + \text{Order}((1-x)^3)$  has `ldegree(x)` 1, not 0.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_equal\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::log\\_series\(\)](#), [mul\\_series\(\)](#), and [power\\_const\(\)](#).

**6.138.3.6 coeff()**

```
ex GiNaC::pseries::coeff (
    const ex & s,
    int n = 1 ) const [override], [virtual]
```

Return coefficient of degree n in power series if s is the expansion variable.

If the expansion point is nonzero, by definition the n=1 coefficient in s of  $a+b*(s-z)+c*(s-z)^2+\text{Order}((s-z)^3)$  is b (assuming the expansion took place in the s in the first place). If s is not the expansion variable, an attempt is made to convert the series to a polynomial and return the corresponding coefficient from there.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::ex::coeff\(\)](#), [coeff\(\)](#), [convert\\_to\\_poly\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [n](#), [seq](#), and [var](#).

Referenced by [coeff\(\)](#), [GiNaC::log\\_series\(\)](#), [mul\\_series\(\)](#), [op\(\)](#), [power\\_const\(\)](#), and [read\\_archive\(\)](#).

**6.138.3.7 collect()**

```
ex GiNaC::pseries::collect (
    const ex & s,
    bool distributed = false ) const [override], [virtual]
```

Does nothing.

Reimplemented from [GiNaC::basic](#).

**6.138.3.8 eval()**

```
ex GiNaC::pseries::eval ( ) const [override], [virtual]
```

Perform coefficient-wise automatic term rewriting rules in this class.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

**6.138.3.9 evalf()**

```
ex GiNaC::pseries::evalf ( ) const [override], [virtual]
```

Evaluate coefficients numerically.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [point](#), [seq](#), and [var](#).

**6.138.3.10 series()**

```
ex GiNaC::pseries::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Re-expansion of a pseries object.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [options](#), [order](#), [point](#), [r](#), [seq](#), [GiNaC::ex::series\(\)](#), and [var](#).

**6.138.3.11 subs()**

```
ex GiNaC::pseries::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [convert\\_to\\_poly\(\)](#), [m](#), [options](#), [point](#), [seq](#), [GiNaC::ex::subs\(\)](#), and [var](#).

**6.138.3.12 normal()**

```
ex GiNaC::pseries::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for pseries.

It normalizes each coefficient and replaces the series by a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::ex::normal\(\)](#), [point](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [seq](#), and [var](#).

### 6.138.3.13 `expand()`

```
ex GiNaC::pseries::expand (
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of `ex::expand()` for a power series.

It expands all the terms individually and returns the resulting series as a new pseries.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::ex::is\\_zero\(\)](#), [options](#), [point](#), [seq](#), and [var](#).

### 6.138.3.14 `conjugate()`

```
ex GiNaC::pseries::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::ex::info\(\)](#), [point](#), [GiNaC::info\\_flags::real](#), [seq](#), and [var](#).

### 6.138.3.15 `real_part()`

```
ex GiNaC::pseries::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::info\(\)](#), [point](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [seq](#), and [var](#).

### 6.138.3.16 `imag_part()`

```
ex GiNaC::pseries::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::info\(\)](#), [point](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [seq](#), and [var](#).

**6.138.3.17 eval\_integ()**

```
ex GiNaC::pseries::eval_integ ( ) const [override], [virtual]
```

Evaluate integrals, if result is known.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::eval\\_integ\(\)](#), [point](#), [seq](#), and [var](#).

**6.138.3.18 evalm()**

```
ex GiNaC::pseries::evalm ( ) const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::evalm\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [seq](#), and [var](#).

**6.138.3.19 archive()**

```
void GiNaC::pseries::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), [point](#), [seq](#), and [var](#).

**6.138.3.20 read\_archive()**

```
void GiNaC::pseries::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [n](#), [point](#), [seq](#), and [var](#).



**6.138.3.21 derivative()**

```
ex GiNaC::pseries::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a power series.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [c](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [seq](#), and [var](#).

**6.138.3.22 get\_var()**

```
ex GiNaC::pseries::get_var ( ) const [inline]
```

Get the expansion variable.

References [var](#).

**6.138.3.23 get\_point()**

```
ex GiNaC::pseries::get_point ( ) const [inline]
```

Get the expansion point.

References [point](#).

**6.138.3.24 convert\_to\_poly()**

```
ex GiNaC::pseries::convert_to_poly (
    bool no_order = false ) const
```

Convert the pseries object to an ordinary polynomial.

Parameters

<i>no_order</i>	flag: discard higher order terms
-----------------	----------------------------------

References [GiNaC::ex::coeff\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [GiNaC::pow\(\)](#), [seq](#), and [var](#).

Referenced by [coeff\(\)](#), [series\(\)](#), and [subs\(\)](#).

#### 6.138.3.25 `is_compatible_to()`

```
bool GiNaC::pseries::is_compatible_to (
    const pseries & other ) const [inline]
```

Check whether series is compatible to another series (expansion variable and point are the same).

References [GiNaC::ex::is\\_equal\(\)](#), [point](#), and [var](#).

Referenced by [add\\_series\(\)](#), and [mul\\_series\(\)](#).

#### 6.138.3.26 `is_zero()`

```
bool GiNaC::pseries::is_zero ( ) const [inline]
```

Check whether series has the value zero.

References [seq](#).

Referenced by [power\\_const\(\)](#).

#### 6.138.3.27 `is_terminating()`

```
bool GiNaC::pseries::is_terminating ( ) const
```

Returns true if there is no order term, i.e.

the series terminates and false otherwise.

References [GiNaC::is\\_order\\_function\(\)](#), and [seq](#).

Referenced by [GiNaC::is\\_terminating\(\)](#), and [GiNaC::log\\_series\(\)](#).

#### 6.138.3.28 `coeffop()`

```
ex GiNaC::pseries::coeffop (
    size_t i ) const
```

Get coefficients and exponents.

References [nops\(\)](#), and [seq](#).

**6.138.3.29 exponop()**

```
ex GiNaC::pseries::exponop (
    size_t i ) const
```

References [nops\(\)](#), and [seq](#).

**6.138.3.30 add\_series()**

```
ex GiNaC::pseries::add_series (
    const pseries & other ) const
```

Add one series object to another, producing a pseries object that represents the sum.

**Parameters**

<i>other</i>	pseries object to add with
--------------	----------------------------

**Returns**

the sum as a pseries

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [is\\_compatible\\_to\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [seq](#), and [var](#).

Referenced by [GiNaC::log\\_series\(\)](#).

**6.138.3.31 mul\_const()**

```
ex GiNaC::pseries::mul_const (
    const numeric & other ) const
```

Multiply a pseries object with a numeric constant, producing a pseries object that represents the product.

**Parameters**

<i>other</i>	constant to multiply with
--------------	---------------------------

**Returns**

the product as a pseries

References [GiNaC::numeric::coeff\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [seq](#), and [var](#).

Referenced by [GiNaC::mul::series\(\)](#).

**6.138.3.32 mul\_series()**

```
ex GiNaC::pseries::mul_series (
    const pseries & other ) const
```

Multiply one pseries object to another, producing a pseries object that represents the product.

**Parameters**

<i>other</i>	pseries object to multiply with
--------------	---------------------------------

**Returns**

the product as a pseries

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [degree\(\)](#), [GiNaC::ex::find\(\)](#), [is\\_compatible\\_to\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [ldegree\(\)](#), [point](#), [seq](#), and [var](#).

Referenced by [GiNaC::mul::series\(\)](#).

**6.138.3.33 power\_const()**

```
ex GiNaC::pseries::power_const (
    const numeric & p,
    int deg ) const
```

Compute the p-th power of a series.

**Parameters**

<i>p</i>	power to compute
<i>deg</i>	truncation order of series calculation

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [coeff\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [is\\_zero\(\)](#), [ldegree\(\)](#), [point](#), [GiNaC::pow\(\)](#), [GiNaC::numeric::real\(\)](#), [seq](#), [GiNaC::to\\_int\(\)](#), and [var](#).

**6.138.3.34 shift\_exponents()**

```
pseries GiNaC::pseries::shift_exponents (
    int deg ) const
```

Return a new pseries object with the powers shifted by deg.

References [point](#), [seq](#), and [var](#).

### 6.138.3.35 print\_series()

```
void GiNaC::pseries::print_series (
    const print\_context & c,
    const char * openbrace,
    const char * closebrace,
    const char * mul_sym,
    const char * pow_sym,
    unsigned level ) const [protected]
```

References [GiNaC::\\_ex1](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [point](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [precedence\(\)](#), [GiNaC::basic::print\(\)](#), [GiNaC::ex::print\(\)](#), [seq](#), and [var](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\(\)](#).

### 6.138.3.36 do\_print()

```
void GiNaC::pseries::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_series\(\)](#).

### 6.138.3.37 do\_print\_latex()

```
void GiNaC::pseries::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_series\(\)](#).

### 6.138.3.38 do\_print\_tree()

```
void GiNaC::pseries::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [point](#), [GiNaC::ex::print\(\)](#), [seq](#), and [var](#).

**6.138.3.39 do\_print\_python()**

```
void GiNaC::pseries::do_print_python (
    const print\_python & c,
    unsigned level ) const [protected]
```

References [c](#), and [print\\_series\(\)](#).

**6.138.3.40 do\_print\_python\_repr()**

```
void GiNaC::pseries::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), [point](#), [GiNaC::ex::print\(\)](#), [seq](#), and [var](#).

**6.138.4 Member Data Documentation****6.138.4.1 seq**

```
epvector GiNaC::pseries::seq [protected]
```

Vector of {coefficient, power} pairs.

Referenced by [add\\_series\(\)](#), [archive\(\)](#), [coeff\(\)](#), [coeffop\(\)](#), [conjugate\(\)](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [exponop\(\)](#), [imag\\_part\(\)](#), [is\\_terminating\(\)](#), [is\\_zero\(\)](#), [ldegree\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [nops\(\)](#), [normal\(\)](#), [op\(\)](#), [power\\_const\(\)](#), [print\\_series\(\)](#), [pseries\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [shift\\_exponents\(\)](#), and [subs\(\)](#).

**6.138.4.2 var**

```
ex GiNaC::pseries::var [protected]
```

Series variable (holds a symbol)

Referenced by [add\\_series\(\)](#), [archive\(\)](#), [coeff\(\)](#), [conjugate\(\)](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [get\\_var\(\)](#), [imag\\_part\(\)](#), [is\\_compatible\\_to\(\)](#), [ldegree\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [normal\(\)](#), [op\(\)](#), [power\\_const\(\)](#), [print\\_series\(\)](#), [pseries\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [shift\\_exponents\(\)](#), and [subs\(\)](#).

### 6.138.4.3 point

`ex GiNaC::pseries::point` [protected]

Expansion point.

Referenced by [add\\_series\(\)](#), [archive\(\)](#), [conjugate\(\)](#), [convert\\_to\\_poly\(\)](#), [derivative\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [get\\_point\(\)](#), [imag\\_part\(\)](#), [is\\_compatible\\_to\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [normal\(\)](#), [op\(\)](#), [power\\_const\(\)](#), [print\\_series\(\)](#), [pseries\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [shift\\_exponents\(\)](#), and [subs\(\)](#).

The documentation for this class was generated from the following files:

- [pseries.h](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.139 GiNaC::psi1\_SERIAL Class Reference

Polylogarithm and multiple polylogarithm.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.139.1 Detailed Description

Polylogarithm and multiple polylogarithm.

Nielsen's generalized polylogarithm. Harmonic polylogarithm. Gamma-function. Beta-function. Psi-function (aka digamma-function).

### 6.139.2 Member Data Documentation

#### 6.139.2.1 serial

```
unsigned GiNaC::psi1_SERIAL::serial
```

 [static]

**Initial value:**

```
=
    function::register_new(function_options("psi", 1).
        eval_func(psi1_eval).
        evalf_func(psi1_evalf).
        derivative_func(psi1_deriv).
        series_func(psi1_series).
        latex_name("\\psi").
        overloaded(2))
```

Referenced by [GiNaC::psi\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_gamma.cpp](#)

## 6.140 GiNaC::psi2\_SERIAL Class Reference

Derivatives of Psi-function (aka polygamma-functions).

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.140.1 Detailed Description

Derivatives of Psi-function (aka polygamma-functions).

### 6.140.2 Member Data Documentation

#### 6.140.2.1 serial

```
unsigned GiNaC::psi2_SERIAL::serial [static]
```

#### Initial value:

```
=  
    function::register_new(function_options("psi", 2).  
        eval_func(psi2_eval).  
        evalf_func(psi2_evalf).  
        derivative_func(psi2_deriv).  
        series_func(psi2_series).  
        latex_name("\\psi").  
        overloaded(2))
```

Referenced by [GiNaC::psi\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_gamma.cpp](#)

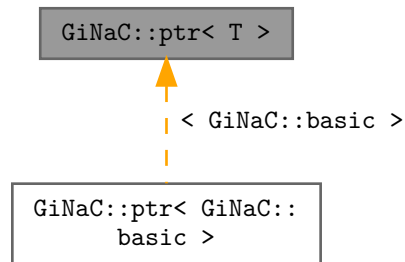


## 6.141 GiNaC::ptr< T > Class Template Reference

Class of (intrusively) reference-counted pointers that support copy-on-write semantics.

```
#include <ptr.h>
```

Inheritance diagram for GiNaC::ptr< T >:



### Public Member Functions

- `ptr` (T \*t) noexcept  
*Bind ptr to newly created object, start reference counting.*
- `ptr` (T &t) noexcept  
*Bind ptr to existing reference-counted object.*
- `ptr` (const `ptr` &other) noexcept
- `~ptr` ()
- `ptr` & `operator=` (const `ptr` &other)
- T & `operator*` () const noexcept
- T \* `operator->` () const noexcept
- void `makewritable` ()  
*Announce your intention to modify the object bound to this ptr.*
- void `swap` (`ptr` &other) noexcept  
*Swap the bound object of this ptr with another ptr.*
- template<class U >  
bool `operator==` (const `ptr`< U > &rhs) const noexcept
- template<class U >  
bool `operator!=` (const `ptr`< U > &rhs) const noexcept

### Private Attributes

- T \* `p`

## Friends

- struct `std::less< ptr< T > >`
- `T * get_pointer (const ptr &x)` noexcept
- `template<class U >`  
`bool operator== (const ptr &lhs, const U *rhs)` noexcept
- `template<class U >`  
`bool operator!= (const ptr &lhs, const U *rhs)` noexcept
- `template<class U >`  
`bool operator== (const U *lhs, const ptr &rhs)` noexcept
- `template<class U >`  
`bool operator!= (const U *lhs, const ptr &rhs)` noexcept
- `std::ostream & operator<< (std::ostream &os, const ptr< T > &rhs)`

### 6.141.1 Detailed Description

```
template<class T>
class GiNaC::ptr< T >
```

Class of (intrusively) reference-counted pointers that support copy-on-write semantics.

Requirements for T: must support the refcounted interface (usually by being derived from refcounted) T\* T↔  
 ::duplicate() member function (only if makewriteable() is used)

### 6.141.2 Constructor & Destructor Documentation

#### 6.141.2.1 ptr() [1/3]

```
template<class T >
GiNaC::ptr< T >::ptr (
    T * t ) [inline], [noexcept]
```

Bind ptr to newly created object, start reference counting.

References [GINAC\\_ASSERT](#), and [GiNaC::ptr< T >::p](#).

#### 6.141.2.2 ptr() [2/3]

```
template<class T >
GiNaC::ptr< T >::ptr (
    T & t ) [inline], [explicit], [noexcept]
```

Bind ptr to existing reference-counted object.

References [GiNaC::ptr< T >::p](#).

### 6.141.2.3 ptr() [3/3]

```
template<class T >
GiNaC::ptr< T >::ptr (
    const ptr< T > & other ) [inline], [noexcept]
```

References [GiNaC::ptr< T >::p](#).

### 6.141.2.4 ~ptr()

```
template<class T >
GiNaC::ptr< T >::~~ptr ( ) [inline]
```

References [GiNaC::ptr< T >::p](#).

## 6.141.3 Member Function Documentation

### 6.141.3.1 operator=()

```
template<class T >
ptr & GiNaC::ptr< T >::operator= (
    const ptr< T > & other ) [inline]
```

References [GiNaC::ptr< T >::p](#).

### 6.141.3.2 operator\*()

```
template<class T >
T & GiNaC::ptr< T >::operator* ( ) const [inline], [noexcept]
```

References [GiNaC::ptr< T >::p](#).

### 6.141.3.3 operator->()

```
template<class T >
T * GiNaC::ptr< T >::operator-> ( ) const [inline], [noexcept]
```

References [GiNaC::ptr< T >::p](#).

#### 6.141.3.4 makewritable()

```
template<class T >
void GiNaC::ptr< T >::makewritable ( ) [inline]
```

Announce your intention to modify the object bound to this ptr.

This ensures that the object is not shared by any other ptrs.

References [GiNaC::ptr< T >::p](#).

#### 6.141.3.5 swap()

```
template<class T >
void GiNaC::ptr< T >::swap (
    ptr< T > & other ) [inline], [noexcept]
```

Swap the bound object of this ptr with another ptr.

References [GiNaC::ptr< T >::p](#).

#### 6.141.3.6 operator==()

```
template<class T >
template<class U >
bool GiNaC::ptr< T >::operator== (
    const ptr< U > & rhs ) const [inline], [noexcept]
```

References [GiNaC::ptr< T >::get\\_pointer](#), [GiNaC::ptr< T >::p](#), and [GiNaC::rhs\(\)](#).

#### 6.141.3.7 operator!=(())

```
template<class T >
template<class U >
bool GiNaC::ptr< T >::operator!= (
    const ptr< U > & rhs ) const [inline], [noexcept]
```

References [GiNaC::ptr< T >::get\\_pointer](#), [GiNaC::ptr< T >::p](#), and [GiNaC::rhs\(\)](#).

### 6.141.4 Friends And Related Function Documentation

#### 6.141.4.1 std::less< ptr< T > >

```
template<class T >
friend struct std::less< ptr< T > > [friend]
```

#### 6.141.4.2 get\_pointer

```
template<class T >
T * get_pointer (
    const ptr< T > & x ) [friend]
```

Referenced by [GiNaC::ptr< T >::operator!=\(\)](#), and [GiNaC::ptr< T >::operator==\(\)](#).

#### 6.141.4.3 operator== [1/2]

```
template<class T >
template<class U >
bool operator== (
    const ptr< T > & lhs,
    const U * rhs ) [friend]
```

#### 6.141.4.4 operator"!= [1/2]

```
template<class T >
template<class U >
bool operator!= (
    const ptr< T > & lhs,
    const U * rhs ) [friend]
```

#### 6.141.4.5 operator== [2/2]

```
template<class T >
template<class U >
bool operator== (
    const U * lhs,
    const ptr< T > & rhs ) [friend]
```

**6.141.4.6 operator"!= [2/2]**

```
template<class T >
template<class U >
bool operator!= (
    const U * lhs,
    const ptr< T > & rhs ) [friend]
```

**6.141.4.7 operator<<**

```
template<class T >
std::ostream & operator<< (
    std::ostream & os,
    const ptr< T > & rhs ) [friend]
```

**6.141.5 Member Data Documentation****6.141.5.1 p**

```
template<class T >
T* GiNaC::ptr< T >::p [private]
```

Referenced by [GiNaC::ptr< T >::makewritable\(\)](#), [GiNaC::ptr< T >::operator!=\(\)](#), [GiNaC::ptr< T >::operator\\*\(\)](#), [GiNaC::ptr< T >::operator->\(\)](#), [GiNaC::ptr< T >::operator=\(\)](#), [GiNaC::ptr< T >::operator==\(\)](#), [GiNaC::ptr< T >::ptr\(\)](#), [GiNaC::ptr< T >::swap\(\)](#), and [GiNaC::ptr< T >::~~ptr\(\)](#).

The documentation for this class was generated from the following file:

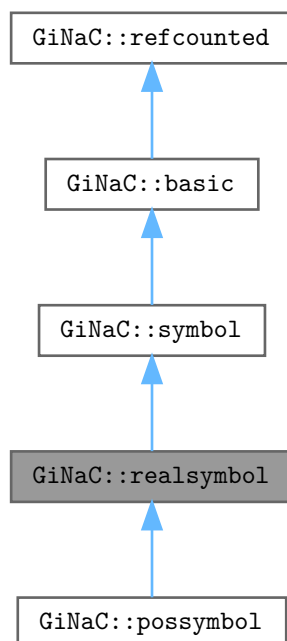
- [ptr.h](#)

**6.142 GiNaC::realsymbol Class Reference**

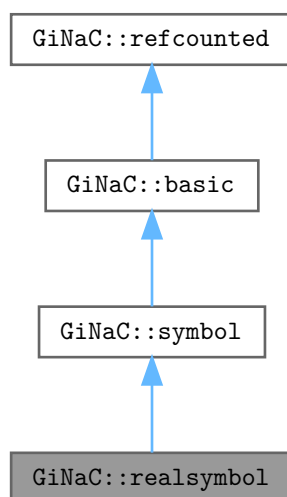
Specialization of symbol to real domain.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::realsymbol:



Collaboration diagram for GiNaC::realsymbol:



## Public Member Functions

- [realsymbol](#) ()
- [realsymbol](#) (const std::string &initname)
- [realsymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [realsymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

## Public Member Functions inherited from [GiNaC::symbol](#)

- [symbol](#) (const std::string &initname)
- [symbol](#) (const std::string &initname, const std::string &texname)
- bool [info](#) (unsigned inf) const override
 

*Information about the object.*
- [ex eval](#) () const override
 

*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override
 

*Evaluate object numerically.*
- [ex series](#) (const [relational](#) &s, int [order](#), unsigned [options](#)=0) const override
 

*Implementation of [ex::series\(\)](#) for symbols.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override
 

*Substitute a set of objects by arbitrary expressions.*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override
 

*Implementation of [ex::normal\(\)](#) for symbols.*
- [ex to\\_rational](#) ([exmap](#) &repl) const override
 

*Implementation of [ex::to\\_rational\(\)](#) for symbols.*
- [ex to\\_polynomial](#) ([exmap](#) &repl) const override
 

*Implementation of [ex::to\\_polynomial\(\)](#) for symbols.*
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_polynomial](#) (const [ex](#) &var) const override
 

*Check whether this is a polynomial in the given variables.*
- void [archive](#) ([archive\\_node](#) &n) const override
 

*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override
 

*Read (a.k.a.*
- virtual unsigned [get\\_domain](#) () const
- void [set\\_name](#) (const std::string &n)
- void [set\\_TeX\\_name](#) (const std::string &n)
- std::string [get\\_name](#) () const
- std::string [get\\_TeX\\_name](#) () const



Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

#### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Additional Inherited Members

#### Protected Member Functions inherited from [GiNaC::symbol](#)

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for single differentiation of a symbol.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

#### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

**Protected Attributes inherited from [GiNaC::symbol](#)**

- unsigned [serial](#)  
*unique serial number for comparison*
- std::string [name](#)  
*printname of this symbol*
- std::string [TeX\\_name](#)  
*LaTeX name of this symbol.*

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

**6.142.1 Detailed Description**

Specialization of symbol to real domain.

**6.142.2 Constructor & Destructor Documentation****6.142.2.1 [realsymbol\(\)](#) [1/3]**

```
GiNaC::realsymbol::realsymbol ( )
```

Referenced by [duplicate\(\)](#).

**6.142.2.2 [realsymbol\(\)](#) [2/3]**

```
GiNaC::realsymbol::realsymbol (
    const std::string & initname ) [explicit]
```

**6.142.2.3 [realsymbol\(\)](#) [3/3]**

```
GiNaC::realsymbol::realsymbol (
    const std::string & initname,
    const std::string & texname )
```

## 6.142.3 Member Function Documentation

### 6.142.3.1 `get_domain()`

```
unsigned GiNaC::realsymbol::get_domain ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

Reimplemented in [GiNaC::possymbol](#).

References [GiNaC::domain::real](#).

### 6.142.3.2 `conjugate()`

```
ex GiNaC::realsymbol::conjugate ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

### 6.142.3.3 `real_part()`

```
ex GiNaC::realsymbol::real_part ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

### 6.142.3.4 `imag_part()`

```
ex GiNaC::realsymbol::imag_part ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

### 6.142.3.5 `duplicate()`

```
realsymbol * GiNaC::realsymbol::duplicate ( ) const [inline], [override], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::possymbol](#).

References [GiNaC::status\\_flags::dynallocated](#), [realsymbol\(\)](#), and [GiNaC::basic::setflag\(\)](#).

The documentation for this class was generated from the following files:

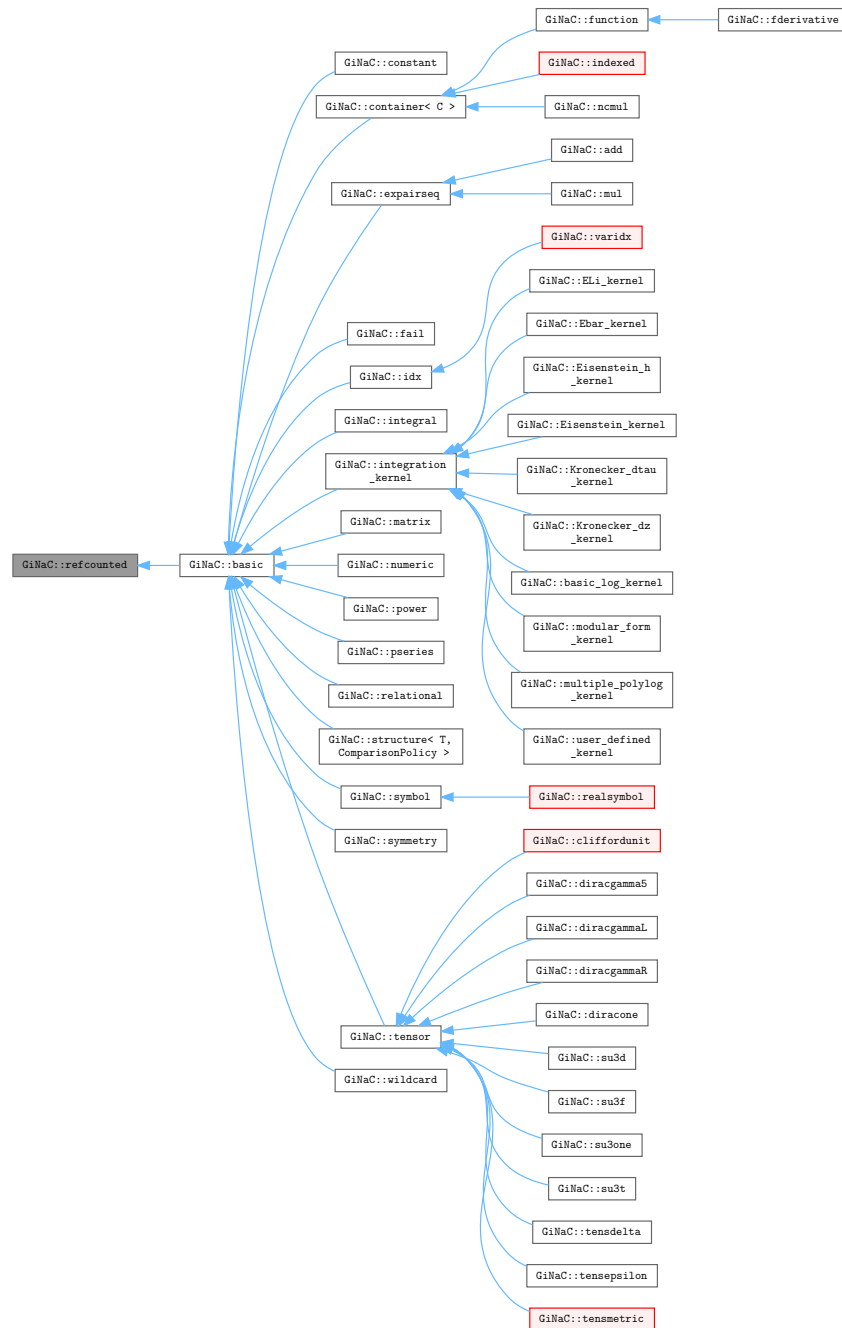
- [symbol.h](#)
- [symbol.cpp](#)

## 6.143 GiNaC::refcounted Class Reference

Base class for reference-counted objects.

```
#include <ptr.h>
```

Inheritance diagram for GiNaC::refcounted:



### Public Member Functions

- [refcounted](#) () noexcept

- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Private Attributes

- unsigned int [refcount](#)  
*reference counter*

### 6.143.1 Detailed Description

Base class for reference-counted objects.

### 6.143.2 Constructor & Destructor Documentation

#### 6.143.2.1 refcounted()

```
GiNaC::refcounted::refcounted ( ) [inline], [noexcept]
```

### 6.143.3 Member Function Documentation

#### 6.143.3.1 add\_reference()

```
unsigned int GiNaC::refcounted::add_reference ( ) [inline], [noexcept]
```

References [refcount](#).

#### 6.143.3.2 remove\_reference()

```
unsigned int GiNaC::refcounted::remove_reference ( ) [inline], [noexcept]
```

References [refcount](#).

### 6.143.3.3 `get_refcount()`

```
unsigned int GiNaC::refcounted::get_refcount ( ) const [inline], [noexcept]
```

References [refcount](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#), and [GiNaC::basic::~~basic\(\)](#).

### 6.143.3.4 `set_refcount()`

```
void GiNaC::refcounted::set_refcount (
    unsigned int r ) [inline], [noexcept]
```

References [r](#), and [refcount](#).

## 6.143.4 Member Data Documentation

### 6.143.4.1 `refcount`

```
unsigned int GiNaC::refcounted::refcount [private]
```

reference counter

Referenced by [add\\_reference\(\)](#), [get\\_refcount\(\)](#), [remove\\_reference\(\)](#), and [set\\_refcount\(\)](#).

The documentation for this class was generated from the following file:

- [ptr.h](#)

## 6.144 `GiNaC::registered_class_options` Class Reference

This class stores information about a registered [GiNaC](#) class.

```
#include <registrar.h>
```

### Public Member Functions

- [registered\\_class\\_options](#) (const char \*n, const char \*p, const std::type\_info &ti)
- const char \* [get\\_name](#) () const
- const char \* [get\\_parent\\_name](#) () const
- std::type\_info const \* [get\\_id](#) () const
- const std::vector< [print\\_func](#) > & [get\\_print\\_dispatch\\_table](#) () const
- template<class Ctx, class T, class C >  
[registered\\_class\\_options](#) & [print\\_func](#) (void f(const T &, const C &c, unsigned))
- template<class Ctx, class T, class C >  
[registered\\_class\\_options](#) & [print\\_func](#) (void(T::\*f)(const C &, unsigned))
- template<class Ctx >  
[registered\\_class\\_options](#) & [print\\_func](#) (const [print\\_func](#) &f)
- void [set\\_print\\_func](#) (unsigned id, const [print\\_func](#) &f)



## Private Attributes

- const char \* [name](#)  
*Class name.*
- const char \* [parent\\_name](#)  
*Name of superclass.*
- std::type\_info const \* [tinfo\\_key](#)  
*Type information key.*
- std::vector< [print\\_functor](#) > [print\\_dispatch\\_table](#)  
*Method table for print() dispatch.*

### 6.144.1 Detailed Description

This class stores information about a registered [GiNaC](#) class.

### 6.144.2 Constructor & Destructor Documentation

#### 6.144.2.1 registered\_class\_options()

```
GiNaC::registered_class_options::registered_class_options (
    const char * n,
    const char * p,
    const std::type_info & ti ) [inline]
```

### 6.144.3 Member Function Documentation

#### 6.144.3.1 get\_name()

```
const char * GiNaC::registered_class_options::get_name ( ) const [inline]
```

References [name](#).

#### 6.144.3.2 get\_parent\_name()

```
const char * GiNaC::registered_class_options::get_parent_name ( ) const [inline]
```

References [parent\\_name](#).

#### 6.144.3.3 `get_id()`

```
std::type_info const * GiNaC::registered_class_options::get_id ( ) const [inline]
```

References [tinfo\\_key](#).

#### 6.144.3.4 `get_print_dispatch_table()`

```
const std::vector< print\_func > & GiNaC::registered_class_options::get_print_dispatch_table  
( ) const [inline]
```

References [print\\_dispatch\\_table](#).

#### 6.144.3.5 `print_func()` [1/3]

```
template<class Ctx , class T , class C >  
registered\_class\_options & GiNaC::registered_class_options::print_func (   
    void fconst T &, const C &c, unsigned ) [inline]
```

References [options](#), and [set\\_print\\_func\(\)](#).

#### 6.144.3.6 `print_func()` [2/3]

```
template<class Ctx , class T , class C >  
registered\_class\_options & GiNaC::registered_class_options::print_func (   
    void(T::*)(const C &, unsigned) f ) [inline]
```

References [options](#), and [set\\_print\\_func\(\)](#).

#### 6.144.3.7 `print_func()` [3/3]

```
template<class Ctx >  
registered\_class\_options & GiNaC::registered_class_options::print_func (   
    const print\_func & f ) [inline]
```

References [options](#), and [set\\_print\\_func\(\)](#).

### 6.144.3.8 set\_print\_func()

```
void GiNaC::registered_class_options::set_print_func (
    unsigned id,
    const print\_functor & f ) [inline]
```

References [print\\_dispatch\\_table](#).

Referenced by [print\\_func\(\)](#).

## 6.144.4 Member Data Documentation

### 6.144.4.1 name

```
const char* GiNaC::registered_class_options::name [private]
```

Class name.

Referenced by [get\\_name\(\)](#).

### 6.144.4.2 parent\_name

```
const char* GiNaC::registered_class_options::parent_name [private]
```

Name of superclass.

Referenced by [get\\_parent\\_name\(\)](#).

### 6.144.4.3 tinfo\_key

```
std::type_info const* GiNaC::registered_class_options::tinfo_key [private]
```

Type information key.

Referenced by [get\\_id\(\)](#).

#### 6.144.4.4 print\_dispatch\_table

```
std::vector<print_functor> GiNaC::registered_class_options::print_dispatch_table [private]
```

Method table for print() dispatch.

Referenced by [get\\_print\\_dispatch\\_table\(\)](#), and [set\\_print\\_func\(\)](#).

The documentation for this class was generated from the following file:

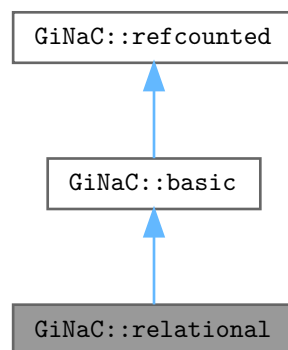
- [registrar.h](#)

### 6.145 GiNaC::relational Class Reference

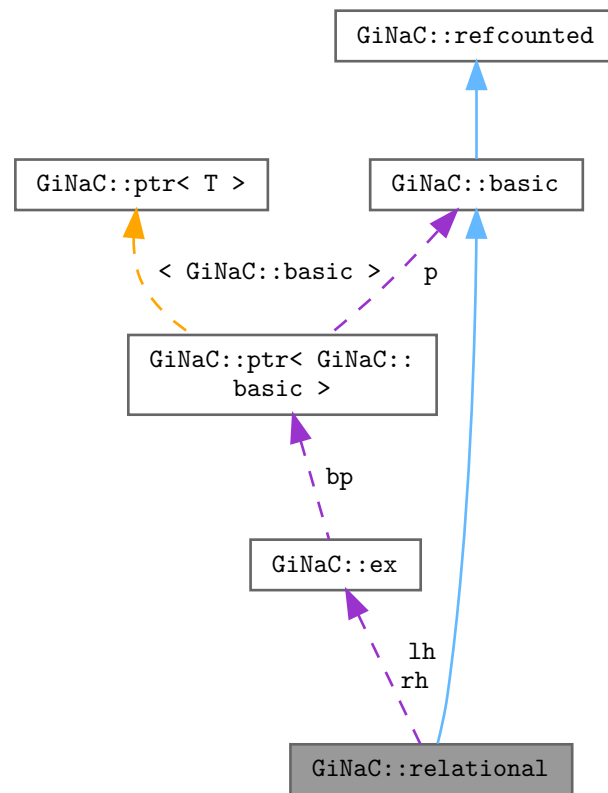
This class holds a relation consisting of two expressions and a logical relation between them.

```
#include <relational.h>
```

Inheritance diagram for GiNaC::relational:



Collaboration diagram for GiNaC::relational:



## Classes

- struct [safe\\_bool\\_helper](#)

## Public Types

- enum [operators](#) {  
[equal](#) , [not\\_equal](#) , [less](#) , [less\\_or\\_equal](#) ,  
[greater](#) , [greater\\_or\\_equal](#) }

## Public Member Functions

- [relational](#) (const [ex](#) &[lhs](#), const [ex](#) &[rhs](#), [operators](#) oper=[equal](#))
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- size\_t [nops](#) () const override

- *Number of operands/members.*
- `ex op` (`size_t i`) `const` override
  - Return operand/member at position  $i$ .*
- `ex map` (`map_function &f`) `const` override
  - Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex subs` (`const exmap &m`, unsigned `options=0`) `const` override
  - Substitute a set of objects by arbitrary expressions.*
- `void archive` (`archive_node &n`) `const` override
  - Save (a.k.a.*
- `void read_archive` (`const archive_node &n`, `lst &symbols`) override
  - Read (a.k.a.*
- `ex canonical` () `const`
  - Returns an equivalent relational with zero right-hand side.*
- `ex lhs` () `const`
- `ex rhs` () `const`
- `operator safe_bool` () `const`
  - Cast the relational into a Boolean, mainly for evaluation within an if-statement.*
- `safe_bool operator!` () `const`

### Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic` ()
  - basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (`const basic &other`)
- `const basic & operator=` (`const basic &other`)
  - basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate` () `const`
  - Create a clone of this object on the heap.*
- `virtual ex eval` () `const`
  - Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf` () `const`
  - Evaluate object numerically.*
- `virtual ex evalm` () `const`
  - Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ` () `const`
  - Evaluate integrals, if result is known.*
- `virtual ex eval_indexed` (`const basic &i`) `const`
  - Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print` (`const print_context &c`, unsigned `level=0`) `const`
  - Output to stream.*
- `virtual void dbgprint` () `const`
  - Little wrapper around `print` to be called within a debugger.*
- `virtual void dbgprinttree` () `const`
  - Little wrapper around `printtree` to be called within a debugger.*
- `virtual unsigned precedence` () `const`
  - Return relative operator precedence (for parenthezing output).*
- `virtual bool info` (unsigned `inf`) `const`
  - Information about the object.*
- `virtual size_t nops` () `const`
  - Number of operands/members.*

- virtual `ex op` (`size_t i`) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex &index`) const
- virtual `ex operator[]` (`size_t i`) const
- virtual `ex & let_op` (`size_t i`)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex &index`)
- virtual `ex & operator[]` (`size_t i`)
- virtual bool `has` (const `ex &other`, unsigned `options=0`) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex &pattern`, `exmap &repls`) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap &m`, unsigned `options=0`) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function &f`) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor &v`) const
- virtual bool `is_polynomial` (const `ex &var`) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex &s`) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex &s`) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex &s`, int `n=1`) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options=0`) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex &s`, bool `distributed=false`) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational &r`, int `order`, unsigned `options=0`) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap &repl`, `exmap &rev_lookup`, `lst &modifier`) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap &repl`) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap &repl`) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric &xi`) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex &self`, const `ex &other`) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex &self`, const `numeric &other`) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator self`, `exvector::iterator other`, `exvector &v`) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const

- virtual `ex conjugate ()` const
- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- `ex eval_ncmul` (const `exvector` &v) const override
- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const



### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- [ex](#) lh
- [ex](#) rh
- [operators](#) o

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### Private Types

- typedef void(safe\_bool\_helper::\* [safe\\_bool](#)) ()

### Private Member Functions

- [safe\\_bool](#) [make\\_safe\\_bool](#) (bool) const

## 6.145.1 Detailed Description

This class holds a relation consisting of two expressions and a logical relation between them.

## 6.145.2 Member Typedef Documentation

### 6.145.2.1 safe\_bool

```
typedef void(safe_bool_helper::* GiNaC::relational::safe_bool) () [private]
```

## 6.145.3 Member Enumeration Documentation

### 6.145.3.1 operators

```
enum GiNaC::relational::operators
```

Enumerator

equal	
not_equal	
less	
less_or_equal	
greater	
greater_or_equal	

## 6.145.4 Constructor & Destructor Documentation

### 6.145.4.1 relational()

```
GiNaC::relational::relational (
    const ex & lhs,
    const ex & rhs,
    operators oper = equal )
```

## 6.145.5 Member Function Documentation

### 6.145.5.1 precedence()

```
unsigned GiNaC::relational::precedence ( ) const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\(\)](#).

### 6.145.5.2 info()

```
bool GiNaC::relational::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [equal](#), [greater](#), [greater\\_or\\_equal](#), [less](#), [less\\_or\\_equal](#), [not\\_equal](#), [o](#), [GiNaC::info\\_flags::relation](#), [GiNaC::info\\_flags::relation\\_equal](#), [GiNaC::info\\_flags::relation\\_greater](#), [GiNaC::info\\_flags::relation\\_greater\\_or\\_equal](#), [GiNaC::info\\_flags::relation\\_less](#), [GiNaC::info\\_flags::relation\\_less\\_or\\_equal](#), and [GiNaC::info\\_flags::relation\\_not\\_equal](#).

Referenced by [operator safe\\_bool\(\)](#).

### 6.145.5.3 nops()

```
size_t GiNaC::relational::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.145.5.4 op()

```
ex GiNaC::relational::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position *i*.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [lh](#), and [rh](#).

#### 6.145.5.5 map()

```
ex GiNaC::relational::map (
    map_function & f ) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [lh](#), [o](#), and [rh](#).

#### 6.145.5.6 subs()

```
ex GiNaC::relational::subs (
    const exmap & m,
    unsigned options = 0 ) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [lh](#), [m](#), [o](#), [options](#), [rh](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

#### 6.145.5.7 archive()

```
void GiNaC::relational::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [lh](#), [n](#), [o](#), and [rh](#).

#### 6.145.5.8 read\_archive()

```
void GiNaC::relational::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [lh](#), [n](#), [o](#), and [rh](#).

### 6.145.5.9 canonical()

```
ex GiNaC::relational::canonical ( ) const
```

Returns an equivalent relational with zero right-hand side.

References [GiNaC::\\_ex0](#), [lh](#), [o](#), and [rh](#).

### 6.145.5.10 eval\_ncmul()

```
ex GiNaC::relational::eval_ncmul (
    const exvector & v ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::eval\\_ncmul\(\)](#), and [lh](#).

### 6.145.5.11 match\_same\_type()

```
bool GiNaC::relational::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [o](#).

### 6.145.5.12 return\_type()

```
unsigned GiNaC::relational::return_type ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [lh](#), [GiNaC::ex::return\\_type\(\)](#), and [rh](#).

**6.145.5.13 return\_type\_tinfo()**

```
return_type_t GiNaC::relational::return_type_tinfo ( ) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [lh](#), [GiNaC::ex::return\\_type\\_tinfo\(\)](#), and [rh](#).

**6.145.5.14 calchash()**

```
unsigned GiNaC::relational::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [equal](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [greater](#), [greater\\_or\\_equal](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [less](#), [less\\_or\\_equal](#), [lh](#), [GiNaC::make\\_hash\\_seed\(\)](#), [not\\_equal](#), [o](#), [rh](#), [GiNaC::rotate\\_left\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**6.145.5.15 do\_print()**

```
void GiNaC::relational::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

References [c](#), [lh](#), [o](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::print\\_operator\(\)](#), and [rh](#).

**6.145.5.16 do\_print\_python\_repr()**

```
void GiNaC::relational::do_print_python_repr (
    const print_python_repr & c,
    unsigned level ) const [protected]
```

References [c](#), [lh](#), [o](#), [GiNaC::ex::print\(\)](#), [GiNaC::print\\_operator\(\)](#), and [rh](#).

**6.145.5.17 lhs()**

```
ex GiNaC::relational::lhs ( ) const [inline]
```

References [lh](#).

Referenced by [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::tan\\_series\(\)](#), and [GiNaC::tanh\\_series\(\)](#).

**6.145.5.18 rhs()**

```
ex GiNaC::relational::rhs ( ) const [inline]
```

References [rh](#).

Referenced by [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), and [GiNaC::log\\_series\(\)](#).

**6.145.5.19 make\_safe\_bool()**

```
relational::safe_bool GiNaC::relational::make_safe_bool (
    bool cond ) const [private]
```

References [GiNaC::relational::safe\\_bool\\_helper::nonnull\(\)](#).

Referenced by [operator safe\\_bool\(\)](#), and [operator!\(\)](#).

**6.145.5.20 operator safe\_bool()**

```
GiNaC::relational::operator relational::safe_bool ( ) const
```

Cast the relational into a Boolean, mainly for evaluation within an if-statement.

Note that  $(a < b) == \text{false}$  does not imply  $(a \geq b) == \text{true}$  in the general symbolic case. A false result means the comparison is either false or undecidable (except of course for  $!=$ , where true means either unequal or undecidable).

References [GiNaC::\\_num0\\_p](#), [equal](#), [greater](#), [greater\\_or\\_equal](#), [GiNaC::ex::info\(\)](#), [info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [less](#), [less\\_or\\_equal](#), [lh](#), [make\\_safe\\_bool\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [not\\_equal](#), [o](#), [GiNaC::info\\_flags::positive](#), and [rh](#).

**6.145.5.21 operator"!()**

```
relational::safe_bool GiNaC::relational::operator! ( ) const [inline]
```

References [make\\_safe\\_bool\(\)](#).

## 6.145.6 Member Data Documentation

### 6.145.6.1 lh

`ex GiNaC::relational::lh` [protected]

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonical\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [eval\\_ncmul\(\)](#), [lhs\(\)](#), [map\(\)](#), [op\(\)](#), [operator safe\\_bool\(\)](#), [read\\_archive\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), and [subs\(\)](#).

### 6.145.6.2 rh

`ex GiNaC::relational::rh` [protected]

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonical\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [map\(\)](#), [op\(\)](#), [operator safe\\_bool\(\)](#), [read\\_archive\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [rhs\(\)](#), and [subs\(\)](#).

### 6.145.6.3 o

`operators GiNaC::relational::o` [protected]

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonical\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [info\(\)](#), [map\(\)](#), [match\\_same\\_type\(\)](#), [operator safe\\_bool\(\)](#), [read\\_archive\(\)](#), and [subs\(\)](#).

The documentation for this class was generated from the following files:

- [relational.h](#)
- [relational.cpp](#)

## 6.146 GiNaC::remember\_strategies Class Reference

Strategies how to clean up the function remember cache.

```
#include <flags.h>
```

### Public Types

- enum { [delete\\_never](#) , [delete\\_lru](#) , [delete\\_lfu](#) , [delete\\_cyclic](#) }



## 6.146.1 Detailed Description

Strategies how to clean up the function remember cache.

See also

[remember\\_table](#)

## 6.146.2 Member Enumeration Documentation

### 6.146.2.1 anonymous enum

anonymous enum

## Enumerator

delete_never	Let table grow indefinitely.
delete_lru	Least recently used.
delete_lfu	Least frequently used.
delete_cyclic	First (oldest) one in list.

The documentation for this class was generated from the following file:

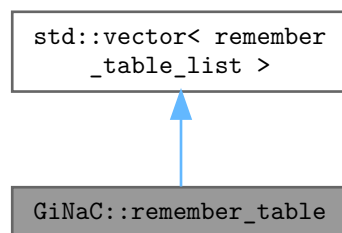
- [flags.h](#)

## 6.147 GiNaC::remember\_table Class Reference

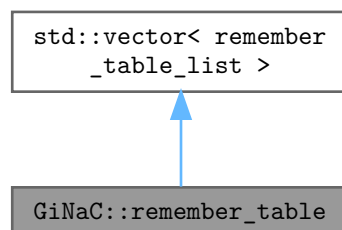
The remember table is organized like an n-fold associative cache in a microprocessor.

```
#include <remember.h>
```

Inheritance diagram for GiNaC::remember\_table:



Collaboration diagram for GiNaC::remember\_table:



## Public Member Functions

- [remember\\_table](#) ()
- [remember\\_table](#) (unsigned s, unsigned as, unsigned strat)
- bool [lookup\\_entry](#) (function const &f, [ex](#) &result) const
- void [add\\_entry](#) (function const &f, [ex](#) const &result)
- void [clear\\_all\\_entries](#) ()
- void [show\\_statistics](#) (std::ostream &os, unsigned level) const

## Static Public Member Functions

- static std::vector< [remember\\_table](#) > & [remember\\_tables](#) ()

## Protected Member Functions

- void [init\\_table](#) ()

## Protected Attributes

- unsigned [table\\_size](#)
- unsigned [max\\_assoc\\_size](#)
- unsigned [remember\\_strategy](#)

### 6.147.1 Detailed Description

The remember table is organized like an n-fold associative cache in a microprocessor.

The table has a width of 's' (which is rounded to `table_size`, some power of 2 near 's', internally) and a depth of 'as' (unless you choose that entries are never discarded). The place where an entry is stored depends on the hashvalue of the parameters of the function (this corresponds to the address of byte to be cached). The '`log_2(table_size)`' least significant bits of this hashvalue give the slot in which the entry will be stored or looked up. Each slot can take up to 'as' entries. If a slot is full, an older entry is removed by one of the following strategies:

- oldest entry (the first one in the list)
- least recently used (the one with the lowest 'last\_access')
- least frequently used (the one with the lowest 'successful\_hits') or all entries are kept which means that the table grows indefinitely.

### 6.147.2 Constructor & Destructor Documentation

#### 6.147.2.1 `remember_table()` [1/2]

```
GiNaC::remember_table::remember_table ( )
```

References [GiNaC::remember\\_strategies::delete\\_never](#), [max\\_assoc\\_size](#), [remember\\_strategy](#), and [table\\_size](#).

### 6.147.2.2 `remember_table()` [2/2]

```
GiNaC::remember_table::remember_table (
    unsigned s,
    unsigned as,
    unsigned strat )
```

References [init\\_table\(\)](#), [GiNaC::log2\(\)](#), and [table\\_size](#).

## 6.147.3 Member Function Documentation

### 6.147.3.1 `lookup_entry()`

```
bool GiNaC::remember_table::lookup_entry (
    function const & f,
    ex & result ) const
```

References [GiNaC::basic::gethash\(\)](#), [GINAC\\_ASSERT](#), and [table\\_size](#).

### 6.147.3.2 `add_entry()`

```
void GiNaC::remember_table::add_entry (
    function const & f,
    ex const & result )
```

References [GiNaC::basic::gethash\(\)](#), [GINAC\\_ASSERT](#), and [table\\_size](#).

### 6.147.3.3 `clear_all_entries()`

```
void GiNaC::remember_table::clear_all_entries ( )
```

References [init\\_table\(\)](#).

### 6.147.3.4 `show_statistics()`

```
void GiNaC::remember_table::show_statistics (
    std::ostream & os,
    unsigned level ) const
```

### 6.147.3.5 remember\_tables()

```
std::vector< remember_table > & GiNaC::remember_table::remember_tables ( ) [static]
```

Referenced by [GiNaC::function::lookup\\_remember\\_table\(\)](#), [GiNaC::function::register\\_new\(\)](#), and [GiNaC::function::store\\_remember\\_t](#).

### 6.147.3.6 init\_table()

```
void GiNaC::remember_table::init_table ( ) [protected]
```

References [max\\_assoc\\_size](#), [remember\\_strategy](#), and [table\\_size](#).

Referenced by [clear\\_all\\_entries\(\)](#), and [remember\\_table\(\)](#).

## 6.147.4 Member Data Documentation

### 6.147.4.1 table\_size

```
unsigned GiNaC::remember_table::table_size [protected]
```

Referenced by [add\\_entry\(\)](#), [init\\_table\(\)](#), [lookup\\_entry\(\)](#), and [remember\\_table\(\)](#).

### 6.147.4.2 max\_assoc\_size

```
unsigned GiNaC::remember_table::max_assoc_size [protected]
```

Referenced by [init\\_table\(\)](#), and [remember\\_table\(\)](#).

### 6.147.4.3 remember\_strategy

```
unsigned GiNaC::remember_table::remember_strategy [protected]
```

Referenced by [init\\_table\(\)](#), and [remember\\_table\(\)](#).

The documentation for this class was generated from the following files:

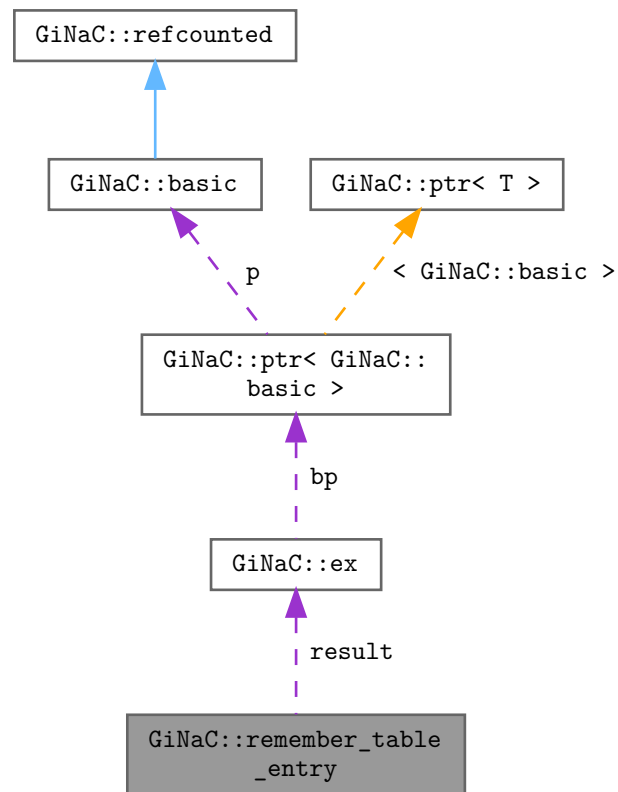
- [remember.h](#)
- [remember.cpp](#)

## 6.148 GiNaC::remember\_table\_entry Class Reference

A single entry in the remember table of a function.

```
#include <remember.h>
```

Collaboration diagram for GiNaC::remember\_table\_entry:



### Public Member Functions

- `remember_table_entry` (function const &f, ex const &r)
- `bool is_equal` (function const &f) const
- `ex get_result` () const
- `unsigned long get_last_access` () const
- `unsigned long get_successful_hits` () const

### Protected Attributes

- `unsigned hashvalue`
- `exvector seq`
- `ex result`
- `unsigned long last_access`
- `unsigned successful_hits`

## Static Protected Attributes

- static unsigned long [access\\_counter](#) = 0

### 6.148.1 Detailed Description

A single entry in the remember table of a function.

Needs to be a friend of class function to access 'seq'. 'last\_access' and 'successful\_hits' are updated at each successful 'is\_equal'.

### 6.148.2 Constructor & Destructor Documentation

#### 6.148.2.1 remember\_table\_entry()

```
GiNaC::remember_table_entry::remember_table_entry (
    function const & f,
    ex const & r )
```

References [access\\_counter](#), [last\\_access](#), and [successful\\_hits](#).

### 6.148.3 Member Function Documentation

#### 6.148.3.1 is\_equal()

```
bool GiNaC::remember_table_entry::is_equal (
    function const & f ) const
```

References [access\\_counter](#), [GiNaC::basic::gethash\(\)](#), [GINAC\\_ASSERT](#), [hashvalue](#), [is\\_equal\(\)](#), [last\\_access](#), [GiNaC::container\\_storage< C >::seq](#), [seq](#), and [successful\\_hits](#).

Referenced by [is\\_equal\(\)](#).

#### 6.148.3.2 get\_result()

```
ex GiNaC::remember_table_entry::get_result ( ) const [inline]
```

References [result](#).

### 6.148.3.3 `get_last_access()`

```
unsigned long GiNaC::remember_table_entry::get_last_access ( ) const [inline]
```

References [last\\_access](#).

### 6.148.3.4 `get_successful_hits()`

```
unsigned long GiNaC::remember_table_entry::get_successful_hits ( ) const [inline]
```

References [successful\\_hits](#).

## 6.148.4 Member Data Documentation

### 6.148.4.1 `hashvalue`

```
unsigned GiNaC::remember_table_entry::hashvalue [protected]
```

Referenced by [is\\_equal\(\)](#).

### 6.148.4.2 `seq`

```
exvector GiNaC::remember_table_entry::seq [protected]
```

Referenced by [is\\_equal\(\)](#).

### 6.148.4.3 `result`

```
ex GiNaC::remember_table_entry::result [protected]
```

Referenced by [get\\_result\(\)](#).

### 6.148.4.4 `last_access`

```
unsigned long GiNaC::remember_table_entry::last_access [mutable], [protected]
```

Referenced by [get\\_last\\_access\(\)](#), [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).



#### 6.148.4.5 successful\_hits

```
unsigned GiNaC::remember_table_entry::successful_hits [mutable], [protected]
```

Referenced by [get\\_successful\\_hits\(\)](#), [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

#### 6.148.4.6 access\_counter

```
unsigned long GiNaC::remember_table_entry::access_counter = 0 [static], [protected]
```

Referenced by [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

The documentation for this class was generated from the following files:

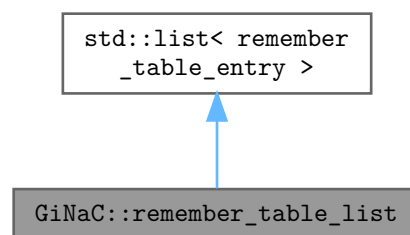
- [remember.h](#)
- [remember.cpp](#)

## 6.149 GiNaC::remember\_table\_list Class Reference

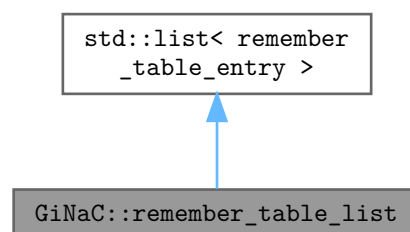
A list of entries in the remember table having some least significant bits of the hashvalue in common.

```
#include <remember.h>
```

Inheritance diagram for GiNaC::remember\_table\_list:



Collaboration diagram for GiNaC::remember\_table\_list:



## Public Member Functions

- [remember\\_table\\_list](#) (unsigned *as*, unsigned *strat*)
- void [add\\_entry](#) ([function](#) const &*f*, [ex](#) const &*result*)
- bool [lookup\\_entry](#) ([function](#) const &*f*, [ex](#) &*result*) const

## Protected Attributes

- unsigned [max\\_assoc\\_size](#)
- unsigned [remember\\_strategy](#)

### 6.149.1 Detailed Description

A list of entries in the remember table having some least significant bits of the hashvalue in common.

### 6.149.2 Constructor & Destructor Documentation

#### 6.149.2.1 [remember\\_table\\_list\(\)](#)

```
GiNaC::remember_table_list::remember_table_list (
    unsigned as,
    unsigned strat )
```

References [max\\_assoc\\_size](#), and [remember\\_strategy](#).

### 6.149.3 Member Function Documentation

#### 6.149.3.1 [add\\_entry\(\)](#)

```
void GiNaC::remember_table_list::add_entry (
    function const & f,
    ex const & result )
```

References [GiNaC::remember\\_strategies::delete\\_cyclic](#), [GiNaC::remember\\_strategies::delete\\_ifu](#), [GiNaC::remember\\_strategies::delete\\_never](#), [GINAC\\_ASSERT](#), [max\\_assoc\\_size](#), and [remember\\_strategy](#).

#### 6.149.3.2 [lookup\\_entry\(\)](#)

```
bool GiNaC::remember_table_list::lookup_entry (
    function const & f,
    ex & result ) const
```

## 6.149.4 Member Data Documentation

### 6.149.4.1 max\_assoc\_size

unsigned GiNaC::remember\_table\_list::max\_assoc\_size [protected]

Referenced by [add\\_entry\(\)](#), and [remember\\_table\\_list\(\)](#).

### 6.149.4.2 remember\_strategy

unsigned GiNaC::remember\_table\_list::remember\_strategy [protected]

Referenced by [add\\_entry\(\)](#), and [remember\\_table\\_list\(\)](#).

The documentation for this class was generated from the following files:

- [remember.h](#)
- [remember.cpp](#)

## 6.150 GiNaC::return\_type\_t Struct Reference

To distinguish between different kinds of non-commutative objects.

```
#include <registrar.h>
```

### Public Member Functions

- bool [operator<](#) (const [return\\_type\\_t](#) &other) const  
*Strict weak ordering (so one can put [return\\_type\\_t](#)'s into a STL container).*
- bool [operator==](#) (const [return\\_type\\_t](#) &other) const
- bool [operator!=](#) (const [return\\_type\\_t](#) &other) const

### Public Attributes

- std::type\_info const \* [tinfo](#)  
*to distinguish between non-commutative objects of different type.*
- unsigned [rl](#)  
*to distinguish between non-commutative objects of the same type.*

### 6.150.1 Detailed Description

To distinguish between different kinds of non-commutative objects.

## 6.150.2 Member Function Documentation

### 6.150.2.1 `operator<()`

```
bool GiNaC::return_type_t::operator< (
    const return\_type\_t & other ) const [inline]
```

Strict weak ordering (so one can put [return\\_type\\_t](#)'s into a STL container).

References [rl](#), and [tinfo](#).

### 6.150.2.2 `operator==()`

```
bool GiNaC::return_type_t::operator== (
    const return\_type\_t & other ) const [inline]
```

References [rl](#), and [tinfo](#).

Referenced by [operator!=\(\)](#).

### 6.150.2.3 `operator"!=()`

```
bool GiNaC::return_type_t::operator!= (
    const return\_type\_t & other ) const [inline]
```

References [operator==\(\)](#).

## 6.150.3 Member Data Documentation

### 6.150.3.1 `tinfo`

```
std::type_info const* GiNaC::return_type_t::tinfo
```

to distinguish between non-commutative objects of different type.

Referenced by [GiNaC::is\\_clifford\\_tinfo\(\)](#), [GiNaC::is\\_color\\_tinfo\(\)](#), [GiNaC::make\\_return\\_type\\_t\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [GiNaC::basic::return\\_type\\_tinfo\(\)](#).

### 6.150.3.2 rl

```
unsigned GiNaC::return_type_t::rl
```

to distinguish between non-commutative objects of the same type.

Think of gamma matrices with different representation labels.

Referenced by [GiNaC::get\\_representation\\_label\(\)](#), [GiNaC::make\\_return\\_type\\_t\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [GiNaC::basic::return\\_type\\_tinfo\(\)](#).

The documentation for this struct was generated from the following file:

- [registrar.h](#)

## 6.151 GiNaC::return\_types Class Reference

```
#include <flags.h>
```

### Public Types

- enum { [commutative](#) , [noncommutative](#) , [noncommutative\\_composite](#) }

### 6.151.1 Member Enumeration Documentation

#### 6.151.1.1 anonymous enum

```
anonymous enum
```

##### Enumerator

<a href="#">commutative</a>	
<a href="#">noncommutative</a>	
<a href="#">noncommutative_composite</a>	

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.152 GiNaC::relational::safe\_bool\_helper Struct Reference

### Public Member Functions

- void [nonnull](#) ()

## 6.152.1 Member Function Documentation

### 6.152.1.1 `nonnull()`

```
void GiNaC::relational::safe_bool_helper::nonnull ( ) [inline]
```

Referenced by [GiNaC::relational::make\\_safe\\_bool\(\)](#).

The documentation for this struct was generated from the following file:

- [relational.h](#)

## 6.153 GiNaC::scalar\_products Class Reference

Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).

```
#include <indexed.h>
```

### Public Member Functions

- void [add](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &sp)  
*Register scalar product pair and its value.*
- void [add](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim, const [ex](#) &sp)  
*Register scalar product pair and its value for a specific space dimension.*
- void [add\\_vectors](#) (const [lst](#) &l, const [ex](#) &dim=[wild](#)())  
*Register list of vectors.*
- void [clear](#) ()  
*Clear all registered scalar products.*
- bool [is\\_defined](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim) const  
*Check whether scalar product pair is defined.*
- [ex](#) [evaluate](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim) const  
*Return value of defined scalar product pair.*
- void [debugprint](#) () const

### Protected Attributes

- [smap](#) [spm](#)

### 6.153.1 Detailed Description

Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).

See also

[simplify\\_indexed](#)

## 6.153.2 Member Function Documentation

### 6.153.2.1 add() [1/2]

```
void GiNaC::scalar_products::add (
    const ex & v1,
    const ex & v2,
    const ex & sp )
```

Register scalar product pair and its value.

References [spm](#).

### 6.153.2.2 add() [2/2]

```
void GiNaC::scalar_products::add (
    const ex & v1,
    const ex & v2,
    const ex & dim,
    const ex & sp )
```

Register scalar product pair and its value for a specific space dimension.

References [spm](#).

### 6.153.2.3 add\_vectors()

```
void GiNaC::scalar_products::add_vectors (
    const lst & l,
    const ex & dim = wild\(\) )
```

Register list of vectors.

This adds all possible pairs of products  $a.i * b.i$  with the value  $a*b$  (note that this is not a scalar vector product but an ordinary product of scalars).

### 6.153.2.4 clear()

```
void GiNaC::scalar_products::clear ( )
```

Clear all registered scalar products.

References [spm](#).

#### 6.153.2.5 is\_defined()

```
bool GiNaC::scalar_products::is_defined (
    const ex & v1,
    const ex & v2,
    const ex & dim ) const
```

Check whether scalar product pair is defined.

References [spm](#).

#### 6.153.2.6 evaluate()

```
ex GiNaC::scalar_products::evaluate (
    const ex & v1,
    const ex & v2,
    const ex & dim ) const
```

Return value of defined scalar product pair.

References [spm](#).

#### 6.153.2.7 debugprint()

```
void GiNaC::scalar_products::debugprint ( ) const
```

References [k](#), and [spm](#).

### 6.153.3 Member Data Documentation

#### 6.153.3.1 spm

```
spm GiNaC::scalar_products::spm [protected]
```

Referenced by [add\(\)](#), [clear\(\)](#), [debugprint\(\)](#), [evaluate\(\)](#), and [is\\_defined\(\)](#).

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)



## 6.154 GiNaC::series\_options Class Reference

Flags to control series expansion.

```
#include <flags.h>
```

### Public Types

- enum { [suppress\\_branchcut](#) = 0x0001 }

#### 6.154.1 Detailed Description

Flags to control series expansion.

#### 6.154.2 Member Enumeration Documentation

##### 6.154.2.1 anonymous enum

anonymous enum

##### Enumerator

<a href="#">suppress_branchcut</a>	Suppress branch cuts in series expansion. Branch cuts manifest themselves as step functions, if this option is not passed. If it is passed and expansion at a point on a cut is performed, then the analytic continuation of the function is expanded.
------------------------------------	--

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.155 GiNaC::solve\_algo Class Reference

Switch to control algorithm for linear system solving.

```
#include <flags.h>
```

### Public Types

- enum {  
[automatic](#) , [gauss](#) , [divfree](#) , [bareiss](#) ,  
[markowitz](#) }

### 6.155.1 Detailed Description

Switch to control algorithm for linear system solving.

### 6.155.2 Member Enumeration Documentation

#### 6.155.2.1 anonymous enum

anonymous enum

##### Enumerator

automatic	Let the system choose. A heuristics is applied for automatic determination of a suitable algorithm.
gauss	<p>Gauss elimination. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$ <p>This algorithm is well-suited for numerical matrices but generally suffers from the expensive division (and computation of GCDs) at each step.</p>
divfree	<p>Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$ <p>This algorithm is only there for the purpose of cross-checks. It suffers from exponential intermediate expression swell. Use it only for small systems.</p>
bareiss	<p>Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is <i>delayed</i> until it can be carried out without computing GCDs. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = (m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}) / m_{k-1,k-1}^{(k-1)}$ <p>(We have set <math>m_{-1,-1}^{(-1)} = 1</math> in order to avoid a case distinction in above formula.) It can be shown that nothing more than polynomial long division is needed for carrying out the division. This is generally the fastest algorithm for solving linear systems. In contrast to division-free elimination it only has a linear expression swell. For two-dimensional systems, the two algorithms are equivalent, however.</p>
markowitz	Markowitz-ordered Gaussian elimination. Same as the usual Gaussian elimination, but with additional effort spent on selecting pivots that minimize fill-in. Faster than the methods above for large sparse matrices (particularly with symbolic coefficients), otherwise slightly slower than Gaussian elimination.

The documentation for this class was generated from the following file:

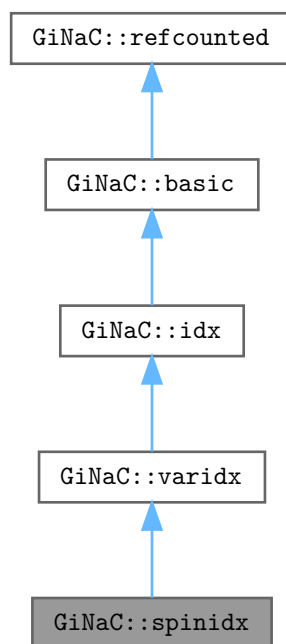
- [flags.h](#)

## 6.156 GiNaC::spinidx Class Reference

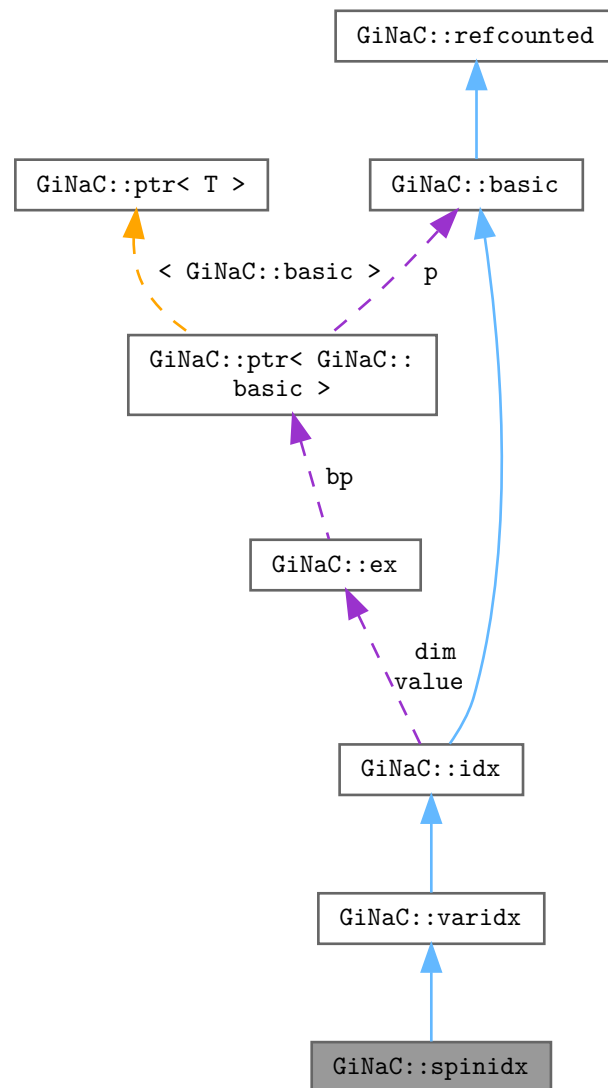
This class holds a spinor index that can be dotted or undotted and that also has a variance.

```
#include <idx.h>
```

Inheritance diagram for GiNaC::spinidx:



Collaboration diagram for `GiNaC::spinidx`:



## Public Member Functions

- `spinidx` (const `ex` &`v`, const `ex` &`dim`=2, bool `covariant`=false, bool `dotted`=false)  
Construct index with given value, dimension, variance and dot.
- bool `is_dummy_pair_same_type` (const `basic` &`other`) const override  
Check whether the index forms a dummy index pair with another index of the same type.
- `ex conjugate` () const override
- void `archive` (`archive_node` &`n`) const override  
Save (serialize) the object into archive node.
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
Load (deserialize) the object from an archive node.

- `bool is_dotted () const`  
*Check whether the index is dotted.*
- `bool is_undotted () const`  
*Check whether the index is not dotted.*
- `ex toggle_dot () const`  
*Make a new index with the same value and variance but the opposite dottedness.*
- `ex toggle_variance_dot () const`  
*Make a new index with the same value but opposite variance and dottedness.*

#### Public Member Functions inherited from `GiNaC::varidx`

- `varidx (const ex &v, const ex &dim, bool covariant=false)`  
*Construct index with given value, dimension and variance.*
- `bool is_dummy_pair_same_type (const basic &other) const override`  
*Check whether the index forms a dummy index pair with another index of the same type.*
- `void archive (archive_node &n) const override`  
*Save (serialize) the object into archive node.*
- `void read_archive (const archive_node &n, lst &syms) override`  
*Load (deserialize) the object from an archive node.*
- `bool is_covariant () const`  
*Check whether the index is covariant.*
- `bool is_contravariant () const`  
*Check whether the index is contravariant (not covariant).*
- `ex toggle_variance () const`  
*Make a new index with the same value but the opposite variance.*

#### Public Member Functions inherited from `GiNaC::idx`

- `idx (const ex &v, const ex &dim)`  
*Construct index with given value and dimension.*
- `bool info (unsigned inf) const override`  
*Information about the object.*
- `size_t nops () const override`  
*Number of operands/members.*
- `ex op (size_t i) const override`  
*Return operand/member at position i.*
- `ex map (map_function &f) const override`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex evalf () const override`  
*By default, `basic::evalf` would evaluate the index value but we don't want a.1 to become a.*
- `ex subs (const exmap &m, unsigned options=0) const override`  
*Substitute a set of objects by arbitrary expressions.*
- `void archive (archive_node &n) const override`  
*Save (serialize) the object into archive node.*
- `void read_archive (const archive_node &n, lst &syms) override`  
*Load (deserialize) the object from an archive node.*
- `virtual bool is_dummy_pair_same_type (const basic &other) const`  
*Check whether the index forms a dummy index pair with another index of the same type.*
- `ex get_value () const`

- *Get value of index.*
- `bool is_numeric () const`  
*Check whether the index is numeric.*
- `bool is_symbolic () const`  
*Check whether the index is symbolic.*
- `ex get_dim () const`  
*Get dimension of index space.*
- `bool is_dim_numeric () const`  
*Check whether the dimension is numeric.*
- `bool is_dim_symbolic () const`  
*Check whether the dimension is symbolic.*
- `ex replace_dim (const ex &new_dim) const`  
*Make a new index with the same value but a different dimension.*
- `ex minimal_dim (const idx &other) const`  
*Return the minimum of the dimensions of this and another index.*

### Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate () const`  
*Create a clone of this object on the heap.*
- `virtual ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf () const`  
*Evaluate object numerically.*
- `virtual ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- `virtual ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- `virtual void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- `virtual void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- `virtual unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- `virtual bool info (unsigned inf) const`  
*Information about the object.*
- `virtual size_t nops () const`  
*Number of operands/members.*
- `virtual ex op (size_t i) const`  
*Return operand/member at position i.*
- `virtual ex operator[] (const ex &index) const`

- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const

- `template<class T >`  
`void print_dispatch (const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `virtual void archive (archive_node &n) const`  
*Save (serialize) the object into archive node.*
- `virtual void read_archive (const archive_node &n, lst &syms)`  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level (const exmap &m, unsigned options) const`  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1) const`  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare (const basic &other) const`  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal (const basic &other) const`  
*Test for syntactic equality.*
- `const basic & hold () const`  
*Stop further evaluation.*
- `unsigned gethash () const`
- `const basic & setflag (unsigned f) const`  
*Set some `status_flags`.*
- `const basic & clearflag (unsigned f) const`  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted () noexcept`
- `unsigned int add_reference () noexcept`
- `unsigned int remove_reference () noexcept`
- `unsigned int get_refcount () const noexcept`
- `void set_refcount (unsigned int r) noexcept`

### Protected Member Functions

- `bool match_same_type (const basic &other) const override`  
*Returns true if the attributes of two objects are similar enough for a match.*
- `void do_print (const print_context &c, unsigned level) const`
- `void do_print_latex (const print_latex &c, unsigned level) const`
- `void do_print_tree (const print_tree &c, unsigned level) const`

### Protected Member Functions inherited from `GiNaC::varidx`

- `bool match_same_type (const basic &other) const override`  
*Returns true if the attributes of two objects are similar enough for a match.*
- `void do_print (const print_context &c, unsigned level) const`
- `void do_print_tree (const print_tree &c, unsigned level) const`



**Protected Member Functions inherited from GiNaC::idx**

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an index always returns 0.*
- `bool match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `unsigned calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void print_index` (const `print_context` &c, unsigned level) const
- `void do_print` (const `print_context` &c, unsigned level) const
- `void do_print_csrc` (const `print_csrc` &c, unsigned level) const
- `void do_print_latex` (const `print_latex` &c, unsigned level) const
- `void do_print_tree` (const `print_tree` &c, unsigned level) const

**Protected Member Functions inherited from GiNaC::basic**

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &v) const
- `virtual bool match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- `virtual int compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- `virtual unsigned calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- `void do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- `void do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- `void do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

**Protected Attributes**

- `bool dotted`

**Protected Attributes inherited from GiNaC::varidx**

- `bool covariant`  
 *$x.\mu$ , default is contravariant:  $x \sim \mu$*

### Protected Attributes inherited from `GiNaC::idx`

- `ex value`  
*Expression that constitutes the index (numeric or symbolic name)*
- `ex dim`  
*Dimension of space (can be symbolic or numeric)*

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 6.156.1 Detailed Description

This class holds a spinor index that can be dotted or undotted and that also has a variance.

This is used in the Weyl-van-der-Waerden formalism where the dot indicates complex conjugation. There is an associated (asymmetric) metric tensor that can be used to raise/lower spinor indices.

## 6.156.2 Constructor & Destructor Documentation

### 6.156.2.1 `spinidx()`

```
GiNaC::spinidx::spinidx (
    const ex & v,
    const ex & dim = 2,
    bool covariant = false,
    bool dotted = false )
```

Construct index with given value, dimension, variance and dot.

#### Parameters

<code>v</code>	Value of index (numeric or symbolic)
<code>dim</code>	Dimension of index space (numeric or symbolic)
<code>covariant</code>	Make covariant index (default is contravariant)
<code>dotted</code>	Make covariant dotted (default is undotted)

**Returns**

newly constructed index

### 6.156.3 Member Function Documentation

#### 6.156.3.1 is\_dummy\_pair\_same\_type()

```
bool GiNaC::spinidx::is_dummy_pair_same_type (
    const basic & other ) const [override], [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented from [GiNaC::varidx](#).

References [dotted](#).

#### 6.156.3.2 conjugate()

```
ex GiNaC::spinidx::conjugate ( ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [toggle\\_dot\(\)](#).

#### 6.156.3.3 archive()

```
void GiNaC::spinidx::archive (
    archive_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::varidx](#).

References [dotted](#), and [n](#).

#### 6.156.3.4 read\_archive()

```
void GiNaC::spinidx::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

##### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::varidx](#).

References [dotted](#), and [n](#).

#### 6.156.3.5 match\_same\_type()

```
bool GiNaC::spinidx::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

##### See also

[basic::match](#)

Reimplemented from [GiNaC::varidx](#).

References [dotted](#), and [GINAC\\_ASSERT](#).

#### 6.156.3.6 is\_dotted()

```
bool GiNaC::spinidx::is_dotted ( ) const [inline]
```

Check whether the index is dotted.

References [dotted](#).

### 6.156.3.7 is\_undotted()

```
bool GiNaC::spinidx::is_undotted ( ) const [inline]
```

Check whether the index is not dotted.

References [dotted](#).

### 6.156.3.8 toggle\_dot()

```
ex GiNaC::spinidx::toggle_dot ( ) const
```

Make a new index with the same value and variance but the opposite dottedness.

References [GiNaC::basic::clearflag\(\)](#), [dotted](#), [GiNaC::basic::duplicate\(\)](#), and [GiNaC::status\\_flags::hash\\_calculated](#).

Referenced by [conjugate\(\)](#).

### 6.156.3.9 toggle\_variance\_dot()

```
ex GiNaC::spinidx::toggle_variance_dot ( ) const
```

Make a new index with the same value but opposite variance and dottedness.

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::varidx::covariant](#), [dotted](#), [GiNaC::basic::duplicate\(\)](#), and [GiNaC::status\\_flags::hash\\_calculated](#).

### 6.156.3.10 do\_print()

```
void GiNaC::spinidx::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::varidx::covariant](#), [dotted](#), and [GiNaC::idx::print\\_index\(\)](#).

### 6.156.3.11 do\_print\_latex()

```
void GiNaC::spinidx::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), [dotted](#), and [GiNaC::idx::print\\_index\(\)](#).

### 6.156.3.12 do\_print\_tree()

```
void GiNaC::spinidx::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::varidx::covariant](#), [GiNaC::idx::dim](#), [dotted](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::print\(\)](#), and [GiNaC::idx::value](#).

## 6.156.4 Member Data Documentation

### 6.156.4.1 dotted

```
bool GiNaC::spinidx::dotted [protected]
```

Referenced by [archive\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [is\\_dotted\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [is\\_undotted\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [toggle\\_dot\(\)](#), and [toggle\\_variance\\_dot\(\)](#).

The documentation for this class was generated from the following files:

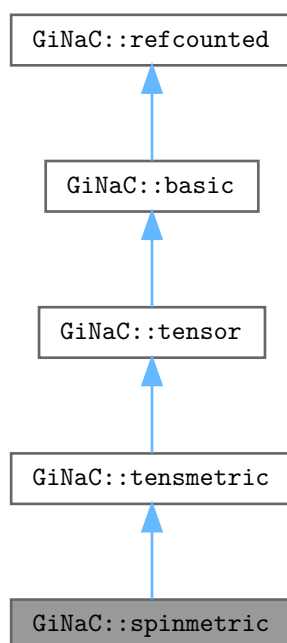
- [idx.h](#)
- [idx.cpp](#)

## 6.157 GiNaC::spinmetric Class Reference

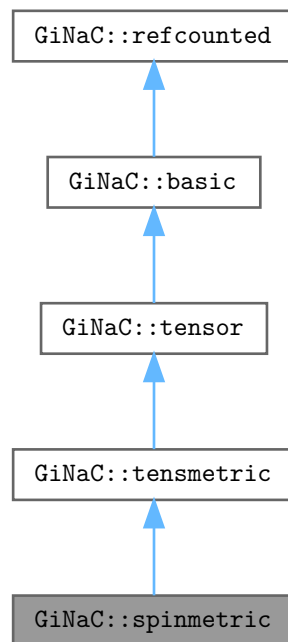
This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::spinmetric:



Collaboration diagram for `GiNaC::spinmetric`:



## Public Member Functions

- `bool info` (unsigned inf) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed spinor metric with something else.*
- `bool info` (unsigned inf) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed metric tensor with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*



Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

#### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

#### Protected Member Functions

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

#### Protected Member Functions inherited from [GiNaC::tensmetric](#)

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- unsigned [return\\_type](#) () const override

#### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex](#) [eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex](#) [derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.157.1 Detailed Description

This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.

If indexed, it must have exactly two indices of the same type which must be of class `spinidx` or a subclass and have dimension 2.

## 6.157.2 Member Function Documentation

### 6.157.2.1 `info()`

```
bool GiNaC::spinmetric::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::info\\_flags::real](#).

### 6.157.2.2 `eval_indexed()`

```
ex GiNaC::spinmetric::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed metric tensor.

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), and [GiNaC::basic::op\(\)](#).

### 6.157.2.3 contract\_with()

```
bool GiNaC::spinmetric::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed spinor metric with something else.

Reimplemented from [GiNaC::tensmetric](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex\\_2](#), [GiNaC::delta\\_tensor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_dummy\\_pair\(\)](#), and [GiNaC::idx::is\\_symbolic\(\)](#).

### 6.157.2.4 do\_print()

```
void GiNaC::spinmetric::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.157.2.5 do\_print\_latex()

```
void GiNaC::spinmetric::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

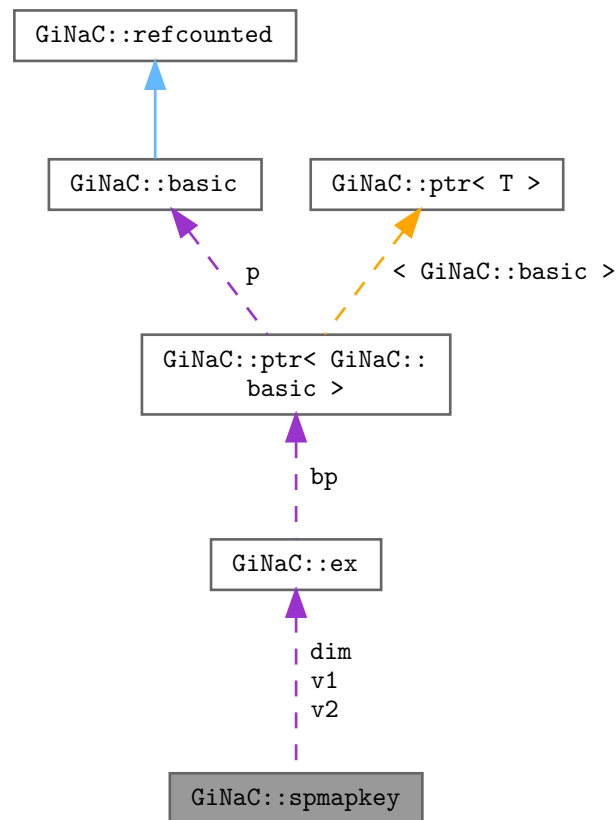
The documentation for this class was generated from the following files:

- [tensor.h](#)
- [tensor.cpp](#)

## 6.158 GiNaC::spmapkey Class Reference

```
#include <indexed.h>
```

Collaboration diagram for `GiNaC::spmapkey`:



## Public Member Functions

- `spmapkey ()`
- `spmapkey (const ex &v1, const ex &v2, const ex &dim=wild())`
- `bool operator== (const spmapkey &other) const`
- `bool operator< (const spmapkey &other) const`
- `void debugprint () const`

## Protected Attributes

- `ex v1`
- `ex v2`
- `ex dim`

### 6.158.1 Constructor & Destructor Documentation

### 6.158.1.1 spmapkey() [1/2]

```
GiNaC::spmapkey::spmapkey ( ) [inline]
```

### 6.158.1.2 spmapkey() [2/2]

```
GiNaC::spmapkey::spmapkey (
    const ex & v1,
    const ex & v2,
    const ex & dim = wild\(\) )
```

References [GiNaC::ex::compare\(\)](#), [GiNaC::ex::op\(\)](#), [v1](#), and [v2](#).

## 6.158.2 Member Function Documentation

### 6.158.2.1 operator==( )

```
bool GiNaC::spmapkey::operator==(
    const spmapkey & other ) const
```

References [dim](#), [GiNaC::ex::is\\_equal\(\)](#), [v1](#), and [v2](#).

### 6.158.2.2 operator<( )

```
bool GiNaC::spmapkey::operator< (
    const spmapkey & other ) const
```

References [GiNaC::ex::compare\(\)](#), [dim](#), [v1](#), and [v2](#).

### 6.158.2.3 debugprint( )

```
void GiNaC::spmapkey::debugprint ( ) const
```

References [dim](#), [v1](#), and [v2](#).

## 6.158.3 Member Data Documentation

### 6.158.3.1 v1

`ex GiNaC::spmapkey::v1 [protected]`

Referenced by [debugprint\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [spmapkey\(\)](#).

### 6.158.3.2 v2

`ex GiNaC::spmapkey::v2 [protected]`

Referenced by [debugprint\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [spmapkey\(\)](#).

### 6.158.3.3 dim

`ex GiNaC::spmapkey::dim [protected]`

Referenced by [debugprint\(\)](#), [operator<\(\)](#), and [operator==\(\)](#).

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 6.159 GiNaC::status\_flags Class Reference

Flags to store information about the state of an object.

```
#include <flags.h>
```

### Public Types

- enum {  
    [dynallocated](#) = 0x0001 , [evaluated](#) = 0x0002 , [expanded](#) = 0x0004 , [hash\\_calculated](#) = 0x0008 ,  
    [not\\_shareable](#) = 0x0010 , [has\\_indices](#) = 0x0020 , [has\\_no\\_indices](#) = 0x0040 , [is\\_positive](#) = 0x0080 ,  
    [is\\_negative](#) = 0x0100 , [purely\\_indefinite](#) = 0x0200 }

### 6.159.1 Detailed Description

Flags to store information about the state of an object.

See also

[basic::flags](#)

### 6.159.2 Member Enumeration Documentation

#### 6.159.2.1 anonymous enum

anonymous enum



## Enumerator

dynallocated	heap-allocated (i.e. created by new if we want to be clever and bypass the stack, See also <a href="#">ex::construct_from_basic()</a> )
evaluated	<a href="#">.eval()</a> has already done its job
expanded	<a href="#">.expand(0)</a> has already done its job (other <a href="#">expand()</a> options ignore this flag)
hash_calculated	<a href="#">.calchash()</a> has already done its job
not_shareable	don't share instances of this object between different expressions unless explicitly asked to (used by <a href="#">ex::compare()</a> )
has_indices	
has_no_indices	
is_positive	
is_negative	
purely_indefinite	

The documentation for this class was generated from the following file:

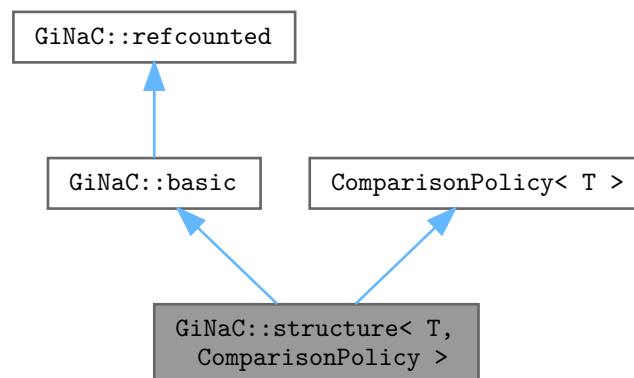
- [flags.h](#)

## 6.160 GiNaC::structure< T, ComparisonPolicy > Class Template Reference

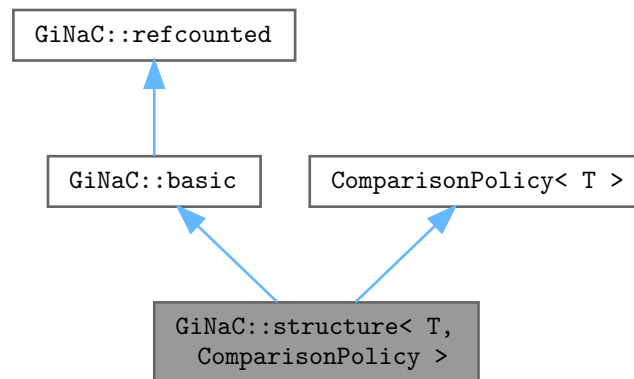
Wrapper template for making [GiNaC](#) classes out of C++ structures.

```
#include <structure.h>
```

Inheritance diagram for GiNaC::structure< T, ComparisonPolicy >:



Collaboration diagram for `GiNaC::structure< T, ComparisonPolicy >`:



## Public Member Functions

- `structure` (const T &t)  
*Construct structure as a copy of a given C++ structure.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex evalm` () const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex eval_indexed` (const `basic` &i) const override  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `void print` (const `print_context` &c, unsigned level=0) const override  
*Output to stream.*
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex operator[]` (const `ex` &index) const override
- `ex operator[]` (size\_t i) const override
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex & operator[]` (const `ex` &index) override
- `ex & operator[]` (size\_t i) override
- bool `has` (const `ex` &other, unsigned `options`=0) const override  
*Test for occurrence of a pattern.*
- bool `match` (const `ex` &pattern, `exmap` &repl\_lst) const override  
*Check whether the expression matches a given pattern.*
- `ex subs` (const `exmap` &m, unsigned `options`=0) const override

- Substitute a set of objects by arbitrary expressions.*

  - `ex map` (`map_function` &`f`) const override

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `int degree` (const `ex` &`s`) const override

*Return degree of highest power in object `s`.*
- `int ldegree` (const `ex` &`s`) const override

*Return degree of lowest power in object `s`.*
- `ex coeff` (const `ex` &`s`, int `n=1`) const override

*Return coefficient of degree `n` in object `s`.*
- `ex expand` (unsigned `options=0`) const override

*Expand expression, i.e.*
- `ex collect` (const `ex` &`s`, bool `distributed=false`) const override

*Sort expanded expression in terms of powers of some object(s).*
- `ex series` (const `relational` &`r`, int `order`, unsigned `options=0`) const override

*Default implementation of `ex::series()`.*
- `ex normal` (`exmap` &`repl`, `exmap` &`rev_lookup`, `lst` &`modifier`) const override

*Default implementation of `ex::normal()`.*
- `ex to_rational` (`exmap` &`repl`) const override

*Default implementation of `ex::to_rational()`.*
- `ex to_polynomial` (`exmap` &`repl`) const override
- `numeric integer_content` () const override
- `ex smod` (const `numeric` &`xi`) const override

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient` () const override

*Implementation `ex::max_coefficient()`.*
- `exvector get_free_indices` () const override

*Return a vector containing the free indices of an expression.*
- `ex add_indexed` (const `ex` &`self`, const `ex` &`other`) const override

*Add two indexed expressions.*
- `ex scalar_mul_indexed` (const `ex` &`self`, const `numeric` &`other`) const override

*Multiply an indexed expression with a scalar.*
- `bool contract_with` (`exvector::iterator` `self`, `exvector::iterator` `other`, `exvector` &`v`) const override

*Try to contract two indexed expressions that appear in the same product.*
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- const `T` \* `operator->` () const
- `T` & `get_struct` ()
- const `T` & `get_struct` () const

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()

*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)

*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const

*Create a clone of this object on the heap.*
- virtual `ex eval` () const

*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const

- Evaluate object numerically.*

  - virtual `ex evalm () const`
- Evaluate sums, products and integer powers of matrices.*

  - virtual `ex eval_integ () const`
- Evaluate integrals, if result is known.*

  - virtual `ex eval_indexed (const basic &i) const`
- Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*

  - virtual void `print (const print_context &c, unsigned level=0) const`
- Output to stream.*

  - virtual void `dbgprint () const`
- Little wrapper around print to be called within a debugger.*

  - virtual void `dbgprinttree () const`
- Little wrapper around printtree to be called within a debugger.*

  - virtual unsigned `precedence () const`
- Return relative operator precedence (for parenthesizing output).*

  - virtual bool `info (unsigned inf) const`
- Information about the object.*

  - virtual size\_t `nops () const`
- Number of operands/members.*

  - virtual `ex op (size_t i) const`
- Return operand/member at position i.*

  - virtual `ex operator[] (const ex &index) const`
  - virtual `ex operator[] (size_t i) const`
  - virtual `ex &let_op (size_t i)`
- Return modifiable operand/member at position i.*

  - virtual `ex &operator[] (const ex &index)`
  - virtual `ex &operator[] (size_t i)`
- Test for occurrence of a pattern.*

  - virtual bool `has (const ex &other, unsigned options=0) const`
- Check whether the expression matches a given pattern.*

  - virtual bool `match (const ex &pattern, exmap &repls) const`
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map (map_function &f) const`
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept (GiNaC::visitor &v) const`
  - virtual bool `is_polynomial (const ex &var) const`
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree (const ex &s) const`
- Return degree of highest power in object s.*

  - virtual int `ldegree (const ex &s) const`
- Return degree of lowest power in object s.*

  - virtual `ex coeff (const ex &s, int n=1) const`
- Return coefficient of degree n in object s.*

  - virtual `ex expand (unsigned options=0) const`
- Expand expression, i.e.*

  - virtual `ex collect (const ex &s, bool distributed=false) const`
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series (const relational &r, int order, unsigned options=0) const`
- Default implementation of ex::series().*

  - virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`

- *Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &`repl`) const
- *Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &`repl`) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &`xi`) const
- *Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const
- *Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const
- *Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &`self`, const `ex` &`other`) const
- *Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &`self`, const `numeric` &`other`) const
- *Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` `self`, `exvector::iterator` `other`, `exvector` &`v`) const
- *Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
- void `print_dispatch` (const `print_context` &`c`, unsigned `level`) const
- *Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &`ri`, const `print_context` &`c`, unsigned `level`) const
- *Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &`n`) const
- *Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &`n`, `lst` &`syms`)
- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &`m`, unsigned `options`) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &`s`, unsigned `nth=1`) const
- *Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &`other`) const
- *Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &`other`) const
- *Test for syntactic equality.*
- const `basic` & `hold` () const
- *Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned `f`) const
- *Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned `f`) const
- *Clear some `status_flags`.*

#### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &`v`) const override
- bool `match_same_type` (const `basic` &`other`) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex derivative` (const `symbol` &`s`) const override  
*Default implementation of `ex::diff()`.*
- bool `is_equal_same_type` (const `basic` &`other`) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &`v`) const
- virtual bool `match_same_type` (const `basic` &`other`) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &`s`) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &`other`) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &`other`) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &`c`, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &`c`, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &`c`, unsigned level) const  
*Python parsable output to stream.*

## Static Private Member Functions

- static const char \* `get_class_name` ()

## Private Attributes

- T `obj`

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 6.160.1 Detailed Description

```
template<class T, template< class > class ComparisonPolicy>
class GiNaC::structure< T, ComparisonPolicy >
```

Wrapper template for making [GiNaC](#) classes out of C++ structures.

## 6.160.2 Constructor & Destructor Documentation

### 6.160.2.1 structure()

```
template<class T , template< class > class ComparisonPolicy>
GiNaC::structure< T, CP >::structure (
    const T & t ) [inline]
```

Construct structure as a copy of a given C++ structure.

Default constructor.

## 6.160.3 Member Function Documentation

### 6.160.3.1 get\_class\_name()

```
template<class T , template< class > class ComparisonPolicy>
static const char * GiNaC::structure< T, ComparisonPolicy >::get_class_name ( ) [inline],
[static], [private]
```

### 6.160.3.2 eval()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval ( ) const [inline], [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

**6.160.3.3 evalm()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::evalm ( ) const [inline], [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

**6.160.3.4 eval\_ncmul()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval_ncmul (
    const exvector & v ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::hold\\_ncmul\(\)](#).

**6.160.3.5 eval\_indexed()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval_indexed (
    const basic & i ) const [inline], [override], [virtual]
```

Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

**6.160.3.6 print()**

```
template<class T , template< class > class ComparisonPolicy>
void GiNaC::structure< T, ComparisonPolicy >::print (
    const print_context & c,
    unsigned level = 0 ) const [inline], [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

**Parameters**

<i>c</i>	print context object that describes the output formatting
<i>level</i>	value that is used to identify the precedence or indentation level for placing parentheses and formatting



Reimplemented from [GiNaC::basic](#).

References [c](#).

### 6.160.3.7 precedence()

```
template<class T , template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::precedence ( ) const [inline], [override],
[virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

### 6.160.3.8 info()

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::info (
    unsigned inf ) const [inline], [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

### 6.160.3.9 nops()

```
template<class T , template< class > class ComparisonPolicy>
size_t GiNaC::structure< T, ComparisonPolicy >::nops ( ) const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.160.3.10 op()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::op (
    size_t i ) const [inline], [override], [virtual]
```

Return operand/member at position *i*.

Reimplemented from [GiNaC::basic](#).

**6.160.3.11 operator[]()** [1/4]

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::operator[] (
    const ex & index ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.12 operator[]()** [2/4]

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::operator[] (
    size_t i ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.13 let\_op()**

```
template<class T , template< class > class ComparisonPolicy>
ex & GiNaC::structure< T, ComparisonPolicy >::let_op (
    size_t i ) [inline], [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

**6.160.3.14 operator[]()** [3/4]

```
template<class T , template< class > class ComparisonPolicy>
ex & GiNaC::structure< T, ComparisonPolicy >::operator[] (
    const ex & index ) [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.15 operator[]()** [4/4]

```
template<class T , template< class > class ComparisonPolicy>
ex & GiNaC::structure< T, ComparisonPolicy >::operator[] (
    size_t i ) [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.16 has()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::has (
    const ex & pattern,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [options](#).

**6.160.3.17 match()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::match (
    const ex & pattern,
    exmap & repl_lst ) const [inline], [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to *repl\_lst*.

Reimplemented from [GiNaC::basic](#).

**6.160.3.18 match\_same\_type()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::match_same_type (
    const basic & other ) const [inline], [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.19 subs()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::subs (
    const exmap & m,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [m](#), and [options](#).

**6.160.3.20 map()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::map (
    map_function & f ) const [inline], [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

**6.160.3.21 degree()**

```
template<class T , template< class > class ComparisonPolicy>
int GiNaC::structure< T, ComparisonPolicy >::degree (
    const ex & s ) const [inline], [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

**6.160.3.22 ldegree()**

```
template<class T , template< class > class ComparisonPolicy>
int GiNaC::structure< T, ComparisonPolicy >::ldegree (
    const ex & s ) const [inline], [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

**6.160.3.23 coeff()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::coeff (
    const ex & s,
    int n = 1 ) const [inline], [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [n](#).

**6.160.3.24 expand()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::expand (
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [options](#).

**6.160.3.25 collect()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::collect (
    const ex & s,
    bool distributed = false ) const [inline], [override], [virtual]
```

Sort expanded expression in terms of powers of some object(s).

**Parameters**

<i>s</i>	object(s) to sort in
<i>distributed</i>	recursive or distributed form (only used when s is a list)

Reimplemented from [GiNaC::basic](#).

**6.160.3.26 derivative()**

```
template<class T , template< class > class ComparisonPolicy>
```

```
ex GiNaC::structure< T, ComparisonPolicy >::derivative (
    const symbol & s ) const [inline], [override], [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

### 6.160.3.27 series()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [options](#), [order](#), and [r](#).

### 6.160.3.28 normal()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [inline], [override], [virtual]
```

Default implementation of [ex::normal\(\)](#).

It normalizes the children and replaces the object with a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.29 to\_rational()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::to_rational (
    exmap & repl ) const [inline], [override], [virtual]
```

Default implementation of [ex::to\\_rational\(\)](#).

This replaces the object with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

**6.160.3.30 to\_polynomial()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::to_polynomial (
    exmap & repl ) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.31 integer\_content()**

```
template<class T , template< class > class ComparisonPolicy>
numeric GiNaC::structure< T, ComparisonPolicy >::integer_content ( ) const [inline], [override],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.32 smod()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::smod (
    const numeric & xi ) const [inline], [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

**Parameters**

$x_i$	modulus
-------	---------

**Returns**

mapped polynomial

**See also**

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.33 max\_coefficient()**

```
template<class T , template< class > class ComparisonPolicy>
numeric GiNaC::structure< T, ComparisonPolicy >::max_coefficient ( ) const [inline], [override],
[virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

**See also**

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.34 get\_free\_indices()**

```
template<class T , template< class > class ComparisonPolicy>
exvector GiNaC::structure< T, ComparisonPolicy >::get_free_indices ( ) const [inline], [override],
[virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

**6.160.3.35 add\_indexed()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::add_indexed (
    const ex & self,
    const ex & other ) const [inline], [override], [virtual]
```

Add two indexed expressions.

They are guaranteed to be of class indexed (or a subclass) and their indices are compatible. This function is used internally by [simplify\\_indexed\(\)](#).



## Parameters

<i>self</i>	First indexed expression; its base object is *this
<i>other</i>	Second indexed expression

## Returns

sum of self and other

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.36 scalar\_mul\_indexed()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::scalar_mul_indexed (
    const ex & self,
    const numeric & other ) const [inline], [override], [virtual]
```

Multiply an indexed expression with a scalar.

This function is used internally by [simplify\\_indexed\(\)](#).

## Parameters

<i>self</i>	Indexed expression; its base object is *this
<i>other</i>	Numeric value

## Returns

product of self and other

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**6.160.3.37 contract\_with()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [inline], [override], [virtual]
```

Try to contract two indexed expressions that appear in the same product.

If a contraction exists, the function overwrites one or both of the expressions and returns true. Otherwise it returns false. It is guaranteed that both expressions are of class indexed (or a subclass) and that at least one dummy index has been found. This functions is used internally by [simplify\\_indexed\(\)](#).

**Parameters**

<i>self</i>	Pointer to first indexed expression; its base object is *this
<i>other</i>	Pointer to second indexed expression
<i>v</i>	The complete vector of factors

**Returns**

true if the contraction was successful, false otherwise

**See also**

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.38 return\_type()**

```
template<class T , template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::return_type ( ) const [inline], [override],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

**6.160.3.39 return\_type\_tinfo()**

```
template<class T , template< class > class ComparisonPolicy>
return_type_t GiNaC::structure< T, ComparisonPolicy >::return_type_tinfo ( ) const [inline],
[override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [r](#).

**6.160.3.40 is\_equal\_same\_type()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::is_equal_same_type (
    const basic & other ) const [inline], [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::structure< T, ComparisonPolicy >::obj](#).

**6.160.3.41 calchash()**

```
template<class T , template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::calchash ( ) const [inline], [override],
[protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

**6.160.3.42 operator->()**

```
template<class T , template< class > class ComparisonPolicy>
const T * GiNaC::structure< T, ComparisonPolicy >::operator-> ( ) const [inline]
```

References [GiNaC::structure< T, ComparisonPolicy >::obj](#).

**6.160.3.43 get\_struct() [1/2]**

```
template<class T , template< class > class ComparisonPolicy>
T & GiNaC::structure< T, ComparisonPolicy >::get_struct ( ) [inline]
```

References [GiNaC::structure< T, ComparisonPolicy >::obj](#).

#### 6.160.3.44 `get_struct()` [2/2]

```
template<class T , template< class > class ComparisonPolicy>
const T & GiNaC::structure< T, ComparisonPolicy >::get_struct ( ) const [inline]
```

References [GiNaC::structure< T, ComparisonPolicy >::obj](#).

### 6.160.4 Member Data Documentation

#### 6.160.4.1 `obj`

```
template<class T , template< class > class ComparisonPolicy>
T GiNaC::structure< T, ComparisonPolicy >::obj [private]
```

Referenced by [GiNaC::structure< T, ComparisonPolicy >::get\\_struct\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::is\\_equal\\_same\\_t](#) and [GiNaC::structure< T, ComparisonPolicy >::operator->\(\)](#).

The documentation for this class was generated from the following file:

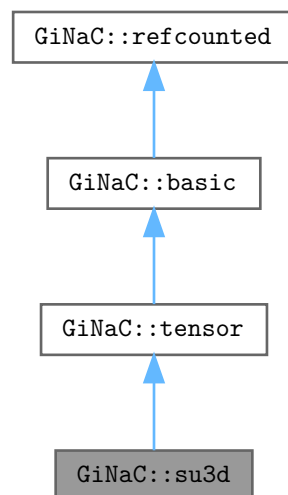
- [structure.h](#)

## 6.161 GiNaC::su3d Class Reference

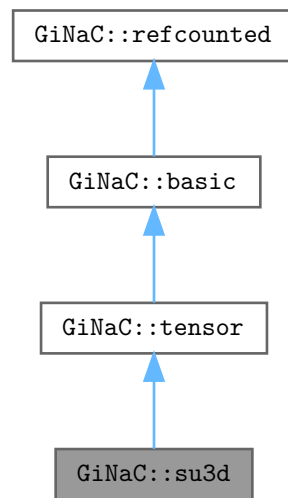
This class represents the tensor of symmetric su(3) structure constants.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3d:



Collaboration diagram for GiNaC::su3d:



## Public Member Functions

- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of indexed symmetric structure constant.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed symmetric structure constant with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*

- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual void `dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprintree () const`  
*Little wrapper around printree to be called within a debugger.*
- virtual unsigned `precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf) const`  
*Information about the object.*
- virtual size\_t `nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v) const`
- virtual bool `is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0) const`  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0) const`  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*

- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

#### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::tensor](#)

- unsigned [return\\_type](#) () const override

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.161.1 Detailed Description

This class represents the tensor of symmetric su(3) structure constants.



## 6.161.2 Member Function Documentation

### 6.161.2.1 eval\_indexed()

```
ex GiNaC::su3d::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of indexed symmetric structure constant.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_6](#), [CMPINDICES](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::sqrt\(\)](#), and [std::swap\(\)](#).

### 6.161.2.2 contract\_with()

```
bool GiNaC::su3d::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed symmetric structure constant with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_T\(\)](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::op\(\)](#), and [GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#).

### 6.161.2.3 return\_type()

```
unsigned GiNaC::su3d::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 6.161.2.4 do\_print()

```
void GiNaC::su3d::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.161.2.5 do\_print\_latex()

```
void GiNaC::su3d::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

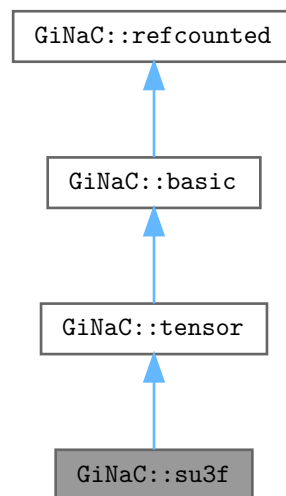
- [color.h](#)
- [color.cpp](#)

## 6.162 GiNaC::su3f Class Reference

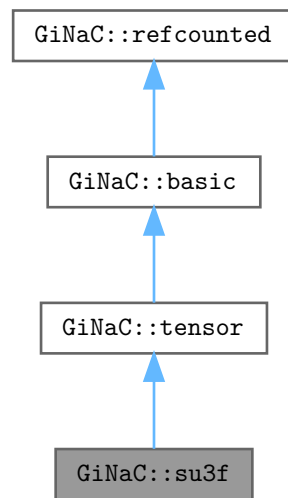
This class represents the tensor of antisymmetric su(3) structure constants.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3f:



Collaboration diagram for GiNaC::su3f:



## Public Member Functions

- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of indexed antisymmetric structure constant.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed antisymmetric structure constant with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- `const basic & operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate` () const  
*Create a clone of this object on the heap.*
- `virtual ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf` () const  
*Evaluate object numerically.*

- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprintree ()` const  
*Little wrapper around printree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*

- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

#### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- unsigned `return_type` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.162.1 Detailed Description

This class represents the tensor of antisymmetric  $\text{su}(3)$  structure constants.

## 6.162.2 Member Function Documentation

### 6.162.2.1 eval\_indexed()

```
ex GiNaC::su3f::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of indexed antisymmetric structure constant.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex\\_1\\_2](#), [CMPINDICES](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::sqrt\(\)](#), and [std::swap\(\)](#).

### 6.162.2.2 contract\_with()

```
bool GiNaC::su3f::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed antisymmetric structure constant with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_T\(\)](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::I](#), [GiNaC::ex::op\(\)](#), and [GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#).

### 6.162.2.3 return\_type()

```
unsigned GiNaC::su3f::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 6.162.2.4 do\_print()

```
void GiNaC::su3f::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.162.2.5 do\_print\_latex()

```
void GiNaC::su3f::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

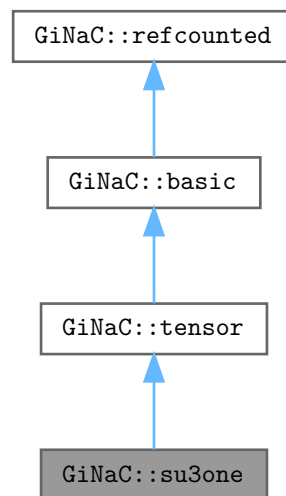
- [color.h](#)
- [color.cpp](#)

## 6.163 GiNaC::su3one Class Reference

This class represents the su(3) unity element.

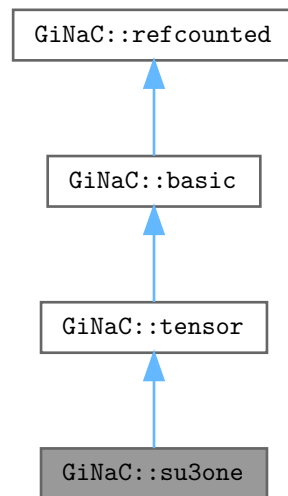
```
#include <color.h>
```

Inheritance diagram for GiNaC::su3one:





Collaboration diagram for GiNaC::su3one:



## Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Public Member Functions inherited from [GiNaC::tensor](#)

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthesizing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*

- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*

- `ex subs_one_level` (const `exmap` &`m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &`s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &`other`) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &`other`) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- `const basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 6.163.1 Detailed Description

This class represents the  $su(3)$  unity element.

## 6.163.2 Member Function Documentation

### 6.163.2.1 `do_print()`

```
void GiNaC::su3one::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.163.2.2 do\_print\_latex()

```
void GiNaC::su3one::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following file:

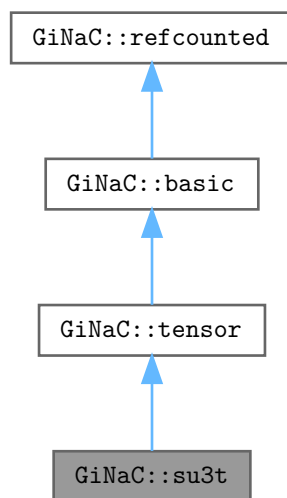
- [color.h](#)

## 6.164 GiNaC::su3t Class Reference

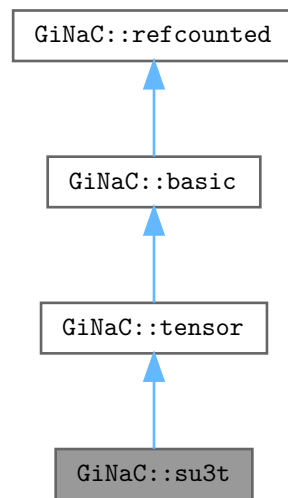
This class represents an  $su(3)$  generator.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3t:



Collaboration diagram for `GiNaC::su3t`:



## Public Member Functions

- bool `contract_with` (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of generator with something else.*

## Public Member Functions inherited from [GiNaC::tensor](#)

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*

- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex &let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex &operator[] (const ex &index)`
- virtual `ex &operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept



## Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.164.1 Detailed Description

This class represents an  $su(3)$  generator.

### 6.164.2 Member Function Documentation

### 6.164.2.1 contract\_with()

```
bool GiNaC::su3t::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of generator with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_ONE\(\)](#), [GiNaC::color\\_trace\(\)](#), and [GINAC\\_ASSERT](#).

### 6.164.2.2 do\_print()

```
void GiNaC::su3t::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

### 6.164.2.3 do\_print\_latex()

```
void GiNaC::su3t::do_print_latex (
    const print_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

- [color.h](#)
- [color.cpp](#)

## 6.165 GiNaC::subs\_options Class Reference

Flags to control the behavior of [subs\(\)](#).

```
#include <flags.h>
```

### Public Types

- enum {  
[no\\_pattern](#) = 0x0001 , [subs\\_no\\_pattern](#) = 0x0001 , [algebraic](#) = 0x0002 , [subs\\_algebraic](#) = 0x0002 ,  
[pattern\\_is\\_product](#) = 0x0004 , [pattern\\_is\\_not\\_product](#) = 0x0008 , [no\\_index\\_renaming](#) = 0x0010 ,  
[really\\_subs\\_idx](#) = 0x0020 }

### 6.165.1 Detailed Description

Flags to control the behavior of [subs\(\)](#).

### 6.165.2 Member Enumeration Documentation

#### 6.165.2.1 anonymous enum

```
anonymous enum
```

## Enumerator

no_pattern	disable pattern matching
subs_no_pattern	
algebraic	enable algebraic substitutions
subs_algebraic	
pattern_is_product	used internally by <a href="#">expairseq::subschildren()</a>
pattern_is_not_product	used internally by <a href="#">expairseq::subschildren()</a>
no_index_renaming	
really_subs_idx	

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.166 GiNaC::sy\_is\_less Class Reference

### Public Member Functions

- [sy\\_is\\_less](#) (exvector::iterator v\_)
- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

### Private Attributes

- exvector::iterator [v](#)

### 6.166.1 Constructor & Destructor Documentation

#### 6.166.1.1 sy\_is\_less()

```
GiNaC::sy_is_less::sy_is_less (
    exvector::iterator v_ ) [inline]
```

### 6.166.2 Member Function Documentation

#### 6.166.2.1 operator>()

```
bool GiNaC::sy_is_less::operator() (
    const ex & lh,
    const ex & rh ) const [inline]
```

References [GINAC\\_ASSERT](#), and [v](#).

## 6.166.3 Member Data Documentation

### 6.166.3.1 v

`exvector::iterator GiNaC::sy_is_less::v` [private]

Referenced by [operator\(\)\(\)](#).

The documentation for this class was generated from the following file:

- [symmetry.cpp](#)

## 6.167 GiNaC::sy\_swap Class Reference

### Public Member Functions

- [sy\\_swap](#) (`exvector::iterator v_`, `bool &s`)
- `void` [operator\(\)](#) (`const` [ex](#) &lh, `const` [ex](#) &rh)

### Public Attributes

- `bool` & [swapped](#)

### Private Attributes

- `exvector::iterator` [v](#)

## 6.167.1 Constructor & Destructor Documentation

### 6.167.1.1 sy\_swap()

```
GiNaC::sy_swap::sy_swap (  
    exvector::iterator v_,  
    bool & s ) [inline]
```

## 6.167.2 Member Function Documentation

### 6.167.2.1 operator>()

```
void GiNaC::sy_swap::operator() (
    const ex & lh,
    const ex & rh ) [inline]
```

References [GINAC\\_ASSERT](#), [swapped](#), and [v](#).

## 6.167.3 Member Data Documentation

### 6.167.3.1 v

```
exvector::iterator GiNaC::sy_swap::v [private]
```

Referenced by [operator>\(\)](#).

### 6.167.3.2 swapped

```
bool& GiNaC::sy_swap::swapped
```

Referenced by [operator>\(\)](#).

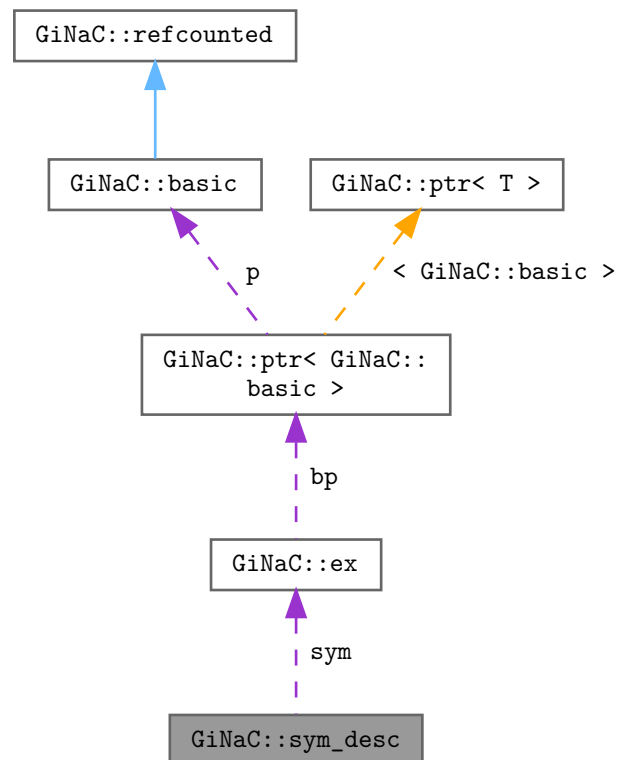
The documentation for this class was generated from the following file:

- [symmetry.cpp](#)

## 6.168 GiNaC::sym\_desc Struct Reference

This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".

Collaboration diagram for `GiNaC::sym_desc`:



## Public Member Functions

- `sym_desc` (const `ex` &s)  
*Initialize symbol, leave other variables uninitialized.*
- bool `operator<` (const `sym_desc` &x) const  
*Comparison operator for sorting.*

## Public Attributes

- `ex sym`  
*Reference to symbol.*
- int `deg_a`  
*Highest degree of symbol in polynomial "a".*
- int `deg_b`  
*Highest degree of symbol in polynomial "b".*
- int `ldeg_a`  
*Lowest degree of symbol in polynomial "a".*
- int `ldeg_b`  
*Lowest degree of symbol in polynomial "b".*

- int [max\\_deg](#)  
*Maximum of deg\_a and deg\_b (Used for sorting)*
- size\_t [max\\_lcnops](#)  
*Maximum number of terms of leading coefficient of symbol in both polynomials.*

### 6.168.1 Detailed Description

This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".

A vector of these structures with information about all symbols in two polynomials can be created with the function [get\\_symbol\\_stats\(\)](#).

See also

[get\\_symbol\\_stats](#)

### 6.168.2 Constructor & Destructor Documentation

#### 6.168.2.1 sym\_desc()

```
GiNaC::sym_desc::sym_desc (
    const ex & s ) [inline]
```

Initialize symbol, leave other variables uninitialized.

### 6.168.3 Member Function Documentation

#### 6.168.3.1 operator<()

```
bool GiNaC::sym_desc::operator< (
    const sym\_desc & x ) const [inline]
```

Comparison operator for sorting.

References [max\\_deg](#), [max\\_lcnops](#), and [x](#).

### 6.168.4 Member Data Documentation

#### 6.168.4.1 sym

```
ex GiNaC::sym_desc::sym
```

Reference to symbol.

#### 6.168.4.2 deg\_a

```
int GiNaC::sym_desc::deg_a
```

Highest degree of symbol in polynomial "a".

#### 6.168.4.3 deg\_b

```
int GiNaC::sym_desc::deg_b
```

Highest degree of symbol in polynomial "b".

#### 6.168.4.4 ldeg\_a

```
int GiNaC::sym_desc::ldeg_a
```

Lowest degree of symbol in polynomial "a".

#### 6.168.4.5 ldeg\_b

```
int GiNaC::sym_desc::ldeg_b
```

Lowest degree of symbol in polynomial "b".

#### 6.168.4.6 max\_deg

```
int GiNaC::sym_desc::max_deg
```

Maximum of deg\_a and deg\_b (Used for sorting)

Referenced by [operator<\(\)](#).



#### 6.168.4.7 max\_lcnops

```
size_t GiNaC::sym_desc::max_lcnops
```

Maximum number of terms of leading coefficient of symbol in both polynomials.

Referenced by [operator<\(\)](#).

The documentation for this struct was generated from the following file:

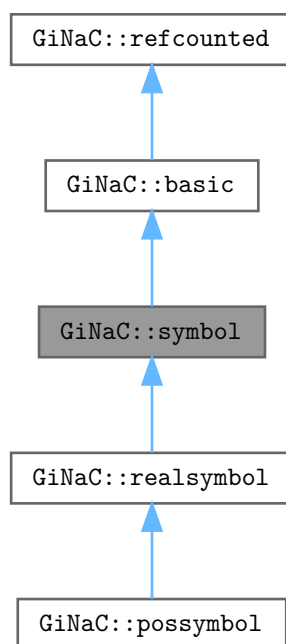
- [normal.cpp](#)

## 6.169 GiNaC::symbol Class Reference

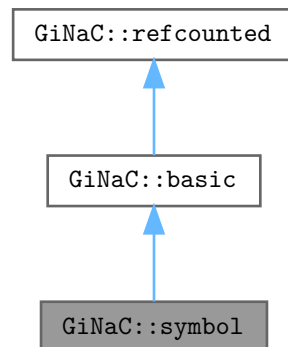
Basic CAS symbol.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::symbol:



Collaboration diagram for GiNaC::symbol:



## Public Member Functions

- [symbol](#) (const std::string &initname)
- [symbol](#) (const std::string &initname, const std::string &texname)
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex series](#) (const [relational](#) &s, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series\(\)](#) for symbols.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override  
*Implementation of [ex::normal\(\)](#) for symbols.*
- [ex to\\_rational](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_rational\(\)](#) for symbols.*
- [ex to\\_polynomial](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_polynomial\(\)](#) for symbols.*
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_polynomial](#) (const [ex](#) &var) const override  
*Check whether this is a polynomial in the given variables.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- virtual unsigned [get\\_domain](#) () const
- void [set\\_name](#) (const std::string &n)
- void [set\\_TeX\\_name](#) (const std::string &n)
- std::string [get\\_name](#) () const
- std::string [get\\_TeX\\_name](#) () const

Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for single differentiation of a symbol.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex\\_derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- unsigned [serial](#)  
*unique serial number for comparison*
- std::string [name](#)  
*printname of this symbol*
- std::string [TeX\\_name](#)  
*LaTeX name of this symbol.*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## Static Private Attributes

- static unsigned [next\\_serial](#) = 0

### 6.169.1 Detailed Description

Basic CAS symbol.

It has a name because it must know how to output itself.

### 6.169.2 Constructor & Destructor Documentation

#### 6.169.2.1 [symbol\(\)](#) [1/2]

```
GiNaC::symbol::symbol (
    const std::string & initname ) [explicit]
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GiNaC::basic::setflag\(\)](#).

#### 6.169.2.2 [symbol\(\)](#) [2/2]

```
GiNaC::symbol::symbol (
    const std::string & initname,
    const std::string & texname )
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GiNaC::basic::setflag\(\)](#).

## 6.169.3 Member Function Documentation

### 6.169.3.1 info()

```
bool GiNaC::symbol::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::expanded](#), [get\\_domain\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::domain::positive](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::domain::real](#), [GiNaC::info\\_flags::real](#), and [GiNaC::info\\_flags::symbol](#).

Referenced by [GiNaC::conjugate\\_expl\\_derivative\(\)](#), [GiNaC::imag\\_part\\_expl\\_derivative\(\)](#), and [GiNaC::real\\_part\\_expl\\_derivative\(\)](#).

### 6.169.3.2 eval()

```
ex GiNaC::symbol::eval ( ) const [inline], [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.3 evalf()

```
ex GiNaC::symbol::evalf ( ) const [inline], [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

#### 6.169.3.4 series()

```
ex GiNaC::symbol::series (
    const relational & r,
    int order,
    unsigned options = 0 ) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for symbols.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GINAC\\_ASSERT](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [order](#), and [r](#).

#### 6.169.3.5 subs()

```
ex GiNaC::symbol::subs (
    const exmap & m,
    unsigned options = 0 ) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [m](#), [options](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

#### 6.169.3.6 normal()

```
ex GiNaC::symbol::normal (
    exmap & repl,
    exmap & rev_lookup,
    lst & modifier ) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for symbols.

This returns the unmodified symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#).



### 6.169.3.7 to\_rational()

```
ex GiNaC::symbol::to_rational (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for symbols.

This returns the unmodified symbol.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.8 to\_polynomial()

```
ex GiNaC::symbol::to_polynomial (
    exmap & repl ) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for symbols.

This returns the unmodified symbol.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.9 conjugate()

```
ex GiNaC::symbol::conjugate ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::realsymbol](#).

### 6.169.3.10 real\_part()

```
ex GiNaC::symbol::real_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::realsymbol](#).

### 6.169.3.11 imag\_part()

```
ex GiNaC::symbol::imag_part ( ) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::realsymbol](#).

### 6.169.3.12 is\_polynomial()

```
bool GiNaC::symbol::is_polynomial (
    const ex & var ) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.13 archive()

```
void GiNaC::symbol::archive (
    archive\_node & n ) const [override], [virtual]
```

Save (a.k.a.

Archive the object.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), [name](#), and [TeX\\_name](#).

### 6.169.3.14 read\_archive()

```
void GiNaC::symbol::read_archive (
    const archive\_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

Read object from [archive\\_node](#).

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container< C >::append\(\)](#), [GiNaC::status\\_flags::dynallocated](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [n](#), [name](#), [next\\_serial](#), [serial](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

### 6.169.3.15 derivative()

```
ex GiNaC::symbol::derivative (
    const symbol & s ) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for single differentiation of a symbol.

It returns 1 or 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), and [GiNaC::basic::compare\\_same\\_type\(\)](#).

### 6.169.3.16 is\_equal\_same\_type()

```
bool GiNaC::symbol::is_equal_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [serial](#).

Referenced by [series\(\)](#).

### 6.169.3.17 calchash()

```
unsigned GiNaC::symbol::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

#### 6.169.3.18 `get_domain()`

```
virtual unsigned GiNaC::symbol::get_domain ( ) const [inline], [virtual]
```

Reimplemented in [GiNaC::realsymbol](#), and [GiNaC::possymbol](#).

References [GiNaC::domain::complex](#).

Referenced by [do\\_print\\_tree\(\)](#), and [info\(\)](#).

#### 6.169.3.19 `set_name()`

```
void GiNaC::symbol::set_name (
    const std::string & n ) [inline]
```

References [n](#), and [name](#).

#### 6.169.3.20 `set_TeX_name()`

```
void GiNaC::symbol::set_TeX_name (
    const std::string & n ) [inline]
```

References [n](#), and [TeX\\_name](#).

#### 6.169.3.21 `get_name()`

```
std::string GiNaC::symbol::get_name ( ) const
```

References [name](#), and [serial](#).

Referenced by [do\\_print\(\)](#), and [get\\_TeX\\_name\(\)](#).

#### 6.169.3.22 `get_TeX_name()`

```
std::string GiNaC::symbol::get_TeX_name ( ) const
```

References [GiNaC::get\\_default\\_TeX\\_name\(\)](#), [get\\_name\(\)](#), and [TeX\\_name](#).

### 6.169.3.23 do\_print()

```
void GiNaC::symbol::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [get\\_name\(\)](#).

### 6.169.3.24 do\_print\_latex()

```
void GiNaC::symbol::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::get\\_default\\_TeX\\_name\(\)](#), [name](#), [serial](#), and [TeX\\_name](#).

### 6.169.3.25 do\_print\_tree()

```
void GiNaC::symbol::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [get\\_domain\(\)](#), [GiNaC::basic::hashvalue](#), [name](#), and [serial](#).

### 6.169.3.26 do\_print\_python\_repr()

```
void GiNaC::symbol::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), [name](#), [serial](#), and [TeX\\_name](#).

## 6.169.4 Member Data Documentation

### 6.169.4.1 serial

```
unsigned GiNaC::symbol::serial [protected]
```

unique serial number for comparison

Referenced by [calchash\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_name\(\)](#), [is\\_equal\\_same\\_type\(\)](#), and [read\\_archive\(\)](#).

#### 6.169.4.2 name

```
std::string GiNaC::symbol::name [mutable], [protected]
```

printname of this symbol

Referenced by [archive\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_name\(\)](#), [read\\_archive\(\)](#), and [set\\_name\(\)](#).

#### 6.169.4.3 TeX\_name

```
std::string GiNaC::symbol::TeX_name [protected]
```

LaTeX name of this symbol.

Referenced by [archive\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [get\\_TeX\\_name\(\)](#), [read\\_archive\(\)](#), and [set\\_TeX\\_name\(\)](#).

#### 6.169.4.4 next\_serial

```
unsigned GiNaC::symbol::next_serial = 0 [static], [private]
```

Referenced by [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

- [symbol.h](#)
- [normal.cpp](#)
- [pseries.cpp](#)
- [symbol.cpp](#)

## 6.170 GiNaC::symbolset Class Reference

### Public Member Functions

- [symbolset](#) (const [ex](#) &e)
- bool [has](#) (const [ex](#) &e) const

### Private Member Functions

- void [insert\\_symbols](#) (const [ex](#) &e)

## Private Attributes

- [exset s](#)

## 6.170.1 Constructor & Destructor Documentation

### 6.170.1.1 symbolset()

```
GiNaC::symbolset::symbolset (
    const ex & e ) [inline], [explicit]
```

References [insert\\_symbols\(\)](#).

## 6.170.2 Member Function Documentation

### 6.170.2.1 insert\_symbols()

```
void GiNaC::symbolset::insert_symbols (
    const ex & e ) [inline], [private]
```

References [insert\\_symbols\(\)](#), and [s](#).

Referenced by [insert\\_symbols\(\)](#), and [symbolset\(\)](#).

### 6.170.2.2 has()

```
bool GiNaC::symbolset::has (
    const ex & e ) const [inline]
```

References [s](#).

Referenced by [GiNaC::lsolve\(\)](#).

## 6.170.3 Member Data Documentation

### 6.170.3.1 s

```
exset GiNaC::symbolset::s [private]
```

Referenced by [has\(\)](#), and [insert\\_symbols\(\)](#).

The documentation for this class was generated from the following file:

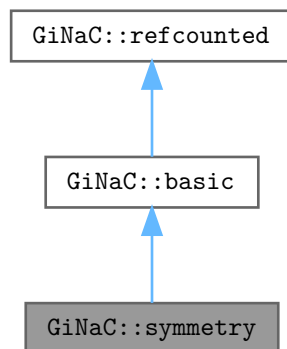
- [inifcns.cpp](#)

## 6.171 GiNaC::symmetry Class Reference

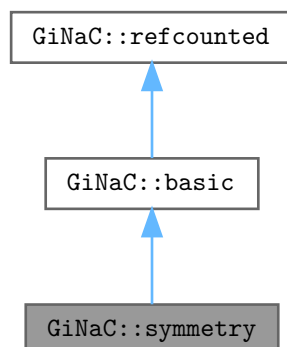
This class describes the symmetry of a group of indices.

```
#include <symmetry.h>
```

Inheritance diagram for GiNaC::symmetry:



Collaboration diagram for GiNaC::symmetry:





## Public Types

- enum `symmetry_type` { `none` , `symmetric` , `antisymmetric` , `cyclic` }
- Type of symmetry.*

## Public Member Functions

- `symmetry` (unsigned i)  
*Create leaf node that represents one index.*
- `symmetry` (`symmetry_type` t, const `symmetry` &c1, const `symmetry` &c2)  
*Create node with two children.*
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*
- `symmetry_type` `get_type` () const  
*Get symmetry type.*
- void `set_type` (`symmetry_type` t)  
*Set symmetry type.*
- `symmetry` & `add` (const `symmetry` &c)  
*Add child node, check index sets for consistency.*
- void `validate` (unsigned n)  
*Verify that all indices of this node are in the range [0..n-1].*
- bool `has_symmetry` () const  
*Check whether this node actually represents any kind of symmetry.*
- bool `has_nonsymmetric` () const  
*Check whether this node involves anything non symmetric.*
- bool `has_cyclic` () const  
*Check whether this node involves a cyclic symmetry.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex` `eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex` `evalf` () const  
*Evaluate object numerically.*
- virtual `ex` `evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex` `eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex` `eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const

- *Output to stream.*  
virtual void `dbgprint` () const
- *Little wrapper around print to be called within a debugger.*  
virtual void `dbgprinttree` () const
- *Little wrapper around printtree to be called within a debugger.*  
virtual unsigned `precedence` () const
- *Return relative operator precedence (for parenthezing output).*  
virtual bool `info` (unsigned inf) const
- *Information about the object.*  
virtual size\_t `nops` () const
- *Number of operands/members.*  
virtual `ex op` (size\_t i) const
- *Return operand/member at position i.*  
virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

- Implementation *ex::max\_coefficient()*.
- virtual [exvector get\\_free\\_indices](#) () const  
Return a vector containing the free indices of an expression.
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
Add two indexed expressions.
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
Multiply an indexed expression with a scalar.
- virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const  
Try to contract two indexed expressions that appear in the same product.
- virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
Like [print\(\)](#), but dispatch to the specified class.
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
Like [print\(\)](#), but dispatch to the specified class.
- virtual void [archive](#) ([archive\\_node](#) &n) const  
Save (serialize) the object into archive node.
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
Load (deserialize) the object from an archive node.
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
Helper function for [subs\(\)](#).
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
Default interface of nth derivative *ex::diff(s, n)*.
- int [compare](#) (const [basic](#) &other) const  
Compare objects syntactically to establish canonical ordering.
- bool [is\\_equal](#) (const [basic](#) &other) const  
Test for syntactic equality.
- const [basic](#) & [hold](#) () const  
Stop further evaluation.
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
Set some *status\_flags*.
- const [basic](#) & [clearflag](#) (unsigned f) const  
Clear some *status\_flags*.

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [calchash](#) () const override  
Compute the hash value of an object and if it makes sense to store it in the objects *status\_flags*, do so.
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Private Attributes

- [symmetry\\_type](#) type  
*Type of symmetry described by this node.*
- std::set< unsigned > [indices](#)  
*Sorted union set of all indices handled by this node.*
- [exvector children](#)  
*Vector of child nodes.*

### Friends

- class [sy\\_is\\_less](#)
- class [sy\\_swap](#)
- int [canonicalize](#) (exvector::iterator v, const [symmetry](#) &symm)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*

### Additional Inherited Members

#### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.171.1 Detailed Description

This class describes the symmetry of a group of indices.

These objects can be grouped into a tree to form complex mixed symmetries.

### 6.171.2 Member Enumeration Documentation

#### 6.171.2.1 symmetry\_type

```
enum GiNaC::symmetry::symmetry_type
```

Type of symmetry.

Enumerator

none	no symmetry properties
symmetric	totally symmetric
antisymmetric	totally antisymmetric
cyclic	cyclic symmetry

### 6.171.3 Constructor & Destructor Documentation

#### 6.171.3.1 symmetry() [1/2]

```
GiNaC::symmetry::symmetry (
    unsigned i )
```

Create leaf node that represents one index.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [indices](#), and [GiNaC::basic::setflag\(\)](#).

#### 6.171.3.2 symmetry() [2/2]

```
GiNaC::symmetry::symmetry (
    symmetry_type t,
    const symmetry & c1,
    const symmetry & c2 )
```

Create node with two children.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GiNaC::basic::setflag\(\)](#).

## 6.171.4 Member Function Documentation

### 6.171.4.1 archive()

```
void GiNaC::symmetry::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

Archive the object.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [children](#), [indices](#), [n](#), and [type](#).

### 6.171.4.2 read\_archive()

```
void GiNaC::symmetry::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

Construct object from [archive\\_node](#).

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [indices](#), [n](#), and [type](#).

### 6.171.4.3 calchash()

```
unsigned GiNaC::symmetry::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [children](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [indices](#), [GiNaC::make\\_hash\\_seed\(\)](#), [none](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::basic::setflag\(\)](#), and [type](#).

#### 6.171.4.4 `get_type()`

```
symmetry_type GiNaC::symmetry::get_type ( ) const [inline]
```

Get symmetry type.

References [type](#).

#### 6.171.4.5 `set_type()`

```
void GiNaC::symmetry::set_type (
    symmetry_type t ) [inline]
```

Set symmetry type.

References [type](#).

Referenced by [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_cycl\(\)](#), and [GiNaC::sy\\_symm\(\)](#).

#### 6.171.4.6 `add()`

```
symmetry & GiNaC::symmetry::add (
    const symmetry & c )
```

Add child node, check index sets for consistency.

References [c](#), [children](#), [GINAC\\_ASSERT](#), [indices](#), [none](#), and [type](#).

Referenced by [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_cycl\(\)](#), [GiNaC::sy\\_none\(\)](#), and [GiNaC::sy\\_symm\(\)](#).

#### 6.171.4.7 `validate()`

```
void GiNaC::symmetry::validate (
    unsigned n )
```

Verify that all indices of this node are in the range [0..n-1].

This function throws an exception if the verification fails. If the top node has a type != none and no children, add all indices in the range [0..n-1] as children.

References [indices](#), [n](#), [none](#), and [type](#).

#### 6.171.4.8 `has_symmetry()`

```
bool GiNaC::symmetry::has_symmetry ( ) const [inline]
```

Check whether this node actually represents any kind of symmetry.

References [children](#), [none](#), and [type](#).

#### 6.171.4.9 `has_nonsymmetric()`

```
bool GiNaC::symmetry::has_nonsymmetric ( ) const
```

Check whether this node involves anything non symmetric.

References [antisymmetric](#), [children](#), [cyclic](#), [has\\_nonsymmetric\(\)](#), and [type](#).

Referenced by [has\\_nonsymmetric\(\)](#).

#### 6.171.4.10 `has_cyclic()`

```
bool GiNaC::symmetry::has_cyclic ( ) const
```

Check whether this node involves a cyclic symmetry.

References [children](#), [cyclic](#), [has\\_cyclic\(\)](#), and [type](#).

Referenced by [has\\_cyclic\(\)](#).

#### 6.171.4.11 `do_print()`

```
void GiNaC::symmetry::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [antisymmetric](#), [c](#), [children](#), [cyclic](#), [indices](#), [none](#), [symmetric](#), and [type](#).

#### 6.171.4.12 `do_print_tree()`

```
void GiNaC::symmetry::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [antisymmetric](#), [c](#), [children](#), [cyclic](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [indices](#), [none](#), [symmetric](#), and [type](#).



## 6.171.5 Friends And Related Function Documentation

### 6.171.5.1 sy\_is\_less

```
friend class sy_is_less [friend]
```

### 6.171.5.2 sy\_swap

```
friend class sy_swap [friend]
```

### 6.171.5.3 canonicalize

```
int canonicalize (
    exvector::iterator v,
    const symmetry & symm ) [friend]
```

Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.

#### Parameters

<i>v</i>	Start of expression vector
<i>symm</i>	Root node of symmetry tree

#### Returns

the overall sign introduced by the reordering (+1, -1 or 0) or numeric\_limits<int>::max() if nothing changed

## 6.171.6 Member Data Documentation

### 6.171.6.1 type

```
symmetry_type GiNaC::symmetry::type [private]
```

Type of symmetry described by this node.

Referenced by [add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_type\(\)](#), [has\\_cyclic\(\)](#), [has\\_nonsymmetric\(\)](#), [has\\_symmetry\(\)](#), [read\\_archive\(\)](#), [set\\_type\(\)](#), and [validate\(\)](#).

### 6.171.6.2 indices

```
std::set<unsigned> GiNaC::symmetry::indices [private]
```

Sorted union set of all indices handled by this node.

Referenced by [add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [read\\_archive\(\)](#), [symmetry\(\)](#), and [validate\(\)](#).

### 6.171.6.3 children

```
exvector GiNaC::symmetry::children [private]
```

Vector of child nodes.

Referenced by [add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [has\\_cyclic\(\)](#), [has\\_nonsymmetric\(\)](#), and [has\\_symmetry\(\)](#).

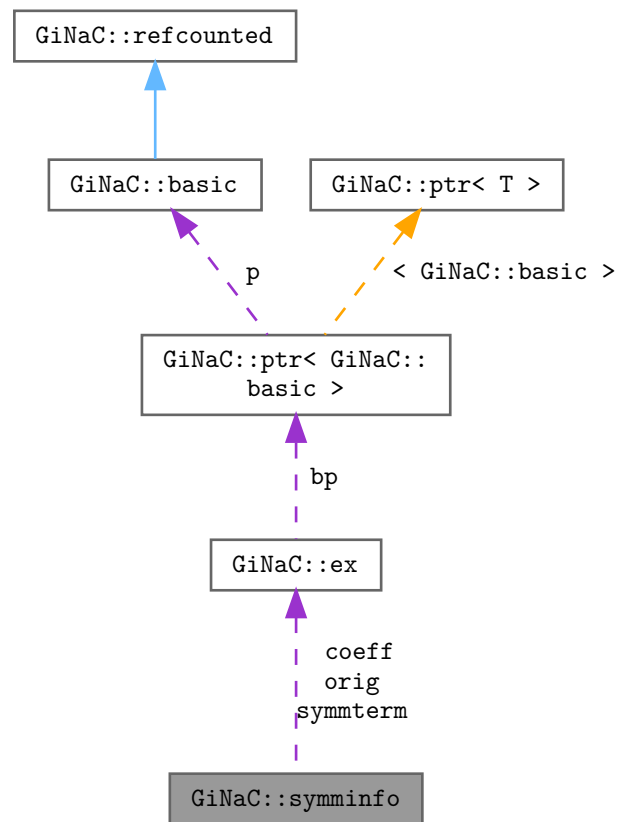
The documentation for this class was generated from the following files:

- [symmetry.h](#)
- [symmetry.cpp](#)

## 6.172 GiNaC::symminfo Class Reference

This structure stores the individual symmetrized terms obtained during the simplification of sums.

Collaboration diagram for GiNaC::symminfo:



## Public Member Functions

- [symminfo](#) ()
- [symminfo](#) (const [ex](#) &symmterm\_, const [ex](#) &orig\_, size\_t num\_)

## Public Attributes

- [ex symmterm](#)  
*symmetrized term*
- [ex coeff](#)  
*coefficient of symmetrized term*
- [ex orig](#)  
*original term*
- [size\\_t num](#)  
*how many symmetrized terms resulted from the original term*

### 6.172.1 Detailed Description

This structure stores the individual symmetrized terms obtained during the simplification of sums.

### 6.172.2 Constructor & Destructor Documentation

#### 6.172.2.1 `symminfo()` [1/2]

```
GiNaC::symminfo::symminfo ( ) [inline]
```

#### 6.172.2.2 `symminfo()` [2/2]

```
GiNaC::symminfo::symminfo (
    const ex & symmterm_,
    const ex & orig_,
    size_t num_ ) [inline]
```

References [coeff](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [symmterm](#).

### 6.172.3 Member Data Documentation

#### 6.172.3.1 `symmterm`

```
ex GiNaC::symminfo::symmterm
```

symmetrized term

Referenced by [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm::operator\(\)\(\)](#), and [symminfo\(\)](#).

#### 6.172.3.2 `coeff`

```
ex GiNaC::symminfo::coeff
```

coefficient of symmetrized term

Referenced by [symminfo\(\)](#).

### 6.172.3.3 orig

`ex` `GiNaC::symminfo::orig`

original term

Referenced by [GiNaC::symminfo\\_is\\_less\\_by\\_orig::operator\(\)\(\)](#).

### 6.172.3.4 num

`size_t` `GiNaC::symminfo::num`

how many symmetrized terms resulted from the original term

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.173 GiNaC::symminfo\_is\_less\_by\_orig Class Reference

### Public Member Functions

- `bool` [operator\(\)](#) (const [symminfo](#) &si1, const [symminfo](#) &si2) const

### 6.173.1 Member Function Documentation

#### 6.173.1.1 operator()()

```
bool GiNaC::symminfo_is_less_by_orig::operator() (
    const symminfo & si1,
    const symminfo & si2 ) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::symminfo::orig](#).

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.174 GiNaC::symminfo\_is\_less\_by\_symmterm Class Reference

### Public Member Functions

- `bool` [operator\(\)](#) (const [symminfo](#) &si1, const [symminfo](#) &si2) const

## 6.174.1 Member Function Documentation

### 6.174.1.1 `operator()()`

```
bool GiNaC::symminfo_is_less_by_symmterm::operator() (
    const symminfo & si1,
    const symminfo & si2 ) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::symminfo::symmterm](#).

The documentation for this class was generated from the following file:

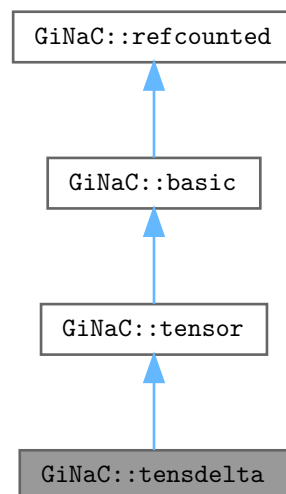
- [indexed.cpp](#)

## 6.175 GiNaC::tensdelta Class Reference

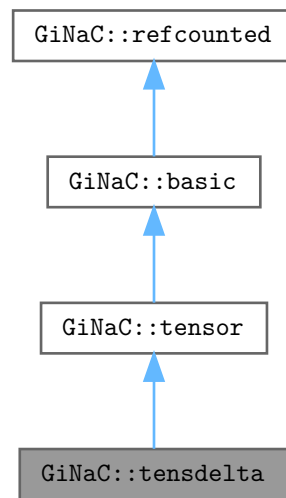
This class represents the delta tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensdelta:



Collaboration diagram for GiNaC::tensdelta:



## Public Member Functions

- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed delta tensor.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed delta tensor with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*

- virtual `ex evalf ()` const  
*Evaluate object numerically.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthesizing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*



- virtual `ex normal` (`exmap &repl`, `exmap &rev_lookup`, `lst &modifier`) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap &repl`) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap &repl`) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric &xi`) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex &self`, const `ex &other`) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex &self`, const `numeric &other`) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator self`, `exvector::iterator other`, `exvector &v`) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context &c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info &ri`, const `print_context &c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node &n`) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node &n`, `lst &syms`)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap &m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol &s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic &other`) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic &other`) const  
*Test for syntactic equality.*
- const `basic & hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- const `basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- unsigned [return\\_type](#) () const override

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

#### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.175.1 Detailed Description

This class represents the delta tensor.

If indexed, it must have exactly two indices of the same type.

### 6.175.2 Member Function Documentation

#### 6.175.2.1 info()

```
bool GiNaC::tensdelta::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

#### 6.175.2.2 eval\_indexed()

```
ex GiNaC::tensdelta::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed delta tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::idx::get\\_dim\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [m](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::ex::subs\(\)](#).

#### 6.175.2.3 contract\_with()

```
bool GiNaC::tensdelta::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed delta tensor with something else.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

#### 6.175.2.4 return\_type()

```
unsigned GiNaC::tensdelta::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::tensor](#).

References [GiNaC::return\\_types::commutative](#).

#### 6.175.2.5 do\_print()

```
void GiNaC::tensdelta::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

#### 6.175.2.6 do\_print\_latex()

```
void GiNaC::tensdelta::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

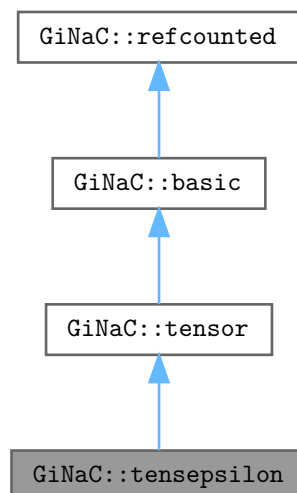
- [tensor.h](#)
- [tensor.cpp](#)

## 6.176 GiNaC::tensepsilon Class Reference

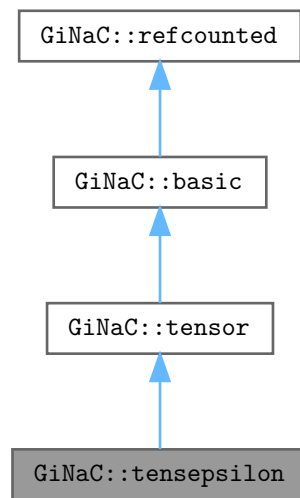
This class represents the totally antisymmetric epsilon tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensepsilon:



Collaboration diagram for GiNaC::tensepsilon:



## Public Member Functions

- [tensepsilon](#) (bool [minkowski](#), bool [pos\\_sig](#))
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval\\_indexed](#) (const [basic](#) &i) const override  
*Automatic symbolic evaluation of an indexed epsilon tensor.*
- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of epsilon tensor with something else.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &[syms](#)) override  
*Read (a.k.a.*

## Public Member Functions inherited from [GiNaC::tensor](#)

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions.*

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)

- basic assignment operator: the other object might be of a derived class.*

  - virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
  - virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
  - virtual `ex evalf () const`  
*Evaluate object numerically.*
  - virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
  - virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
  - virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
  - virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
  - virtual void `dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
  - virtual void `dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
  - virtual unsigned `precedence () const`  
*Return relative operator precedence (for parenthezing output).*
  - virtual bool `info (unsigned inf) const`  
*Information about the object.*
  - virtual size\_t `nops () const`  
*Number of operands/members.*
  - virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
  - virtual `ex operator[] (const ex &index) const`
  - virtual `ex operator[] (size_t i) const`
  - virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
  - virtual `ex & operator[] (const ex &index)`
  - virtual `ex & operator[] (size_t i)`
  - virtual bool `has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
  - virtual bool `match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
  - virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
  - virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
  - virtual void `accept (GiNaC::visitor &v) const`
  - virtual bool `is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
  - virtual int `degree (const ex &s) const`  
*Return degree of highest power in object s.*
  - virtual int `ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
  - virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
  - virtual `ex expand (unsigned options=0) const`

- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const

*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const

*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const

*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const

*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const

*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*
- const `basic` & `hold` () const

*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const

*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const

*Clear some `status_flags`.*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- unsigned [return\\_type](#) () const override

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Private Attributes

- bool [minkowski](#)  
*If true, tensor is in Minkowski-type space.*
- bool [pos\\_sig](#)  
*If true, the metric is assumed to be  $\text{diag}(-1, 1, 1 \dots)$ .*



## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.176.1 Detailed Description

This class represents the totally antisymmetric epsilon tensor.

If indexed, all indices must be of the same type and their number must be equal to the dimension of the index space.

## 6.176.2 Constructor & Destructor Documentation

### 6.176.2.1 tensepsilon()

```
GiNaC::tensepsilon::tensepsilon (
    bool minkowski,
    bool pos_sig )
```

## 6.176.3 Member Function Documentation

### 6.176.3.1 info()

```
bool GiNaC::tensepsilon::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

### 6.176.3.2 eval\_indexed()

```
ex GiNaC::tensepsilon::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed epsilon tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_zero\(\)](#), [minkowski](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::permutation\\_sign\(\)](#), [pos\\_sig](#), and [x](#).

### 6.176.3.3 contract\_with()

```
bool GiNaC::tensepsilon::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of epsilon tensor with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::lorentz\\_g\(\)](#), [GiNaC::metric\\_tensor\(\)](#), [minkowski](#), [pos\\_sig](#), and [GiNaC::ex::simplify\\_indexed\(\)](#).

### 6.176.3.4 archive()

```
void GiNaC::tensepsilon::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [minkowski](#), [n](#), and [pos\\_sig](#).

### 6.176.3.5 read\_archive()

```
void GiNaC::tensepsilon::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [minkowski](#), [n](#), and [pos\\_sig](#).

### 6.176.3.6 return\_type()

```
unsigned GiNaC::tensepsilon::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::tensor](#).

References [GiNaC::return\\_types::commutative](#).

### 6.176.3.7 do\_print()

```
void GiNaC::tensepsilon::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

### 6.176.3.8 do\_print\_latex()

```
void GiNaC::tensepsilon::do_print_latex (
    const print\_latex & c,
    unsigned level ) const [protected]
```

## 6.176.4 Member Data Documentation

### 6.176.4.1 minkowski

```
bool GiNaC::tensepsilon::minkowski [private]
```

If true, tensor is in Minkowski-type space.

Otherwise it is in a Euclidean space.

Referenced by [archive\(\)](#), [contract\\_with\(\)](#), [eval\\_indexed\(\)](#), and [read\\_archive\(\)](#).

### 6.176.4.2 pos\_sig

```
bool GiNaC::tensepsilon::pos_sig [private]
```

If true, the metric is assumed to be  $\text{diag}(-1, 1, 1, \dots)$ .

Otherwise it is  $\text{diag}(1, -1, -1, \dots)$ . This is only relevant if `minkowski = true`.

Referenced by [archive\(\)](#), [contract\\_with\(\)](#), [eval\\_indexed\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

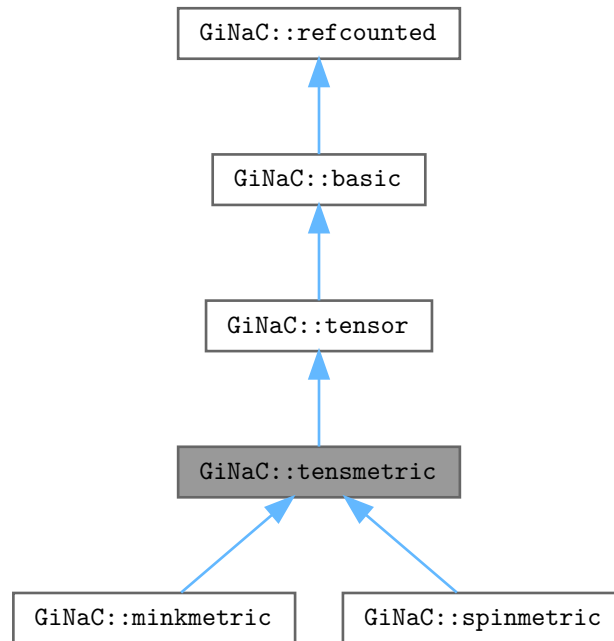
- [tensor.h](#)
- [tensor.cpp](#)

## 6.177 GiNaC::tensmetric Class Reference

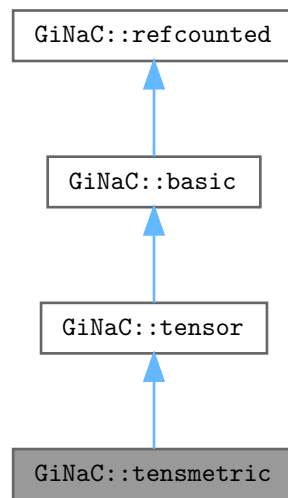
This class represents a general metric tensor which can be used to raise/lower indices.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensmetric:



Collaboration diagram for GiNaC::tensmetric:



## Public Member Functions

- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed metric tensor with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*

- virtual `ex evalf ()` const  
*Evaluate object numerically.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*

- virtual `ex normal` (`exmap &repl`, `exmap &rev_lookup`, `lst &modifier`) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap &repl`) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap &repl`) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric &xi`) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex &self`, const `ex &other`) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex &self`, const `numeric &other`) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator self`, `exvector::iterator other`, `exvector &v`) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context &c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info &ri`, const `print_context &c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node &n`) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node &n`, `lst &syms`)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap &m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol &s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic &other`) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic &other`) const  
*Test for syntactic equality.*
- const `basic & hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- const `basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- unsigned [return\\_type](#) () const override

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

#### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*



## 6.177.1 Detailed Description

This class represents a general metric tensor which can be used to raise/lower indices.

If indexed, it must have exactly two indices of the same type which must be of class `varidx` or a subclass.

## 6.177.2 Member Function Documentation

### 6.177.2.1 `info()`

```
bool GiNaC::tensmetric::info (
    unsigned inf ) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::minkmetric](#), and [GiNaC::spinmetric](#).

References [GiNaC::info\\_flags::real](#).

### 6.177.2.2 `eval_indexed()`

```
ex GiNaC::tensmetric::eval_indexed (
    const basic & i ) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed metric tensor.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::minkmetric](#), and [GiNaC::spinmetric](#).

References [GiNaC::delta\\_tensor\(\)](#), [GiNaC::idx::get\\_dim\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::varidx::is\\_covariant\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [m](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::basic::subs\(\)](#).

### 6.177.2.3 `contract_with()`

```
bool GiNaC::tensmetric::contract_with (
    exvector::iterator self,
    exvector::iterator other,
    exvector & v ) const [override], [virtual]
```

Contraction of an indexed metric tensor with something else.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::spinmetric](#).

References [GINAC\\_ASSERT](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

### 6.177.2.4 `return_type()`

```
unsigned GiNaC::tensmetric::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::tensor](#).

Reimplemented in [GiNaC::minkmetric](#).

References [GiNaC::return\\_types::commutative](#).

### 6.177.2.5 `do_print()`

```
void GiNaC::tensmetric::do_print (
    const print_context & c,
    unsigned level ) const [protected]
```

The documentation for this class was generated from the following files:

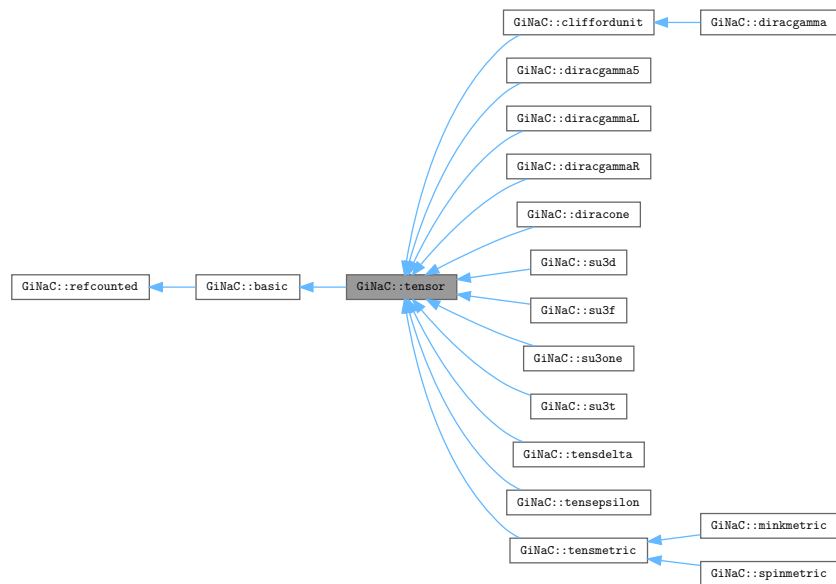
- [tensor.h](#)
- [tensor.cpp](#)

## 6.178 GiNaC::tensor Class Reference

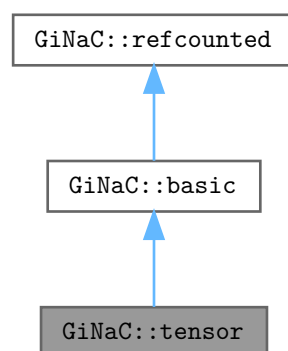
This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensor:



Collaboration diagram for GiNaC::tensor:



### Public Member Functions

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [return\\_type](#) () const override

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

#### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.178.1 Detailed Description

This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.

They are represented without indices. To attach indices to them, wrap them in an object of class [indexed](#).

### 6.178.2 Member Function Documentation

#### 6.178.2.1 `return_type()`

```
unsigned GiNaC::tensor::return_type ( ) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::tensdelta](#), [GiNaC::tensmetric](#), [GiNaC::minkmetric](#), and [GiNaC::tensepsilon](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

#### 6.178.2.2 `replace_contr_index()`

```
bool GiNaC::tensor::replace_contr_index (
    exvector::iterator self,
    exvector::iterator other ) const
```

Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).

References [GiNaC::\\_ex1](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::idx::is\\_symbolic\(\)](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::tensdelta::contract\\_with\(\)](#), and [GiNaC::tensmetric::contract\\_with\(\)](#).

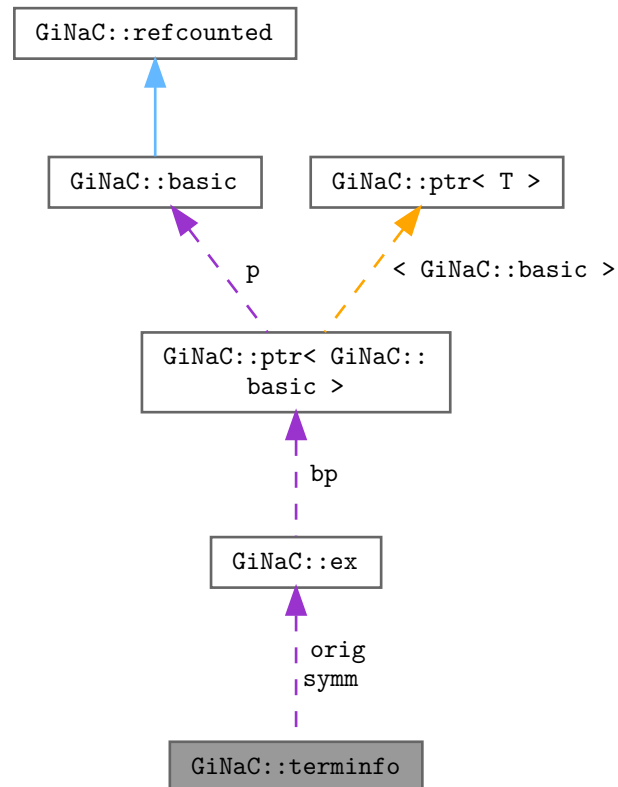
The documentation for this class was generated from the following files:

- [tensor.h](#)
- [tensor.cpp](#)

## 6.179 GiNaC::terminfo Class Reference

This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.

Collaboration diagram for GiNaC::terminfo:



### Public Member Functions

- `terminfo` (const `ex` &`orig_`, const `ex` &`symm_`)

### Public Attributes

- `ex orig`  
*original term*
- `ex symm`  
*symmetrized term*

#### 6.179.1 Detailed Description

This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.



## 6.179.2 Constructor & Destructor Documentation

### 6.179.2.1 terminfo()

```
GiNaC::terminfo::terminfo (
    const ex & orig_,
    const ex & symm_ ) [inline]
```

## 6.179.3 Member Data Documentation

### 6.179.3.1 orig

[ex](#) GiNaC::terminfo::orig

original term

### 6.179.3.2 symm

[ex](#) GiNaC::terminfo::symm

symmetrized term

Referenced by [GiNaC::terminfo\\_is\\_less::operator\(\)\(\)](#).

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.180 GiNaC::terminfo\_is\_less Class Reference

### Public Member Functions

- bool [operator\(\)](#) (const [terminfo](#) &ti1, const [terminfo](#) &ti2) const

### 6.180.1 Member Function Documentation

### 6.180.1.1 operator>()

```
bool GiNaC::terminfo_is_less::operator() (
    const terminfo & ti1,
    const terminfo & ti2 ) const [inline]
```

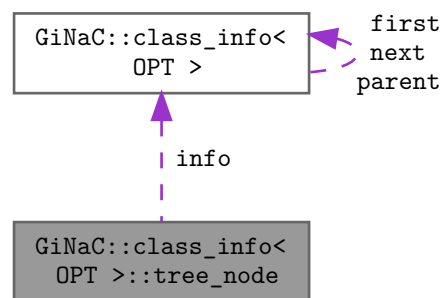
References [GiNaC::ex::compare\(\)](#), and [GiNaC::terminfo::symm](#).

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.181 GiNaC::class\_info< OPT >::tree\_node Struct Reference

Collaboration diagram for GiNaC::class\_info< OPT >::tree\_node:



### Public Member Functions

- [tree\\_node](#) ([class\\_info](#) \*)
- void [add\\_child](#) ([tree\\_node](#) \*n)

### Public Attributes

- std::vector< [tree\\_node](#) \* > [children](#)
- [class\\_info](#) \* [info](#)

### 6.181.1 Constructor & Destructor Documentation

### 6.181.1.1 tree\_node()

```
template<class OPT >
GiNaC::class_info< OPT >::tree_node::tree_node (
    class_info * i ) [inline]
```

## 6.181.2 Member Function Documentation

### 6.181.2.1 add\_child()

```
template<class OPT >
void GiNaC::class_info< OPT >::tree_node::add_child (
    tree_node * n ) [inline]
```

References [GiNaC::class\\_info< OPT >::tree\\_node::children](#), and [n](#).

## 6.181.3 Member Data Documentation

### 6.181.3.1 children

```
template<class OPT >
std::vector<tree_node *> GiNaC::class_info< OPT >::tree_node::children
```

Referenced by [GiNaC::class\\_info< OPT >::tree\\_node::add\\_child\(\)](#).

### 6.181.3.2 info

```
template<class OPT >
class_info* GiNaC::class_info< OPT >::tree_node::info
```

The documentation for this struct was generated from the following file:

- [class\\_info.h](#)

## 6.182 GiNaC::unarchive\_table\_t Class Reference

```
#include <archive.h>
```

## Public Member Functions

- [unarchive\\_table\\_t](#) ()
- [~unarchive\\_table\\_t](#) ()
- [synthesize\\_func](#) [find](#) (const std::string &classname) const
- void [insert](#) (const std::string &classname, [synthesize\\_func](#) f)

## Static Private Attributes

- static int [usecount](#) = 0
- static [unarchive\\_map\\_t](#) \* [unarch\\_map](#) = nullptr

## 6.182.1 Constructor & Destructor Documentation

### 6.182.1.1 [unarchive\\_table\\_t](#)()

`GiNaC::unarchive_table_t::unarchive_table_t ( )`

References [unarch\\_map](#), and [usecount](#).

### 6.182.1.2 [~unarchive\\_table\\_t](#)()

`GiNaC::unarchive_table_t::~~unarchive_table_t ( )`

References [unarch\\_map](#), and [usecount](#).

## 6.182.2 Member Function Documentation

### 6.182.2.1 [find](#)()

`synthesize\_func GiNaC::unarchive_table_t::find (`  
    `const std::string & classname ) const`

References [unarch\\_map](#).

Referenced by [GiNaC::find\\_factory\\_fcn](#)().

### 6.182.2.2 insert()

```
void GiNaC::unarchive_table_t::insert (
    const std::string & classname,
    synthesize_func f )
```

References [unarch\\_map](#).

## 6.182.3 Member Data Documentation

### 6.182.3.1 usecount

```
int GiNaC::unarchive_table_t::usecount = 0 [static], [private]
```

Referenced by [unarchive\\_table\\_t\(\)](#), and [~unarchive\\_table\\_t\(\)](#).

### 6.182.3.2 unarch\_map

```
unarchive\_map\_t * GiNaC::unarchive_table_t::unarch_map = nullptr [static], [private]
```

Referenced by [find\(\)](#), [insert\(\)](#), [unarchive\\_table\\_t\(\)](#), and [~unarchive\\_table\\_t\(\)](#).

The documentation for this class was generated from the following files:

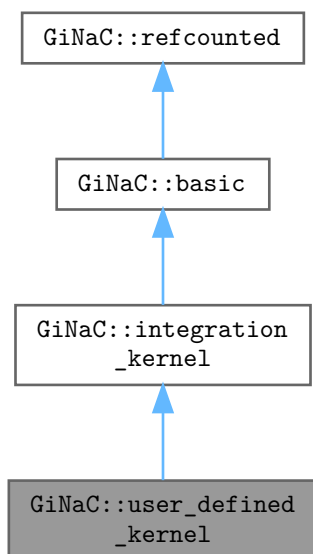
- [archive.h](#)
- [archive.cpp](#)

## 6.183 GiNaC::user\_defined\_kernel Class Reference

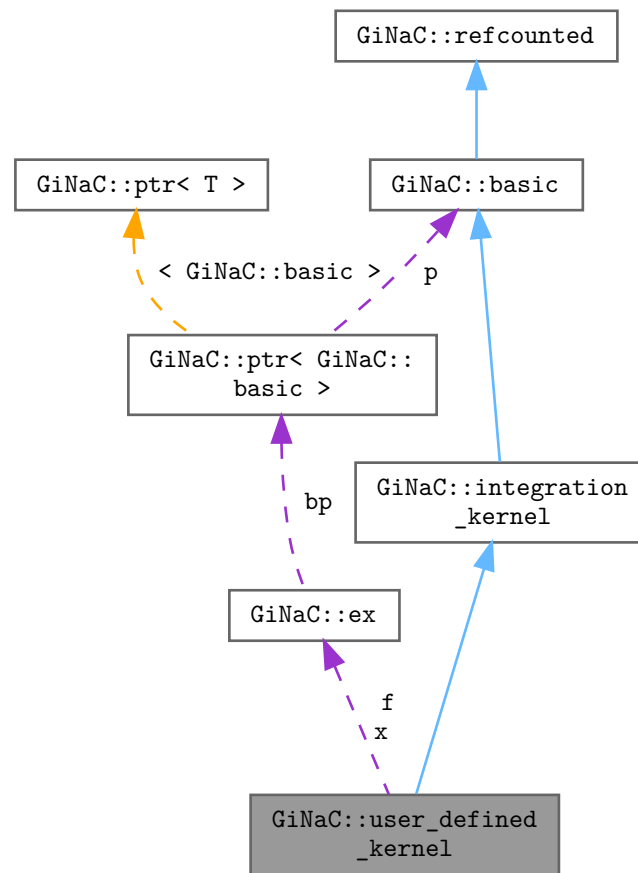
A user-defined integration kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::user\_defined\_kernel:



Collaboration diagram for GiNaC::user\_defined\_kernel:



## Public Member Functions

- `user_defined_kernel` (const `ex` &`f`, const `ex` &`x`)
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position i.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position i.*
- `bool is_numeric (void)` const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series` (const `ex` &`x`, int `order`) const override  
*Returns the Laurent series, starting possibly with the pole term.*

### Public Member Functions inherited from [GiNaC::integration\\_kernel](#)

- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Default implementation of [ex::series\(\)](#).*
- virtual bool [has\\_trailing\\_zero](#) (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool [is\\_numeric](#) (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual [ex Laurent\\_series](#) (const [ex](#) &x, int [order](#)) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual [ex get\\_numerical\\_value](#) (const [ex](#) &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t [get\\_cache\\_size](#) (void) const  
*Returns the current size of the cache.*
- void [set\\_cache\\_step](#) (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- [ex get\\_series\\_coeff](#) (int i) const  
*Wrapper around [series\\_coeff\(i\)](#), converts [cl\\_N](#) to numeric.*
- [cln::cl\\_N series\\_coeff](#) (int i) const  
*Subclasses have either to implement [series\\_coeff\\_impl](#) or the two methods [Laurent\\_series](#) and [uses\\_Laurent\\_series](#).*

### Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class [ex](#) will delete objects of derived classes via a [basic\\*](#).*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic \\* duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around [print](#) to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around [printtree](#) to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*



- virtual `size_t nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v)` const
- virtual `bool is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl)` const
- virtual `numeric integer_content ()` const
- virtual `ex smod (const numeric &xi)` const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*

- virtual unsigned `return_type` () const
- virtual `return_type_t` `return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- bool `uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- void `do_print` (const `print_context` &c, unsigned level) const

**Protected Member Functions inherited from [GiNaC::integration\\_kernel](#)**

- virtual bool [uses\\_Laurent\\_series](#) () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- virtual `cln::cl_N` [series\\_coeff\\_impl](#) (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex` [get\\_numerical\\_value\\_impl](#) (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void [do\\_print](#) (const `print_context` &c, unsigned level) const

**Protected Member Functions inherited from [GiNaC::basic](#)**

- [basic](#) ()
- virtual `ex` [eval\\_ncmul](#) (const `exvector` &v) const
- virtual bool [match\\_same\\_type](#) (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex` [derivative](#) (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int [compare\\_same\\_type](#) (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

**Protected Attributes**

- `ex` f
- `ex` x

**Protected Attributes inherited from [GiNaC::integration\\_kernel](#)**

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl_N >` [series\\_vec](#)

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
*of type `status_flags`*
- unsigned [hashvalue](#)  
*hash value*

### 6.183.1 Detailed Description

A user-defined integration kernel.

The input is an expression  $f$ , depending on a variable  $x$ . It is assumed that  $f$  has a Laurent expansion around  $x = 0$  and maximally a simple pole at  $x = 0$ .

### 6.183.2 Constructor & Destructor Documentation

#### 6.183.2.1 user\_defined\_kernel()

```
GiNaC::user_defined_kernel::user_defined_kernel (
    const ex & f,
    const ex & x )
```

### 6.183.3 Member Function Documentation

#### 6.183.3.1 nops()

```
size_t GiNaC::user_defined_kernel::nops ( ) const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.183.3.2 op()

```
ex GiNaC::user_defined_kernel::op (
    size_t i ) const [override], [virtual]
```

Return operand/member at position  $i$ .

Reimplemented from [GiNaC::basic](#).

References [f](#), and [x](#).

**6.183.3.3 let\_op()**

```
ex & GiNaC::user_defined_kernel::let_op (
    size_t i ) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [f](#), and [x](#).

**6.183.3.4 is\_numeric()**

```
bool GiNaC::user_defined_kernel::is_numeric (
    void ) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [f](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**6.183.3.5 Laurent\_series()**

```
ex GiNaC::user_defined_kernel::Laurent_series (
    const ex & x,
    int order ) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [f](#), [order](#), [GiNaC::ex::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**6.183.3.6 uses\_Laurent\_series()**

```
bool GiNaC::user_defined_kernel::uses_Laurent_series ( ) const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.183.3.7 do\_print()

```
void GiNaC::user_defined_kernel::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), [f](#), [GiNaC::ex::print\(\)](#), and [x](#).

## 6.183.4 Member Data Documentation

### 6.183.4.1 f

```
ex GiNaC::user_defined_kernel::f [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

### 6.183.4.2 x

```
ex GiNaC::user_defined_kernel::x [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

The documentation for this class was generated from the following files:

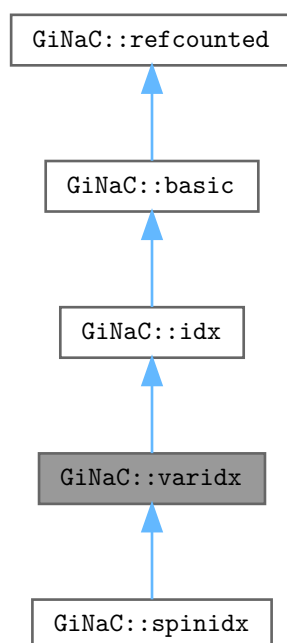
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.184 GiNaC::varidx Class Reference

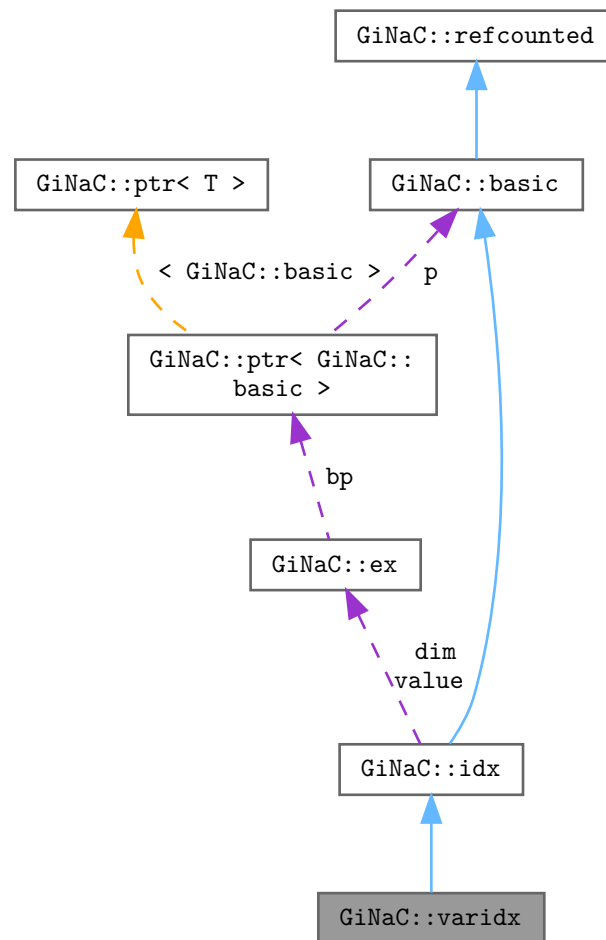
This class holds an index with a variance (co- or contravariant).

```
#include <idx.h>
```

Inheritance diagram for GiNaC::varidx:



Collaboration diagram for `GiNaC::varidx`:



## Public Member Functions

- `varidx` (const `ex` &`v`, const `ex` &`dim`, bool `covariant`=false)  
Construct index with given value, dimension and variance.
- bool `is_dummy_pair_same_type` (const `basic` &`other`) const override  
Check whether the index forms a dummy index pair with another index of the same type.
- void `archive` (`archive_node` &`n`) const override  
Save (serialize) the object into archive node.
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
Load (deserialize) the object from an archive node.
- bool `is_covariant` () const  
Check whether the index is covariant.
- bool `is_contravariant` () const  
Check whether the index is contravariant (not covariant).
- `ex toggle_variance` () const  
Make a new index with the same value but the opposite variance.



Public Member Functions inherited from [GiNaC::idx](#)

- [idx](#) (const [ex](#) &v, const [ex](#) &dim)  
*Construct index with given value and dimension.*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- size\_t [nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex map](#) (map\_function &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- [ex evalf](#) () const override  
*By default, [basic::evalf](#) would evaluate the index value but we don't want a.1 to become a.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (serialize) the object into archive node.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, lst &syms) override  
*Load (deserialize) the object from an archive node.*
- virtual bool [is\\_dummy\\_pair\\_same\\_type](#) (const [basic](#) &other) const  
*Check whether the index forms a dummy index pair with another index of the same type.*
- [ex get\\_value](#) () const  
*Get value of index.*
- bool [is\\_numeric](#) () const  
*Check whether the index is numeric.*
- bool [is\\_symbolic](#) () const  
*Check whether the index is symbolic.*
- [ex get\\_dim](#) () const  
*Get dimension of index space.*
- bool [is\\_dim\\_numeric](#) () const  
*Check whether the dimension is numeric.*
- bool [is\\_dim\\_symbolic](#) () const  
*Check whether the dimension is symbolic.*
- [ex replace\\_dim](#) (const [ex](#) &new\_dim) const  
*Make a new index with the same value but a different dimension.*
- [ex minimal\\_dim](#) (const [idx](#) &other) const  
*Return the minimum of the dimensions of this and another index.*

Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*

- virtual `ex evalf ()` const  
*Evaluate object numerically.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthesizing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*

- virtual `ex normal` (`exmap &repl`, `exmap &rev_lookup`, `lst &modifier`) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap &repl`) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap &repl`) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric &xi`) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex &self`, const `ex &other`) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex &self`, const `numeric &other`) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator self`, `exvector::iterator other`, `exvector &v`) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context &c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info &ri`, const `print_context &c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node &n`) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node &n`, `lst &syms`)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap &m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol &s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic &other`) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic &other`) const  
*Test for syntactic equality.*
- const `basic & hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- const `basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::idx](#)

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for an index always returns 0.*
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [print\\_index](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex\\_derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- bool [covariant](#)

*x.mu, default is contravariant:  $x \sim \mu$*

### Protected Attributes inherited from [GiNaC::idx](#)

- [ex value](#)

*Expression that constitutes the index (numeric or symbolic name)*

- [ex dim](#)

*Dimension of space (can be symbolic or numeric)*

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)

*of type [status\\_flags](#)*

- unsigned [hashvalue](#)

*hash value*

## 6.184.1 Detailed Description

This class holds an index with a variance (co- or contravariant).

There is an associated metric tensor that can be used to raise/lower indices.

## 6.184.2 Constructor & Destructor Documentation

### 6.184.2.1 `varidx()`

```
GiNaC::varidx::varidx (
    const ex & v,
    const ex & dim,
    bool covariant = false )
```

Construct index with given value, dimension and variance.

#### Parameters

<i>v</i>	Value of index (numeric or symbolic)
<i>dim</i>	Dimension of index space (numeric or symbolic)
<i>covariant</i>	Make covariant index (default is contravariant)

#### Returns

newly constructed index

### 6.184.3 Member Function Documentation

#### 6.184.3.1 `is_dummy_pair_same_type()`

```
bool GiNaC::varidx::is_dummy_pair_same_type (
    const basic & other ) const [override], [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented from [GiNaC::idx](#).

Reimplemented in [GiNaC::spinidx](#).

References [covariant](#).

#### 6.184.3.2 `archive()`

```
void GiNaC::varidx::archive (
    archive\_node & n ) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::idx](#).

Reimplemented in [GiNaC::spinidx](#).

References [covariant](#), and [n](#).

### 6.184.3.3 read\_archive()

```
void GiNaC::varidx::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::idx](#).

Reimplemented in [GiNaC::spinidx](#).

References [covariant](#), and [n](#).

### 6.184.3.4 match\_same\_type()

```
bool GiNaC::varidx::match_same_type (
    const basic & other ) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

#### See also

[basic::match](#)

Reimplemented from [GiNaC::idx](#).

Reimplemented in [GiNaC::spinidx](#).

References [covariant](#), and [GINAC\\_ASSERT](#).

**6.184.3.5 is\_covariant()**

```
bool GiNaC::varidx::is_covariant ( ) const [inline]
```

Check whether the index is covariant.

References [covariant](#).

Referenced by [GiNaC::tensmetric::eval\\_indexed\(\)](#).

**6.184.3.6 is\_contravariant()**

```
bool GiNaC::varidx::is_contravariant ( ) const [inline]
```

Check whether the index is contravariant (not covariant).

References [covariant](#).

**6.184.3.7 toggle\_variance()**

```
ex GiNaC::varidx::toggle_variance ( ) const
```

Make a new index with the same value but the opposite variance.

References [GiNaC::basic::clearflag\(\)](#), [covariant](#), [GiNaC::basic::duplicate\(\)](#), and [GiNaC::status\\_flags::hash\\_calculated](#).

**6.184.3.8 do\_print()**

```
void GiNaC::varidx::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), [covariant](#), and [GiNaC::idx::print\\_index\(\)](#).

**6.184.3.9 do\_print\_tree()**

```
void GiNaC::varidx::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [covariant](#), [GiNaC::idx::dim](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::print\(\)](#), and [GiNaC::idx::value](#).



## 6.184.4 Member Data Documentation

### 6.184.4.1 covariant

```
bool GiNaC::varidx::covariant [protected]
```

x.mu, default is contravariant:  $x \sim \mu$

Referenced by [archive\(\)](#), [do\\_print\(\)](#), [GiNaC::spinidx::do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [is\\_contravariant\(\)](#), [is\\_covariant\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [toggle\\_variance\(\)](#), and [GiNaC::spinidx::toggle\\_variance\\_dot\(\)](#).

The documentation for this class was generated from the following files:

- [idx.h](#)
- [idx.cpp](#)

## 6.185 GiNaC::visitor Class Reference

Degenerate base class for visitors.

```
#include <basic.h>
```

### Protected Member Functions

- virtual [~visitor\(\)](#)

### 6.185.1 Detailed Description

Degenerate base class for visitors.

basic and derivative classes support Robert C. Martin's Acyclic Visitor pattern (cf. <http://condor.depaul.edu/dmumaugh/OOT/Design-Principles/acv.pdf> or chapter 10 of Andrei Alexandrescu's "Modern C++ Design").

### 6.185.2 Constructor & Destructor Documentation

#### 6.185.2.1 ~visitor()

```
virtual GiNaC::visitor::~~visitor ( ) [inline], [protected], [virtual]
```

The documentation for this class was generated from the following file:

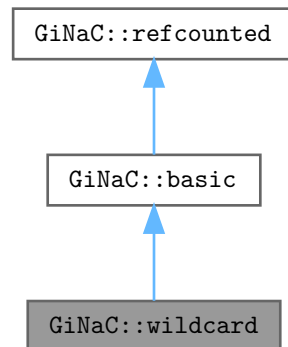
- [basic.h](#)

## 6.186 GiNaC::wildcard Class Reference

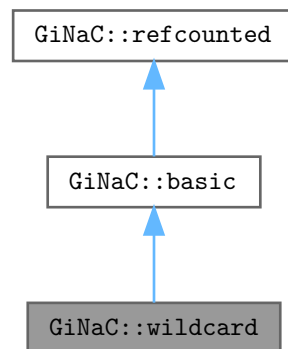
This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).

```
#include <wildcard.h>
```

Inheritance diagram for GiNaC::wildcard:



Collaboration diagram for GiNaC::wildcard:



### Public Member Functions

- [wildcard](#) (unsigned [label](#))  
*Construct wildcard with specified label.*
- bool [match](#) (const [ex](#) &pattern, [exmap](#) &repl\_lst) const override  
*Check whether the expression matches a given pattern.*

- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*
- unsigned `get_label` () const

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const

- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const

*Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const

*Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_info` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
  - template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const

*Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

*Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const

- *Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- unsigned [label](#)  
*Label used to distinguish different wildcards.*

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.186.1 Detailed Description

This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).

An integer label is used to identify different wildcards.

## 6.186.2 Constructor & Destructor Documentation

### 6.186.2.1 wildcard()

```
GiNaC::wildcard::wildcard (
    unsigned label )
```

Construct wildcard with specified label.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GiNaC::basic::setflag\(\)](#).

## 6.186.3 Member Function Documentation

### 6.186.3.1 match()

```
bool GiNaC::wildcard::match (
    const ex & pattern,
    exmap & repl_lst ) const [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to `repl_lst`.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::is\\_equal\(\)](#).

### 6.186.3.2 archive()

```
void GiNaC::wildcard::archive (
    archive_node & n ) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [label](#), and [n](#).

### 6.186.3.3 read\_archive()

```
void GiNaC::wildcard::read_archive (
    const archive_node & n,
    lst & syms ) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [label](#), [n](#), and [GiNaC::basic::setflag\(\)](#).

### 6.186.3.4 calchash()

```
unsigned GiNaC::wildcard::calchash ( ) const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [label](#), [GiNaC::make\\_hash\\_seed\(\)](#), and [GiNaC::basic::setflag\(\)](#).

### 6.186.3.5 get\_label()

```
unsigned GiNaC::wildcard::get_label ( ) const [inline]
```

References [label](#).

### 6.186.3.6 do\_print()

```
void GiNaC::wildcard::do_print (
    const print\_context & c,
    unsigned level ) const [protected]
```

References [c](#), and [label](#).

### 6.186.3.7 do\_print\_tree()

```
void GiNaC::wildcard::do_print_tree (
    const print\_tree & c,
    unsigned level ) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), and [label](#).

### 6.186.3.8 do\_print\_python\_repr()

```
void GiNaC::wildcard::do_print_python_repr (
    const print\_python\_repr & c,
    unsigned level ) const [protected]
```

References [c](#), and [label](#).

## 6.186.4 Member Data Documentation

### 6.186.4.1 label

```
unsigned GiNaC::wildcard::label [private]
```

Label used to distinguish different wildcards.

Referenced by [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_label\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

- [wildcard.h](#)
- [wildcard.cpp](#)



## 6.187 GiNaC::zeta1\_SERIAL Class Reference

Complex conjugate.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.187.1 Detailed Description

Complex conjugate.

Real part. Imaginary part. Absolute value. Step function. Complex sign. Eta function:  $\log(a*b) == \log(a) + \log(b) + \eta(a, b)$ . Sine. Cosine. Tangent. Exponential function. Natural logarithm. Inverse sine (arc sine). Inverse cosine (arc cosine). Inverse tangent (arc tangent). Inverse tangent with two arguments. Hyperbolic Sine. Hyperbolic Cosine. Hyperbolic Tangent. Inverse hyperbolic Sine (area hyperbolic sine). Inverse hyperbolic Cosine (area hyperbolic cosine). Inverse hyperbolic Tangent (area hyperbolic tangent). Dilogarithm. Trilogarithm. Derivatives of Riemann's Zeta-function. Multiple zeta value including Riemann's zeta-function.

### 6.187.2 Member Data Documentation

#### 6.187.2.1 serial

```
unsigned GiNaC::zeta1_SERIAL::serial [static]
```

##### Initial value:

```
= function::register_new(function_options("zeta", 1).
    evalf_func(zetal_evalf).
    eval_func(zetal_eval).
    derivative_func(zetal_deriv).
    print_func<print_latex>(zetal_print_latex).
    do_not_evalf_params().
    overloaded(2))
```

Referenced by [GiNaC::zeta\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 6.188 GiNaC::zeta2\_SERIAL Class Reference

Alternating Euler sum or colored MZV.

```
#include <inifcns.h>
```

## Static Public Attributes

- static unsigned [serial](#)

### 6.188.1 Detailed Description

Alternating Euler sum or colored MZV.

### 6.188.2 Member Data Documentation

#### 6.188.2.1 [serial](#)

```
unsigned GiNaC::zeta2_SERIAL::serial [static]
```

##### Initial value:

```
= function::register_new(function_options("zeta", 2).  
    evalf_func(zeta2_evalf).  
    eval_func(zeta2_eval).  
    derivative_func(zeta2_deriv).  
    print_func<print_latex>(zeta2_print_latex).  
    do_not_evalf_params().  
    overloaded(2))
```

Referenced by [GiNaC::zeta\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## Chapter 7

# File Documentation

### 7.1 add.cpp File Reference

Implementation of [GiNaC](#)'s sums of expressions.

```
#include "add.h"
#include "mul.h"
#include "archive.h"
#include "operators.h"
#include "matrix.h"
#include "utils.h"
#include "clifford.h"
#include "ncmul.h"
#include "compiler.h"
#include <iostream>
#include <limits>
#include <stdexcept>
#include <string>
```

#### Namespaces

- namespace [GiNaC](#)

#### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (add, expairseq, print\_func< print\_context >(&add::do\_print). print\_func< print\_latex >(&add::do\_print\_latex). print\_func< print\_csrc >(&add::do\_print\_csrc). print\_func< print\_tree >(&add::do\_print\_tree). print\_func< print\_python\_repr >(&add::do\_print\_python\_repr)) add
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (add)

#### 7.1.1 Detailed Description

Implementation of [GiNaC](#)'s sums of expressions.

## 7.2 add.h File Reference

Interface to [GiNaC](#)'s sums of expressions.

```
#include "expairseq.h"
```

### Classes

- class [GiNaC::add](#)  
*Sum of expressions.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (add)

### 7.2.1 Detailed Description

Interface to [GiNaC](#)'s sums of expressions.

## 7.3 archive.cpp File Reference

Archiving of [GiNaC](#) expressions.

```
#include "archive.h"  
#include "registrar.h"  
#include "ex.h"  
#include "lst.h"  
#include "version.h"  
#include <iostream>  
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

## Functions

- static void [GiNaC::write\\_unsigned](#) (std::ostream &os, unsigned val)  
*Write unsigned integer quantity to stream.*
- static unsigned [GiNaC::read\\_unsigned](#) (std::istream &is)  
*Read unsigned integer quantity from stream.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const archive\_node &n)  
*Write [archive\\_node](#) to binary data stream.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const archive &ar)  
*Write archive to binary data stream.*
- std::istream & [GiNaC::operator>>](#) (std::istream &is, archive\_node &n)  
*Read [archive\\_node](#) from binary data stream.*
- std::istream & [GiNaC::operator>>](#) (std::istream &is, archive &ar)  
*Read archive from binary data stream.*
- static synthesizer\_func [GiNaC::find\\_factory\\_fcn](#) (const std::string &name)

### 7.3.1 Detailed Description

Archiving of [GiNaC](#) expressions.

## 7.4 archive.h File Reference

Archiving of [GiNaC](#) expressions.

```
#include "ex.h"
#include <iosfwd>
#include <map>
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::archive\\_node](#)  
*This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).*
- struct [GiNaC::archive\\_node::property\\_info](#)  
*Information about a stored property.*
- struct [GiNaC::archive\\_node::property](#)  
*Archived property (data type, name and associated data)*
- struct [GiNaC::archive\\_node::archive\\_node\\_cit\\_range](#)
- class [GiNaC::unarchive\\_table\\_t](#)
- class [GiNaC::archive](#)  
*This class holds archived versions of [GiNaC](#) expressions (class ex).*
- struct [GiNaC::archive::archived\\_ex](#)  
*Archived expression descriptor.*

## Namespaces

- namespace [GiNaC](#)

## Macros

- #define [GINAC\\_DECLARE\\_UNARCHIVER](#)(classname)  
*Helper macros to register a class with (un)archiving (a.k.a.*
- #define [GINAC\\_BIND\\_UNARCHIVER](#)(classname)

## Typedefs

- typedef unsigned [GiNaC::archive\\_node\\_id](#)  
*Numerical ID value to refer to an [archive\\_node](#).*
- typedef unsigned [GiNaC::archive\\_atom](#)  
*Numerical ID value to refer to a string.*
- typedef basic <(\* [GiNaC::synthesize\\_func](#)) ()
- typedef std::map< std::string, synthesize\_func > [GiNaC::unarchive\\_map\\_t](#)

## Functions

- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const archive &ar)  
*Write archive to binary data stream.*
- std::istream & [GiNaC::operator>>](#) (std::istream &is, archive &ar)  
*Read archive from binary data stream.*

## Variables

- static unarchive\_table\_t [GiNaC::unarch\\_table\\_instance](#)

### 7.4.1 Detailed Description

Archiving of [GiNaC](#) expressions.

### 7.4.2 Macro Definition Documentation

## 7.4.2.1 GINAC\_DECLARE\_UNARCHIVER

```
#define GINAC_DECLARE_UNARCHIVER(  
    classname )
```

**Value:**

```
class classname ## _unarchiver  
{  
    static int usecount;  
public:  
    static GiNaC::basic* create();  
    classname ## _unarchiver();  
    ~ classname ## _unarchiver();  
};  
static classname ## _unarchiver classname ## _unarchiver_instance
```

Helper macros to register a class with (un)archiving (a.k.a. (de)serialization).

Usage: put

```
GINAC_DECLARE_UNARCHIVER(myclass);
```

into the header file (in the global or namespace scope), and

```
GINAC_BIND_UNARCHIVER(myclass);
```

into the source file.

Effect: the 'myclass' (being a class derived directly or indirectly from [GiNaC::basic](#)) can be archived and unarchived.

Note: you need to use GINAC\_{DECLARE,BIND}\_UNARCHIVER incantations in order to make your class (un)archivable *even if your class does not overload 'read\_archive' method*. Sorry for inconvenience.

How it works:

The 'basic' class has a 'read\_archive' virtual method which reads an expression from archive. Derived classes can overload that method. There's a small problem, though. On unarchiving all we have is a set of named byte streams. In C++ the class name (as written in the source code) has nothing to do with its actual type. Thus, we need establish a correspondence ourselves. To do so we maintain a 'class\_name' => 'function\_pointer' table (see the `unarchive_table_t` class above). Every function in this table is supposed to create a new object of the 'class\_name' type. The 'archive\_node' class uses that table to construct an object of correct type. Next it invokes `read_archive` virtual method of newly created object, which does the actual job.

Note: this approach is very simple-minded (it does not handle classes with same names from different namespaces, multiple inheritance, etc), but it happens to work surprisingly well.

## 7.4.2.2 GINAC\_BIND\_UNARCHIVER

```
#define GINAC_BIND_UNARCHIVER(  
    classname )
```

**Value:**

```
classname ## _unarchiver::classname ## _unarchiver()  
{  
    static GiNaC::unarchive_table_t table;  
    if (usecount++ == 0) {  
        table.insert(std::string(#classname),  
            &(classname ## _unarchiver::create));  
    }  
}  
GiNaC::basic* classname ## _unarchiver::create()  
{  
    return new classname();  
}  
classname ## _unarchiver::~~ classname ## _unarchiver() { }  
int classname ## _unarchiver::usecount = 0
```

## 7.5 `assertion.h` File Reference

Assertion macro definition.

### Macros

- `#define GINAC_ASSERT(X) ((void)0)`  
*Assertion macro for checking invariances.*

### 7.5.1 Detailed Description

Assertion macro definition.

### 7.5.2 Macro Definition Documentation

#### 7.5.2.1 GINAC\_ASSERT

```
#define GINAC_ASSERT(  
    X ) ((void)0)
```

Assertion macro for checking invariances.

## 7.6 `basic.cpp` File Reference

Implementation of [GiNaC](#)'s ABC.

```
#include "basic.h"  
#include "ex.h"  
#include "numeric.h"  
#include "power.h"  
#include "add.h"  
#include "symbol.h"  
#include "lst.h"  
#include "ncmul.h"  
#include "relational.h"  
#include "operators.h"  
#include "wildcard.h"  
#include "archive.h"  
#include "utils.h"  
#include "hash_seed.h"  
#include "inifcns.h"  
#include <iostream>  
#include <stdexcept>  
#include <typeinfo>
```



## Classes

- struct [GiNaC::evalf\\_map\\_function](#)  
Function object to be applied by [basic::evalf\(\)](#).
- struct [GiNaC::evalm\\_map\\_function](#)  
Function object to be applied by [basic::evalm\(\)](#).
- struct [GiNaC::eval\\_integ\\_map\\_function](#)  
Function object to be applied by [basic::eval\\_integ\(\)](#).
- struct [GiNaC::derivative\\_map\\_function](#)  
Function object to be applied by [basic::derivative\(\)](#).
- struct [GiNaC::expand\\_map\\_function](#)  
Function object to be applied by [basic::expand\(\)](#).

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (basic, void, print\_func< print\_context >(&basic::do\_print). print\_func< print\_tree >(&basic::do\_print\_tree). print\_func< print\_python\_repr >(&basic::do\_print\_python\_repr)) basic  
*basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by duplicate()), so it can copy the tinfo\_key and the hash value.*

## Variables

- [GiNaC::evalm\\_map\\_function](#) [GiNaC::map\\_evalm](#)
- [GiNaC::eval\\_integ\\_map\\_function](#) [GiNaC::map\\_eval\\_integ](#)

### 7.6.1 Detailed Description

Implementation of [GiNaC](#)'s ABC.

## 7.7 basic.h File Reference

Interface to [GiNaC](#)'s ABC.

```
#include "flags.h"
#include "ptr.h"
#include "assertion.h"
#include "registrar.h"
#include <stddef>
#include <map>
#include <set>
#include <typeinfo>
#include <vector>
#include <utility>
```

## Classes

- struct [GiNaC::map\\_function](#)  
*Function object for map().*
- class [GiNaC::visitor](#)  
*Degenerate base class for visitors.*
- class [GiNaC::basic](#)  
*This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.*

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef std::vector< ex > [GiNaC::exvector](#)
- typedef std::set< ex, ex\_is\_less > [GiNaC::exset](#)
- typedef std::map< ex, ex, ex\_is\_less > [GiNaC::exmap](#)

## Functions

- template<class T >  
bool [GiNaC::is\\_a](#) (const basic &obj)  
*Check if obj is a T, including base classes.*
- template<class T >  
bool [GiNaC::is\\_exactly\\_a](#) (const basic &obj)  
*Check if obj is a T, not including base classes.*
- template<class B , typename... Args>  
B & [GiNaC::dynallocate](#) (Args &&... args)  
*Constructs a new (class basic or derived) B object on the heap.*
- template<class B >  
B & [GiNaC::dynallocate](#) (std::initializer\_list< ex > il)  
*Constructs a new (class basic or derived) B object on the heap.*

### 7.7.1 Detailed Description

Interface to [GiNaC](#)'s ABC.

## 7.8 class\_info.h File Reference

Helper templates to provide per-class information for class hierarchies.

```
#include <cstddef>
#include <cstring>
#include <iomanip>
#include <iostream>
#include <map>
#include <stdexcept>
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::class\\_info< OPT >](#)
- struct [GiNaC::class\\_info< OPT >::tree\\_node](#)

## Namespaces

- namespace [GiNaC](#)

### 7.8.1 Detailed Description

Helper templates to provide per-class information for class hierarchies.

## 7.9 clifford.cpp File Reference

Implementation of [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

```
#include "clifford.h"
#include "ex.h"
#include "idx.h"
#include "ncmul.h"
#include "symbol.h"
#include "numeric.h"
#include "symmetry.h"
#include "lst.h"
#include "relational.h"
#include "operators.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "matrix.h"
#include "archive.h"
#include "utils.h"
#include <stdexcept>
```

## Classes

- struct [GiNaC::is\\_not\\_a\\_clifford](#)  
*Predicate for finding non-clifford objects.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (clifford, indexed, print\_func< print\_dflt >(&clifford::do\_print\_dflt). print\_func< print\_latex >(&clifford::do\_print\_latex). print\_func< print\_tree >(&clifford::do\_print\_tree)) GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT(diracone
- [GiNaC::print\\_func< print\\_dflt >](#) (&diracone::do\_print). print\_func< print\_latex >(&diracone
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (clifford)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (cliffordunit)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracone)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracgamma)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracgamma5)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracgammaL)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (diracgammaR)
- static bool [GiNaC::is\\_dirac\\_slash](#) (const ex &seq0)
- static void [GiNaC::base\\_and\\_index](#) (const ex &c, ex &b, ex &i)  
*This function decomposes  $\gamma_{\sim\mu} \rightarrow (1, \mu)$  and  $a_l \rightarrow (a.ix, ix)$*
- ex [GiNaC::dirac\\_ONE](#) (unsigned char rl=0)  
*Create a Clifford unity object.*
- static unsigned [GiNaC::get\\_dim\\_uint](#) (const ex &e)
- ex [GiNaC::clifford\\_unit](#) (const ex &mu, const ex &metr, unsigned char rl=0)  
*Create a Clifford unit object.*
- ex [GiNaC::dirac\\_gamma](#) (const ex &mu, unsigned char rl=0)  
*Create a Dirac gamma object.*
- ex [GiNaC::dirac\\_gamma5](#) (unsigned char rl=0)  
*Create a Dirac gamma5 object.*
- ex [GiNaC::dirac\\_gammaL](#) (unsigned char rl=0)  
*Create a Dirac gammaL object.*
- ex [GiNaC::dirac\\_gammaR](#) (unsigned char rl=0)  
*Create a Dirac gammaR object.*
- ex [GiNaC::dirac\\_slash](#) (const ex &e, const ex &dim, unsigned char rl=0)  
*Create a term of the form  $e_{\mu} * \gamma_{\sim\mu}$  with a unique index  $\mu$ .*
- static unsigned char [GiNaC::get\\_representation\\_label](#) (const return\_type\_t &ti)  
*Extract representation label from tinfo key (as returned by return\_type\_tinfo()).*
- static ex [GiNaC::trace\\_string](#) (exvector::const\_iterator ix, size\_t num)  
*Take trace of a string of an even number of Dirac gammas given a vector of indices.*
- ex [GiNaC::dirac\\_trace](#) (const ex &e, const std::set< unsigned char > &rls, const ex &trONE=4)  
*Calculate dirac traces over the specified set of representation labels.*
- ex [GiNaC::dirac\\_trace](#) (const ex &e, const lst &rls, const ex &trONE=4)  
*Calculate dirac traces over the specified list of representation labels.*
- ex [GiNaC::dirac\\_trace](#) (const ex &e, unsigned char rl=0, const ex &trONE=4)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- ex [GiNaC::canonicalize\\_clifford](#) (const ex &e)  
*Bring all products of clifford objects in an expression into a canonical order.*
- ex [GiNaC::clifford\\_star\\_bar](#) (const ex &e, bool do\_bar, unsigned options)  
*An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).*
- ex [GiNaC::clifford\\_prime](#) (const ex &e)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- ex [GiNaC::remove\\_dirac\\_ONE](#) (const ex &e, unsigned char rl=0, unsigned options=0)  
*Replaces [dirac\\_ONE](#)'s (with a representation\_label no less than rl) in e with 1.*
- int [GiNaC::clifford\\_max\\_label](#) (const ex &e, bool ignore\_ONE=false)  
*Returns the maximal representation label of a clifford object if e contains at least one, otherwise returns -1.*
- ex [GiNaC::clifford\\_norm](#) (const ex &e)

- Calculation of the norm in the Clifford algebra.*

  - ex [GiNaC::clifford\\_inverse](#) (const ex &e)
- Calculation of the inverse in the Clifford algebra.*

  - ex [GiNaC::lst\\_to\\_clifford](#) (const ex &v, const ex &mu, const ex &metr, unsigned char rl=0)
- List or vector conversion into the Clifford vector.*

  - ex [GiNaC::lst\\_to\\_clifford](#) (const ex &v, const ex &e)
  - static ex [GiNaC::get\\_clifford\\_comp](#) (const ex &e, const ex &c, bool root=true)
- Auxiliary structure to define a function for stripping one Clifford unit from vectors.*

  - lst [GiNaC::clifford\\_to\\_lst](#) (const ex &e, const ex &c, bool algebraic=true)
- An inverse function to [lst\\_to\\_clifford\(\)](#).*

  - ex [GiNaC::clifford\\_moebius\\_map](#) (const ex &a, const ex &b, const ex &c, const ex &d, const ex &v, const ex &G, unsigned char rl=0)
- Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b|c d) in linear spaces with arbitrary signature.*

  - ex [GiNaC::clifford\\_moebius\\_map](#) (const ex &M, const ex &v, const ex &G, unsigned char rl=0)
- The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*

## Variables

- [GiNaC::tensor](#)

## 7.9.1 Detailed Description

Implementation of [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

## 7.10 clifford.h File Reference

Interface to [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

```
#include "indexed.h"
#include "tensor.h"
#include "symbol.h"
#include "idx.h"
#include <set>
```

## Classes

- class [GiNaC::clifford](#)

*This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).*
- class [GiNaC::diracone](#)

*This class represents the Clifford algebra unity element.*
- class [GiNaC::cliffordunit](#)

*This class represents the Clifford algebra generators (units).*
- class [GiNaC::diracgamma](#)

*This class represents the Dirac gamma Lorentz vector.*
- class [GiNaC::diracgamma5](#)

*This class represents the Dirac gamma5 object which anticommutes with all other gammas.*
- class [GiNaC::diracgammaL](#)

*This class represents the Dirac gammaL object which behaves like 1/2 (1-gamma5).*
- class [GiNaC::diracgammaR](#)

*This class represents the Dirac gammaL object which behaves like 1/2 (1+gamma5).*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (clifford)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracone)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (cliffordunit)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracgamma)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracgamma5)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracgammaL)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (diracgammaR)
- bool [GiNaC::is\\_clifford\\_tinfo](#) (const return\_type\_t &ti)  
*Check whether a given [return\\_type\\_t](#) object (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a clifford object (with an arbitrary representation label).*
- ex [GiNaC::dirac\\_ONE](#) (unsigned char rl=0)  
*Create a Clifford unity object.*
- ex [GiNaC::clifford\\_unit](#) (const ex &mu, const ex &metr, unsigned char rl=0)  
*Create a Clifford unit object.*
- ex [GiNaC::dirac\\_gamma](#) (const ex &mu, unsigned char rl=0)  
*Create a Dirac gamma object.*
- ex [GiNaC::dirac\\_gamma5](#) (unsigned char rl=0)  
*Create a Dirac gamma5 object.*
- ex [GiNaC::dirac\\_gammaL](#) (unsigned char rl=0)  
*Create a Dirac gammaL object.*
- ex [GiNaC::dirac\\_gammaR](#) (unsigned char rl=0)  
*Create a Dirac gammaR object.*
- ex [GiNaC::dirac\\_slash](#) (const ex &e, const ex &dim, unsigned char rl=0)  
*Create a term of the form  $e_{\mu} * \gamma_{\sim\mu}$  with a unique index  $\mu$ .*
- ex [GiNaC::dirac\\_trace](#) (const ex &e, const std::set< unsigned char > &rls, const ex &trONE=4)  
*Calculate dirac traces over the specified set of representation labels.*
- ex [GiNaC::dirac\\_trace](#) (const ex &e, const lst &rl, const ex &trONE=4)  
*Calculate dirac traces over the specified list of representation labels.*
- ex [GiNaC::dirac\\_trace](#) (const ex &e, unsigned char rl=0, const ex &trONE=4)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- ex [GiNaC::canonicalize\\_clifford](#) (const ex &e)  
*Bring all products of clifford objects in an expression into a canonical order.*
- ex [GiNaC::clifford\\_prime](#) (const ex &e)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- ex [GiNaC::clifford\\_star\\_bar](#) (const ex &e, bool do\_bar, unsigned options)  
*An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).*
- ex [GiNaC::clifford\\_bar](#) (const ex &e)  
*Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.*
- ex [GiNaC::clifford\\_star](#) (const ex &e)  
*Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.*
- ex [GiNaC::remove\\_dirac\\_ONE](#) (const ex &e, unsigned char rl=0, unsigned options=0)  
*Replaces [dirac\\_ONE](#)'s (with a [representation\\_label](#) no less than  $rl$ ) in  $e$  with 1.*
- int [GiNaC::clifford\\_max\\_label](#) (const ex &e, bool ignore\_ONE=false)  
*Returns the maximal representation label of a clifford object if  $e$  contains at least one, otherwise returns -1.*
- ex [GiNaC::clifford\\_norm](#) (const ex &e)

- *Calculation of the norm in the Clifford algebra.*  
ex [GiNaC::clifford\\_inverse](#) (const ex &e)
- *Calculation of the inverse in the Clifford algebra.*  
ex [GiNaC::lst\\_to\\_clifford](#) (const ex &v, const ex &mu, const ex &metr, unsigned char rl=0)
- *List or vector conversion into the Clifford vector.*  
ex [GiNaC::lst\\_to\\_clifford](#) (const ex &v, const ex &e)
- *List or vector conversion into the Clifford vector.*  
lst [GiNaC::clifford\\_to\\_lst](#) (const ex &e, const ex &c, bool algebraic=true)
- *An inverse function to [lst\\_to\\_clifford\(\)](#).*  
ex [GiNaC::clifford\\_moebius\\_map](#) (const ex &a, const ex &b, const ex &c, const ex &d, const ex &v, const ex &G, unsigned char rl=0)
- *Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b|c d) in linear spaces with arbitrary signature.*  
ex [GiNaC::clifford\\_moebius\\_map](#) (const ex &M, const ex &v, const ex &G, unsigned char rl=0)
- *The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*

### 7.10.1 Detailed Description

Interface to [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

## 7.11 color.cpp File Reference

Implementation of [GiNaC](#)'s color (SU(3) Lie algebra) objects.

```
#include "color.h"
#include "idx.h"
#include "ncmul.h"
#include "symmetry.h"
#include "operators.h"
#include "numeric.h"
#include "mul.h"
#include "power.h"
#include "symbol.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

### Macros

- #define [TEST\\_PERMUTATION](#)(A, B, C, P)
- #define [CMPINDICES](#)(A, B, C) ((v[0] == (A)) && (v[1] == (B)) && (v[2] == (C)))

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (su3one, tensor, print\_func< print\_dflt >(&su3one::do\_print). print\_func< print\_latex >(&su3one::do\_print\_latex)) GINAC\_IMPLEMENT\_↵ REGISTERED\_CLASS\_OPT(su3t
- [GiNaC::print\\_func< print\\_dflt >](#) (&su3t::do\_print). print\_func< print\_latex >(&su3t
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (color)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (su3one)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (su3t)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (su3f)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (su3d)
- static ex [GiNaC::permute\\_free\\_index\\_to\\_front](#) (const exvector &iv3, const exvector &iv2, int &sig)
 

*Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.*
- ex [GiNaC::color\\_ONE](#) (unsigned char rl=0)
 

*Create the su(3) unity element.*
- ex [GiNaC::color\\_T](#) (const ex &a, unsigned char rl=0)
 

*Create an su(3) generator.*
- ex [GiNaC::color\\_f](#) (const ex &a, const ex &b, const ex &c)
 

*Create an su(3) antisymmetric structure constant.*
- ex [GiNaC::color\\_d](#) (const ex &a, const ex &b, const ex &c)
 

*Create an su(3) symmetric structure constant.*
- ex [GiNaC::color\\_h](#) (const ex &a, const ex &b, const ex &c)
 

*This returns the linear combination d.a.b.c+1\*f.a.b.c.*
- static bool [GiNaC::is\\_color\\_tinfo](#) (const return\_type\_t &ti)
 

*Check whether a given tinfo key (as returned by return\_type\_tinfo()) is that of a color object (with an arbitrary representation label).*
- static unsigned char [GiNaC::get\\_representation\\_label](#) (const return\_type\_t &ti)
 

*Extract representation label from tinfo key (as returned by return\_type\_tinfo()).*
- ex [GiNaC::color\\_trace](#) (const ex &e, const std::set< unsigned char > &rls)
 

*Calculate color traces over the specified set of representation labels.*
- ex [GiNaC::color\\_trace](#) (const ex &e, const lst &rls)
 

*Calculate color traces over the specified list of representation labels.*
- ex [GiNaC::color\\_trace](#) (const ex &e, unsigned char rl=0)
 

*Calculate the trace of an expression containing color objects with a specified representation label.*

### 7.11.1 Detailed Description

Implementation of [GiNaC](#)'s color (SU(3) Lie algebra) objects.

### 7.11.2 Macro Definition Documentation



## 7.11.2.1 TEST\_PERMUTATION

```
#define TEST_PERMUTATION(
    A,
    B,
    C,
    P )
```

## Value:

```
if (iv3[B].is_equal(iv2[0]) && iv3[C].is_equal(iv2[1])) { \
    sig = P; \
    return iv3[A]; \
}
```

## 7.11.2.2 CMPINDICES

```
#define CMPINDICES(
    A,
    B,
    C ) ((v[0] == (A)) && (v[1] == (B)) && (v[2] == (C)))
```

## 7.12 color.h File Reference

Interface to [GiNaC](#)'s color (SU(3) Lie algebra) objects.

```
#include "indexed.h"
#include "tensor.h"
#include <set>
```

## Classes

- class [GiNaC::color](#)  
*This class holds a generator  $T_a$  or the unity element of the Lie algebra of SU(3), as used for calculations in quantum chromodynamics.*
- class [GiNaC::su3one](#)  
*This class represents the su(3) unity element.*
- class [GiNaC::su3t](#)  
*This class represents an su(3) generator.*
- class [GiNaC::su3f](#)  
*This class represents the tensor of antisymmetric su(3) structure constants.*
- class [GiNaC::su3d](#)  
*This class represents the tensor of symmetric su(3) structure constants.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- `GiNaC::GINAC_DECLARE_UNARCHIVER` (color)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (su3one)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (su3t)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (su3f)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (su3d)
- ex `GiNaC::color_ONE` (unsigned char rl=0)  
*Create the su(3) unity element.*
- ex `GiNaC::color_T` (const ex &a, unsigned char rl=0)  
*Create an su(3) generator.*
- ex `GiNaC::color_f` (const ex &a, const ex &b, const ex &c)  
*Create an su(3) antisymmetric structure constant.*
- ex `GiNaC::color_d` (const ex &a, const ex &b, const ex &c)  
*Create an su(3) symmetric structure constant.*
- ex `GiNaC::color_h` (const ex &a, const ex &b, const ex &c)  
*This returns the linear combination d.a.b.c+l\*f.a.b.c.*
- ex `GiNaC::color_trace` (const ex &e, const std::set< unsigned char > &rls)  
*Calculate color traces over the specified set of representation labels.*
- ex `GiNaC::color_trace` (const ex &e, const lst &rls)  
*Calculate color traces over the specified list of representation labels.*
- ex `GiNaC::color_trace` (const ex &e, unsigned char rl=0)  
*Calculate the trace of an expression containing color objects with a specified representation label.*

### 7.12.1 Detailed Description

Interface to `GiNaC`'s color (SU(3) Lie algebra) objects.

## 7.13 compiler.h File Reference

Definition of optimizing macros.

### Macros

- `#define unlikely(cond)` (cond)
- `#define likely(cond)` (cond)
- `#define attribute_deprecated`

### 7.13.1 Detailed Description

Definition of optimizing macros.

### 7.13.2 Macro Definition Documentation

### 7.13.2.1 unlikely

```
#define unlikely(  
    cond ) (cond)
```

### 7.13.2.2 likely

```
#define likely(  
    cond ) (cond)
```

### 7.13.2.3 attribute\_deprecated

```
#define attribute_deprecated
```

## 7.14 constant.cpp File Reference

Implementation of [GiNaC](#)'s constant types and some special constants.

```
#include "constant.h"  
#include "numeric.h"  
#include "ex.h"  
#include "archive.h"  
#include "utils.h"  
#include "inifcns.h"  
#include <iostream>  
#include <stdexcept>  
#include <string>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (constant, basic, print\_func< print\_context >(&constant::do\_print). print\_func< print\_latex >(&constant::do\_print\_latex). print\_func< print\_tree >(&constant::do\_print\_tree). print\_func< print\_python\_repr >(&constant::do\_print\_python\_repr)) constant
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (constant)

## Variables

- const constant [GiNaC::Pi](#) ("Pi", PiEvalf, "\\pi", domain::positive)  
*Pi.*
- const constant [GiNaC::Euler](#) ("Euler", EulerEvalf, "\\gamma\_E", domain::positive)  
*Euler's constant.*
- const constant [GiNaC::Catalan](#) ("Catalan", CatalanEvalf, "G", domain::positive)  
*Catalan's constant.*

### 7.14.1 Detailed Description

Implementation of [GiNaC](#)'s constant types and some special constants.

## 7.15 constant.h File Reference

Interface to [GiNaC](#)'s constant types and some special constants.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include <string>
```

## Classes

- class [GiNaC::constant](#)  
*This class holds constants, symbols with specific numerical value.*

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef `ex(* GiNaC::evalffunctype) ()`

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (constant)

### 7.15.1 Detailed Description

Interface to [GiNaC](#)'s constant types and some special constants.

## 7.16 container.h File Reference

Wrapper template for making [GiNaC](#) classes out of STL containers.

```
#include "ex.h"
#include "print.h"
#include "archive.h"
#include "assertion.h"
#include "compiler.h"
#include <algorithm>
#include <iterator>
#include <list>
#include <memory>
#include <stdexcept>
#include <vector>
```

### Classes

- class [GiNaC::container\\_storage< C >](#)  
*Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.*
- class [GiNaC::container< C >](#)  
*Wrapper template for making [GiNaC](#) classes out of STL containers.*

### Namespaces

- namespace [GiNaC](#)

#### 7.16.1 Detailed Description

Wrapper template for making [GiNaC](#) classes out of STL containers.

## 7.17 crc32.h File Reference

CRC32 hash function.

### Namespaces

- namespace [GiNaC](#)

### Functions

- static unsigned [GiNaC::crc32](#) (const char \*c, const unsigned len, const unsigned crcinit)

### Variables

- static unsigned const [GiNaC::crctab](#) [256]

### 7.17.1 Detailed Description

CRC32 hash function.

Shamelessly stolen from GNU coreutils (cksum.c).

## 7.18 ex.cpp File Reference

Implementation of [GiNaC](#)'s light-weight expression handles.

```
#include "ex.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "numeric.h"
#include "matrix.h"
#include "power.h"
#include "lst.h"
#include "relational.h"
#include "utils.h"
#include <iostream>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

### 7.18.1 Detailed Description

Implementation of [GiNaC](#)'s light-weight expression handles.

## 7.19 ex.h File Reference

Interface to [GiNaC](#)'s light-weight expression handles.

```
#include "basic.h"
#include "ptr.h"
#include <functional>
#include <iosfwd>
#include <iterator>
#include <memory>
#include <stack>
```

## Classes

- class [GiNaC::library\\_init](#)  
*Helper class to initialize the library.*
- class [GiNaC::ex](#)  
*Lightweight wrapper for [GiNaC](#)'s symbolic objects.*
- class [GiNaC::const\\_iterator](#)
- struct [GiNaC::internal::\\_iter\\_rep](#)
- class [GiNaC::const\\_preorder\\_iterator](#)
- class [GiNaC::const\\_postorder\\_iterator](#)
- struct [GiNaC::ex\\_is\\_less](#)
- struct [GiNaC::ex\\_is\\_equal](#)
- struct [GiNaC::op0\\_is\\_equal](#)
- struct [GiNaC::ex\\_swap](#)
- class [GiNaC::pointer\\_to\\_map\\_function](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >](#)
- struct [std::hash< GiNaC::ex >](#)  
*Specialization of `std::hash()` for `ex` objects.*
- struct [std::equal\\_to< GiNaC::ex >](#)  
*Specialization of `std::equal_to()` for `ex` objects.*

## Namespaces

- namespace [GiNaC](#)
- namespace [GiNaC::internal](#)
- namespace [std](#)

## Functions

- bool [GiNaC::are\\_ex\\_trivially\\_equal](#) (const ex &e1, const ex &e2)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const exvector &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const exset &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const exmap &e)
- size\_t [GiNaC::nops](#) (const ex &thisex)
- ex [GiNaC::expand](#) (const ex &thisex, unsigned [options](#)=0)
- ex [GiNaC::conjugate](#) (const ex &thisex)
- ex [GiNaC::real\\_part](#) (const ex &thisex)
- ex [GiNaC::imag\\_part](#) (const ex &thisex)
- bool [GiNaC::has](#) (const ex &thisex, const ex &pattern, unsigned [options](#)=0)
- bool [GiNaC::find](#) (const ex &thisex, const ex &pattern, exset &found)
- bool [GiNaC::is\\_polynomial](#) (const ex &thisex, const ex &vars)
- int [GiNaC::degree](#) (const ex &thisex, const ex &s)
- int [GiNaC::ldegree](#) (const ex &thisex, const ex &s)
- ex [GiNaC::coeff](#) (const ex &thisex, const ex &s, int [n](#)=1)
- ex [GiNaC::numer](#) (const ex &thisex)

- ex [GiNaC::denom](#) (const ex &thisex)
- ex [GiNaC::numer\\_denom](#) (const ex &thisex)
- ex [GiNaC::normal](#) (const ex &thisex)
- ex [GiNaC::to\\_rational](#) (const ex &thisex, exmap &repl)
- ex [GiNaC::to\\_polynomial](#) (const ex &thisex, exmap &repl)
- ex [GiNaC::collect](#) (const ex &thisex, const ex &s, bool distributed=false)
- ex [GiNaC::eval](#) (const ex &thisex)
- ex [GiNaC::evalf](#) (const ex &thisex)
- ex [GiNaC::evalm](#) (const ex &thisex)
- ex [GiNaC::eval\\_integ](#) (const ex &thisex)
- ex [GiNaC::diff](#) (const ex &thisex, const symbol &s, unsigned nth=1)
- ex [GiNaC::series](#) (const ex &thisex, const ex &r, int [order](#), unsigned [options](#)=0)
- bool [GiNaC::match](#) (const ex &thisex, const ex &pattern, exmap &repl\_lst)
- ex [GiNaC::simplify\\_indexed](#) (const ex &thisex, unsigned [options](#)=0)
- ex [GiNaC::simplify\\_indexed](#) (const ex &thisex, const scalar\_products &sp, unsigned [options](#)=0)
- ex [GiNaC::symmetrize](#) (const ex &thisex)
- ex [GiNaC::symmetrize](#) (const ex &thisex, const lst &l)
- ex [GiNaC::antisymmetrize](#) (const ex &thisex)
- ex [GiNaC::antisymmetrize](#) (const ex &thisex, const lst &l)
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &thisex)
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &thisex, const lst &l)
- ex [GiNaC::op](#) (const ex &thisex, size\_t i)
- ex [GiNaC::lhs](#) (const ex &thisex)
- ex [GiNaC::rhs](#) (const ex &thisex)
- bool [GiNaC::is\\_zero](#) (const ex &thisex)
- void [GiNaC::swap](#) (ex &e1, ex &e2)
- ex [GiNaC::subs](#) (const ex &thisex, const exmap &m, unsigned [options](#)=0)
- ex [GiNaC::subs](#) (const ex &thisex, const lst &ls, const lst &lr, unsigned [options](#)=0)
- ex [GiNaC::subs](#) (const ex &thisex, const ex &e, unsigned [options](#)=0)
- template<class T >  
 bool [GiNaC::is\\_a](#) (const ex &obj)  
*Check if ex is a handle to a T, including base classes.*
- template<class T >  
 bool [GiNaC::is\\_exactly\\_a](#) (const ex &obj)  
*Check if ex is a handle to a T, not including base classes.*
- template<class T >  
 const T & [GiNaC::ex\\_to](#) (const ex &e)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- template<> void [std::swap](#) ([GiNaC::ex](#) &a, [GiNaC::ex](#) &b)  
*Specialization of [std::swap\(\)](#) for ex objects.*

## Variables

- static library\_init [GiNaC::library\\_initializer](#)  
*For construction of flyweights, etc.*
- const basic \* [GiNaC::\\_num0\\_bp](#)

### 7.19.1 Detailed Description

Interface to [GiNaC](#)'s light-weight expression handles.



## 7.20 excompiler.cpp File Reference

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

```
#include "excompiler.h"
#include "ex.h"
#include "lst.h"
#include "operators.h"
#include "relational.h"
#include "symbol.h"
#include <cstdlib>
#include <fstream>
#include <ios>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- void [GiNaC::compile\\_ex](#) (const `ex` &expr, const symbol &sym, FUNCP\_1P &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const `ex` &expr, const symbol &sym1, const symbol &sym2, FUNCP\_2P &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const lst &exprs, const lst &symbols, FUNCP\_CUBA &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::link\\_ex](#) (const std::string filename, FUNCP\_1P &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_1P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, FUNCP\_2P &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_2P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, FUNCP\_CUBA &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_CUBA to the contained function.*
- void [GiNaC::unlink\\_ex](#) (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*

#### 7.20.1 Detailed Description

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

## 7.21 excompiler.h File Reference

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

```
#include "lst.h"
#include <string>
```

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef double(\* [GiNaC::FUNCP\\_1P](#)) (double)  
*Function pointer with one function parameter.*
- typedef double(\* [GiNaC::FUNCP\\_2P](#)) (double, double)  
*Function pointer with two function parameters.*
- typedef void(\* [GiNaC::FUNCP\\_CUBA](#)) (const int \*, const double[], const int \*, double[])  
*Function pointer for use with the CUBA library ( <http://www.feynarts.de/cuba>).*

## Functions

- void [GiNaC::compile\\_ex](#) (const ex &expr, const symbol &sym, FUNCP\_1P &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const ex &expr, const symbol &sym1, const symbol &sym2, FUNCP\_2P &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const lst &exprs, const lst &syms, FUNCP\_CUBA &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::link\\_ex](#) (const std::string filename, FUNCP\_1P &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_1P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, FUNCP\_2P &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_2P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, FUNCP\_CUBA &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_CUBA to the contained function.*
- void [GiNaC::unlink\\_ex](#) (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*

### 7.21.1 Detailed Description

Functions to facilitate the conversion of a ex to a function pointer suited for fast numerical integration.

## 7.22 expair.cpp File Reference

Implementation of expression pairs (building blocks of expairseq).

```
#include "expair.h"
#include "operators.h"
#include <iostream>
```

## Namespaces

- namespace [GiNaC](#)

### 7.22.1 Detailed Description

Implementation of expression pairs (building blocks of `expairseq`).

## 7.23 `expair.h` File Reference

Definition of expression pairs (building blocks of `expairseq`).

```
#include "ex.h"
#include "numeric.h"
#include "print.h"
```

### Classes

- class [GiNaC::expair](#)  
*A pair of expressions.*
- struct [GiNaC::expair\\_is\\_less](#)  
*Function object for insertion into third argument of STL's `sort()` etc.*
- struct [GiNaC::expair\\_rest\\_is\\_less](#)  
*Function object not caring about the numerical coefficients for insertion into third argument of STL's `sort()`.*
- struct [GiNaC::expair\\_swap](#)

### Namespaces

- namespace [GiNaC](#)

### Functions

- void [GiNaC::swap](#) (`expair &e1`, `expair &e2`)

### 7.23.1 Detailed Description

Definition of expression pairs (building blocks of `expairseq`).

## 7.24 expairseq.cpp File Reference

Implementation of sequences of expression pairs.

```
#include "expairseq.h"
#include "lst.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "relational.h"
#include "wildcard.h"
#include "archive.h"
#include "operators.h"
#include "utils.h"
#include "hash_seed.h"
#include "indexed.h"
#include "compiler.h"
#include <algorithm>
#include <iostream>
#include <iterator>
#include <memory>
#include <stdexcept>
#include <string>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (expairseq, basic, print\_func< print\_context >(&expairseq::do\_print). print\_func< print\_tree >(&expairseq::do\_print\_tree)) class epp\_is\_less
- `epvector *` [GiNaC::conjugateepvector](#) (const epvector &)

*Complex conjugate every element of an epvector.*

#### 7.24.1 Detailed Description

Implementation of sequences of expression pairs.

## 7.25 expairseq.h File Reference

Interface to sequences of expression pairs.

```
#include "expair.h"
#include "indexed.h"
#include <vector>
```

## Classes

- class [GiNaC::exprseq](#)  
*A sequence of class expair.*
- class [GiNaC::make\\_flat\\_inserter](#)  
*Class to handle the renaming of dummy indices.*

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef `std::vector< expair >` [GiNaC::epvector](#)  
*expair-vector*
- typedef `epvector::iterator` [GiNaC::epp](#)  
*expair-vector pointer*

## Functions

- `epvector *` [GiNaC::conjugateepvector](#) (const epvector &)  
*Complex conjugate every element of an epvector.*

### 7.25.1 Detailed Description

Interface to sequences of expression pairs.

## 7.26 exprseq.cpp File Reference

Implementation of [GiNaC](#)'s exprseq.

```
#include "exprseq.h"
```

## Namespaces

- namespace [GiNaC](#)

### 7.26.1 Detailed Description

Implementation of [GiNaC](#)'s exprseq.

## 7.27 exprseq.h File Reference

Definition of [GiNaC's](#) `exprseq`.

```
#include "container.h"  
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef container< std::vector > [GiNaC::exprseq](#)

### 7.27.1 Detailed Description

Definition of [GiNaC's](#) `exprseq`.

## 7.28 factor.cpp File Reference

Polynomial factorization (implementation).

```
#include "factor.h"  
#include "ex.h"  
#include "numeric.h"  
#include "operators.h"  
#include "inifcns.h"  
#include "symbol.h"  
#include "relational.h"  
#include "power.h"  
#include "mul.h"  
#include "normal.h"  
#include "add.h"  
#include <type_traits>  
#include <algorithm>  
#include <limits>  
#include <list>  
#include <vector>  
#include <stack>  
#include <cln/cln.h>
```

### Namespaces

- namespace [GiNaC](#)

## Macros

- `#define DCOUT(str)`
- `#define DCOUTVAR(var)`
- `#define DCOUT2(str, var)`
- `#define USE_SAME_DEGREE_FACTOR`

## Functions

- `ex GiNaC::factor` (const ex &poly, unsigned options)

*Interface function to the outside world.*

### 7.28.1 Detailed Description

Polynomial factorization (implementation).

The interface function `factor()` at the end of this file is defined in the `GiNaC` namespace. All other utility functions and classes are defined in an additional anonymous namespace.

Factorization starts by doing a square free factorization and making the coefficients integer. Then, depending on the number of free variables it proceeds either in dedicated univariate or multivariate factorization code.

Univariate factorization does a modular factorization via Berlekamp's algorithm and distinct degree factorization. Hensel lifting is used at the end.

Multivariate factorization uses the univariate factorization (applying a evaluation homomorphism first) and Hensel lifting raises the answer to the multivariate domain. The Hensel lifting code is completely distinct from the code used by the univariate factorization.

Algorithms used can be found in [Wan] An Improved Multivariate Polynomial Factoring Algorithm, P.S.Wang, Mathematics of Computation, Vol. 32, No. 144 (1978) 1215–1231. [GCL] Algorithms for Computer Algebra, K.O.Geddes, S.R.Czapor, G.Labahn, Springer Verlag, 1992. [Mig] Some Useful Bounds, M.Mignotte, In "Computer Algebra, Symbolic and Algebraic Computation" (B.Buchberger et al., eds.), pp. 259-263, Springer-Verlag, New York, 1982.

### 7.28.2 Macro Definition Documentation

#### 7.28.2.1 DCOUT

```
#define DCOUT(  
    str )
```

#### 7.28.2.2 DCOUTVAR

```
#define DCOUTVAR(  
    var )
```

### 7.28.2.3 DCOUT2

```
#define DCOUT2(
    str,
    var )
```

### 7.28.2.4 USE\_SAME\_DEGREE\_FACTOR

```
#define USE_SAME_DEGREE_FACTOR
```

## 7.28.3 Variable Documentation

### 7.28.3.1 value

```
const bool value = false [static]
```

Referenced by [GiNaC::archive\\_node::add\\_bool\(\)](#), [GiNaC::archive\\_node::add\\_ex\(\)](#), [GiNaC::archive\\_node::add\\_string\(\)](#), [GiNaC::archive\\_node::add\\_unsigned\(\)](#), [GiNaC::Li2\\_\(\)](#), [GiNaC::matrix::set\(\)](#), and [GiNaC::subsvalue\(\)](#).

### 7.28.3.2 r

```
size_t r [private]
```

Referenced by [GiNaC::matrix::charpoly\(\)](#), [GiNaC::collect\\_common\\_factors\(\)](#), [GiNaC::ex::content\(\)](#), [GiNaC::numeric::csgn\(\)](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [GiNaC::matrix::echelon\\_form\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::idx\\_symmetrization\(\)](#), [GiNaC::matrix::inverse\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::matrix::mul\(\)](#), [GiNaC::matrix::mul\\_scalar\(\)](#), [GiNaC::numeric::numer\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::numeric::print\\_numeric\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::matrix::rank\(\)](#), [GiNaC::reduced\\_matrix\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::return\\_type\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::fderivative::series\(\)](#), [GiNaC::function::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::integration\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::refcounted::set\\_refcount\(\)](#), [GiNaC::matrix::solve\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::sr\\_gcd\(\)](#), [GiNaC::numeric::step\(\)](#), [GiNaC::sub\\_matrix\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::matrix::subs\(\)](#), [GiNaC::symbolic\\_matrix\(\)](#), [GiNaC::matrix::trace\(\)](#), [GiNaC::matrix::transpose\(\)](#), [GiNaC::unit\\_matrix\(\)](#), and [GiNaC::zeta1\\_evalf\(\)](#).



## 7.28.3.3 c

size\_t c [private]

Referenced by [GiNaC::abs\\_print\\_csrc\\_float\(\)](#), [GiNaC::abs\\_print\\_latex\(\)](#), [GiNaC::symmetry::add\(\)](#), [GiNaC::bernoulli\(\)](#), [GiNaC::matrix::charpoly\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::color\\_d\(\)](#), [GiNaC::color\\_f\(\)](#), [GiNaC::color\\_h\(\)](#), [GiNaC::expairseq::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [GiNaC::expairseq::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::conjugate\\_print\\_latex\(\)](#), [GiNaC::ex::content\(\)](#), [GiNaC::crc32\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [GiNaC::add::do\\_print\(\)](#), [GiNaC::basic::do\\_print\(\)](#), [GiNaC::constant::do\\_print\(\)](#), [GiNaC::container< C >::do\\_print\(\)](#), [GiNaC::expairseq::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::idx::do\\_print\(\)](#), [GiNaC::varidx::do\\_print\(\)](#), [GiNaC::spinidx::do\\_print\(\)](#), [GiNaC::indexed::do\\_print\(\)](#), [GiNaC::integral::do\\_print\(\)](#), [GiNaC::integration\\_kernel::do\\_print\(\)](#), [GiNaC::basic\\_log\\_kernel::do\\_print\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::do\\_print\(\)](#), [GiNaC::ELi\\_kernel::do\\_print\(\)](#), [GiNaC::Ebar\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::do\\_print\(\)](#), [GiNaC::modular\\_form\\_kernel::do\\_print\(\)](#), [GiNaC::user\\_defined\\_kernel::do\\_print\(\)](#), [GiNaC::matrix::do\\_print\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::ncmul::do\\_print\(\)](#), [GiNaC::numeric::do\\_print\(\)](#), [GiNaC::pseries::do\\_print\(\)](#), [GiNaC::relational::do\\_print\(\)](#), [GiNaC::symbol::do\\_print\(\)](#), [GiNaC::symmetry::do\\_print\(\)](#), [GiNaC::wildcard::do\\_print\(\)](#), [GiNaC::ncmul::do\\_print\\_csrc\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::numeric::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::numeric::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::clifford::do\\_print\\_dfft\(\)](#), [GiNaC::power::do\\_print\\_dfft\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::add::do\\_print\\_latex\(\)](#), [GiNaC::clifford::do\\_print\\_latex\(\)](#), [GiNaC::constant::do\\_print\\_latex\(\)](#), [GiNaC::idx::do\\_print\\_latex\(\)](#), [GiNaC::spinidx::do\\_print\\_latex\(\)](#), [GiNaC::indexed::do\\_print\\_latex\(\)](#), [GiNaC::integral::do\\_print\\_latex\(\)](#), [GiNaC::matrix::do\\_print\\_latex\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [GiNaC::numeric::do\\_print\\_latex\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::pseries::do\\_print\\_latex\(\)](#), [GiNaC::symbol::do\\_print\\_latex\(\)](#), [GiNaC::container< C >::do\\_print\\_python\(\)](#), [GiNaC::power::do\\_print\\_python\(\)](#), [GiNaC::pseries::do\\_print\\_python\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::basic::do\\_print\\_python\\_repr\(\)](#), [GiNaC::constant::do\\_print\\_python\\_repr\(\)](#), [GiNaC::container< C >::do\\_print\\_python\\_repr\(\)](#), [GiNaC::matrix::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::numeric::do\\_print\\_python\\_repr\(\)](#), [GiNaC::power::do\\_print\\_python\\_repr\(\)](#), [GiNaC::pseries::do\\_print\\_python\\_repr\(\)](#), [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#), [GiNaC::symbol::do\\_print\\_python\\_repr\(\)](#), [GiNaC::wildcard::do\\_print\\_python\\_repr\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [GiNaC::matrix::echelon\\_form\(\)](#), [GiNaC::EllipticE\\_print\\_latex\(\)](#), [GiNaC::EllipticK\\_print\\_latex\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::factorial\\_print\\_dfft\\_latex\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::imag\\_part\\_print\\_latex\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::matrix::inverse\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::matrix::mul\(\)](#), [GiNaC::matrix::mul\\_scalar\(\)](#), [GiNaC::print\\_functor::operator\(\)](#), [GiNaC::print\\_ptrfun\\_handler< T, C >::operator\(\)](#), [GiNaC::print\\_memfun\\_handler< T, C >::operator\(\)](#), [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::ex::primpart\(\)](#), [GiNaC::basic::print\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::fderivative::print\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::print\(\)](#), [GiNaC::expair::print\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::basic::print\\_dispatch\(\)](#), [GiNaC::matrix::print\\_elements\(\)](#), [GiNaC::idx::print\\_index\(\)](#), [GiNaC::indexed::print\\_indexed\(\)](#), [GiNaC::print\\_integer\\_csrc\(\)](#), [GiNaC::numeric::print\\_numeric\(\)](#), [GiNaC::print\\_operator\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::power::print\\_power\(\)](#), [GiNaC::print\\_real\\_cl\\_N\(\)](#), [GiNaC::print\\_real\\_csrc\(\)](#), [GiNaC::print\\_real\\_number\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::expairseq::printseq\(\)](#), [GiNaC::container< C >::printseq\(\)](#), [GiNaC::numeric::read\\_archive\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::real\\_part\\_print\\_latex\(\)](#), [GiNaC::reduced\\_matrix\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), [GiNaC::set\\_print\\_context\(\)](#), [GiNaC::matrix::solve\(\)](#), [GiNaC::sr\\_gcd\(\)](#), [GiNaC::sub\\_matrix\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::matrix::subs\(\)](#), [GiNaC::symbolic\\_matrix\(\)](#), [GiNaC::matrix::transpose\(\)](#), [GiNaC::ex::unit\(\)](#), [GiNaC::unit\\_matrix\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [GiNaC::zeta1\\_print\\_latex\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

## 7.28.3.4 m

mvec m [private]

Referenced by [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::charpoly\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [GiNaC::cols\(\)](#), [GiNaC::convert\\_H\\_to\\_Li\(\)](#), [GiNaC::determinant\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::evalf\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::fibonacci\(\)](#), [GiNaC::H\\_deriv\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::H\\_series\(\)](#), [GiNaC::inverse\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::Order\\_eval\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::rank\(\)](#), [GiNaC::reduced\\_matrix\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::rows\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::smod\(\)](#), [GiNaC::sub\\_matrix\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::basic::subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::container< C >::subs\(\)](#), [GiNaC::expairseq::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::numeric::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::pseries::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::subs\(\)](#), [GiNaC::symbol::subs\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [GiNaC::container< C >::subschildren\(\)](#), [GiNaC::expairseq::subschildren\(\)](#), [GiNaC::tgamma\\_series\(\)](#), [GiNaC::trace\(\)](#), [GiNaC::transpose\(\)](#), [GiNaC::zeta1\\_deriv\(\)](#), [GiNaC::zeta1\\_eval\(\)](#), [GiNaC::zeta1\\_print\\_latex\(\)](#), [GiNaC::zeta2\\_deriv\(\)](#), [GiNaC::zeta2\\_eval\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

### 7.28.3.5 lr

```
umodpoly lr[2] [private]
```

Referenced by [GiNaC::subs\(\)](#), and [GiNaC::ex::subs\(\)](#).

### 7.28.3.6 cache

```
vector<vector<umodpoly> > cache [private]
```

### 7.28.3.7 factors

```
upvec factors [private]
```

Referenced by [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::sqrfree\(\)](#), and [GiNaC::sqrfree\\_yun\(\)](#).

### 7.28.3.8 one

```
umodpoly one [private]
```

Referenced by [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#).

**7.28.3.9 n**

size\_t n [private]

Referenced by [GiNaC::class\\_info< OPT >::tree\\_node::add\\_child\(\)](#), [GiNaC::archive::add\\_node\(\)](#), [GiNaC::basic::archive\(\)](#), [GiNaC::clifford::archive\(\)](#), [GiNaC::color::archive\(\)](#), [GiNaC::constant::archive\(\)](#), [GiNaC::container< C >::archive\(\)](#), [GiNaC::expairseq::archive\(\)](#), [GiNaC::fderivative::archive\(\)](#), [GiNaC::function::archive\(\)](#), [GiNaC::idx::archive\(\)](#), [GiNaC::varidx::archive\(\)](#), [GiNaC::spinidx::archive\(\)](#), [GiNaC::indexed::archive\(\)](#), [GiNaC::integral::archive\(\)](#), [GiNaC::matrix::archive\(\)](#), [GiNaC::numeric::archive\(\)](#), [GiNaC::power::archive\(\)](#), [GiNaC::pseries::archive\(\)](#), [GiNaC::relational::archive\(\)](#), [GiNaC::symbol::archive\(\)](#), [GiNaC::symmetry::archive\(\)](#), [GiNaC::minkmetric::archive\(\)](#), [GiNaC::tensepsilon::archive\(\)](#), [GiNaC::wildcard::archive\(\)](#), [GiNaC::archive::archive\(\)](#), [GiNaC::bernoulli\(\)](#), [GiNaC::binomial\(\)](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::numeric::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::coeff\(\)](#), [GiNaC::coeff\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::const\\_postorder\\_iterator::const\\_postorder\\_iterator\(\)](#), [GiNaC::const\\_preorder\\_iterator::const\\_preorder\\_iterator\(\)](#), [GiNaC::composition\\_generator::coolmulti::coolmulti\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::dirichlet\\_character\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [GiNaC::doublefactorial\(\)](#), [GiNaC::class\\_info< OPT >::dump\\_tree\(\)](#), [GiNaC::matrix::echelon\\_form\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::fibonacci\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::function\\_options::function\\_options\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::ifactor\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [GiNaC::log2\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::basic::map\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::const\\_iterator::operator+\(\)](#), [GiNaC::const\\_iterator::operator+=\(\)](#), [GiNaC::const\\_iterator::operator-\(\)](#), [GiNaC::const\\_iterator::operator-=\(\)](#), [GiNaC::const\\_iterator::operator\[\]\(\)](#), [GiNaC::primitive\\_dirichlet\\_character\(\)](#), [GiNaC::psi2\\_deriv\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_evalf\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::clifford::read\\_archive\(\)](#), [GiNaC::color::read\\_archive\(\)](#), [GiNaC::container< C >::read\\_archive\(\)](#), [GiNaC::constant::read\\_archive\(\)](#), [GiNaC::expairseq::read\\_archive\(\)](#), [GiNaC::fderivative::read\\_archive\(\)](#), [GiNaC::function::read\\_archive\(\)](#), [GiNaC::idx::read\\_archive\(\)](#), [GiNaC::varidx::read\\_archive\(\)](#), [GiNaC::spinidx::read\\_archive\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [GiNaC::integral::read\\_archive\(\)](#), [GiNaC::matrix::read\\_archive\(\)](#), [GiNaC::numeric::read\\_archive\(\)](#), [GiNaC::power::read\\_archive\(\)](#), [GiNaC::pseries::read\\_archive\(\)](#), [GiNaC::relational::read\\_archive\(\)](#), [GiNaC::symbol::read\\_archive\(\)](#), [GiNaC::symmetry::read\\_archive\(\)](#), [GiNaC::minkmetric::read\\_archive\(\)](#), [GiNaC::tensepsilon::read\\_archive\(\)](#), [GiNaC::wildcard::read\\_archive\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::S\\_deriv\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::symbol::set\\_name\(\)](#), [GiNaC::function\\_options::set\\_name\(\)](#), [GiNaC::symbol::set\\_TeX\\_name\(\)](#), [GiNaC::matrix::solve\(\)](#), [GiNaC::sqrfree\\_parrfrac\(\)](#), [GiNaC::function\\_options::test\\_and\\_set\\_nparams\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [GiNaC::ex::traverse\\_postorder\(\)](#), [GiNaC::ex::traverse\\_preorder\(\)](#), [GiNaC::symmetry::validate\(\)](#), [GiNaC::write\\_real\\_float\(\)](#), [GiNaC::zetaderiv\\_deriv\(\)](#), and [GiNaC::zetaderiv\\_eval\(\)](#).

**7.28.3.10 len**

size\_t len [private]

Referenced by [GiNaC::crc32\(\)](#).

**7.28.3.11 last**

size\_t last [private]

Referenced by [GiNaC::antisymmetrize\(\)](#), [GiNaC::expairseq::combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [GiNaC::expairseq::construct\\_from\\_permutation\(\)](#), [GiNaC::cyclic\\_permutation\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::permutation\\_sign\(\)](#), [GiNaC::shaker\\_sort\(\)](#), [GiNaC::expairseq::subchildren\(\)](#), [GiNaC::symm\(\)](#), [GiNaC::symmetrize\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

### 7.28.3.12 k

```
vector<int> k [private]
```

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::binomial\(\)](#), [GiNaC::scalar\\_products::debugprint\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::EllipticE\\_deriv\(\)](#), [GiNaC::EllipticE\\_eval\(\)](#), [GiNaC::EllipticE\\_evalf\(\)](#), [GiNaC::EllipticE\\_print\\_latex\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_deriv\(\)](#), [GiNaC::EllipticK\\_eval\(\)](#), [GiNaC::EllipticK\\_evalf\(\)](#), [GiNaC::EllipticK\\_print\\_latex\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::get\\_sign\(\)](#), [GiNaC::log2\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::basic\\_partition\\_generator::mpartition2::mpartition2\(\)](#), [GiNaC::basic\\_partition\\_generator::composition\\_generator::coolmulti::next\\_permutation\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::operator++\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::ELi\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Ebar\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::sqrfree\\_parfrac\(\)](#).

### 7.28.3.13 poly

```
const ex poly
```

Referenced by [GiNaC::matrix::charpoly\(\)](#), and [GiNaC::factor\(\)](#).

### 7.28.3.14 x

```
const ex x
```

Referenced by [GiNaC::abs\(\)](#), [GiNaC::acos\(\)](#), [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acos\\_deriv\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acos\\_evalf\(\)](#), [GiNaC::acosh\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::acosh\\_deriv\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::acosh\\_evalf\(\)](#), [GiNaC::adaptivesimpson\(\)](#), [GiNaC::asin\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asin\\_deriv\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asin\\_evalf\(\)](#), [GiNaC::asin\\_info\(\)](#), [GiNaC::asinh\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::asinh\\_deriv\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::asinh\\_evalf\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::atan2\\_deriv\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan2\\_evalf\(\)](#), [GiNaC::atan2\\_info\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atan\\_deriv\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_evalf\(\)](#), [GiNaC::atan\\_info\(\)](#), [GiNaC::atanh\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::atanh\\_deriv\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::atanh\\_evalf\(\)](#), [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::beta\\_deriv\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::beta\\_evalf\(\)](#), [GiNaC::binomial\\_conjugate\(\)](#), [GiNaC::binomial\\_eval\(\)](#), [GiNaC::binomial\\_evalf\(\)](#), [GiNaC::binomial\\_real\\_part\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::lanczos\\_coeffs::calc\\_lanczos\\_A\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::expairseq::conjugate\(\)](#), [GiNaC::matrix::conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::ex::content\(\)](#), [GiNaC::convert\\_H\\_to\\_Li\(\)](#), [GiNaC::cos\(\)](#), [GiNaC::cos\\_conjugate\(\)](#), [GiNaC::cos\\_deriv\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cos\\_evalf\(\)](#), [GiNaC::cos\\_imag\\_part\(\)](#), [GiNaC::cos\\_real\\_part\(\)](#), [GiNaC::cosh\(\)](#), [GiNaC::cosh\\_conjugate\(\)](#), [GiNaC::cosh\\_deriv\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::cosh\\_evalf\(\)](#), [GiNaC::cosh\\_imag\\_part\(\)](#), [GiNaC::cosh\\_real\\_part\(\)](#), [GiNaC::csgn\(\)](#), [GiNaC::decomp\\_rational\(\)](#), [GiNaC::denom\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::eta\\_conjugate\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [GiNaC::eta\\_imag\\_part\(\)](#), [GiNaC::eta\\_series\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::exp\\_conjugate\(\)](#), [GiNaC::exp\\_deriv\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::exp\\_evalf\(\)](#), [GiNaC::exp\\_imag\\_part\(\)](#), [GiNaC::exp\\_info\(\)](#), [GiNaC::exp\\_power\(\)](#), [GiNaC::exp\\_real\\_part\(\)](#), [GiNaC::factorial\\_conjugate\(\)](#), [GiNaC::factorial\\_eval\(\)](#), [GiNaC::factorial\\_evalf\(\)](#), [GiNaC::factorial\\_print\\_dflt\\_latex\(\)](#), [GiNaC::factorial\\_real\\_part\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::G\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [GiNaC::guess\\_precision\(\)](#), [GiNaC::H\\_deriv\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::H\\_series\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::hasindex\(\)](#),

[GiNaC::haswild\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::imag\(\)](#), [GiNaC::interpolate\(\)](#), [GiNaC::inverse\(\)](#), [GiNaC::is\\_cinteger\(\)](#),  
[GiNaC::is\\_crational\(\)](#), [GiNaC::is\\_even\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::is\\_negative\(\)](#), [GiNaC::is\\_nonneg\\_integer\(\)](#),  
[GiNaC::is\\_odd\(\)](#), [GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::is\\_positive\(\)](#), [GiNaC::is\\_prime\(\)](#), [GiNaC::is\\_rational\(\)](#),  
[GiNaC::is\\_real\(\)](#), [GiNaC::is\\_the\\_function\(\)](#), [GiNaC::is\\_the\\_function< G\\_SERIAL >\(\)](#), [GiNaC::is\\_the\\_function< iterated\\_integral\\_SERIAL >\(\)](#),  
[GiNaC::is\\_the\\_function< psi\\_SERIAL >\(\)](#), [GiNaC::is\\_the\\_function< zeta\\_SERIAL >\(\)](#), [GiNaC::is\\_zero\(\)](#),  
[GiNaC::isqrt\(\)](#), [GiNaC::Eisenstein\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::Laurent\\_series\(\)](#),  
[GiNaC::lgamma\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::lgamma\\_deriv\(\)](#), [GiNaC::lgamma\\_eval\(\)](#), [GiNaC::lgamma\\_evalf\(\)](#),  
[GiNaC::Li2\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_deriv\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_evalf\(\)](#), [GiNaC::Li2\\_projection\(\)](#),  
[GiNaC::Li2\\_series\(\)](#), [GiNaC::Li3\\_eval\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#),  
[GiNaC::Li\\_series\(\)](#), [GiNaC::log\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::log\\_deriv\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_evalf\(\)](#),  
[GiNaC::log\\_imag\\_part\(\)](#), [GiNaC::log\\_info\(\)](#), [GiNaC::log\\_real\\_part\(\)](#), [GiNaC::make\\_real\\_float\(\)](#), [GiNaC::matrix::matrix\(\)](#),  
[GiNaC::numer\(\)](#), [GiNaC::sym\\_desc::operator<\(\)](#), [GiNaC::Order\\_conjugate\(\)](#), [GiNaC::Order\\_eval\(\)](#), [GiNaC::Order\\_imag\\_part\(\)](#),  
[GiNaC::Order\\_real\\_part\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::pow\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::ex::primpart\(\)](#),  
[GiNaC::print\\_integer\\_csrc\(\)](#), [GiNaC::print\\_real\\_cl\\_N\(\)](#), [GiNaC::print\\_real\\_csrc\(\)](#), [GiNaC::print\\_real\\_number\(\)](#),  
[GiNaC::print\\_sym\\_pow\(\)](#), [GiNaC::psi1\\_deriv\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi1\\_evalf\(\)](#), [GiNaC::psi2\\_deriv\(\)](#),  
[GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_evalf\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::read\\_real\\_float\(\)](#), [GiNaC::real\(\)](#), [GiNaC::rem\(\)](#),  
[GiNaC::S\\_deriv\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::sin\(\)](#),  
[GiNaC::sin\\_conjugate\(\)](#), [GiNaC::sin\\_deriv\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sin\\_evalf\(\)](#), [GiNaC::sin\\_imag\\_part\(\)](#),  
[GiNaC::sin\\_real\\_part\(\)](#), [GiNaC::sinh\(\)](#), [GiNaC::sinh\\_conjugate\(\)](#), [GiNaC::sinh\\_deriv\(\)](#), [GiNaC::sinh\\_eval\(\)](#),  
[GiNaC::sinh\\_evalf\(\)](#), [GiNaC::sinh\\_imag\\_part\(\)](#), [GiNaC::sinh\\_real\\_part\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\(\)](#),  
[GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::sqrfree\\_yun\(\)](#), [GiNaC::sqrt\(\)](#), [GiNaC::sr\\_gcd\(\)](#), [GiNaC::step\(\)](#), [GiNaC::tan\(\)](#),  
[GiNaC::tan\\_conjugate\(\)](#), [GiNaC::tan\\_deriv\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tan\\_evalf\(\)](#), [GiNaC::tan\\_imag\\_part\(\)](#),  
[GiNaC::tan\\_real\\_part\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\(\)](#), [GiNaC::tanh\\_conjugate\(\)](#), [GiNaC::tanh\\_deriv\(\)](#),  
[GiNaC::tanh\\_eval\(\)](#), [GiNaC::tanh\\_evalf\(\)](#), [GiNaC::tanh\\_imag\\_part\(\)](#), [GiNaC::tanh\\_real\\_part\(\)](#), [GiNaC::tanh\\_series\(\)](#),  
[GiNaC::tgamma\(\)](#), [GiNaC::tgamma\\_conjugate\(\)](#), [GiNaC::tgamma\\_deriv\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [GiNaC::tgamma\\_evalf\(\)](#),  
[GiNaC::to\\_double\(\)](#), [GiNaC::to\\_int\(\)](#), [GiNaC::to\\_long\(\)](#), [GiNaC::trig\\_info\(\)](#), [GiNaC::ex::unit\(\)](#), [GiNaC::unit\\_matrix\(\)](#),  
[GiNaC::ex::unitcontprim\(\)](#), [GiNaC::zeta\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), [GiNaC::zeta2\\_evalf\(\)](#), [GiNaC::zetaderiv\\_deriv\(\)](#),  
and [GiNaC::zetaderiv\\_eval\(\)](#).

### 7.28.3.15 evalpoint

```
int evalpoint
```

### 7.28.3.16 R

```
cl_modint_ring R
```

### 7.28.3.17 syms\_wox

```
const exset syms_wox
```

#### 7.28.3.18 unit

`ex unit`

Referenced by [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::ex::primpart\(\)](#), and [GiNaC::ex::unitcontprim\(\)](#).

#### 7.28.3.19 cont

`ex cont`

Referenced by [GiNaC::ex::content\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), and [GiNaC::container< C >::real\\_part\(\)](#).

#### 7.28.3.20 pp

`ex pp`

#### 7.28.3.21 vn

`ex vn`

#### 7.28.3.22 vnlst

`exvector vnlst`

#### 7.28.3.23 modulus

`numeric modulus`

#### 7.28.3.24 syms

`exset syms`

Referenced by [GiNaC::lsolve\(\)](#).

### 7.28.3.25 options

unsigned options

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::csgn\\_series\(\)](#), [GiNaC::determinant\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::basic::expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::add::expand\(\)](#), [GiNaC::expairseq::expand\(\)](#), [GiNaC::function::expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::factor\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::basic::has\(\)](#), [GiNaC::mul::has\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::has\(\)](#), [GiNaC::ex::has\(\)](#), [GiNaC::has\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expand\\_map\\_function::operator\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::registered\\_class\\_options::print\\_func\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::fderivative::series\(\)](#), [GiNaC::function::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::set\\_print\\_context\(\)](#), [GiNaC::set\\_print\\_func\(\)](#), [GiNaC::set\\_print\\_options\(\)](#), [GiNaC::simplify\\_indexed\(\)](#), [GiNaC::step\\_series\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::basic::subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::container< C >::subs\(\)](#), [GiNaC::expairseq::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::matrix::subs\(\)](#), [GiNaC::numeric::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::pseries::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::subs\(\)](#), [GiNaC::symbol::subs\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [GiNaC::container< C >::subschildren\(\)](#), [GiNaC::expairseq::subschildren\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_series\(\)](#), and [GiNaC::tgamma\\_series\(\)](#).

## 7.29 factor.h File Reference

Polynomial factorization.

### Namespaces

- namespace [GiNaC](#)

### Functions

- ex [GiNaC::factor](#) (const ex &poly, unsigned options)

*Interface function to the outside world.*

### 7.29.1 Detailed Description

Polynomial factorization.

## 7.30 fail.cpp File Reference

Implementation of class signaling failure of operation.

```
#include "fail.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (fail, basic, print\_func< print\_context >(&fail←→::do\_print). print\_func< print\_tree >(&fail::do\_print\_tree)) [GINAC\\_BIND\\_UNARCHIVER](#)(fail)

### 7.30.1 Detailed Description

Implementation of class signaling failure of operation.

Considered somewhat obsolete (most of this can be replaced by exceptions).

## 7.31 fail.h File Reference

Interface to class signaling failure of operation.

```
#include "basic.h"  
#include "archive.h"
```

## Classes

- class [GiNaC::fail](#)

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (fail)

### 7.31.1 Detailed Description

Interface to class signaling failure of operation.

Considered obsolete somewhat obsolete (most of this can be replaced by exceptions).



## 7.32 fderivative.cpp File Reference

Implementation of abstract derivatives of functions.

```
#include "fderivative.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (fderivative, function, print\_func< print\_context >(&fderivative::do\_print). print\_func< print\_latex >(&fderivative::do\_print\_latex). print\_func< print\_csrc >(&fderivative::do\_print\_csrc). print\_func< print\_tree >(&fderivative::do\_print\_tree)) fderivative
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (fderivative)

#### 7.32.1 Detailed Description

Implementation of abstract derivatives of functions.

## 7.33 fderivative.h File Reference

Interface to abstract derivatives of functions.

```
#include "function.h"
#include <set>
```

### Classes

- class [GiNaC::fderivative](#)  
*This class represents the (abstract) derivative of a symbolic function.*

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef std::multiset< unsigned > [GiNaC::paramset](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (fderivative)

### 7.33.1 Detailed Description

Interface to abstract derivatives of functions.

## 7.34 flags.h File Reference

Collection of all flags used through the [GiNaC](#) framework.

## Classes

- class [GiNaC::expand\\_options](#)  
*Flags to control the behavior of [expand\(\)](#).*
- class [GiNaC::has\\_options](#)  
*Flags to control the behavior of [has\(\)](#).*
- class [GiNaC::subs\\_options](#)  
*Flags to control the behavior of [subs\(\)](#).*
- class [GiNaC::domain](#)  
*Domain of an object.*
- class [GiNaC::series\\_options](#)  
*Flags to control series expansion.*
- class [GiNaC::determinant\\_algo](#)  
*Switch to control algorithm for determinant computation.*
- class [GiNaC::solve\\_algo](#)  
*Switch to control algorithm for linear system solving.*
- class [GiNaC::status\\_flags](#)  
*Flags to store information about the state of an object.*
- class [GiNaC::info\\_flags](#)  
*Possible attributes an object can have.*
- class [GiNaC::return\\_types](#)
- class [GiNaC::remember\\_strategies](#)  
*Strategies how to clean up the function remember cache.*
- class [GiNaC::factor\\_options](#)  
*Flags to control the polynomial factorization.*

## Namespaces

- namespace [GiNaC](#)

### 7.34.1 Detailed Description

Collection of all flags used through the [GiNaC](#) framework.

## 7.35 function.cpp File Reference

Implementation of class of symbolic functions.

```
#include "function.h"
#include "operators.h"
#include "fderivative.h"
#include "ex.h"
#include "lst.h"
#include "symmetry.h"
#include "print.h"
#include "power.h"
#include "archive.h"
#include "inifcns.h"
#include "utils.h"
#include "hash_seed.h"
#include "remember.h"
#include <iostream>
#include <limits>
#include <list>
#include <stdexcept>
#include <string>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (function)

#### 7.35.1 Detailed Description

Implementation of class of symbolic functions.

## 7.36 function.h File Reference

Interface to class of symbolic functions.

```
#include "exprseq.h"
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::function\\_options](#)
- class [GiNaC::do\\_taylor](#)

*Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.*

- class [GiNaC::function](#)

*The class function is used to implement builtin functions like sin, cos... and user defined functions.*

## Namespaces

- namespace [GiNaC](#)

## Macros

- #define [DECLARE\\_FUNCTION\\_1P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_2P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_3P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_4P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_5P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_6P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_7P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_8P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_9P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_10P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_11P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_12P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_13P](#)(NAME)
- #define [DECLARE\\_FUNCTION\\_14P](#)(NAME)
- #define [REGISTER\\_FUNCTION](#)(NAME, OPT)
- #define [is\\_ex\\_the\\_function](#)(OBJ, FUNCNAME) ([GiNaC::is\\_the\\_function](#)<FUNCNAME##\_SERIAL>(OBJ))

## Typedefs

- typedef ex(\* [GiNaC::eval\\_funcp](#)) ()
- typedef ex(\* [GiNaC::evalf\\_funcp](#)) ()
- typedef ex(\* [GiNaC::conjugate\\_funcp](#)) ()
- typedef ex(\* [GiNaC::real\\_part\\_funcp](#)) ()
- typedef ex(\* [GiNaC::imag\\_part\\_funcp](#)) ()
- typedef ex(\* [GiNaC::expand\\_funcp](#)) ()
- typedef ex(\* [GiNaC::derivative\\_funcp](#)) ()
- typedef ex(\* [GiNaC::expl\\_derivative\\_funcp](#)) ()
- typedef ex(\* [GiNaC::power\\_funcp](#)) ()
- typedef ex(\* [GiNaC::series\\_funcp](#)) ()
- typedef void(\* [GiNaC::print\\_funcp](#)) ()
- typedef bool(\* [GiNaC::info\\_funcp](#)) ()
- typedef ex(\* [GiNaC::eval\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::evalf\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::conjugate\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::real\\_part\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::imag\\_part\\_funcp\\_1](#)) (const ex &)
- typedef ex(\* [GiNaC::expand\\_funcp\\_1](#)) (const ex &, unsigned)

- [illegible]



[illegible]







- Generated by Doxygen

- typedef ex(\* [GiNaC::evalf\\_funcp\\_exvector](#)) (const exvector &)
- typedef ex(\* [GiNaC::conjugate\\_funcp\\_exvector](#)) (const exvector &)
- typedef ex(\* [GiNaC::real\\_part\\_funcp\\_exvector](#)) (const exvector &)
- typedef ex(\* [GiNaC::imag\\_part\\_funcp\\_exvector](#)) (const exvector &)
- typedef ex(\* [GiNaC::expand\\_funcp\\_exvector](#)) (const exvector &, unsigned)
- typedef ex(\* [GiNaC::derivative\\_funcp\\_exvector](#)) (const exvector &, unsigned)
- typedef ex(\* [GiNaC::expl\\_derivative\\_funcp\\_exvector](#)) (const exvector &, const symbol &)
- typedef ex(\* [GiNaC::power\\_funcp\\_exvector](#)) (const exvector &, const ex &)
- typedef ex(\* [GiNaC::series\\_funcp\\_exvector](#)) (const exvector &, const relational &, int, unsigned)
- typedef void(\* [GiNaC::print\\_funcp\\_exvector](#)) (const exvector &, const print\_context &)
- typedef bool(\* [GiNaC::info\\_funcp\\_exvector](#)) (const exvector &, unsigned)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (function)
- template<typename T >  
bool [GiNaC::is\\_the\\_function](#) (const ex &x)

### 7.36.1 Detailed Description

Interface to class of symbolic functions.

### 7.36.2 Macro Definition Documentation

#### 7.36.2.1 DECLARE\_FUNCTION\_1P

```
#define DECLARE_FUNCTION_1P (
    NAME )
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 1; \
template< typename T1 > const GiNaC::function NAME( const T1 & p1 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1) ); \
}
```

#### 7.36.2.2 DECLARE\_FUNCTION\_2P

```
#define DECLARE_FUNCTION_2P (
    NAME )
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 2; \
template< typename T1, typename T2 > const GiNaC::function NAME( const T1 & p1, const T2 & p2 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2) ); \
}
```

## 7.36.2.3 DECLARE\_FUNCTION\_3P

```
#define DECLARE_FUNCTION_3P (
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 3; \
template< typename T1, typename T2, typename T3 > const GiNaC::function NAME( const T1 & p1, const T2 & p2,
    const T3 & p3 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3) ); \
}
```

## 7.36.2.4 DECLARE\_FUNCTION\_4P

```
#define DECLARE_FUNCTION_4P (
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 4; \
template< typename T1, typename T2, typename T3, typename T4 > const GiNaC::function NAME( const T1 & p1,
    const T2 & p2, const T3 & p3, const T4 & p4 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4)
    ); \
}
```

## 7.36.2.5 DECLARE\_FUNCTION\_5P

```
#define DECLARE_FUNCTION_5P (
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 5; \
template< typename T1, typename T2, typename T3, typename T4, typename T5 > const GiNaC::function NAME(
    const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5) ); \
}
```

## 7.36.2.6 DECLARE\_FUNCTION\_6P

```
#define DECLARE_FUNCTION_6P (
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 6; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
    T6 & p6 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
    GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6) ); \
}
```

### 7.36.2.7 DECLARE\_FUNCTION\_7P

```
#define DECLARE_FUNCTION_7P (
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 7; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7 > const
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
        T6 & p6, const T7 & p7 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
        GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7) ); \
}
```

### 7.36.2.8 DECLARE\_FUNCTION\_8P

```
#define DECLARE_FUNCTION_8P (
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 8; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8 > const GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4,
    const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
        GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8) ); \
}
```

### 7.36.2.9 DECLARE\_FUNCTION\_9P

```
#define DECLARE_FUNCTION_9P (
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 9; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9 > const GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3,
    const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
        GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9) ); \
}
```

### 7.36.2.10 DECLARE\_FUNCTION\_10P

```
#define DECLARE_FUNCTION_10P (
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 10; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
    typename T8, typename T9, typename T10 > const GiNaC::function NAME( const T1 & p1, const T2 & p2,
    const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 &
    p9, const T10 & p10 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
        GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
        GiNaC::ex(p10) ); \
}
```

### 7.36.2.11 DECLARE\_FUNCTION\_11P

```
#define DECLARE_FUNCTION_11P(  
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 11; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,  
    typename T8, typename T9, typename T10, typename T11 > const GiNaC::function NAME( const T1 & p1,  
    const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 &  
    p8, const T9 & p9, const T10 & p10, const T11 & p11 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),  
        GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),  
        GiNaC::ex(p10), GiNaC::ex(p11) ); \
}
```

### 7.36.2.12 DECLARE\_FUNCTION\_12P

```
#define DECLARE_FUNCTION_12P(  
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 12; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,  
    typename T8, typename T9, typename T10, typename T11, typename T12 > const GiNaC::function NAME( const  
    T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7,  
    const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12 & p12 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),  
        GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),  
        GiNaC::ex(p10), GiNaC::ex(p11), GiNaC::ex(p12) ); \
}
```

### 7.36.2.13 DECLARE\_FUNCTION\_13P

```
#define DECLARE_FUNCTION_13P(  
    NAME )
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 13; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,  
    typename T8, typename T9, typename T10, typename T11, typename T12, typename T13 > const  
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const  
    T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12 &  
    p12, const T13 & p13 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),  
        GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),  
        GiNaC::ex(p10), GiNaC::ex(p11), GiNaC::ex(p12), GiNaC::ex(p13) ); \
}
```

### 7.36.2.14 DECLARE\_FUNCTION\_14P

```
#define DECLARE_FUNCTION_14P(  
    NAME )
```

#### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 14; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,  
    typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14 > const  
    GiNaC::function NAME( const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const  
    T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12 &  
    p12, const T13 & p13, const T14 & p14 ) { \
    return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),  
        GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),  
        GiNaC::ex(p10), GiNaC::ex(p11), GiNaC::ex(p12), GiNaC::ex(p13), GiNaC::ex(p14) ); \
}
```

### 7.36.2.15 REGISTER\_FUNCTION

```
#define REGISTER_FUNCTION(  
    NAME,  
    OPT )
```

#### Value:

```
unsigned NAME##_SERIAL::serial = \
    GiNaC::function::register_new(GiNaC::function_options(#NAME, NAME##_NPARAMS).OPT);
```

### 7.36.2.16 is\_ex\_the\_function

```
#define is_ex_the_function(  
    OBJ,  
    FUNCNAME ) (GiNaC::is_the_function<FUNCNAME##_SERIAL>(OBJ))
```

## 7.37 ginac.h File Reference

This include file includes all other public [GiNaC](#) headers.

```
#include "version.h"  
#include "basic.h"  
#include "ex.h"  
#include "normal.h"  
#include "archive.h"  
#include "print.h"  
#include "constant.h"  
#include "fail.h"  
#include "integral.h"  
#include "lst.h"  
#include "matrix.h"  
#include "numeric.h"  
#include "power.h"  
#include "relational.h"
```

```

#include "structure.h"
#include "symbol.h"
#include "pseries.h"
#include "wildcard.h"
#include "symmetry.h"
#include "expair.h"
#include "expairseq.h"
#include "add.h"
#include "mul.h"
#include "exprseq.h"
#include "function.h"
#include "ncmul.h"
#include "inifcns.h"
#include "fderivative.h"
#include "operators.h"
#include "hash_map.h"
#include "idx.h"
#include "indexed.h"
#include "tensor.h"
#include "color.h"
#include "clifford.h"
#include "factor.h"
#include "integration_kernel.h"
#include "excompiler.h"
#include "parser.h"

```

### 7.37.1 Detailed Description

This include file includes all other public [GiNaC](#) headers.

## 7.38 hash\_map.h File Reference

Replacement for `map<>` using hash tables.

```
#include <unordered_map>
```

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- `template<typename T, class Hash = std::hash<ex>, class KeyEqual = std::equal_to<ex>, class Allocator = std::allocator<std::pair<const ex, T>>>`  
`using GiNaC::exhashmap = std::unordered_map< ex, T, Hash, KeyEqual, Allocator >`

### 7.38.1 Detailed Description

Replacement for `map<>` using hash tables.

## 7.39 hash\_seed.h File Reference

Type-specific hash seed.

```
#include <typeinfo>
#include <cstring>
#include "utils.h"
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- static unsigned [GiNaC::make\\_hash\\_seed](#) (const std::type\_info &tinfo)  
*We need a hash function which gives different values for objects of different types.*

### 7.39.1 Detailed Description

Type-specific hash seed.

## 7.40 idx.cpp File Reference

Implementation of [GiNaC](#)'s indices.

```
#include "idx.h"
#include "symbol.h"
#include "lst.h"
#include "relational.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <iostream>
#include <sstream>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)



## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (idx, basic, print\_func< print\_context >(&idx::do\_print). print\_func< print\_latex >(&idx::do\_print\_latex). print\_func< print\_csrc >(&idx::do\_print\_csrc). print\_func< print\_tree >(&idx::do\_print\_tree)) GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT(varidx
- [GiNaC::print\\_func< print\\_context >](#) (&varidx::do\_print). print\_func< print\_latex >(&varidx
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (idx)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (varidx)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (spinidx)
- bool [GiNaC::is\\_dummy\\_pair](#) (const idx &i1, const idx &i2)  
*Check whether two indices form a dummy pair.*
- bool [GiNaC::is\\_dummy\\_pair](#) (const ex &e1, const ex &e2)  
*Check whether two expressions form a dummy index pair.*
- void [GiNaC::find\\_free\\_and\\_dummy](#) (exvector::const\_iterator it, exvector::const\_iterator itend, exvector &out\_free, exvector &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- ex [GiNaC::minimal\\_dim](#) (const ex &dim1, const ex &dim2)  
*Return the minimum of two index dimensions.*

## Variables

- [GiNaC::idx](#)

### 7.40.1 Detailed Description

Implementation of [GiNaC](#)'s indices.

## 7.41 idx.h File Reference

Interface to [GiNaC](#)'s indices.

```
#include "ex.h"
#include "numeric.h"
```

## Classes

- class [GiNaC::idx](#)  
*This class holds one index of an indexed object.*
- class [GiNaC::varidx](#)  
*This class holds an index with a variance (co- or contravariant).*
- class [GiNaC::spinidx](#)  
*This class holds a spinor index that can be dotted or undotted and that also has a variance.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (idx)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (varidx)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (spinidx)
- bool [GiNaC::is\\_dummy\\_pair](#) (const idx &i1, const idx &i2)  
*Check whether two indices form a dummy pair.*
- bool [GiNaC::is\\_dummy\\_pair](#) (const ex &e1, const ex &e2)  
*Check whether two expressions form a dummy index pair.*
- void [GiNaC::find\\_free\\_and\\_dummy](#) (exvector::const\_iterator it, exvector::const\_iterator itend, exvector &out\_free, exvector &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [GiNaC::find\\_free\\_and\\_dummy](#) (const exvector &v, exvector &out\_free, exvector &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [GiNaC::find\\_dummy\\_indices](#) (const exvector &v, exvector &out\_dummy)  
*Given a vector of indices, find the dummy indices.*
- size\_t [GiNaC::count\\_dummy\\_indices](#) (const exvector &v)  
*Count the number of dummy index pairs in an index vector.*
- size\_t [GiNaC::count\\_free\\_indices](#) (const exvector &v)  
*Count the number of dummy index pairs in an index vector.*
- ex [GiNaC::minimal\\_dim](#) (const ex &dim1, const ex &dim2)  
*Return the minimum of two index dimensions.*

### 7.41.1 Detailed Description

Interface to [GiNaC](#)'s indices.

## 7.42 indexed.cpp File Reference

Implementation of [GiNaC](#)'s indexed expressions.

```
#include "indexed.h"
#include "idx.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "power.h"
#include "relational.h"
#include "symmetry.h"
#include "operators.h"
#include "lst.h"
#include "archive.h"
#include "symbol.h"
#include "utils.h"
#include "integral.h"
#include "matrix.h"
#include "inifcns.h"
#include <iostream>
#include <limits>
#include <sstream>
#include <stdexcept>
```

## Classes

- struct [GiNaC::idx\\_is\\_equal\\_ignore\\_dim](#)
- struct [GiNaC::is\\_summation\\_idx](#)
- struct [GiNaC::ex\\_base\\_is\\_less](#)
- class [GiNaC::terminfo](#)

*This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.*

- class [GiNaC::terminfo\\_is\\_less](#)
- class [GiNaC::symminfo](#)

*This structure stores the individual symmetrized terms obtained during the simplification of sums.*

- class [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm](#)
- class [GiNaC::symminfo\\_is\\_less\\_by\\_orig](#)

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (indexed, exprseq, print\_func< print\_context >(&indexed::do\_print). print\_func< print\_latex >(&indexed::do\_print\_latex). print\_func< print\_tree >(&indexed::do\_print\_tree)) indexed
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (indexed)
- static bool [GiNaC::indices\\_consistent](#) (const exvector &v1, const exvector &v2)  
*Check whether two sorted index vectors are consistent (i.e.*
- template<class T >  
size\_t [GiNaC::number\\_of\\_type](#) (const exvector &v)
- template<class T >  
static ex [GiNaC::rename\\_dummy\\_indices](#) (const ex &e, exvector &global\_dummy\_indices, exvector &local\_↵\_dummy\_indices)  
*Rename dummy indices in an expression.*
- static void [GiNaC::find\\_variant\\_indices](#) (const exvector &v, exvector &variant\_indices)  
*Given a set of indices, extract those of class varidx.*
- bool [GiNaC::reposition\\_dummy\\_indices](#) (ex &e, exvector &variant\_dummy\_indices, exvector &moved\_↵indices)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*
- static void [GiNaC::product\\_to\\_exvector](#) (const ex &e, exvector &v, bool &non\_commutative)
- template<class T >  
ex [GiNaC::idx\\_symmetrization](#) (const ex &r, const exvector &local\_dummy\_indices)
- ex [GiNaC::simplify\\_indexed](#) (const ex &e, exvector &free\_indices, exvector &dummy\_indices, const scalar\_↵\_products &sp)  
*Simplify indexed expression, return list of free indices.*
- ex [GiNaC::simplify\\_indexed\\_product](#) (const ex &e, exvector &free\_indices, exvector &dummy\_indices, const scalar\_products &sp)  
*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.*
- bool [GiNaC::hasindex](#) (const ex &x, const ex &sym)
- exvector [GiNaC::get\\_all\\_dummy\\_indices\\_safely](#) (const ex &e)  
*More reliable version of the form.*
- exvector [GiNaC::get\\_all\\_dummy\\_indices](#) (const ex &e)  
*Returns all dummy indices from the exvector.*
- lst [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const exvector &va, const exvector &vb)

- Similar to above, where  $va$  and  $vb$  are the same and the return value is a list of two lists for substitution in  $b$ .*

  - ex [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const exvector &va, const exvector &vb, const ex &b)

*Same as above, where  $va$  and  $vb$  contain the indices of  $a$  and  $b$  and are sorted.*
- ex [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const ex &a, const ex &b)

*Returns  $b$  with all dummy indices, which are common with  $a$ , renamed.*
- ex [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (exvector &va, const ex &b, bool modify\_va=false)

*Returns  $b$  with all dummy indices, which are listed in  $va$ , renamed if `modify_va` is set to `TRUE` all dummy indices of  $b$  will be appended to  $va$ .*
- ex [GiNaC::expand\\_dummy\\_sum](#) (const ex &e, bool subs\_idx=false)

*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*

### 7.42.1 Detailed Description

Implementation of [GiNaC](#)'s indexed expressions.

## 7.43 indexed.h File Reference

Interface to [GiNaC](#)'s indexed expressions.

```
#include "exprseq.h"
#include "wildcard.h"
#include <map>
```

### Classes

- class [GiNaC::indexed](#)

*This class holds an indexed expression.*
- class [GiNaC::spmapkey](#)
- class [GiNaC::scalar\\_products](#)

*Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).*

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef std::map< spmapkey, ex > [GiNaC::spmap](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (indexed)
- [exvector GiNaC::get\\_all\\_dummy\\_indices](#) (const ex &e)  
*Returns all dummy indices from the exvector.*
- [exvector GiNaC::get\\_all\\_dummy\\_indices\\_safely](#) (const ex &e)  
*More reliable version of the form.*
- [ex GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (exvector &va, const ex &b, bool modify\_va=false)  
*Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b will be appended to va.*
- [ex GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const ex &a, const ex &b)  
*Returns b with all dummy indices, which are common with a, renamed.*
- [ex GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const exvector &va, const exvector &vb, const ex &b)  
*Same as above, where va and vb contain the indices of a and b and are sorted.*
- [lst GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const exvector &va, const exvector &vb)  
*Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.*
- [ex GiNaC::expand\\_dummy\\_sum](#) (const ex &e, bool subs\_idx=false)  
*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*

### 7.43.1 Detailed Description

Interface to [GiNaC](#)'s indexed expressions.

## 7.44 inifcns.cpp File Reference

Implementation of [GiNaC](#)'s initially known functions.

```
#include "inifcns.h"
#include "ex.h"
#include "constant.h"
#include "lst.h"
#include "fderivative.h"
#include "matrix.h"
#include "mul.h"
#include "power.h"
#include "operators.h"
#include "relational.h"
#include "pseries.h"
#include "symbol.h"
#include "symmetry.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

## Classes

- class [GiNaC::symbolset](#)

## Namespaces

- namespace [GiNaC](#)

## Functions

- static ex [GiNaC::conjugate\\_evalf](#) (const ex &arg)
- static ex [GiNaC::conjugate\\_eval](#) (const ex &arg)
- static void [GiNaC::conjugate\\_print\\_latex](#) (const ex &arg, const print\_context &c)
- static ex [GiNaC::conjugate\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::conjugate\\_expl\\_derivative](#) (const ex &arg, const symbol &s)
- static ex [GiNaC::conjugate\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::conjugate\\_imag\\_part](#) (const ex &arg)
- static bool [GiNaC::func\\_arg\\_info](#) (const ex &arg, unsigned inf)
- static bool [GiNaC::conjugate\\_info](#) (const ex &arg, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) (conjugate\_function, eval\_func(conjugate\_eval). evalf\_func(conjugate\_evalf). expl\_derivative\_func(conjugate\_expl\_derivative). info\_func(conjugate\_info). print\_func< print\_latex >(conjugate\_print\_latex). conjugate\_func(conjugate\_conjugate). real\_part\_func(conjugate\_real\_part). imag\_part\_func(conjugate\_imag\_part). set\_name("conjugate","conjugate"))
- static ex [GiNaC::real\\_part\\_evalf](#) (const ex &arg)
- static ex [GiNaC::real\\_part\\_eval](#) (const ex &arg)
- static void [GiNaC::real\\_part\\_print\\_latex](#) (const ex &arg, const print\_context &c)
- static ex [GiNaC::real\\_part\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::real\\_part\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::real\\_part\\_imag\\_part](#) (const ex &arg)
- static ex [GiNaC::real\\_part\\_expl\\_derivative](#) (const ex &arg, const symbol &s)
- [GiNaC::REGISTER\\_FUNCTION](#) (real\_part\_function, eval\_func(real\_part\_eval). evalf\_func(real\_part\_evalf). expl\_derivative\_func(real\_part\_expl\_derivative). print\_func< print\_latex >(real\_part\_print\_latex). conjugate\_func(real\_part\_conjugate). real\_part\_func(real\_part\_real\_part). imag\_part\_func(real\_part\_imag\_part). set\_name("real\_part","real\_part"))
- static ex [GiNaC::imag\\_part\\_evalf](#) (const ex &arg)
- static ex [GiNaC::imag\\_part\\_eval](#) (const ex &arg)
- static void [GiNaC::imag\\_part\\_print\\_latex](#) (const ex &arg, const print\_context &c)
- static ex [GiNaC::imag\\_part\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::imag\\_part\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::imag\\_part\\_imag\\_part](#) (const ex &arg)
- static ex [GiNaC::imag\\_part\\_expl\\_derivative](#) (const ex &arg, const symbol &s)
- [GiNaC::REGISTER\\_FUNCTION](#) (imag\_part\_function, eval\_func(imag\_part\_eval). evalf\_func(imag\_part\_evalf). expl\_derivative\_func(imag\_part\_expl\_derivative). print\_func< print\_latex >(imag\_part\_print\_latex). conjugate\_func(imag\_part\_conjugate). real\_part\_func(imag\_part\_real\_part). imag\_part\_func(imag\_part\_imag\_part). set\_name("imag\_part","imag\_part"))
- static ex [GiNaC::abs\\_evalf](#) (const ex &arg)
- static ex [GiNaC::abs\\_eval](#) (const ex &arg)
- static ex [GiNaC::abs\\_expand](#) (const ex &arg, unsigned options)
- static ex [GiNaC::abs\\_expl\\_derivative](#) (const ex &arg, const symbol &s)
- static void [GiNaC::abs\\_print\\_latex](#) (const ex &arg, const print\_context &c)
- static void [GiNaC::abs\\_print\\_csrc\\_float](#) (const ex &arg, const print\_context &c)
- static ex [GiNaC::abs\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::abs\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::abs\\_imag\\_part](#) (const ex &arg)
- static ex [GiNaC::abs\\_power](#) (const ex &arg, const ex &exp)
- bool [GiNaC::abs\\_info](#) (const ex &arg, unsigned inf)

- [GiNaC::REGISTER\\_FUNCTION](#) (abs, eval\_func(abs\_eval). evalf\_func(abs\_evalf). expand\_func(abs\_↵  
expand). expl\_derivative\_func(abs\_expl\_derivative). info\_func(abs\_info). print\_func< print\_latex >(abs\_↵  
\_print\_latex). print\_func< print\_csrc\_float >(abs\_print\_csrc\_float). print\_func< print\_csrc\_double >(abs\_↵  
\_print\_csrc\_float). conjugate\_func(abs\_conjugate). real\_part\_func(abs\_real\_part). imag\_part\_func(abs\_↵  
imag\_part). power\_func(abs\_power))
- static ex [GiNaC::step\\_evalf](#) (const ex &arg)
- static ex [GiNaC::step\\_eval](#) (const ex &arg)
- static ex [GiNaC::step\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::step\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::step\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::step\\_imag\\_part](#) (const ex &arg)
- [GiNaC::REGISTER\\_FUNCTION](#) (step, eval\_func(step\_eval). evalf\_func(step\_evalf). series\_func(step\_↵  
series). conjugate\_func(step\_conjugate). real\_part\_func(step\_real\_part). imag\_part\_func(step\_imag\_part))
- static ex [GiNaC::csgn\\_evalf](#) (const ex &arg)
- static ex [GiNaC::csgn\\_eval](#) (const ex &arg)
- static ex [GiNaC::csgn\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::csgn\\_conjugate](#) (const ex &arg)
- static ex [GiNaC::csgn\\_real\\_part](#) (const ex &arg)
- static ex [GiNaC::csgn\\_imag\\_part](#) (const ex &arg)
- static ex [GiNaC::csgn\\_power](#) (const ex &arg, const ex &exp)
- [GiNaC::REGISTER\\_FUNCTION](#) (csgn, eval\_func(csgn\_eval). evalf\_func(csgn\_evalf). series\_func(csgn\_↵  
series). conjugate\_func(csgn\_conjugate). real\_part\_func(csgn\_real\_part). imag\_part\_func(csgn\_imag\_↵  
part). power\_func(csgn\_power))
- static ex [GiNaC::eta\\_evalf](#) (const ex &x, const ex &y)
- static ex [GiNaC::eta\\_eval](#) (const ex &x, const ex &y)
- static ex [GiNaC::eta\\_series](#) (const ex &x, const ex &y, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::eta\\_conjugate](#) (const ex &x, const ex &y)
- static ex [GiNaC::eta\\_real\\_part](#) (const ex &x, const ex &y)
- static ex [GiNaC::eta\\_imag\\_part](#) (const ex &x, const ex &y)
- [GiNaC::REGISTER\\_FUNCTION](#) (eta, eval\_func(eta\_eval). evalf\_func(eta\_evalf). series\_func(eta\_series).  
latex\_name("\\eta"). set\_symmetry(sy\_symm(0, 1)). conjugate\_func(eta\_conjugate). real\_part\_func(eta\_↵  
real\_part). imag\_part\_func(eta\_imag\_part))
- static ex [GiNaC::Li2\\_evalf](#) (const ex &x)
- static ex [GiNaC::Li2\\_eval](#) (const ex &x)
- static ex [GiNaC::Li2\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::Li2\\_series](#) (const ex &x, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::Li2\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (Li2, eval\_func(Li2\_eval). evalf\_func(Li2\_evalf). derivative\_func(Li2\_deriv).  
series\_func(Li2\_series). conjugate\_func(Li2\_conjugate). latex\_name("\\mathrm{Li}\_2"))
- static ex [GiNaC::Li3\\_eval](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (Li3, eval\_func(Li3\_eval). latex\_name("\\mathrm{Li}\_3"))
- static ex [GiNaC::zetaderiv\\_eval](#) (const ex &n, const ex &x)
- static ex [GiNaC::zetaderiv\\_deriv](#) (const ex &n, const ex &x, unsigned deriv\_param)
- [GiNaC::REGISTER\\_FUNCTION](#) (zetaderiv, eval\_func(zetaderiv\_eval). derivative\_func(zetaderiv\_deriv).  
latex\_name("\\zeta^{\\prime}"))
- static ex [GiNaC::factorial\\_evalf](#) (const ex &x)
- static ex [GiNaC::factorial\\_eval](#) (const ex &x)
- static void [GiNaC::factorial\\_print\\_dflt\\_latex](#) (const ex &x, const print\_context &c)
- static ex [GiNaC::factorial\\_conjugate](#) (const ex &x)
- static ex [GiNaC::factorial\\_real\\_part](#) (const ex &x)
- static ex [GiNaC::factorial\\_imag\\_part](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (factorial, eval\_func(factorial\_eval). evalf\_func(factorial\_evalf). print\_func<  
print\_dflt >(factorial\_print\_dflt\_latex). print\_func< print\_latex >(factorial\_print\_dflt\_latex). conjugate\_↵  
func(factorial\_conjugate). real\_part\_func(factorial\_real\_part). imag\_part\_func(factorial\_imag\_part))
- static ex [GiNaC::binomial\\_evalf](#) (const ex &x, const ex &y)

- static ex [GiNaC::binomial\\_sym](#) (const ex &x, const numeric &y)
- static ex [GiNaC::binomial\\_eval](#) (const ex &x, const ex &y)
- static ex [GiNaC::binomial\\_conjugate](#) (const ex &x, const ex &y)
- static ex [GiNaC::binomial\\_real\\_part](#) (const ex &x, const ex &y)
- static ex [GiNaC::binomial\\_imag\\_part](#) (const ex &x, const ex &y)
- [GiNaC::REGISTER\\_FUNCTION](#) (binomial, eval\_func(binomial\_eval). evalf\_func(binomial\_evalf). conjugate↔\_func(binomial\_conjugate). real\_part\_func(binomial\_real\_part). imag\_part\_func(binomial\_imag\_part))
- static ex [GiNaC::Order\\_eval](#) (const ex &x)
- static ex [GiNaC::Order\\_series](#) (const ex &x, const relational &r, int [order](#), unsigned [options](#))
- static ex [GiNaC::Order\\_conjugate](#) (const ex &x)
- static ex [GiNaC::Order\\_real\\_part](#) (const ex &x)
- static ex [GiNaC::Order\\_imag\\_part](#) (const ex &x)
- static ex [GiNaC::Order\\_expl\\_derivative](#) (const ex &arg, const symbol &s)
- [GiNaC::REGISTER\\_FUNCTION](#) (Order, eval\_func(Order\_eval). series\_func(Order\_series). latex\_↔name("\\mathcal{O}"). expl\_derivative\_func(Order\_expl\_derivative). conjugate\_func(Order\_conjugate). real\_part\_func(Order\_real\_part). imag\_part\_func(Order\_imag\_part))
- ex [GiNaC::lsolve](#) (const ex &eqns, const ex &symbols, unsigned [options](#)=solve\_algo::automatic)  
*Factorial function.*
- const numeric [GiNaC::fsolve](#) (const ex &f, const symbol &x, const numeric &x1, const numeric &x2)  
*Find a real root of real-valued function f(x) numerically within a given interval.*

## Variables

- unsigned [GiNaC::force\\_include\\_tgamma](#) = tgamma\_SERIAL::serial
- unsigned [GiNaC::force\\_include\\_zeta1](#) = zeta1\_SERIAL::serial

### 7.44.1 Detailed Description

Implementation of [GiNaC](#)'s initially known functions.

## 7.45 inifcns.h File Reference

Interface to [GiNaC](#)'s initially known functions.

```
#include "numeric.h"
#include "function.h"
#include "ex.h"
```



## Classes

- class [GiNaC::zeta1\\_SERIAL](#)  
*Complex conjugate.*
- class [GiNaC::zeta2\\_SERIAL](#)  
*Alternating Euler sum or colored MZV.*
- class [GiNaC::G2\\_SERIAL](#)  
*Generalized multiple polylogarithm.*
- class [GiNaC::G3\\_SERIAL](#)  
*Generalized multiple polylogarithm with explicit imaginary parts.*
- class [GiNaC::psi1\\_SERIAL](#)  
*Polylogarithm and multiple polylogarithm.*
- class [GiNaC::psi2\\_SERIAL](#)  
*Derivatives of Psi-function (aka polygamma-functions).*
- class [GiNaC::iterated\\_integral2\\_SERIAL](#)  
*Complete elliptic integral of the first kind.*
- class [GiNaC::iterated\\_integral3\\_SERIAL](#)  
*Iterated integral with explicit truncation.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- `template<typename T1 >`  
function [GiNaC::zeta](#) (const T1 &p1)
- `template<typename T1 , typename T2 >`  
function [GiNaC::zeta](#) (const T1 &p1, const T2 &p2)
- `template<> bool` [GiNaC::is\\_the\\_function< zeta\\_SERIAL >](#) (const ex &x)
- `template<typename T1 , typename T2 >`  
function [GiNaC::G](#) (const T1 &x, const T2 &y)
- `template<typename T1 , typename T2 , typename T3 >`  
function [GiNaC::G](#) (const T1 &x, const T2 &s, const T3 &y)
- `template<> bool` [GiNaC::is\\_the\\_function< G\\_SERIAL >](#) (const ex &x)
- `template<typename T1 >`  
function [GiNaC::psi](#) (const T1 &p1)
- `template<typename T1 , typename T2 >`  
function [GiNaC::psi](#) (const T1 &p1, const T2 &p2)
- `template<> bool` [GiNaC::is\\_the\\_function< psi\\_SERIAL >](#) (const ex &x)
- `template<typename T1 , typename T2 >`  
function [GiNaC::iterated\\_integral](#) (const T1 &kernel\_lst, const T2 &lambda)
- `template<typename T1 , typename T2 , typename T3 >`  
function [GiNaC::iterated\\_integral](#) (const T1 &kernel\_lst, const T2 &lambda, const T3 &N\_trunc)
- `template<> bool` [GiNaC::is\\_the\\_function< iterated\\_integral\\_SERIAL >](#) (const ex &x)
- ex [GiNaC::lsolve](#) (const ex &eqns, const ex &symbols, unsigned [options](#)=solve\_algo::automatic)  
*Factorial function.*
- const numeric [GiNaC::fsolve](#) (const ex &f, const symbol &x, const numeric &x1, const numeric &x2)  
*Find a real root of real-valued function f(x) numerically within a given interval.*
- bool [GiNaC::is\\_order\\_function](#) (const ex &e)  
*Check whether a function is the Order (O(n)) function.*
- ex [GiNaC::convert\\_H\\_to\\_Li](#) (const ex &parameterlst, const ex &arg)  
*Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding [GiNaC](#) functions.*

### 7.45.1 Detailed Description

Interface to [GiNaC](#)'s initially known functions.

## 7.46 inifcns\_elliptic.cpp File Reference

Implementation of some special functions related to elliptic curves.

```
#include "inifcns.h"
#include "add.h"
#include "constant.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "operators.h"
#include "power.h"
#include "pseries.h"
#include "relational.h"
#include "symbol.h"
#include "utils.h"
#include "wildcard.h"
#include "integration_kernel.h"
#include "utils_multi_iterator.h"
#include <cln/cln.h>
#include <sstream>
#include <stdexcept>
#include <vector>
#include <cmath>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- static ex [GiNaC::EllipticK\\_evalf](#) (const ex &k)
- static ex [GiNaC::EllipticK\\_eval](#) (const ex &k)
- static ex [GiNaC::EllipticK\\_deriv](#) (const ex &k, unsigned deriv\_param)
- static ex [GiNaC::EllipticK\\_series](#) (const ex &k, const relational &rel, int [order](#), unsigned [options](#))
- static void [GiNaC::EllipticK\\_print\\_latex](#) (const ex &k, const print\_context &c)
- [GiNaC::REGISTER\\_FUNCTION](#) (EllipticK, evalf\_func(EllipticK\_evalf). eval\_func(EllipticK\_eval). derivative↵\_func(EllipticK\_deriv). series\_func(EllipticK\_series). print\_func< print\_latex >(EllipticK\_print\_latex). do\_↵not\_evalf\_params())
- static ex [GiNaC::EllipticE\\_evalf](#) (const ex &k)
- static ex [GiNaC::EllipticE\\_eval](#) (const ex &k)
- static ex [GiNaC::EllipticE\\_deriv](#) (const ex &k, unsigned deriv\_param)
- static ex [GiNaC::EllipticE\\_series](#) (const ex &k, const relational &rel, int [order](#), unsigned [options](#))
- static void [GiNaC::EllipticE\\_print\\_latex](#) (const ex &k, const print\_context &c)
- [GiNaC::REGISTER\\_FUNCTION](#) (EllipticE, evalf\_func(EllipticE\_evalf). eval\_func(EllipticE\_eval). derivative↵\_func(EllipticE\_deriv). series\_func(EllipticE\_series). print\_func< print\_latex >(EllipticE\_print\_latex). do\_↵not\_evalf\_params())
- static ex [GiNaC::iterated\\_integral\\_evalf\\_impl](#) (const ex &kernel\_lst, const ex &lambda, const ex &N\_trunc)
- static ex [GiNaC::iterated\\_integral2\\_evalf](#) (const ex &kernel\_lst, const ex &lambda)
- static ex [GiNaC::iterated\\_integral3\\_evalf](#) (const ex &kernel\_lst, const ex &lambda, const ex &N\_trunc)
- static ex [GiNaC::iterated\\_integral2\\_eval](#) (const ex &kernel\_lst, const ex &lambda)
- static ex [GiNaC::iterated\\_integral3\\_eval](#) (const ex &kernel\_lst, const ex &lambda, const ex &N\_trunc)

### 7.46.1 Detailed Description

Implementation of some special functions related to elliptic curves.

The functions are: complete elliptic integral of the first kind `EllipticK(k)` complete elliptic integral of the second kind `EllipticE(k)` iterated integral `iterated_integral(a,y)` or `iterated_integral(a,y,N_trunc)`

Some remarks:

- All formulae used can be looked up in the following publication: [WW] Numerical evaluation of iterated integrals related to elliptic Feynman integrals, M.Walden, S.Weinzierl, arXiv:2010.05271
- When these routines and methods are used for scientific work that leads to publication in a scientific journal, please refer to this program as : M.Walden, S.Weinzierl, "Numerical evaluation of iterated integrals related to elliptic Feynman integrals", arXiv:2010.05271
- As these routines build on the core part of [GiNaC](#), it is also polite to acknowledge C. Bauer, A. Frink, R. Kreckel, "Introduction to the GiNaC Framework for Symbolic Computation within the C++ Programming Language", J. Symbolic Computations 33, 1 (2002), cs.sc/0004015

## 7.47 inifcns\_gamma.cpp File Reference

Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff.

```
#include "inifcns.h"
#include "constant.h"
#include "pseries.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "symbol.h"
#include "symmetry.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- static ex [GiNaC::lgamma\\_evalf](#) (const ex &x)
- static ex [GiNaC::lgamma\\_eval](#) (const ex &x)  
*Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.*
- static ex [GiNaC::lgamma\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::lgamma\\_series](#) (const ex &arg, const relational &rel, int order, unsigned options)
- static ex [GiNaC::lgamma\\_conjugate](#) (const ex &x)

- `GiNaC::REGISTER_FUNCTION` (lgamma, eval\_func(lgamma\_eval). evalf\_func(lgamma\_evalf). derivative\_func(lgamma\_deriv). series\_func(lgamma\_series). conjugate\_func(lgamma\_conjugate). latex\_name("\\log \\Gamma"))
- static ex `GiNaC::tgamma_evalf` (const ex &x)
- static ex `GiNaC::tgamma_eval` (const ex &x)  
*Evaluation of  $tgamma(x)$ , the true Gamma function.*
- static ex `GiNaC::tgamma_deriv` (const ex &x, unsigned deriv\_param)
- static ex `GiNaC::tgamma_series` (const ex &arg, const relational &rel, int order, unsigned options)
- static ex `GiNaC::tgamma_conjugate` (const ex &x)
- `GiNaC::REGISTER_FUNCTION` (tgamma, eval\_func(tgamma\_eval). evalf\_func(tgamma\_evalf). derivative\_func(tgamma\_deriv). series\_func(tgamma\_series). conjugate\_func(tgamma\_conjugate). latex\_name("\\Gamma"))
- static ex `GiNaC::beta_evalf` (const ex &x, const ex &y)
- static ex `GiNaC::beta_eval` (const ex &x, const ex &y)
- static ex `GiNaC::beta_deriv` (const ex &x, const ex &y, unsigned deriv\_param)
- static ex `GiNaC::beta_series` (const ex &arg1, const ex &arg2, const relational &rel, int order, unsigned options)
- `GiNaC::REGISTER_FUNCTION` (beta, eval\_func(beta\_eval). evalf\_func(beta\_evalf). derivative\_func(beta\_deriv). series\_func(beta\_series). latex\_name("\\mathrm{B}"). set\_symmetry(sy\_symm(0, 1)))
- static ex `GiNaC::psi1_evalf` (const ex &x)
- static ex `GiNaC::psi1_eval` (const ex &x)  
*Evaluation of digamma-function  $psi(x)$ .*
- static ex `GiNaC::psi1_deriv` (const ex &x, unsigned deriv\_param)
- static ex `GiNaC::psi1_series` (const ex &arg, const relational &rel, int order, unsigned options)
- static ex `GiNaC::psi2_evalf` (const ex &n, const ex &x)
- static ex `GiNaC::psi2_eval` (const ex &n, const ex &x)  
*Evaluation of polygamma-function  $psi(n,x)$ .*
- static ex `GiNaC::psi2_deriv` (const ex &n, const ex &x, unsigned deriv\_param)
- static ex `GiNaC::psi2_series` (const ex &n, const ex &arg, const relational &rel, int order, unsigned options)

### 7.47.1 Detailed Description

Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff.

## 7.48 inifcns\_nstdsums.cpp File Reference

Implementation of some special functions that have a representation as nested sums.

```
#include "inifcns.h"
#include "add.h"
#include "constant.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "operators.h"
#include "power.h"
#include "pseries.h"
#include "relational.h"
#include "symbol.h"
#include "utils.h"
#include "wildcard.h"
#include <cln/cln.h>
```

```
#include <sstream>
#include <stdexcept>
#include <vector>
#include <cmath>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- static ex [GiNaC::G2\\_evalf](#) (const ex &x\_, const ex &y)
- static ex [GiNaC::G2\\_eval](#) (const ex &x\_, const ex &y)
- static ex [GiNaC::G3\\_evalf](#) (const ex &x\_, const ex &s\_, const ex &y)
- static ex [GiNaC::G3\\_eval](#) (const ex &x\_, const ex &s\_, const ex &y)
- static ex [GiNaC::Li\\_evalf](#) (const ex &m\_, const ex &x\_)
- static ex [GiNaC::Li\\_eval](#) (const ex &m\_, const ex &x\_)
- static ex [GiNaC::Li\\_series](#) (const ex &m, const ex &x, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::Li\\_deriv](#) (const ex &m\_, const ex &x\_, unsigned deriv\_param)
- static void [GiNaC::Li\\_print\\_latex](#) (const ex &m\_, const ex &x\_, const print\_context &c)
- [GiNaC::REGISTER\\_FUNCTION](#) (Li, evalf\_func(Li\_evalf). eval\_func(Li\_eval). series\_func(Li\_series). derivative\_func(Li\_deriv). print\_func< print\_latex >(Li\_print\_latex). do\_not\_evalf\_params())
- static ex [GiNaC::S\\_evalf](#) (const ex &n, const ex &p, const ex &x)
- static ex [GiNaC::S\\_eval](#) (const ex &n, const ex &p, const ex &x)
- static ex [GiNaC::S\\_series](#) (const ex &n, const ex &p, const ex &x, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::S\\_deriv](#) (const ex &n, const ex &p, const ex &x, unsigned deriv\_param)
- static void [GiNaC::S\\_print\\_latex](#) (const ex &n, const ex &p, const ex &x, const print\_context &c)
- [GiNaC::REGISTER\\_FUNCTION](#) (S, evalf\_func(S\_evalf). eval\_func(S\_eval). series\_func(S\_series). derivative\_func(S\_deriv). print\_func< print\_latex >(S\_print\_latex). do\_not\_evalf\_params())
- static ex [GiNaC::H\\_evalf](#) (const ex &x1, const ex &x2)
- static ex [GiNaC::H\\_eval](#) (const ex &m\_, const ex &x)
- static ex [GiNaC::H\\_series](#) (const ex &m, const ex &x, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::H\\_deriv](#) (const ex &m\_, const ex &x, unsigned deriv\_param)
- static void [GiNaC::H\\_print\\_latex](#) (const ex &m\_, const ex &x, const print\_context &c)
- [GiNaC::REGISTER\\_FUNCTION](#) (H, evalf\_func(H\_evalf). eval\_func(H\_eval). series\_func(H\_series). derivative\_func(H\_deriv). print\_func< print\_latex >(H\_print\_latex). do\_not\_evalf\_params())
- ex [GiNaC::convert\\_H\\_to\\_Li](#) (const ex &parameterlst, const ex &arg)
 

*Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding [GiNaC](#) functions.*
- static ex [GiNaC::zeta1\\_evalf](#) (const ex &x)
- static ex [GiNaC::zeta1\\_eval](#) (const ex &m)
- static ex [GiNaC::zeta1\\_deriv](#) (const ex &m, unsigned deriv\_param)
- static void [GiNaC::zeta1\\_print\\_latex](#) (const ex &m\_, const print\_context &c)
- static ex [GiNaC::zeta2\\_evalf](#) (const ex &x, const ex &s)
- static ex [GiNaC::zeta2\\_eval](#) (const ex &m, const ex &s\_)
- static ex [GiNaC::zeta2\\_deriv](#) (const ex &m, const ex &s, unsigned deriv\_param)
- static void [GiNaC::zeta2\\_print\\_latex](#) (const ex &m\_, const ex &s\_, const print\_context &c)

### 7.48.1 Detailed Description

Implementation of some special functions that have a representation as nested sums.

The functions are: classical polylogarithm  $\text{Li}(n,x)$  multiple polylogarithm  $\text{Li}(\text{lst}\{m_1,\dots,m_k\},\text{lst}\{x_1,\dots,x_k\})$   $G(\text{lst}\{a_1,\dots,a_k\},y)$  or  $G(\text{lst}\{a_1,\dots,a_k\},\text{lst}\{s_1,\dots,s_k\},y)$  Nielsen's generalized polylogarithm  $S(n,p,x)$  harmonic polylogarithm  $H(m,x)$  or  $H(\text{lst}\{m_1,\dots,m_k\},x)$  multiple zeta value  $\text{zeta}(m)$  or  $\text{zeta}(\text{lst}\{m_1,\dots,m_k\})$  alternating Euler sum  $\text{zeta}(m,s)$  or  $\text{zeta}(\text{lst}\{m_1,\dots,m_k\},\text{lst}\{s_1,\dots,s_k\})$

Some remarks:

- All formulae used can be looked up in the following publications: [Kol] Nielsen's Generalized Polylogarithms, K.S.Kolbig, SIAM J.Math.Anal. 17 (1986), pp. 1232-1258. [Cra] Fast Evaluation of Multiple Zeta Sums, R.E.Crandall, Math.Comp. 67 (1998), pp. 1163-1172. [ReV] Harmonic Polylogarithms, E.Remiddi, J.A.M. Vermaseren, Int.J.Mod.Phys. A15 (2000), pp. 725-754 [BBB] Special Values of Multiple Polylogarithms, J.Borwein, D.Bradley, D.Broadhurst, P.Lisonek, Trans.Amer.Math.Soc. 353/3 (2001), pp. 907-941 [VSW] Numerical evaluation of multiple polylogarithms, J.Vollinga, S.Weinzierl, hep-ph/0410259
- The order of parameters and arguments of Li and zeta is defined according to the nested sums representation. The parameters for H are understood as in [ReV]. They can be in expanded — only 0, 1 and -1 — or in compactified — a string with zeros in front of 1 or -1 is written as a single number — notation.
- All functions can be numerically evaluated with arguments in the whole complex plane. The parameters for Li, zeta and S must be positive integers. If you want to have an alternating Euler sum, you have to give the signs of the parameters as a second argument s to  $\text{zeta}(m,s)$  containing 1 and -1.
- The calculation of classical polylogarithms is speeded up by using Bernoulli numbers and look-up tables. S uses look-up tables as well. The zeta function applies the algorithms in [Cra] and [BBB] for speed up. Multiple polylogarithms use Hoelder convolution [BBB].
- The functions have no means to do a series expansion into nested sums. To do this, you have to convert these functions into the appropriate objects from the nestedsums library, do the expansion and convert the result back.
- Numerical testing of this implementation has been performed by doing a comparison of results between this software and the commercial M..... 4.1. Multiple zeta values have been checked by means of evaluations into simple zeta values. Harmonic polylogarithms have been checked by comparison to  $S(n,p,x)$  for corresponding parameter combinations and by continuity checks around  $|x|=1$  along with comparisons to corresponding zeta functions. Multiple polylogarithms were checked against H and zeta and by means of shuffle and quasi-shuffle relations.

## 7.49 inifcns\_trans.cpp File Reference

Implementation of transcendental (and trigonometric and hyperbolic) functions.

```
#include "inifcns.h"
#include "ex.h"
#include "constant.h"
#include "add.h"
#include "mul.h"
#include "numeric.h"
#include "power.h"
#include "operators.h"
#include "relational.h"
#include "symbol.h"
#include "pseries.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- static ex [GiNaC::exp\\_evalf](#) (const ex &*x*)
- static ex [GiNaC::exp\\_eval](#) (const ex &*x*)
- static ex [GiNaC::exp\\_expand](#) (const ex &*arg*, unsigned [options](#))
- static ex [GiNaC::exp\\_deriv](#) (const ex &*x*, unsigned deriv\_param)
- static ex [GiNaC::exp\\_real\\_part](#) (const ex &*x*)
- static ex [GiNaC::exp\\_imag\\_part](#) (const ex &*x*)
- static ex [GiNaC::exp\\_conjugate](#) (const ex &*x*)
- static ex [GiNaC::exp\\_power](#) (const ex &*x*, const ex &*a*)
- static bool [GiNaC::exp\\_info](#) (const ex &*x*, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) (exp, eval\_func(exp\_eval). evalf\_func(exp\_evalf). info\_func(exp\_info). expand\_func(exp\_expand). derivative\_func(exp\_deriv). real\_part\_func(exp\_real\_part). imag\_part\_↵  
func(exp\_imag\_part). conjugate\_func(exp\_conjugate). power\_func(exp\_power). latex\_name("\\exp"))
- static ex [GiNaC::log\\_evalf](#) (const ex &*x*)
- static ex [GiNaC::log\\_eval](#) (const ex &*x*)
- static ex [GiNaC::log\\_deriv](#) (const ex &*x*, unsigned deriv\_param)
- static ex [GiNaC::log\\_series](#) (const ex &*arg*, const relational &*rel*, int [order](#), unsigned [options](#))
- static ex [GiNaC::log\\_real\\_part](#) (const ex &*x*)
- static ex [GiNaC::log\\_imag\\_part](#) (const ex &*x*)
- static ex [GiNaC::log\\_expand](#) (const ex &*arg*, unsigned [options](#))
- static ex [GiNaC::log\\_conjugate](#) (const ex &*x*)
- static bool [GiNaC::log\\_info](#) (const ex &*x*, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) (log, eval\_func(log\_eval). evalf\_func(log\_evalf). info\_func(log\_info). expand\_func(log\_expand). derivative\_func(log\_deriv). series\_func(log\_series). real\_part\_func(log\_↵  
real\_part). imag\_part\_func(log\_imag\_part). conjugate\_func(log\_conjugate). latex\_name("\\ln"))
- static ex [GiNaC::sin\\_evalf](#) (const ex &*x*)
- static ex [GiNaC::sin\\_eval](#) (const ex &*x*)
- static ex [GiNaC::sin\\_deriv](#) (const ex &*x*, unsigned deriv\_param)
- static ex [GiNaC::sin\\_real\\_part](#) (const ex &*x*)
- static ex [GiNaC::sin\\_imag\\_part](#) (const ex &*x*)
- static ex [GiNaC::sin\\_conjugate](#) (const ex &*x*)
- static bool [GiNaC::trig\\_info](#) (const ex &*x*, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) (sin, eval\_func(sin\_eval). evalf\_func(sin\_evalf). info\_func(trig\_info). derivative\_func(sin\_deriv). real\_part\_func(sin\_real\_part). imag\_part\_func(sin\_imag\_part). conjugate\_↵  
func(sin\_conjugate). latex\_name("\\sin"))
- static ex [GiNaC::cos\\_evalf](#) (const ex &*x*)
- static ex [GiNaC::cos\\_eval](#) (const ex &*x*)
- static ex [GiNaC::cos\\_deriv](#) (const ex &*x*, unsigned deriv\_param)
- static ex [GiNaC::cos\\_real\\_part](#) (const ex &*x*)
- static ex [GiNaC::cos\\_imag\\_part](#) (const ex &*x*)
- static ex [GiNaC::cos\\_conjugate](#) (const ex &*x*)
- [GiNaC::REGISTER\\_FUNCTION](#) (cos, eval\_func(cos\_eval). info\_func(trig\_info). evalf\_func(cos\_evalf). derivative\_func(cos\_deriv). real\_part\_func(cos\_real\_part). imag\_part\_func(cos\_imag\_part). conjugate\_↵  
func(cos\_conjugate). latex\_name("\\cos"))
- static ex [GiNaC::tan\\_evalf](#) (const ex &*x*)
- static ex [GiNaC::tan\\_eval](#) (const ex &*x*)
- static ex [GiNaC::tan\\_deriv](#) (const ex &*x*, unsigned deriv\_param)
- static ex [GiNaC::tan\\_real\\_part](#) (const ex &*x*)
- static ex [GiNaC::tan\\_imag\\_part](#) (const ex &*x*)

- static ex `GiNaC::tan_series` (const ex &x, const relational &rel, int `order`, unsigned `options`)
- static ex `GiNaC::tan_conjugate` (const ex &x)
- `GiNaC::REGISTER_FUNCTION` (tan, eval\_func(tan\_eval). evalf\_func(tan\_evalf). info\_func(trig\_info). derivative\_func(tan\_deriv). series\_func(tan\_series). real\_part\_func(tan\_real\_part). imag\_part\_func(tan\_↵imag\_part). conjugate\_func(tan\_conjugate). latex\_name("\\tan"))
- static ex `GiNaC::asin_evalf` (const ex &x)
- static ex `GiNaC::asin_eval` (const ex &x)
- static ex `GiNaC::asin_deriv` (const ex &x, unsigned deriv\_param)
- static ex `GiNaC::asin_conjugate` (const ex &x)
- static bool `GiNaC::asin_info` (const ex &x, unsigned inf)
- `GiNaC::REGISTER_FUNCTION` (asin, eval\_func(asin\_eval). evalf\_func(asin\_evalf). info\_func(asin\_info). derivative\_func(asin\_deriv). conjugate\_func(asin\_conjugate). latex\_name("\\arcsin"))
- static ex `GiNaC::acos_evalf` (const ex &x)
- static ex `GiNaC::acos_eval` (const ex &x)
- static ex `GiNaC::acos_deriv` (const ex &x, unsigned deriv\_param)
- static ex `GiNaC::acos_conjugate` (const ex &x)
- `GiNaC::REGISTER_FUNCTION` (acos, eval\_func(acos\_eval). evalf\_func(acos\_evalf). info\_func(asin\_info). derivative\_func(acos\_deriv). conjugate\_func(acos\_conjugate). latex\_name("\\arccos"))
- static ex `GiNaC::atan_evalf` (const ex &x)
- static ex `GiNaC::atan_eval` (const ex &x)
- static ex `GiNaC::atan_deriv` (const ex &x, unsigned deriv\_param)
- static ex `GiNaC::atan_series` (const ex &arg, const relational &rel, int `order`, unsigned `options`)
- static ex `GiNaC::atan_conjugate` (const ex &x)
- static bool `GiNaC::atan_info` (const ex &x, unsigned inf)
- `GiNaC::REGISTER_FUNCTION` (atan, eval\_func(atan\_eval). evalf\_func(atan\_evalf). info\_func(atan\_↵info). derivative\_func(atan\_deriv). series\_func(atan\_series). conjugate\_func(atan\_conjugate). latex\_↵name("\\arctan"))
- static ex `GiNaC::atan2_evalf` (const ex &y, const ex &x)
- static ex `GiNaC::atan2_eval` (const ex &y, const ex &x)
- static ex `GiNaC::atan2_deriv` (const ex &y, const ex &x, unsigned deriv\_param)
- static bool `GiNaC::atan2_info` (const ex &y, const ex &x, unsigned inf)
- `GiNaC::REGISTER_FUNCTION` (atan2, eval\_func(atan2\_eval). evalf\_func(atan2\_evalf). info\_func(atan2\_↵info). evalf\_func(atan2\_evalf). derivative\_func(atan2\_deriv))
- static ex `GiNaC::sinh_evalf` (const ex &x)
- static ex `GiNaC::sinh_eval` (const ex &x)
- static ex `GiNaC::sinh_deriv` (const ex &x, unsigned deriv\_param)
- static ex `GiNaC::sinh_real_part` (const ex &x)
- static ex `GiNaC::sinh_imag_part` (const ex &x)
- static ex `GiNaC::sinh_conjugate` (const ex &x)
- `GiNaC::REGISTER_FUNCTION` (sinh, eval\_func(sinh\_eval). evalf\_func(sinh\_evalf). info\_func(atan\_info). derivative\_func(sinh\_deriv). real\_part\_func(sinh\_real\_part). imag\_part\_func(sinh\_imag\_part). conjugate\_↵func(sinh\_conjugate). latex\_name("\\sinh"))
- static ex `GiNaC::cosh_evalf` (const ex &x)
- static ex `GiNaC::cosh_eval` (const ex &x)
- static ex `GiNaC::cosh_deriv` (const ex &x, unsigned deriv\_param)
- static ex `GiNaC::cosh_real_part` (const ex &x)
- static ex `GiNaC::cosh_imag_part` (const ex &x)
- static ex `GiNaC::cosh_conjugate` (const ex &x)
- `GiNaC::REGISTER_FUNCTION` (cosh, eval\_func(cosh\_eval). evalf\_func(cosh\_evalf). info\_func(exp\_↵\_info). derivative\_func(cosh\_deriv). real\_part\_func(cosh\_real\_part). imag\_part\_func(cosh\_imag\_part). conjugate\_func(cosh\_conjugate). latex\_name("\\cosh"))
- static ex `GiNaC::tanh_evalf` (const ex &x)
- static ex `GiNaC::tanh_eval` (const ex &x)
- static ex `GiNaC::tanh_deriv` (const ex &x, unsigned deriv\_param)
- static ex `GiNaC::tanh_series` (const ex &x, const relational &rel, int `order`, unsigned `options`)



- static ex [GiNaC::tanh\\_real\\_part](#) (const ex &x)
- static ex [GiNaC::tanh\\_imag\\_part](#) (const ex &x)
- static ex [GiNaC::tanh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (tanh, eval\_func(tanh\_eval). evalf\_func(tanh\_evalf). info\_func(atan\_↵info). derivative\_func(tanh\_deriv). series\_func(tanh\_series). real\_part\_func(tanh\_real\_part). imag\_part\_↵\_func(tanh\_imag\_part). conjugate\_func(tanh\_conjugate). latex\_name("\\tanh"))
- static ex [GiNaC::asinh\\_evalf](#) (const ex &x)
- static ex [GiNaC::asinh\\_eval](#) (const ex &x)
- static ex [GiNaC::asinh\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::asinh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (asinh, eval\_func(asinh\_eval). evalf\_func(asinh\_evalf). info\_func(atan\_↵info). derivative\_func(asinh\_deriv). conjugate\_func(asinh\_conjugate))
- static ex [GiNaC::acosh\\_evalf](#) (const ex &x)
- static ex [GiNaC::acosh\\_eval](#) (const ex &x)
- static ex [GiNaC::acosh\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::acosh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (acosh, eval\_func(acosh\_eval). evalf\_func(acosh\_evalf). info\_func(asin\_↵info). derivative\_func(acosh\_deriv). conjugate\_func(acosh\_conjugate))
- static ex [GiNaC::atanh\\_evalf](#) (const ex &x)
- static ex [GiNaC::atanh\\_eval](#) (const ex &x)
- static ex [GiNaC::atanh\\_deriv](#) (const ex &x, unsigned deriv\_param)
- static ex [GiNaC::atanh\\_series](#) (const ex &arg, const relational &rel, int [order](#), unsigned [options](#))
- static ex [GiNaC::atanh\\_conjugate](#) (const ex &x)
- [GiNaC::REGISTER\\_FUNCTION](#) (atanh, eval\_func(atanh\_eval). evalf\_func(atanh\_evalf). info\_func(asin\_↵info). derivative\_func(atanh\_deriv). series\_func(atanh\_series). conjugate\_func(atanh\_conjugate))

### 7.49.1 Detailed Description

Implementation of transcendental (and trigonometric and hyperbolic) functions.

## 7.50 integral.cpp File Reference

Implementation of [GiNaC](#)'s symbolic integral.

```
#include "integral.h"
#include "numeric.h"
#include "symbol.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "inifcns.h"
#include "wildcard.h"
#include "archive.h"
#include "registrar.h"
#include "utils.h"
#include "operators.h"
#include "relational.h"
```

### Classes

- struct [GiNaC::error\\_and\\_integral](#)
- struct [GiNaC::error\\_and\\_integral\\_is\\_less](#)

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef map< error\_and\_integral, ex, error\_and\_integral\_is\_less > [GiNaC::lookup\\_map](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (integral, basic, print\_func< print\_dflt >(&integral::do\_print). print\_func< print\_python >(&integral::do\_print). print\_func< print\_latex >(&integral::do\_print\_latex)) integral
- ex [GiNaC::subvalue](#) (const ex &var, const ex &value, const ex &fun)
- ex [GiNaC::adaptivesimpson](#) (const ex &x, const ex &a\_in, const ex &b\_in, const ex &f, const ex &error)  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (integral)

### 7.50.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic integral.

## 7.51 integral.h File Reference

Interface to [GiNaC](#)'s symbolic integral.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

## Classes

- class [GiNaC::integral](#)  
*Symbolic integral.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (integral)
- ex [GiNaC::adaptivesimpson](#) (const ex &x, const ex &a\_in, const ex &b\_in, const ex &f, const ex &error)  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*

### 7.51.1 Detailed Description

Interface to [GiNaC](#)'s symbolic integral.

## 7.52 integration\_kernel.cpp File Reference

Implementation of [GiNaC](#)'s integration kernels for iterated integrals.

```
#include "integration_kernel.h"
#include "add.h"
#include "mul.h"
#include "operators.h"
#include "power.h"
#include "relational.h"
#include "symbol.h"
#include "constant.h"
#include "numeric.h"
#include "function.h"
#include "pseries.h"
#include "utils.h"
#include "inifcns.h"
#include <iostream>
#include <stdexcept>
#include <cln/cln.h>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- ex [GiNaC::ifactor](#) (const numeric &n)  
*Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $lst( lst(p_1, \dots, pr), lst(a_1, \dots, ar) )$  such that  $n = p_1^{a_1} \dots p_r^{a_r}$ .*
- bool [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field](#) (const numeric &n)  
*Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.*
- numeric [GiNaC::kronecker\\_symbol](#) (const numeric &a, const numeric &n)  
*Returns the Kronecker symbol  $a$ : integer  $n$ : integer.*
- numeric [GiNaC::primitive\\_dirichlet\\_character](#) (const numeric &n, const numeric &a)  
*Defines a primitive Dirichlet character through the Kronecker symbol.*
- numeric [GiNaC::dirichlet\\_character](#) (const numeric &n, const numeric &a, const numeric &N)  
*Defines a Dirichlet character through the Kronecker symbol.*
- numeric [GiNaC::generalised\\_Bernoulli\\_number](#) (const numeric &k, const numeric &b)  
*The generalised Bernoulli number.*
- ex [GiNaC::Bernoulli\\_polynomial](#) (const numeric &k, const ex &x)  
*The Bernoulli polynomials.*
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (integration\_kernel, basic, print\_func< print\_↔ context >(&integration\_kernel::do\_print)) integration\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (integration\_kernel)

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (basic\_log\_kernel, integration\_kernel, print\_func< print\_context >(&basic\_log\_kernel::do\_print)) basic\_log\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (basic\_log\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (multiple\_polylog\_kernel, integration\_kernel, print\_func< print\_context >(&multiple\_polylog\_kernel::do\_print)) multiple\_polylog\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (multiple\_polylog\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (ELi\_kernel, integration\_kernel, print\_func< print\_context >(&ELi\_kernel::do\_print)) ELi\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (ELi\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Ebar\_kernel, integration\_kernel, print\_func< print\_context >(&Ebar\_kernel::do\_print)) Ebar\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Ebar\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dtau\_kernel, integration\_kernel, print\_func< print\_context >(&Kronecker\_dtau\_kernel::do\_print)) Kronecker\_dtau\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dtau\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dz\_kernel, integration\_kernel, print\_func< print\_context >(&Kronecker\_dz\_kernel::do\_print)) Kronecker\_dz\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dz\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_kernel, integration\_kernel, print\_func< print\_context >(&Eisenstein\_kernel::do\_print)) Eisenstein\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_h\_kernel, integration\_kernel, print\_func< print\_context >(&Eisenstein\_h\_kernel::do\_print)) Eisenstein\_h\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_h\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (modular\_form\_kernel, integration\_kernel, print\_func< print\_context >(&modular\_form\_kernel::do\_print)) modular\_form\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (modular\_form\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (user\_defined\_kernel, integration\_kernel, print\_func< print\_context >(&user\_defined\_kernel::do\_print)) user\_defined\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (user\_defined\_kernel)

### 7.52.1 Detailed Description

Implementation of [GiNaC](#)'s integration kernels for iterated integrals.

### 7.52.2 Variable Documentation

#### 7.52.2.1 qbar

ex qbar

Referenced by [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::ELi\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::modular\\_form\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 7.52.2.2 order

```
int order
```

Referenced by [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::Laurent\\_series\(\)](#), [GiNaC::user\\_defined\\_kernel::Laurent\\_series\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::Eisenstein\\_kernel::q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::fderivative::series\(\)](#), [GiNaC::function::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::integration\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::structure< T, ComparsionOperator>::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_series\(\)](#), and [GiNaC::tgamma\\_series\(\)](#).

### 7.52.2.3 cache\_vec

```
std::vector<ex> cache_vec [static], [protected]
```

### 7.52.2.4 x

```
symbol x [static], [protected]
```

Referenced by [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), and [GiNaC::integration\\_kernel::series\\_coeff\(\)](#).

## 7.53 integration\_kernel.h File Reference

Interface to [GiNaC](#)'s integration kernels for iterated integrals.

```
#include "basic.h"
#include "archive.h"
#include "numeric.h"
#include "lst.h"
#include <cln/complex.h>
#include <vector>
```

## Classes

- class [GiNaC::integration\\_kernel](#)  
The base class for integration kernels for iterated integrals.
- class [GiNaC::basic\\_log\\_kernel](#)  
The basic integration kernel with a logarithmic singularity at the origin.
- class [GiNaC::multiple\\_polylog\\_kernel](#)  
The integration kernel for multiple polylogarithms.
- class [GiNaC::ELi\\_kernel](#)  
The ELi-kernel.
- class [GiNaC::Ebar\\_kernel](#)  
The Ebar-kernel.
- class [GiNaC::Kronecker\\_dtau\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).
- class [GiNaC::Kronecker\\_dz\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .
- class [GiNaC::Eisenstein\\_kernel](#)  
The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .
- class [GiNaC::Eisenstein\\_h\\_kernel](#)  
The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .
- class [GiNaC::modular\\_form\\_kernel](#)  
A kernel corresponding to a polynomial in Eisenstein series.
- class [GiNaC::user\\_defined\\_kernel](#)  
A user-defined integration kernel.

## Namespaces

- namespace [GiNaC](#)

## Functions

- ex [GiNaC::ifactor](#) (const numeric &n)  
Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $lst(lst(p1,...,pr), lst(a1,...,ar))$  such that  $n = p1^{a1} \dots pr^{ar}$ .
- bool [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field](#) (const numeric &n)  
Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.
- numeric [GiNaC::kronecker\\_symbol](#) (const numeric &a, const numeric &n)  
Returns the Kronecker symbol  $a$ : integer  $n$ : integer.
- numeric [GiNaC::primitive\\_dirichlet\\_character](#) (const numeric &n, const numeric &a)  
Defines a primitive Dirichlet character through the Kronecker symbol.
- numeric [GiNaC::dirichlet\\_character](#) (const numeric &n, const numeric &a, const numeric &N)  
Defines a Dirichlet character through the Kronecker symbol.
- numeric [GiNaC::generalised\\_Bernoulli\\_number](#) (const numeric &k, const numeric &b)  
The generalised Bernoulli number.
- ex [GiNaC::Bernoulli\\_polynomial](#) (const numeric &k, const ex &x)  
The Bernoulli polynomials.
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (integration\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (basic\_log\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (multiple\_polylog\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (ELi\_kernel)

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Ebar\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Kronecker\_dtau\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Kronecker\_dz\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Eisenstein\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (Eisenstein\_h\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (modular\_form\_kernel)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (user\_defined\_kernel)

### 7.53.1 Detailed Description

Interface to [GiNaC](#)'s integration kernels for iterated integrals.

## 7.54 Ist.cpp File Reference

Implementation of [GiNaC](#)'s Ist.

```
#include "lst.h"
#include "archive.h"
```

### Namespaces

- namespace [GiNaC](#)

### Variables

- `template<> GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T(Ist, basic, print_func< print_context >(&Ist::do_print). print_func< print_tree >(&Ist::do_print_tree)) template<> bool Ist GiNaC::GINAC\_BIND\_UNARCHIVER (Ist)`

*Specialization of [container::info\(\)](#) for Ist.*

### 7.54.1 Detailed Description

Implementation of [GiNaC](#)'s Ist.

## 7.55 Ist.h File Reference

Definition of [GiNaC](#)'s Ist.

```
#include "container.h"
#include <list>
```

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef container< std::list > [GiNaC::lst](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (lst)

### 7.55.1 Detailed Description

Definition of [GiNaC](#)'s lst.

## 7.56 matrix.cpp File Reference

Implementation of symbolic matrices.

```
#include "matrix.h"
#include "numeric.h"
#include "lst.h"
#include "idx.h"
#include "indexed.h"
#include "add.h"
#include "power.h"
#include "symbol.h"
#include "operators.h"
#include "normal.h"
#include "archive.h"
#include "utils.h"
#include <algorithm>
#include <iostream>
#include <map>
#include <sstream>
#include <stdexcept>
#include <string>
```

## Namespaces

- namespace [GiNaC](#)



## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (matrix, basic, print\_func< print\_context >(&matrix::do\_print). print\_func< print\_latex >(&matrix::do\_print\_latex). print\_func< print\_tree >(&matrix::do\_print\_tree). print\_func< print\_python\_repr >(&matrix::do\_print\_python\_repr)) matrix  
*Default ctor.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (matrix)
- ex [GiNaC::lst\\_to\\_matrix](#) (const lst &l)  
*Convert list of lists to matrix.*
- ex [GiNaC::diag\\_matrix](#) (const lst &l)  
*Convert list of diagonal elements to matrix.*
- ex [GiNaC::diag\\_matrix](#) (std::initializer\_list< ex > l)
- ex [GiNaC::unit\\_matrix](#) (unsigned r, unsigned c)  
*Create an r times c unit matrix.*
- ex [GiNaC::symbolic\\_matrix](#) (unsigned r, unsigned c, const std::string &base\_name, const std::string &tex\_↵ base\_name)  
*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- ex [GiNaC::reduced\\_matrix](#) (const matrix &m, unsigned r, unsigned c)  
*Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- ex [GiNaC::sub\\_matrix](#) (const matrix &m, unsigned r, unsigned nr, unsigned c, unsigned nc)  
*Return the nr times nc submatrix starting at position r, c of matrix m.*

### 7.56.1 Detailed Description

Implementation of symbolic matrices.

## 7.57 matrix.h File Reference

Interface to symbolic matrices.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include "compiler.h"
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::matrix](#)  
*Symbolic matrices.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- `GiNaC::GINAC_DECLARE_UNARCHIVER` (matrix)
- `size_t GiNaC::nops` (const matrix &m)
- ex `GiNaC::expand` (const matrix &m, unsigned options=0)
- ex `GiNaC::evalf` (const matrix &m)
- unsigned `GiNaC::rows` (const matrix &m)
- unsigned `GiNaC::cols` (const matrix &m)
- matrix `GiNaC::transpose` (const matrix &m)
- ex `GiNaC::determinant` (const matrix &m, unsigned options=determinant\_algo::automatic)
- ex `GiNaC::trace` (const matrix &m)
- ex `GiNaC::charpoly` (const matrix &m, const ex &lambda)
- matrix `GiNaC::inverse` (const matrix &m)
- matrix `GiNaC::inverse` (const matrix &m, unsigned algo)
- unsigned `GiNaC::rank` (const matrix &m)
- unsigned `GiNaC::rank` (const matrix &m, unsigned solve\_algo)
- ex `GiNaC::lst_to_matrix` (const lst &l)
 

*Convert list of lists to matrix.*
- ex `GiNaC::diag_matrix` (const lst &l)
 

*Convert list of diagonal elements to matrix.*
- ex `GiNaC::diag_matrix` (std::initializer\_list< ex > l)
- ex `GiNaC::unit_matrix` (unsigned r, unsigned c)
 

*Create an r times c unit matrix.*
- ex `GiNaC::unit_matrix` (unsigned x)
 

*Create a x times x unit matrix.*
- ex `GiNaC::symbolic_matrix` (unsigned r, unsigned c, const std::string &base\_name, const std::string &tex\_↔  
base\_name)
 

*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- ex `GiNaC::reduced_matrix` (const matrix &m, unsigned r, unsigned c)
 

*Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- ex `GiNaC::sub_matrix` (const matrix &m, unsigned r, unsigned nr, unsigned c, unsigned nc)
 

*Return the nr times nc submatrix starting at position r, c of matrix m.*
- ex `GiNaC::symbolic_matrix` (unsigned r, unsigned c, const std::string &base\_name)
 

*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*

### 7.57.1 Detailed Description

Interface to symbolic matrices.

## 7.58 mul.cpp File Reference

Implementation of `GiNaC`'s products of expressions.

```
#include "mul.h"
#include "add.h"
#include "power.h"
#include "operators.h"
#include "matrix.h"
```

```
#include "indexed.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "symbol.h"
#include "compiler.h"
#include <iostream>
#include <limits>
#include <stdexcept>
#include <vector>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (mul, expairseq, print\_func< print\_context >(&mul::do\_print). print\_func< print\_latex >(&mul::do\_print\_latex). print\_func< print\_csrc >(&mul::do\_print\_csrc). print\_func< print\_tree >(&mul::do\_print\_tree). print\_func< print\_python\_repr >(&mul::do\_print\_python\_repr)) mul
- bool [GiNaC::tryfactsubs](#) (const ex &origfactor, const ex &patternfactor, int &nummatches, exmap &repls)
- bool [GiNaC::algebraic\\_match\\_mul\\_with\\_mul](#) (const mul &e, const ex &pat, exmap &repls, int factor, int &nummatches, const std::vector< bool > &subsed, std::vector< bool > &matched)  
*Checks whether e matches to the pattern pat and the (possibly to be updated) list of replacements repls.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (mul)

### 7.58.1 Detailed Description

Implementation of [GiNaC](#)'s products of expressions.

## 7.59 mul.h File Reference

Interface to [GiNaC](#)'s products of expressions.

```
#include "expairseq.h"
```

## Classes

- class [GiNaC::mul](#)  
*Product of expressions.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (mul)

### 7.59.1 Detailed Description

Interface to [GiNaC](#)'s products of expressions.

## 7.60 ncmul.cpp File Reference

Implementation of [GiNaC](#)'s non-commutative products of expressions.

```
#include "ncmul.h"
#include "ex.h"
#include "add.h"
#include "mul.h"
#include "clifford.h"
#include "matrix.h"
#include "archive.h"
#include "indexed.h"
#include "utils.h"
#include <algorithm>
#include <iostream>
#include <stdexcept>
```

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef std::vector< std::size\_t > [GiNaC::uintvector](#)
- typedef std::vector< unsigned > [GiNaC::unsignedvector](#)
- typedef std::vector< exvector > [GiNaC::exvectorvector](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (ncmul, exprseq, print\_func< print\_context >(&ncmul::do\_print). print\_func< print\_tree >(&ncmul::do\_print\_tree). print\_func< print\_csrc >(&ncmul::do\_print\_csrc). print\_func< print\_python\_repr >(&ncmul::do\_print\_csrc)) ncmul
- ex [GiNaC::reeval\\_ncmul](#) (const exvector &v)
- ex [GiNaC::hold\\_ncmul](#) (const exvector &v)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (ncmul)

### 7.60.1 Detailed Description

Implementation of [GiNaC](#)'s non-commutative products of expressions.

## 7.61 ncmul.h File Reference

Interface to [GiNaC](#)'s non-commutative products of expressions.

```
#include "exprseq.h"
#include "archive.h"
```

### Classes

- class [GiNaC::ncmul](#)  
*Non-commutative product of expressions.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (ncmul)
- ex [GiNaC::reeval\\_ncmul](#) (const exvector &v)
- ex [GiNaC::hold\\_ncmul](#) (const exvector &v)

#### 7.61.1 Detailed Description

Interface to [GiNaC](#)'s non-commutative products of expressions.

## 7.62 normal.cpp File Reference

This file implements several functions that work on univariate and multivariate polynomials and rational functions.

```
#include "normal.h"
#include "basic.h"
#include "ex.h"
#include "add.h"
#include "constant.h"
#include "expairseq.h"
#include "fail.h"
#include "inifcns.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "matrix.h"
#include "pseries.h"
#include "symbol.h"
#include "utils.h"
#include "polynomial/chinrem_gcd.h"
#include <algorithm>
#include <map>
```

## Classes

- struct [GiNaC::sym\\_desc](#)  
*This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".*
- class [GiNaC::gcdheu\\_failed](#)  
*Exception thrown by [heur\\_gcd\(\)](#) to signal failure.*
- struct [GiNaC::normal\\_map\\_function](#)  
*Function object to be applied by [basic::normal\(\)](#).*

## Namespaces

- namespace [GiNaC](#)

## Macros

- `#define` [FAST\\_COMPARE](#) 1
- `#define` [USE\\_REMEMBER](#) 0
- `#define` [USE\\_TRIAL\\_DIVISION](#) 0
- `#define` [STATISTICS](#) 0

## Typedefs

- `typedef std::vector< sym_desc >` [GiNaC::sym\\_desc\\_vec](#)

## Functions

- static bool [GiNaC::get\\_first\\_symbol](#) (const ex &e, ex &x)  
*Return pointer to first symbol found in expression.*
- static void [GiNaC::add\\_symbol](#) (const ex &s, sym\_desc\_vec &v)
- static void [GiNaC::collect\\_symbols](#) (const ex &e, sym\_desc\_vec &v)
- static void [GiNaC::get\\_symbol\\_stats](#) (const ex &a, const ex &b, sym\_desc\_vec &v)  
*Collect statistical information about symbols in polynomials.*
- static numeric [GiNaC::lcmcoeff](#) (const ex &e, const numeric &l)
- static numeric [GiNaC::lcm\\_of\\_coefficients\\_denominators](#) (const ex &e)  
*Compute LCM of denominators of coefficients of a polynomial.*
- static ex [GiNaC::multiply\\_lcm](#) (const ex &e, const numeric &lcm)  
*Bring polynomial from  $Q[X]$  to  $Z[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.*
- ex [GiNaC::quo](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::rem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::decomp\\_rational](#) (const ex &a, const ex &x)  
*Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .*
- ex [GiNaC::prem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::sprem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- bool [GiNaC::divide](#) (const ex &a, const ex &b, ex &q, bool check\_args)

- Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .*
  - static bool [GiNaC::divide\\_in\\_z](#) (const ex &a, const ex &b, ex &q, sym\_desc\_vec::const\_iterator var)
- Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Z[X]$ .*
  - static ex [GiNaC::sr\\_gcd](#) (const ex &a, const ex &b, sym\_desc\_vec::const\_iterator var)
- Compute GCD of multivariate polynomials using the subresultant PRS algorithm.*
  - static ex [GiNaC::interpolate](#) (const ex &gamma, const numeric &xi, const ex &x, int degree\_hint=1)
- $\xi$ -adic polynomial interpolation*
  - static bool [GiNaC::heur\\_gcd\\_z](#) (ex &res, const ex &a, const ex &b, ex \*ca, ex \*cb, sym\_desc\_vec::const\_iterator var)
- Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
  - static bool [GiNaC::heur\\_gcd](#) (ex &res, const ex &a, const ex &b, ex \*ca, ex \*cb, sym\_desc\_vec::const\_iterator var)
- Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
  - static ex [GiNaC::gcd\\_pf\\_pow](#) (const ex &a, const ex &b, ex \*ca, ex \*cb)
  - static ex [GiNaC::gcd\\_pf\\_mul](#) (const ex &a, const ex &b, ex \*ca, ex \*cb)
  - ex [GiNaC::gcd](#) (const ex &a, const ex &b, ex \*ca, ex \*cb, bool check\_args, unsigned options)
- Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $Z[X]$ .*
  - static ex [GiNaC::gcd\\_pf\\_pow\\_pow](#) (const ex &a, const ex &b, ex \*ca, ex \*cb)
  - ex [GiNaC::lcm](#) (const ex &a, const ex &b, bool check\_args)
- Compute LCM (Least Common Multiple) of multivariate polynomials in  $Z[X]$ .*
  - static epvector [GiNaC::sqrfree\\_yun](#) (const ex &a, const symbol &x)
- Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.*
  - ex [GiNaC::sqrfree](#) (const ex &a, const lst &l)
- Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
  - ex [GiNaC::sqrfree\\_parfrac](#) (const ex &a, const symbol &x)
- Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
  - static ex [GiNaC::replace\\_with\\_symbol](#) (const ex &e, exmap &repl, exmap &rev\_lookup, lst &modifier)
- Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
  - static ex [GiNaC::replace\\_with\\_symbol](#) (const ex &e, exmap &repl)
- Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
  - static ex [GiNaC::frac\\_cancel](#) (const ex &n, const ex &d)
- Fraction cancellation.*
  - static ex [GiNaC::find\\_common\\_factor](#) (const ex &e, ex &factor, exmap &repl)
- Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).*
  - ex [GiNaC::collect\\_common\\_factors](#) (const ex &e)
- Collect common factors in sums.*
  - ex [GiNaC::resultant](#) (const ex &e1, const ex &e2, const ex &s)
- Resultant of two expressions  $e_1, e_2$  with respect to symbol  $s$ .*

## 7.62.1 Detailed Description

This file implements several functions that work on univariate and multivariate polynomials and rational functions.

These functions include polynomial quotient and remainder, GCD and LCM computation, square-free factorization and rational function normalization.

## 7.62.2 Macro Definition Documentation

### 7.62.2.1 FAST\_COMPARE

```
#define FAST_COMPARE 1
```

### 7.62.2.2 USE\_REMEMBER

```
#define USE_REMEMBER 0
```

### 7.62.2.3 USE\_TRIAL\_DIVISION

```
#define USE_TRIAL_DIVISION 0
```

### 7.62.2.4 STATISTICS

```
#define STATISTICS 0
```

## 7.63 normal.h File Reference

This file defines several functions that work on univariate and multivariate polynomials and rational functions.

```
#include "lst.h"
```

### Classes

- struct [GiNaC::gcd\\_options](#)  
*Flags to control the behavior of [gcd\(\)](#) and friends.*

### Namespaces

- namespace [GiNaC](#)



## Functions

- ex [GiNaC::quo](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::rem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::decomp\\_rational](#) (const ex &a, const ex &x)  
*Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .*
- ex [GiNaC::prem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- ex [GiNaC::sprem](#) (const ex &a, const ex &b, const ex &x, bool check\_args)  
*Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- bool [GiNaC::divide](#) (const ex &a, const ex &b, ex &q, bool check\_args)  
*Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .*
- ex [GiNaC::gcd](#) (const ex &a, const ex &b, ex \*ca, ex \*cb, bool check\_args, unsigned options)  
*Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $Z[X]$ .*
- ex [GiNaC::lcm](#) (const ex &a, const ex &b, bool check\_args)  
*Compute LCM (Least Common Multiple) of multivariate polynomials in  $Z[X]$ .*
- ex [GiNaC::sqrfree](#) (const ex &a, const lst &l)  
*Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
- ex [GiNaC::sqrfree\\_parfrac](#) (const ex &a, const symbol &x)  
*Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
- ex [GiNaC::collect\\_common\\_factors](#) (const ex &e)  
*Collect common factors in sums.*
- ex [GiNaC::resultant](#) (const ex &e1, const ex &e2, const ex &s)  
*Resultant of two expressions  $e1, e2$  with respect to symbol  $s$ .*

### 7.63.1 Detailed Description

This file defines several functions that work on univariate and multivariate polynomials and rational functions.

These functions include polynomial quotient and remainder, GCD and LCM computation, square-free factorization and rational function normalization.

## 7.64 numeric.cpp File Reference

This file contains the interface to the underlying bignum package.

```
#include "numeric.h"
#include "ex.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include <limits>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
#include <cln/output.h>
#include <cln/integer_io.h>
```

```
#include <cln/integer_ring.h>
#include <cln/rational_io.h>
#include <cln/rational_ring.h>
#include <cln/lfloat_class.h>
#include <cln/lfloat_io.h>
#include <cln/real_io.h>
#include <cln/real_ring.h>
#include <cln/complex_io.h>
#include <cln/complex_ring.h>
#include <cln/numtheory.h>
```

## Classes

- class [GiNaC::lanczos\\_coeffs](#)

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (numeric, basic, print\_func< print\_context >(&numeric::do\_print). print\_func< print\_latex >(&numeric::do\_print\_latex). print\_func< print\_csrc >(&numeric::do\_print\_csrc). print\_func< print\_csrc\_cl\_N >(&numeric::do\_print\_csrc\_cl\_N). print\_func< print\_tree >(&numeric::do\_print\_tree). print\_func< print\_python\_repr >(&numeric::do\_print\_python\_repr))  
numeric  
default ctor.
- static const cln::cl\_F [GiNaC::make\\_real\\_float](#) (const cln::cl\_idecoded\_float &dec)  
Construct a floating point number from sign, mantissa, and exponent.
- static const cln::cl\_F [GiNaC::read\\_real\\_float](#) (std::istream &s)  
Read serialized floating point number.
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (numeric)
- static void [GiNaC::write\\_real\\_float](#) (std::ostream &s, const cln::cl\_R &n)
- static void [GiNaC::print\\_real\\_number](#) (const print\_context &c, const cln::cl\_R &x)  
Helper function to print a real number in a nicer way than is CLN's default.
- static void [GiNaC::print\\_integer\\_csrc](#) (const print\_context &c, const cln::cl\_I &x)  
Helper function to print integer number in C++ source format.
- static void [GiNaC::print\\_real\\_csrc](#) (const print\_context &c, const cln::cl\_R &x)  
Helper function to print real number in C++ source format.
- template<typename T1 , typename T2 >  
static bool [GiNaC::coerce](#) (T1 &dst, const T2 &arg)
- template<> bool [GiNaC::coerce< int, cln::cl\\_I >](#) (int &dst, const cln::cl\_I &arg)  
Check if CLN integer can be converted into int.
- template<> bool [GiNaC::coerce< unsigned int, cln::cl\\_I >](#) (unsigned int &dst, const cln::cl\_I &arg)
- static void [GiNaC::print\\_real\\_cl\\_N](#) (const print\_context &c, const cln::cl\_R &x)  
Helper function to print real number in C++ source format using cl\_N types.
- const numeric [GiNaC::exp](#) (const numeric &x)  
Exponential function.
- const numeric [GiNaC::log](#) (const numeric &x)  
Natural logarithm.

- const numeric [GiNaC::sin](#) (const numeric &x)  
*Numeric sine (trigonometric function).*
- const numeric [GiNaC::cos](#) (const numeric &x)  
*Numeric cosine (trigonometric function).*
- const numeric [GiNaC::tan](#) (const numeric &x)  
*Numeric tangent (trigonometric function).*
- const numeric [GiNaC::asin](#) (const numeric &x)  
*Numeric inverse sine (trigonometric function).*
- const numeric [GiNaC::acos](#) (const numeric &x)  
*Numeric inverse cosine (trigonometric function).*
- const numeric [GiNaC::atan](#) (const numeric &x)  
*Numeric arcustangent.*
- const numeric [GiNaC::atan](#) (const numeric &y, const numeric &x)  
*Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- const numeric [GiNaC::sinh](#) (const numeric &x)  
*Numeric hyperbolic sine (trigonometric function).*
- const numeric [GiNaC::cosh](#) (const numeric &x)  
*Numeric hyperbolic cosine (trigonometric function).*
- const numeric [GiNaC::tanh](#) (const numeric &x)  
*Numeric hyperbolic tangent (trigonometric function).*
- const numeric [GiNaC::asinh](#) (const numeric &x)  
*Numeric inverse hyperbolic sine (trigonometric function).*
- const numeric [GiNaC::acosh](#) (const numeric &x)  
*Numeric inverse hyperbolic cosine (trigonometric function).*
- const numeric [GiNaC::atanh](#) (const numeric &x)  
*Numeric inverse hyperbolic tangent (trigonometric function).*
- static cln::cl\_N [GiNaC::Li2\\_series](#) (const cln::cl\_N &x, const cln::float\_format\_t &prec)  
*Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.*
- static cln::cl\_N [GiNaC::Li2\\_projection](#) (const cln::cl\_N &x, const cln::float\_format\_t &prec)  
*Folds Li2's argument inside a small rectangle to enhance convergence.*
- const cln::cl\_N [GiNaC::Li2\\_](#) (const cln::cl\_N &value)  
*Numeric evaluation of Dilogarithm.*
- const numeric [GiNaC::Li2](#) (const numeric &x)
- const numeric [GiNaC::zeta](#) (const numeric &x)  
*Numeric evaluation of Riemann's Zeta function.*
- static cln::float\_format\_t [GiNaC::guess\\_precision](#) (const cln::cl\_N &x)
- const cln::cl\_N [GiNaC::lgamma](#) (const cln::cl\_N &x)  
*The Gamma function.*
- const numeric [GiNaC::lgamma](#) (const numeric &x)
- const cln::cl\_N [GiNaC::tgamma](#) (const cln::cl\_N &x)
- const numeric [GiNaC::tgamma](#) (const numeric &x)
- const numeric [GiNaC::psi](#) (const numeric &x)  
*The psi function (aka polygamma function).*
- const numeric [GiNaC::psi](#) (const numeric &n, const numeric &x)  
*The psi functions (aka polygamma functions).*
- const numeric [GiNaC::factorial](#) (const numeric &n)  
*Factorial combinatorial function.*
- const numeric [GiNaC::doublefactorial](#) (const numeric &n)  
*The double factorial combinatorial function.*
- const numeric [GiNaC::binomial](#) (const numeric &n, const numeric &k)  
*The Binomial coefficients.*

- const numeric [GiNaC::bernoulli](#) (const numeric &nn)  
*Bernoulli number.*
- const numeric [GiNaC::fibonacci](#) (const numeric &n)  
*Fibonacci number.*
- const numeric [GiNaC::abs](#) (const numeric &x)  
*Absolute value.*
- const numeric [GiNaC::mod](#) (const numeric &a, const numeric &b)  
*Modulus (in positive representation).*
- const numeric [GiNaC::smod](#) (const numeric &a\_, const numeric &b\_)  
*Modulus (in symmetric representation).*
- const numeric [GiNaC::irem](#) (const numeric &a, const numeric &b)  
*Numeric integer remainder.*
- const numeric [GiNaC::irem](#) (const numeric &a, const numeric &b, numeric &q)  
*Numeric integer remainder.*
- const numeric [GiNaC::iquo](#) (const numeric &a, const numeric &b)  
*Numeric integer quotient.*
- const numeric [GiNaC::iquo](#) (const numeric &a, const numeric &b, numeric &r)  
*Numeric integer quotient.*
- const numeric [GiNaC::gcd](#) (const numeric &a, const numeric &b)  
*Greatest Common Divisor.*
- const numeric [GiNaC::lcm](#) (const numeric &a, const numeric &b)  
*Least Common Multiple.*
- const numeric [GiNaC::sqrt](#) (const numeric &x)  
*Numeric square root.*
- const numeric [GiNaC::isqrt](#) (const numeric &x)  
*Integer numeric square root.*
- ex [GiNaC::PiEvalf](#) ()  
*Floating point evaluation of Archimedes' constant Pi.*
- ex [GiNaC::EulerEvalf](#) ()  
*Floating point evaluation of Euler's constant gamma.*
- ex [GiNaC::CatalanEvalf](#) ()  
*Floating point evaluation of Catalan's constant.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const \_numeric\_digits &e)

## Variables

- const numeric [GiNaC::I](#) = numeric(cln::complex(cln::cl\_I(0),cln::cl\_I(1)))  
*Imaginary unit.*
- \_numeric\_digits [GiNaC::Digits](#)  
*Accuracy in decimal digits.*

### 7.64.1 Detailed Description

This file contains the interface to the underlying bignum package.

Its most important design principle is to completely hide the inner working of that other package from the user of [GiNaC](#). It must either provide implementation of arithmetic operators and numerical evaluation of special functions or implement the interface to the bignum package.

## 7.65 numeric.h File Reference

Makes the interface to the underlying bignum package available.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include <cln/complex.h>
#include <stdexcept>
#include <vector>
```

### Classes

- class [GiNaC::numeric\\_digits](#)  
*This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.*
- class [GiNaC::pole\\_error](#)  
*Exception class thrown when a singularity is encountered.*
- class [GiNaC::numeric](#)  
*This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.*

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef void(\* [GiNaC::digits\\_changed\\_callback](#)) (long)  
*Function pointer to implement callbacks in the case 'Digits' gets changed.*

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (numeric)
- const numeric [GiNaC::exp](#) (const numeric &x)  
*Exponential function.*
- const numeric [GiNaC::log](#) (const numeric &x)  
*Natural logarithm.*
- const numeric [GiNaC::sin](#) (const numeric &x)  
*Numeric sine (trigonometric function).*
- const numeric [GiNaC::cos](#) (const numeric &x)  
*Numeric cosine (trigonometric function).*
- const numeric [GiNaC::tan](#) (const numeric &x)  
*Numeric tangent (trigonometric function).*
- const numeric [GiNaC::asin](#) (const numeric &x)  
*Numeric inverse sine (trigonometric function).*
- const numeric [GiNaC::acos](#) (const numeric &x)  
*Numeric inverse cosine (trigonometric function).*
- const numeric [GiNaC::atan](#) (const numeric &x)  
*Numeric arcustangent.*

- const numeric [GiNaC::atan](#) (const numeric &y, const numeric &x)  
*Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- const numeric [GiNaC::sinh](#) (const numeric &x)  
*Numeric hyperbolic sine (trigonometric function).*
- const numeric [GiNaC::cosh](#) (const numeric &x)  
*Numeric hyperbolic cosine (trigonometric function).*
- const numeric [GiNaC::tanh](#) (const numeric &x)  
*Numeric hyperbolic tangent (trigonometric function).*
- const numeric [GiNaC::asinh](#) (const numeric &x)  
*Numeric inverse hyperbolic sine (trigonometric function).*
- const numeric [GiNaC::acosh](#) (const numeric &x)  
*Numeric inverse hyperbolic cosine (trigonometric function).*
- const numeric [GiNaC::atanh](#) (const numeric &x)  
*Numeric inverse hyperbolic tangent (trigonometric function).*
- const numeric [GiNaC::Li2](#) (const numeric &x)
- const numeric [GiNaC::zeta](#) (const numeric &x)  
*Numeric evaluation of Riemann's Zeta function.*
- const numeric [GiNaC::lgamma](#) (const numeric &x)
- const numeric [GiNaC::tgamma](#) (const numeric &x)
- const numeric [GiNaC::psi](#) (const numeric &x)  
*The psi function (aka polygamma function).*
- const numeric [GiNaC::psi](#) (const numeric &n, const numeric &x)  
*The psi functions (aka polygamma functions).*
- const numeric [GiNaC::factorial](#) (const numeric &n)  
*Factorial combinatorial function.*
- const numeric [GiNaC::doublefactorial](#) (const numeric &n)  
*The double factorial combinatorial function.*
- const numeric [GiNaC::binomial](#) (const numeric &n, const numeric &k)  
*The Binomial coefficients.*
- const numeric [GiNaC::bernoulli](#) (const numeric &nn)  
*Bernoulli number.*
- const numeric [GiNaC::fibonacci](#) (const numeric &n)  
*Fibonacci number.*
- const numeric [GiNaC::isqrt](#) (const numeric &x)  
*Integer numeric square root.*
- const numeric [GiNaC::sqrt](#) (const numeric &x)  
*Numeric square root.*
- const numeric [GiNaC::abs](#) (const numeric &x)  
*Absolute value.*
- const numeric [GiNaC::mod](#) (const numeric &a, const numeric &b)  
*Modulus (in positive representation).*
- const numeric [GiNaC::smod](#) (const numeric &a\_, const numeric &b\_)  
*Modulus (in symmetric representation).*
- const numeric [GiNaC::irem](#) (const numeric &a, const numeric &b)  
*Numeric integer remainder.*
- const numeric [GiNaC::irem](#) (const numeric &a, const numeric &b, numeric &q)  
*Numeric integer remainder.*
- const numeric [GiNaC::iquo](#) (const numeric &a, const numeric &b)  
*Numeric integer quotient.*
- const numeric [GiNaC::iquo](#) (const numeric &a, const numeric &b, numeric &r)  
*Numeric integer quotient.*

- const numeric [GiNaC::gcd](#) (const numeric &a, const numeric &b)  
*Greatest Common Divisor.*
- const numeric [GiNaC::lcm](#) (const numeric &a, const numeric &b)  
*Least Common Multiple.*
- const numeric [GiNaC::pow](#) (const numeric &x, const numeric &y)
- const numeric [GiNaC::inverse](#) (const numeric &x)
- numeric [GiNaC::step](#) (const numeric &x)
- int [GiNaC::csgn](#) (const numeric &x)
- bool [GiNaC::is\\_zero](#) (const numeric &x)
- bool [GiNaC::is\\_positive](#) (const numeric &x)
- bool [GiNaC::is\\_negative](#) (const numeric &x)
- bool [GiNaC::is\\_integer](#) (const numeric &x)
- bool [GiNaC::is\\_pos\\_integer](#) (const numeric &x)
- bool [GiNaC::is\\_nonneg\\_integer](#) (const numeric &x)
- bool [GiNaC::is\\_even](#) (const numeric &x)
- bool [GiNaC::is\\_odd](#) (const numeric &x)
- bool [GiNaC::is\\_prime](#) (const numeric &x)
- bool [GiNaC::is\\_rational](#) (const numeric &x)
- bool [GiNaC::is\\_real](#) (const numeric &x)
- bool [GiNaC::is\\_cinteger](#) (const numeric &x)
- bool [GiNaC::is\\_crational](#) (const numeric &x)
- int [GiNaC::to\\_int](#) (const numeric &x)
- long [GiNaC::to\\_long](#) (const numeric &x)
- double [GiNaC::to\\_double](#) (const numeric &x)
- const numeric [GiNaC::real](#) (const numeric &x)
- const numeric [GiNaC::imag](#) (const numeric &x)
- const numeric [GiNaC::numer](#) (const numeric &x)
- const numeric [GiNaC::denom](#) (const numeric &x)
- ex [GiNaC::PiEvalf](#) ()  
*Floating point evaluation of Archimedes' constant Pi.*
- ex [GiNaC::EulerEvalf](#) ()  
*Floating point evaluation of Euler's constant gamma.*
- ex [GiNaC::CatalanEvalf](#) ()  
*Floating point evaluation of Catalan's constant.*

### 7.65.1 Detailed Description

Makes the interface to the underlying bignum package available.

## 7.66 operators.cpp File Reference

Implementation of [GiNaC](#)'s overloaded operators.

```
#include "operators.h"
#include "numeric.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "ncmul.h"
#include "relational.h"
#include "print.h"
#include "utils.h"
#include <iostream>
```

## Namespaces

- namespace [GiNaC](#)

## Enumerations

- enum { [GiNaC::callback\\_registered](#) = 1 }

## Functions

- static const ex [GiNaC::exadd](#) (const ex &lh, const ex &rh)  
*Used internally by [operator+\(\)](#) to add two ex objects.*
- static const ex [GiNaC::exmul](#) (const ex &lh, const ex &rh)  
*Used internally by [operator\\*\(\)](#) to multiply two ex objects.*
- static const ex [GiNaC::exminus](#) (const ex &lh)  
*Used internally by [operator-\(\)](#) and friends to change the sign of an argument.*
- const ex [GiNaC::operator+](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator-](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator\\*](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator/](#) (const ex &lh, const ex &rh)
- const numeric [GiNaC::operator+](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator-](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator\\*](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator/](#) (const numeric &lh, const numeric &rh)
- ex & [GiNaC::operator+=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator-=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator\\*=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator/=](#) (ex &lh, const ex &rh)
- numeric & [GiNaC::operator+=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator-=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator\\*=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator/=](#) (numeric &lh, const numeric &rh)
- const ex [GiNaC::operator+](#) (const ex &lh)
- const ex [GiNaC::operator-](#) (const ex &lh)
- const numeric [GiNaC::operator+](#) (const numeric &lh)
- const numeric [GiNaC::operator-](#) (const numeric &lh)
- ex & [GiNaC::operator++](#) (ex &rh)  
*Expression prefix increment.*
- ex & [GiNaC::operator--](#) (ex &rh)  
*Expression prefix decrement.*
- const ex [GiNaC::operator++](#) (ex &lh, int)  
*Expression postfix increment.*
- const ex [GiNaC::operator--](#) (ex &lh, int)  
*Expression postfix decrement.*
- numeric & [GiNaC::operator++](#) (numeric &rh)  
*Numeric prefix increment.*
- numeric & [GiNaC::operator--](#) (numeric &rh)  
*Numeric prefix decrement.*
- const numeric [GiNaC::operator++](#) (numeric &lh, int)  
*Numeric postfix increment.*
- const numeric [GiNaC::operator--](#) (numeric &lh, int)



*Numeric postfix decrement.*

- const relational [GiNaC::operator==](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator!=](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator<](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator<=](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator>](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator>=](#) (const ex &lh, const ex &rh)
- static int [GiNaC::my\\_ios\\_index](#) ()
- static void [GiNaC::my\\_ios\\_callback](#) (std::ios\_base::event ev, std::ios\_base &s, int i)
- static print\_context \* [GiNaC::get\\_print\\_context](#) (std::ios\_base &s)
- static void [GiNaC::set\\_print\\_context](#) (std::ios\_base &s, const print\_context &c)
- static unsigned [GiNaC::get\\_print\\_options](#) (std::ios\_base &s)
- static void [GiNaC::set\\_print\\_options](#) (std::ostream &s, unsigned options)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const ex &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const exvector &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const exset &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const exmap &e)
- std::istream & [GiNaC::operator>>](#) (std::istream &is, ex &e)
- std::ostream & [GiNaC::dflt](#) (std::ostream &os)
- std::ostream & [GiNaC::latex](#) (std::ostream &os)
- std::ostream & [GiNaC::python](#) (std::ostream &os)
- std::ostream & [GiNaC::python\\_repr](#) (std::ostream &os)
- std::ostream & [GiNaC::tree](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_float](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_double](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_cl\\_N](#) (std::ostream &os)
- std::ostream & [GiNaC::index\\_dimensions](#) (std::ostream &os)
- std::ostream & [GiNaC::no\\_index\\_dimensions](#) (std::ostream &os)

### 7.66.1 Detailed Description

Implementation of [GiNaC](#)'s overloaded operators.

## 7.67 operators.h File Reference

Interface to [GiNaC](#)'s overloaded operators.

```
#include <iosfwd>
```

### Namespaces

- namespace [GiNaC](#)

## Functions

- const ex [GiNaC::operator+](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator-](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator\\*](#) (const ex &lh, const ex &rh)
- const ex [GiNaC::operator/](#) (const ex &lh, const ex &rh)
- const numeric [GiNaC::operator+](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator-](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator\\*](#) (const numeric &lh, const numeric &rh)
- const numeric [GiNaC::operator/](#) (const numeric &lh, const numeric &rh)
- ex & [GiNaC::operator+=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator-=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator\\*=](#) (ex &lh, const ex &rh)
- ex & [GiNaC::operator/=](#) (ex &lh, const ex &rh)
- numeric & [GiNaC::operator+=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator-=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator\\*=](#) (numeric &lh, const numeric &rh)
- numeric & [GiNaC::operator/=](#) (numeric &lh, const numeric &rh)
- const ex [GiNaC::operator+](#) (const ex &lh)
- const ex [GiNaC::operator-](#) (const ex &lh)
- const numeric [GiNaC::operator+](#) (const numeric &lh)
- const numeric [GiNaC::operator-](#) (const numeric &lh)
- ex & [GiNaC::operator++](#) (ex &rh)
- Expression prefix increment.*
- ex & [GiNaC::operator--](#) (ex &rh)
- Expression prefix decrement.*
- const ex [GiNaC::operator++](#) (ex &lh, int)
- Expression postfix increment.*
- const ex [GiNaC::operator--](#) (ex &lh, int)
- Expression postfix decrement.*
- numeric & [GiNaC::operator++](#) (numeric &rh)
- Numeric prefix increment.*
- numeric & [GiNaC::operator--](#) (numeric &rh)
- Numeric prefix decrement.*
- const numeric [GiNaC::operator++](#) (numeric &lh, int)
- Numeric postfix increment.*
- const numeric [GiNaC::operator--](#) (numeric &lh, int)
- Numeric postfix decrement.*
- const relational [GiNaC::operator==](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator!=](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator<](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator<=](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator>](#) (const ex &lh, const ex &rh)
- const relational [GiNaC::operator>=](#) (const ex &lh, const ex &rh)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const ex &e)
- std::istream & [GiNaC::operator>>](#) (std::istream &is, ex &e)
- std::ostream & [GiNaC::dflt](#) (std::ostream &os)
- std::ostream & [GiNaC::latex](#) (std::ostream &os)
- std::ostream & [GiNaC::python](#) (std::ostream &os)
- std::ostream & [GiNaC::python\\_repr](#) (std::ostream &os)
- std::ostream & [GiNaC::tree](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_float](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_double](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_cl\\_N](#) (std::ostream &os)
- std::ostream & [GiNaC::index\\_dimensions](#) (std::ostream &os)
- std::ostream & [GiNaC::no\\_index\\_dimensions](#) (std::ostream &os)

### 7.67.1 Detailed Description

Interface to [GiNaC](#)'s overloaded operators.

## 7.68 power.cpp File Reference

Implementation of [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

```
#include "power.h"
#include "expairseq.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "numeric.h"
#include "constant.h"
#include "operators.h"
#include "inifcns.h"
#include "matrix.h"
#include "indexed.h"
#include "symbol.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "relational.h"
#include "compiler.h"
#include <iostream>
#include <limits>
#include <stdexcept>
#include <vector>
#include <algorithm>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (power, basic, print\_func< print\_dflt >(&power::do\_print\_dflt). print\_func< print\_latex >(&power::do\_print\_latex). print\_func< print\_csrc >(&power::do\_print\_csrc). print\_func< print\_python >(&power::do\_print\_python). print\_func< print\_python\_repr >(&power::do\_print\_python\_repr). print\_func< print\_csrc\_cl\_N >(&power::do\_print\_csrc\_cl\_N)) power
- static void [GiNaC::print\\_sym\\_pow](#) (const print\_context &c, const symbol &x, int exp)
- bool [GiNaC::tryfactsubs](#) (const ex &origfactor, const ex &patternfactor, int &nummatches, exmap &repls)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (power)

### 7.68.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

## 7.69 power.h File Reference

Interface to [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::power](#)

*This class holds a two-component object, a basis and an exponent representing exponentiation.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (power)
- ex [GiNaC::pow](#) (const ex &b, const ex &e)  
*Symbolic exponentiation.*
- template<typename T1, typename T2 >  
ex [GiNaC::pow](#) (const T1 &b, const T2 &e)
- ex [GiNaC::sqrt](#) (const ex &a)  
*Square root expression.*

#### 7.69.1 Detailed Description

Interface to [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

## 7.70 print.cpp File Reference

Implementation of helper classes for expression output.

```
#include "print.h"
#include <iostream>
```

### Namespaces

- namespace [GiNaC](#)

## Variables

- unsigned [GiNaC::next\\_print\\_context\\_id](#) = 0  
Next free ID for [print\\_context](#) types.

### 7.70.1 Detailed Description

Implementation of helper classes for expression output.

## 7.71 print.h File Reference

Definition of helper classes for expression output.

```
#include "class_info.h"
#include <iosfwd>
#include <memory>
#include <string>
```

## Classes

- class [GiNaC::print\\_context\\_options](#)  
*This class stores information about a registered [print\\_context](#) class.*
- class [GiNaC::print\\_options](#)  
*Flags to control the behavior of a [print\\_context](#).*
- class [GiNaC::print\\_context](#)  
*Base class for [print\\_contexts](#).*
- class [GiNaC::print\\_dflt](#)  
*Context for default (ginsh-parsable) output.*
- class [GiNaC::print\\_latex](#)  
*Context for latex-parsable output.*
- class [GiNaC::print\\_python](#)  
*Context for python pretty-print output.*
- class [GiNaC::print\\_python\\_repr](#)  
*Context for python-parsable output.*
- class [GiNaC::print\\_tree](#)  
*Context for tree-like output for debugging.*
- class [GiNaC::print\\_csrc](#)  
*Base context for C source output.*
- class [GiNaC::print\\_csrc\\_float](#)  
*Context for C source output using float precision.*
- class [GiNaC::print\\_csrc\\_double](#)  
*Context for C source output using double precision.*
- class [GiNaC::print\\_csrc\\_cl\\_N](#)  
*Context for C source output using CLN numbers.*
- class [GiNaC::print\\_functor\\_impl](#)  
*Base class for [print\\_functor](#) handlers.*
- class [GiNaC::print\\_ptrfun\\_handler< T, C >](#)  
*[print\\_functor](#) handler for pointer-to-functions of class *T*, context type *C**
- class [GiNaC::print\\_memfun\\_handler< T, C >](#)  
*[print\\_functor](#) handler for member functions of class *T*, context type *C**
- class [GiNaC::print\\_functor](#)  
*This class represents a print method for a certain algebraic class and [print\\_context](#) type.*

## Namespaces

- namespace [GiNaC](#)

## Macros

- `#define GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON(classname)`  
*Common part of `GINAC_DECLARE_PRINT_CONTEXT_BASE` and `GINAC_DECLARE_PRINT_CONTEXT_DERIVED`.*
- `#define GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE(classname)`
- `#define GINAC\_DECLARE\_PRINT\_CONTEXT(classname, supertype)`  
*Macro for inclusion in the declaration of a `print_context` class.*
- `#define GINAC\_IMPLEMENT\_PRINT\_CONTEXT(classname, supertype)`  
*Macro for inclusion in the implementation of each `print_context` class.*

## Typedefs

- `typedef class_info< print_context_options > GiNaC::print\_context\_class\_info`

## Functions

- `template<class T >`  
`bool GiNaC::is\_a (const print_context &obj)`  
*Check if `obj` is a `T`, including base classes.*

### 7.71.1 Detailed Description

Definition of helper classes for expression output.

### 7.71.2 Macro Definition Documentation

#### 7.71.2.1 `GINAC_DECLARE_PRINT_CONTEXT_COMMON`

```
#define GINAC_DECLARE_PRINT_CONTEXT_COMMON(  
    classname )
```

##### Value:

```
public: \
    friend class function_options; \
    friend class registered_class_options; \
    static const GiNaC::print\_context\_class\_info &get_class_info_static(); \
    classname();
```

Common part of `GINAC_DECLARE_PRINT_CONTEXT_BASE` and `GINAC_DECLARE_PRINT_CONTEXT_DERIVED`.

### 7.71.2.2 GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE

```
#define GINAC_DECLARE_PRINT_CONTEXT_BASE(  
    classname )
```

**Value:**

```
GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname) \  
virtual const GiNaC::print_context_class_info &get_class_info() const { return  
    classname::get_class_info_static(); } \  
virtual const char *class_name() const { return classname::get_class_info_static().options.get_name(); }  
\  
virtual classname * duplicate() const { return new classname(*this); } \  
private:
```

### 7.71.2.3 GINAC\_DECLARE\_PRINT\_CONTEXT

```
#define GINAC_DECLARE_PRINT_CONTEXT(  
    classname,  
    supertype )
```

**Value:**

```
GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname) \  
typedef supertype inherited; \  
const GiNaC::print_context_class_info &get_class_info() const override { return  
    classname::get_class_info_static(); } \  
const char *class_name() const override { return classname::get_class_info_static().options.get_name();  
} \  
classname * duplicate() const override { return new classname(*this); } \  
private:
```

Macro for inclusion in the declaration of a `print_context` class.

It declares some functions that are common to all classes derived from 'print\_context' as well as all required stuff for the [GiNaC](#) registry.

### 7.71.2.4 GINAC\_IMPLEMENT\_PRINT\_CONTEXT

```
#define GINAC_IMPLEMENT_PRINT_CONTEXT(  
    classname,  
    supertype )
```

**Value:**

```
const GiNaC::print_context_class_info &classname::get_class_info_static() \  
{ \  
    static GiNaC::print_context_class_info reg_info =  
        GiNaC::print_context_class_info(GiNaC::print_context_options(#classname, #supertype,  
        GiNaC::next_print_context_id++)); \  
    return reg_info; \  
}
```

Macro for inclusion in the implementation of each `print_context` class.

## 7.72 pseries.cpp File Reference

Implementation of class for extended truncated power series and methods for series expansion.

```
#include "pseries.h"
#include "add.h"
#include "inifcns.h"
#include "lst.h"
#include "mul.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "symbol.h"
#include "integral.h"
#include "archive.h"
#include "utils.h"
#include <limits>
#include <numeric>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (pseries, basic, print\_func< print\_context >(&pseries::do\_print). print\_func< print\_latex >(&pseries::do\_print\_latex). print\_func< print\_tree >(&pseries::do\_print\_tree). print\_func< print\_python >(&pseries::do\_print\_python). print\_func< print\_python\_repr >(&pseries::do\_print\_python\_repr)) pseries
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (pseries)

#### 7.72.1 Detailed Description

Implementation of class for extended truncated power series and methods for series expansion.

## 7.73 pseries.h File Reference

Interface to class for extended truncated power series.

```
#include "basic.h"
#include "expairseq.h"
```

### Classes

- class [GiNaC::pseries](#)

*This class holds a extended truncated power series (positive and negative integer powers).*



## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (pseries)
- ex [GiNaC::series\\_to\\_poly](#) (const ex &e)  
*Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.*
- bool [GiNaC::is\\_terminating](#) (const pseries &s)

### 7.73.1 Detailed Description

Interface to class for extended truncated power series.

## 7.74 ptr.h File Reference

Reference-counted pointer template.

```
#include "assertion.h"
#include <cstddef>
#include <functional>
#include <iosfwd>
```

## Classes

- class [GiNaC::refcounted](#)  
*Base class for reference-counted objects.*
- class [GiNaC::ptr< T >](#)  
*Class of (intrusively) reference-counted pointers that support copy-on-write semantics.*
- struct [std::less< GiNaC::ptr< T > >](#)  
*Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.*

## Namespaces

- namespace [GiNaC](#)
- namespace [std](#)

### 7.74.1 Detailed Description

Reference-counted pointer template.

## 7.75 registrar.cpp File Reference

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

```
#include "registrar.h"
#include <map>
#include <stdexcept>
#include <string>
```

### Namespaces

- namespace [GiNaC](#)

#### 7.75.1 Detailed Description

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

## 7.76 registrar.h File Reference

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

```
#include "class_info.h"
#include "print.h"
#include <list>
#include <string>
#include <typeinfo>
#include <vector>
```

### Classes

- struct [GiNaC::return\\_type\\_t](#)  
*To distinguish between different kinds of non-commutative objects.*
- class [GiNaC::registered\\_class\\_options](#)  
*This class stores information about a registered [GiNaC](#) class.*

### Namespaces

- namespace [GiNaC](#)

## Macros

- #define [GINAC\\_DECLARE\\_REGISTERED\\_CLASS\\_COMMON](#)(classname)  
*Common part of GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS and GINAC\_DECLARE\_REGISTERED\_CLASS.*
- #define [GINAC\\_DECLARE\\_REGISTERED\\_CLASS\\_NO\\_CTORS](#)(classname, supertype)  
*Primary macro for inclusion in the declaration of each registered class.*
- #define [GINAC\\_DECLARE\\_REGISTERED\\_CLASS](#)(classname, supertype)  
*Macro for inclusion in the declaration of each registered class.*
- #define [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS](#)(classname, supertype) [GiNaC::registered\\_class\\_info](#) classname::reg\_info = [GiNaC::registered\\_class\\_info](#)([GiNaC::registered\\_class\\_options](#)(#classname, #supertype, typeid(classname)));  
*Macro for inclusion in the implementation of each registered class.*
- #define [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)(classname, supertype, options) [GiNaC::registered\\_class\\_info](#) classname::reg\_info = [GiNaC::registered\\_class\\_info](#)([GiNaC::registered\\_class\\_options](#)(#classname, #supertype, typeid(classname)).options);  
*Macro for inclusion in the implementation of each registered class.*
- #define [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\\_T](#)(classname, supertype, options) [GiNaC::registered\\_class\\_info](#) classname::reg\_info = [GiNaC::registered\\_class\\_info](#)([GiNaC::registered\\_class\\_options](#)(#classname, #supertype, typeid(classname)).options);  
*Macro for inclusion in the implementation of each registered class.*

## Typedefs

- typedef class\_info< registered\_class\_options > [GiNaC::registered\\_class\\_info](#)

## Functions

- template<typename T >  
return\_type\_t [GiNaC::make\\_return\\_type\\_t](#) (const unsigned rl=0)
- template<class Alg, class Ctx, class T, class C >  
void [GiNaC::set\\_print\\_func](#) (void f(const T &, const C &, unsigned))  
*Add or replace a print method.*
- template<class Alg, class Ctx, class T, class C >  
void [GiNaC::set\\_print\\_func](#) (void(T::\*f)(const C &, unsigned))  
*Add or replace a print method.*

### 7.76.1 Detailed Description

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

### 7.76.2 Macro Definition Documentation

### 7.76.2.1 GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON

```
#define GINAC_DECLARE_REGISTERED_CLASS_COMMON(  
    classname )
```

#### Value:

```
private: \
    static GiNaC::registered_class_info reg_info; \
public: \
    static GiNaC::registered_class_info &get_class_info_static() { return reg_info; } \
    class visitor { \
    public: \
        virtual void visit(const classname &) = 0; \
        virtual ~visitor() {}; \
    };
```

Common part of GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS and GINAC\_DECLARE\_REGISTERED\_CLASS.

### 7.76.2.2 GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS

```
#define GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS(  
    classname,  
    supertype )
```

#### Value:

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \
typedef supertype inherited; \
virtual const GiNaC::registered_class_info &get_class_info() const { return  
    classname::get_class_info_static(); } \
virtual GiNaC::registered_class_info &get_class_info() { return classname::get_class_info_static(); } \
virtual const char *class_name() const { return classname::get_class_info_static().options.get_name(); } \
private:
```

Primary macro for inclusion in the declaration of each registered class.

### 7.76.2.3 GINAC\_DECLARE\_REGISTERED\_CLASS

```
#define GINAC_DECLARE_REGISTERED_CLASS(  
    classname,  
    supertype )
```

#### Value:

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \
template<class B, typename... Args> friend B & dynallocate(Args &&... args); \
typedef supertype inherited; \
classname(); \
classname * duplicate() const override { \
    classname * bp = new classname(*this); \
    bp->setflag(GiNaC::status_flags::dynallocated); \
    return bp; \
} \
void accept(GiNaC::visitor & v) const override \
{ \
    if (visitor *p = dynamic_cast<visitor *>(&v)) \
        p->visit(*this); \
    else \
        inherited::accept(v); \
} \
const GiNaC::registered_class_info &get_class_info() const override { return  
    classname::get_class_info_static(); } \
```

```

    GiNaC::registered_class_info &get_class_info() override { return classname::get_class_info_static(); } \
    const char *class_name() const override { return classname::get_class_info_static().options.get_name(); \
    } \
protected: \
    int compare_same_type(const GiNaC::basic & other) const override; \
private:

```

Macro for inclusion in the declaration of each registered class.

It declares some functions that are common to all classes derived from 'basic' as well as all required stuff for the [GiNaC](#) class registry (mainly needed for archiving).

#### 7.76.2.4 GINAC\_IMPLEMENT\_REGISTERED\_CLASS

```

#define GINAC_IMPLEMENT_REGISTERED_CLASS( \
    classname, \
    supertype ) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC \
    #supertype, typeid(classname));

```

Macro for inclusion in the implementation of each registered class.

#### 7.76.2.5 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT

```

#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT( \
    classname, \
    supertype, \
    options ) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC \
    #supertype, typeid(classname)).options);

```

Macro for inclusion in the implementation of each registered class.

Additional options can be specified.

#### 7.76.2.6 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T

```

#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T( \
    classname, \
    supertype, \
    options ) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC \
    #supertype, typeid(classname)).options);

```

Macro for inclusion in the implementation of each registered class.

Additional options can be specified.

## 7.77 relational.cpp File Reference

Implementation of relations between expressions.

```
#include "relational.h"
#include "operators.h"
#include "numeric.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <iostream>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (relational, basic, print\_func< print\_context >(&relational::do\_print). print\_func< print\_tree >(&relational::do\_print\_tree). print\_func< print\_python\_repr >(&relational::do\_print\_python\_repr)) relational
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (relational)
- static void [GiNaC::print\\_operator](#) (const print\_context &c, relational::operators o)

#### 7.77.1 Detailed Description

Implementation of relations between expressions.

## 7.78 relational.h File Reference

Interface to relations between expressions.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::relational](#)  
*This class holds a relation consisting of two expressions and a logical relation between them.*
- struct [GiNaC::relational::safe\\_bool\\_helper](#)

### Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (relational)

### 7.78.1 Detailed Description

Interface to relations between expressions.

## 7.79 remember.cpp File Reference

Implementation of helper classes for using the remember option in [GiNaC](#) functions.

```
#include "function.h"
#include "utils.h"
#include "remember.h"
#include <stdexcept>
```

## Namespaces

- namespace [GiNaC](#)

### 7.79.1 Detailed Description

Implementation of helper classes for using the remember option in [GiNaC](#) functions.

## 7.80 remember.h File Reference

Interface to helper classes for using the remember option in [GiNaC](#) functions.

```
#include <iosfwd>
#include <list>
#include <vector>
```

## Classes

- class [GiNaC::remember\\_table\\_entry](#)  
*A single entry in the remember table of a function.*
- class [GiNaC::remember\\_table\\_list](#)  
*A list of entries in the remember table having some least significant bits of the hashvalue in common.*
- class [GiNaC::remember\\_table](#)  
*The remember table is organized like an n-fold associative cache in a microprocessor.*

## Namespaces

- namespace [GiNaC](#)

### 7.80.1 Detailed Description

Interface to helper classes for using the remember option in [GiNaC](#) functions.

## 7.81 structure.h File Reference

Wrapper template for making [GiNaC](#) classes out of C++ structures.

```
#include "ex.h"
#include "ncmul.h"
#include "numeric.h"
#include "operators.h"
#include "print.h"
#include <functional>
```

## Classes

- class [GiNaC::compare\\_all\\_equal< T >](#)  
*Comparison policy: all structures of one type are equal.*
- class [GiNaC::compare\\_std\\_less< T >](#)  
*Comparison policy: use std::equal\_to/std::less (defaults to operators == and <) to compare structures.*
- class [GiNaC::compare\\_bitwise< T >](#)  
*Comparison policy: use bit-wise comparison to compare structures.*
- class [GiNaC::structure< T, ComparisonPolicy >](#)  
*Wrapper template for making [GiNaC](#) classes out of C++ structures.*

## Namespaces

- namespace [GiNaC](#)

### 7.81.1 Detailed Description

Wrapper template for making [GiNaC](#) classes out of C++ structures.



## 7.82 symbol.cpp File Reference

Implementation of [GiNaC](#)'s symbolic objects.

```
#include "symbol.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include "inifcns.h"
#include <map>
#include <stdexcept>
#include <string>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (symbol, basic, print\_func< print\_context >(&symbol::do\_print). print\_func< print\_latex >(&symbol::do\_print\_latex). print\_func< print\_tree >(&symbol::do\_print\_tree). print\_func< print\_python\_repr >(&symbol::do\_print\_python\_repr)) symbol
- static const std::string & [GiNaC::get\\_default\\_TeX\\_name](#) (const std::string &name)  
*Return default TeX name for symbol.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (symbol)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (realsymbol)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (possymbol)

### 7.82.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic objects.

## 7.83 symbol.h File Reference

Interface to [GiNaC](#)'s symbolic objects.

```
#include "basic.h"
#include "ex.h"
#include "ptr.h"
#include "archive.h"
#include <string>
#include <typeinfo>
```

## Classes

- class [GiNaC::symbol](#)  
*Basic CAS symbol.*
- class [GiNaC::realsymbol](#)  
*Specialization of symbol to real domain.*
- class [GiNaC::possymbol](#)  
*Specialization of symbol to real positive domain.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (symbol)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (realsymbol)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (possymbol)

### 7.83.1 Detailed Description

Interface to [GiNaC](#)'s symbolic objects.

## 7.84 symmetry.cpp File Reference

Implementation of [GiNaC](#)'s symmetry definitions.

```
#include "symmetry.h"
#include "lst.h"
#include "add.h"
#include "numeric.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <functional>
#include <iostream>
#include <limits>
#include <stdexcept>
```

## Classes

- class [GiNaC::sy\\_is\\_less](#)
- class [GiNaC::sy\\_swap](#)

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (symmetry, basic, print\_func< print\_context >(&symmetry::do\_print). print\_func< print\_tree >(&symmetry::do\_print\_tree)) symmetry
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (symmetry)
- static const symmetry & [GiNaC::index0](#) ()
- static const symmetry & [GiNaC::index1](#) ()
- static const symmetry & [GiNaC::index2](#) ()
- static const symmetry & [GiNaC::index3](#) ()
- const symmetry & [GiNaC::not\\_symmetric](#) ()
- const symmetry & [GiNaC::symmetric2](#) ()
- const symmetry & [GiNaC::symmetric3](#) ()
- const symmetry & [GiNaC::symmetric4](#) ()
- const symmetry & [GiNaC::antisymmetric2](#) ()
- const symmetry & [GiNaC::antisymmetric3](#) ()
- const symmetry & [GiNaC::antisymmetric4](#) ()
- int [GiNaC::canonicalize](#) (exvector::iterator v, const symmetry &symm)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- static ex [GiNaC::symm](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator [last](#), bool asymmetric)
- ex [GiNaC::symmetrize](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Symmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::antisymmetrize](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*

### 7.84.1 Detailed Description

Implementation of [GiNaC](#)'s symmetry definitions.

## 7.85 symmetry.h File Reference

Interface to [GiNaC](#)'s symmetry definitions.

```
#include "ex.h"
#include "archive.h"
#include <set>
```

## Classes

- class [GiNaC::symmetry](#)  
*This class describes the symmetry of a group of indices.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (symmetry)
- symmetry [GiNaC::sy\\_none](#) ()
- symmetry [GiNaC::sy\\_none](#) (const symmetry &c1, const symmetry &c2)
- symmetry [GiNaC::sy\\_none](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3)
- symmetry [GiNaC::sy\\_none](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3, const symmetry &c4)
- symmetry [GiNaC::sy\\_symm](#) ()
- symmetry [GiNaC::sy\\_symm](#) (const symmetry &c1, const symmetry &c2)
- symmetry [GiNaC::sy\\_symm](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3)
- symmetry [GiNaC::sy\\_symm](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3, const symmetry &c4)
- symmetry [GiNaC::sy\\_anti](#) ()
- symmetry [GiNaC::sy\\_anti](#) (const symmetry &c1, const symmetry &c2)
- symmetry [GiNaC::sy\\_anti](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3)
- symmetry [GiNaC::sy\\_anti](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3, const symmetry &c4)
- symmetry [GiNaC::sy\\_cycl](#) ()
- symmetry [GiNaC::sy\\_cycl](#) (const symmetry &c1, const symmetry &c2)
- symmetry [GiNaC::sy\\_cycl](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3)
- symmetry [GiNaC::sy\\_cycl](#) (const symmetry &c1, const symmetry &c2, const symmetry &c3, const symmetry &c4)
- const symmetry & [GiNaC::not\\_symmetric](#) ()
- const symmetry & [GiNaC::symmetric2](#) ()
- const symmetry & [GiNaC::symmetric3](#) ()
- const symmetry & [GiNaC::symmetric4](#) ()
- const symmetry & [GiNaC::antisymmetric2](#) ()
- const symmetry & [GiNaC::antisymmetric3](#) ()
- const symmetry & [GiNaC::antisymmetric4](#) ()
- int [GiNaC::canonicalize](#) (exvector::iterator v, const symmetry &symm)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- ex [GiNaC::symmetrize](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator last)  
*Symmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::symmetrize](#) (const ex &e, const exvector &v)  
*Symmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::antisymmetrize](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator last)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::antisymmetrize](#) (const ex &e, const exvector &v)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &e, exvector::const\_iterator first, exvector::const\_iterator last)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- ex [GiNaC::symmetrize\\_cyclic](#) (const ex &e, const exvector &v)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*

### 7.85.1 Detailed Description

Interface to [GiNaC](#)'s symmetry definitions.

## 7.86 tensor.cpp File Reference

Implementation of [GiNaC](#)'s special tensors.

```
#include "tensor.h"
#include "idx.h"
#include "indexed.h"
#include "symmetry.h"
#include "relational.h"
#include "operators.h"
#include "lst.h"
#include "numeric.h"
#include "matrix.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
#include <stdexcept>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (tensdelta, tensor, print\_func< print\_dflt >(&tensdelta::do\_print). print\_func< print\_latex >(&tensdelta::do\_print\_latex)) [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)(tensmetric
- [GiNaC::print\\_func< print\\_dflt >](#) (&tensmetric::do\_print). print\_func< print\_latex >(&tensmetric
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (minkmetric)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (tensepsilon)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (tensdelta)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (tensmetric)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (spinmetric)
- ex [GiNaC::delta\\_tensor](#) (const ex &i1, const ex &i2)  
*Create a delta tensor with specified indices.*
- ex [GiNaC::metric\\_tensor](#) (const ex &i1, const ex &i2)  
*Create a symmetric metric tensor with specified indices.*
- ex [GiNaC::lorentz\\_g](#) (const ex &i1, const ex &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- ex [GiNaC::spinor\\_metric](#) (const ex &i1, const ex &i2)  
*Create a spinor metric tensor with specified indices.*
- ex [GiNaC::epsilon\\_tensor](#) (const ex &i1, const ex &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- ex [GiNaC::epsilon\\_tensor](#) (const ex &i1, const ex &i2, const ex &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- ex [GiNaC::lorentz\\_eps](#) (const ex &i1, const ex &i2, const ex &i3, const ex &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*

### 7.86.1 Detailed Description

Implementation of [GiNaC](#)'s special tensors.

## 7.87 tensor.h File Reference

Interface to [GiNaC](#)'s special tensors.

```
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::tensor](#)  
*This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.*
- class [GiNaC::tensdelta](#)  
*This class represents the delta tensor.*
- class [GiNaC::tensmetric](#)  
*This class represents a general metric tensor which can be used to raise/lower indices.*
- class [GiNaC::minkmetric](#)  
*This class represents a Minkowski metric tensor.*
- class [GiNaC::spinmetric](#)  
*This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.*
- class [GiNaC::tensepsilon](#)  
*This class represents the totally antisymmetric epsilon tensor.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (tensdelta)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (tensmetric)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (minkmetric)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (spinmetric)
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (tensepsilon)
- ex [GiNaC::delta\\_tensor](#) (const ex &i1, const ex &i2)  
*Create a delta tensor with specified indices.*
- ex [GiNaC::metric\\_tensor](#) (const ex &i1, const ex &i2)  
*Create a symmetric metric tensor with specified indices.*
- ex [GiNaC::lorentz\\_g](#) (const ex &i1, const ex &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- ex [GiNaC::spinor\\_metric](#) (const ex &i1, const ex &i2)  
*Create a spinor metric tensor with specified indices.*
- ex [GiNaC::epsilon\\_tensor](#) (const ex &i1, const ex &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- ex [GiNaC::epsilon\\_tensor](#) (const ex &i1, const ex &i2, const ex &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- ex [GiNaC::lorentz\\_eps](#) (const ex &i1, const ex &i2, const ex &i3, const ex &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*

### 7.87.1 Detailed Description

Interface to [GiNaC](#)'s special tensors.

## 7.88 utils.cpp File Reference

Implementation of several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

```
#include "ex.h"
#include "numeric.h"
#include "utils.h"
#include "version.h"
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- unsigned [GiNaC::log2](#) (unsigned n)  
*Integer binary logarithm.*
- const numeric [GiNaC::multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient  $n!/(p1!*p2!*...*pk!)$  where  $n = p1+p2+...+pk$ , i.e.*

### Variables

- const int [GiNaC::version\\_major](#) = GINACLIB\_MAJOR\_VERSION
- const int [GiNaC::version\\_minor](#) = GINACLIB\_MINOR\_VERSION
- const int [GiNaC::version\\_micro](#) = GINACLIB\_MICRO\_VERSION
- const numeric \* [GiNaC::\\_num\\_120\\_p](#)
- const ex [GiNaC::\\_ex\\_120](#) = ex(\*\_num\_120\_p)
- const numeric \* [GiNaC::\\_num\\_60\\_p](#)
- const ex [GiNaC::\\_ex\\_60](#) = ex(\*\_num\_60\_p)
- const numeric \* [GiNaC::\\_num\\_48\\_p](#)
- const ex [GiNaC::\\_ex\\_48](#) = ex(\*\_num\_48\_p)
- const numeric \* [GiNaC::\\_num\\_30\\_p](#)
- const ex [GiNaC::\\_ex\\_30](#) = ex(\*\_num\_30\_p)
- const numeric \* [GiNaC::\\_num\\_25\\_p](#)
- const ex [GiNaC::\\_ex\\_25](#) = ex(\*\_num\_25\_p)
- const numeric \* [GiNaC::\\_num\\_24\\_p](#)
- const ex [GiNaC::\\_ex\\_24](#) = ex(\*\_num\_24\_p)
- const numeric \* [GiNaC::\\_num\\_20\\_p](#)
- const ex [GiNaC::\\_ex\\_20](#) = ex(\*\_num\_20\_p)
- const numeric \* [GiNaC::\\_num\\_18\\_p](#)
- const ex [GiNaC::\\_ex\\_18](#) = ex(\*\_num\_18\_p)
- const numeric \* [GiNaC::\\_num\\_15\\_p](#)
- const ex [GiNaC::\\_ex\\_15](#) = ex(\*\_num\_15\_p)

- const numeric \* [GiNaC::\\_num\\_12\\_p](#)
- const ex [GiNaC::\\_ex\\_12](#) = ex(\*\_num\_12\_p)
- const numeric \* [GiNaC::\\_num\\_11\\_p](#)
- const ex [GiNaC::\\_ex\\_11](#) = ex(\*\_num\_11\_p)
- const numeric \* [GiNaC::\\_num\\_10\\_p](#)
- const ex [GiNaC::\\_ex\\_10](#) = ex(\*\_num\_10\_p)
- const numeric \* [GiNaC::\\_num\\_9\\_p](#)
- const ex [GiNaC::\\_ex\\_9](#) = ex(\*\_num\_9\_p)
- const numeric \* [GiNaC::\\_num\\_8\\_p](#)
- const ex [GiNaC::\\_ex\\_8](#) = ex(\*\_num\_8\_p)
- const numeric \* [GiNaC::\\_num\\_7\\_p](#)
- const ex [GiNaC::\\_ex\\_7](#) = ex(\*\_num\_7\_p)
- const numeric \* [GiNaC::\\_num\\_6\\_p](#)
- const ex [GiNaC::\\_ex\\_6](#) = ex(\*\_num\_6\_p)
- const numeric \* [GiNaC::\\_num\\_5\\_p](#)
- const ex [GiNaC::\\_ex\\_5](#) = ex(\*\_num\_5\_p)
- const numeric \* [GiNaC::\\_num\\_4\\_p](#)
- const ex [GiNaC::\\_ex\\_4](#) = ex(\*\_num\_4\_p)
- const numeric \* [GiNaC::\\_num\\_3\\_p](#)
- const ex [GiNaC::\\_ex\\_3](#) = ex(\*\_num\_3\_p)
- const numeric \* [GiNaC::\\_num\\_2\\_p](#)
- const ex [GiNaC::\\_ex\\_2](#) = ex(\*\_num\_2\_p)
- const numeric \* [GiNaC::\\_num\\_1\\_p](#)
- const ex [GiNaC::\\_ex\\_1](#) = ex(\*\_num\_1\_p)
- const numeric \* [GiNaC::\\_num\\_1\\_2\\_p](#)
- const ex [GiNaC::\\_ex\\_1\\_2](#) = ex(\*\_num\_1\_2\_p)
- const numeric \* [GiNaC::\\_num\\_1\\_3\\_p](#)
- const ex [GiNaC::\\_ex\\_1\\_3](#) = ex(\*\_num\_1\_3\_p)
- const numeric \* [GiNaC::\\_num\\_1\\_4\\_p](#)
- const ex [GiNaC::\\_ex\\_1\\_4](#) = ex(\*\_num\_1\_4\_p)
- const numeric \* [GiNaC::\\_num0\\_p](#)
- const ex [GiNaC::\\_ex0](#) = ex(\*\_num0\_p)
- const numeric \* [GiNaC::\\_num1\\_4\\_p](#)
- const ex [GiNaC::\\_ex1\\_4](#) = ex(\*\_num1\_4\_p)
- const numeric \* [GiNaC::\\_num1\\_3\\_p](#)
- const ex [GiNaC::\\_ex1\\_3](#) = ex(\*\_num1\_3\_p)
- const numeric \* [GiNaC::\\_num1\\_2\\_p](#)
- const ex [GiNaC::\\_ex1\\_2](#) = ex(\*\_num1\_2\_p)
- const numeric \* [GiNaC::\\_num1\\_p](#)
- const ex [GiNaC::\\_ex1](#) = ex(\*\_num1\_p)
- const numeric \* [GiNaC::\\_num2\\_p](#)
- const ex [GiNaC::\\_ex2](#) = ex(\*\_num2\_p)
- const numeric \* [GiNaC::\\_num3\\_p](#)
- const ex [GiNaC::\\_ex3](#) = ex(\*\_num3\_p)
- const numeric \* [GiNaC::\\_num4\\_p](#)
- const ex [GiNaC::\\_ex4](#) = ex(\*\_num4\_p)
- const numeric \* [GiNaC::\\_num5\\_p](#)
- const ex [GiNaC::\\_ex5](#) = ex(\*\_num5\_p)
- const numeric \* [GiNaC::\\_num6\\_p](#)
- const ex [GiNaC::\\_ex6](#) = ex(\*\_num6\_p)
- const numeric \* [GiNaC::\\_num7\\_p](#)
- const ex [GiNaC::\\_ex7](#) = ex(\*\_num7\_p)
- const numeric \* [GiNaC::\\_num8\\_p](#)
- const ex [GiNaC::\\_ex8](#) = ex(\*\_num8\_p)
- const numeric \* [GiNaC::\\_num9\\_p](#)



- const ex [GiNaC::\\_ex9](#) = ex(\*\_num9\_p)
- const numeric \* [GiNaC::\\_num10\\_p](#)
- const ex [GiNaC::\\_ex10](#) = ex(\*\_num10\_p)
- const numeric \* [GiNaC::\\_num11\\_p](#)
- const ex [GiNaC::\\_ex11](#) = ex(\*\_num11\_p)
- const numeric \* [GiNaC::\\_num12\\_p](#)
- const ex [GiNaC::\\_ex12](#) = ex(\*\_num12\_p)
- const numeric \* [GiNaC::\\_num15\\_p](#)
- const ex [GiNaC::\\_ex15](#) = ex(\*\_num15\_p)
- const numeric \* [GiNaC::\\_num18\\_p](#)
- const ex [GiNaC::\\_ex18](#) = ex(\*\_num18\_p)
- const numeric \* [GiNaC::\\_num20\\_p](#)
- const ex [GiNaC::\\_ex20](#) = ex(\*\_num20\_p)
- const numeric \* [GiNaC::\\_num24\\_p](#)
- const ex [GiNaC::\\_ex24](#) = ex(\*\_num24\_p)
- const numeric \* [GiNaC::\\_num25\\_p](#)
- const ex [GiNaC::\\_ex25](#) = ex(\*\_num25\_p)
- const numeric \* [GiNaC::\\_num30\\_p](#)
- const ex [GiNaC::\\_ex30](#) = ex(\*\_num30\_p)
- const numeric \* [GiNaC::\\_num48\\_p](#)
- const ex [GiNaC::\\_ex48](#) = ex(\*\_num48\_p)
- const numeric \* [GiNaC::\\_num60\\_p](#)
- const ex [GiNaC::\\_ex60](#) = ex(\*\_num60\_p)
- const numeric \* [GiNaC::\\_num120\\_p](#)
- const ex [GiNaC::\\_ex120](#) = ex(\*\_num120\_p)

### 7.88.1 Detailed Description

Implementation of several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

## 7.89 utils.h File Reference

Interface to several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

```
#include "assertion.h"
#include <functional>
#include <cstdlib>
#include <string>
```

## Classes

- class [GiNaC::dunno](#)  
*Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold()*
- class [GiNaC::basic\\_partition\\_generator](#)  
*Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.*
- struct [GiNaC::basic\\_partition\\_generator::mpartition2](#)
- class [GiNaC::partition\\_with\\_zero\\_parts\\_generator](#)  
*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.*
- class [GiNaC::partition\\_generator](#)  
*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.*
- class [GiNaC::composition\\_generator](#)  
*Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.*
- struct [GiNaC::composition\\_generator::coolmulti](#)
- struct [GiNaC::composition\\_generator::coolmulti::element](#)

## Namespaces

- namespace [GiNaC](#)

## Macros

- `#define DEFAULT\_CTOR(classname) classname::classname() { setflag(status_flags::evaluated | status_↔ flags::expanded); }`
- `#define DEFAULT\_COMPARE(classname)`
- `#define DEFAULT\_PRINT(classname, text)`
- `#define DEFAULT\_PRINT\_LATEX(classname, text, latex)`

## Functions

- unsigned [GiNaC::log2](#) (unsigned  $n$ )  
*Integer binary logarithm.*
- unsigned [GiNaC::rotate\\_left](#) (unsigned  $n$ )  
*Rotate bits of unsigned value by one bit to the left.*
- `template<class T >`  
int [GiNaC::compare\\_pointers](#) (const T \*a, const T \*b)  
*Compare two pointers (just to establish some sort of canonical order).*
- unsigned [GiNaC::golden\\_ratio\\_hash](#) (uintptr\_t  $n$ )  
*Truncated multiplication with golden ratio, for computing hash values.*
- `template<class It >`  
int [GiNaC::permutation\\_sign](#) (It first, It last)
- `template<class It , class Cmp , class Swap >`  
int [GiNaC::permutation\\_sign](#) (It first, It last, Cmp comp, Swap swapit)
- `template<class It , class Cmp , class Swap >`  
void [GiNaC::shaker\\_sort](#) (It first, It last, Cmp comp, Swap swapit)
- `template<class It , class Swap >`  
void [GiNaC::cyclic\\_permutation](#) (It first, It last, It new\_first, Swap swapit)
- const numeric [GiNaC::multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient  $n!/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.*

## 7.89.1 Detailed Description

Interface to several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

## 7.89.2 Macro Definition Documentation

### 7.89.2.1 DEFAULT\_CTOR

```
#define DEFAULT_CTOR(  
    classname ) classname::classname() { setflag(status_flags::evaluated | status_↔  
flags::expanded); }
```

### 7.89.2.2 DEFAULT\_COMPARE

```
#define DEFAULT_COMPARE(  
    classname )
```

#### Value:

```
int classname::compare_same_type(const basic & other) const \  
{ \  
    /* by default, the objects are always identical */ \  
    return 0; \  
}
```

### 7.89.2.3 DEFAULT\_PRINT

```
#define DEFAULT_PRINT(  
    classname,  
    text )
```

#### Value:

```
void classname::do_print(const print_context & c, unsigned level) const \  
{ \  
    c.s « text; \  
}
```

### 7.89.2.4 DEFAULT\_PRINT\_LATEX

```
#define DEFAULT_PRINT_LATEX(  
    classname,  
    text,  
    latex )
```

#### Value:

```
DEFAULT_PRINT(classname, text) \  
void classname::do_print_latex(const print_latex & c, unsigned level) const \  
{ \  
    c.s « latex; \  
}
```

## 7.90 utils\_multi\_iterator.h File Reference

Utilities for summing over multiple indices.

```
#include <cstdint>
#include <vector>
#include <ostream>
#include <iterator>
```

### Classes

- class [GiNaC::has\\_distance< T >](#)  
*SFINAE test for distance.*
- class [GiNaC::basic\\_multi\\_iterator< T >](#)  
*basic\_multi\_iterator is a base class.*
- class [GiNaC::multi\\_iterator\\_ordered< T >](#)  
*The class multi\_iterator\_ordered defines a multi\_iterator ( $i_1, i_2, \dots, i_k$ ), such that.*
- class [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#)  
*The class multi\_iterator\_ordered\_eq defines a multi\_iterator ( $i_1, i_2, \dots, i_k$ ), such that.*
- class [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >](#)  
*The class multi\_iterator\_ordered\_eq\_indv defines a multi\_iterator ( $i_1, i_2, \dots, i_k$ ), such that.*
- class [GiNaC::multi\\_iterator\\_counter< T >](#)  
*The class multi\_iterator\_counter defines a multi\_iterator ( $i_1, i_2, \dots, i_k$ ), such that.*
- class [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#)  
*The class multi\_iterator\_counter\_indv defines a multi\_iterator ( $i_1, i_2, \dots, i_k$ ), such that.*
- class [GiNaC::multi\\_iterator\\_permutation< T >](#)  
*The class multi\_iterator\_permutation defines a multi\_iterator ( $i_1, i_2, \dots, i_k$ ), for which.*
- class [GiNaC::multi\\_iterator\\_shuffle< T >](#)  
*The class multi\_iterator\_shuffle defines a multi\_iterator, which runs over all shuffles of a and b.*
- class [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#)  
*The class multi\_iterator\_shuffle\_prime defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).*

### Namespaces

- namespace [GiNaC](#)

### Functions

- `template<typename T >`  
`std::enable_if< has_distance< T >::value, typename std::iterator_traits< T >::difference_type >::type`  
`GiNaC::format\_index\_value (const T &a, const T &b)`  
*For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.*
- `template<typename T >`  
`std::enable_if<!has_distance< T >::value, T >::type GiNaC::format\_index\_value (const T &a, const T &b)`  
*For all other cases we simply print the value.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const basic_multi_iterator< T > &v)`

*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_ordered< T > &v)`

*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_ordered_eq< T > &v)`

*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_ordered_eq_indv< T > &v)`

*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_counter< T > &v)`

*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_counter_indv< T > &v)`

*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_permutation< T > &v)`

*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_shuffle< T > &v)`

*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_shuffle_prime< T > &v)`

*Output operator.*

### 7.90.1 Detailed Description

Utilities for summing over multiple indices.

## 7.91 version.h File Reference

[GiNaC](#) library version information.

### Namespaces

- namespace [GiNaC](#)

### Macros

- `#define GINACLIB\_MAJOR\_VERSION 1`
- `#define GINACLIB\_MINOR\_VERSION 8`
- `#define GINACLIB\_MICRO\_VERSION 6`
- `#define GINAC\_LT\_CURRENT 12`
- `#define GINAC\_LT\_REVISION 5`
- `#define GINAC\_LT\_AGE 1`
- `#define GINACLIB\_ARCHIVE\_VERSION 3`
- `#define GINACLIB\_ARCHIVE\_AGE 3`
- `#define GINACLIB\_STR\_HELPER(x) #x`
- `#define GINACLIB\_STR(x) GINACLIB\_STR\_HELPER(x)`
- `#define GINACLIB\_VERSION`

### 7.91.1 Detailed Description

[GiNaC](#) library version information.

## 7.91.2 Macro Definition Documentation

### 7.91.2.1 GINACLIB\_MAJOR\_VERSION

```
#define GINACLIB_MAJOR_VERSION 1
```

### 7.91.2.2 GINACLIB\_MINOR\_VERSION

```
#define GINACLIB_MINOR_VERSION 8
```

### 7.91.2.3 GINACLIB\_MICRO\_VERSION

```
#define GINACLIB_MICRO_VERSION 6
```

### 7.91.2.4 GINAC\_LT\_CURRENT

```
#define GINAC_LT_CURRENT 12
```

### 7.91.2.5 GINAC\_LT\_REVISION

```
#define GINAC_LT_REVISION 5
```

### 7.91.2.6 GINAC\_LT\_AGE

```
#define GINAC_LT_AGE 1
```

### 7.91.2.7 GINACLIB\_ARCHIVE\_VERSION

```
#define GINACLIB_ARCHIVE_VERSION 3
```

### 7.91.2.8 GINACLIB\_ARCHIVE\_AGE

```
#define GINACLIB_ARCHIVE_AGE 3
```

### 7.91.2.9 GINACLIB\_STR\_HELPER

```
#define GINACLIB_STR_HELPER(  
    x ) #x
```

### 7.91.2.10 GINACLIB\_STR

```
#define GINACLIB_STR(  
    x ) GINACLIB_STR_HELPER(x)
```

### 7.91.2.11 GINACLIB\_VERSION

```
#define GINACLIB_VERSION
```

#### Value:

```
GINACLIB_STR(GINACLIB_MAJOR_VERSION) "." \\  
GINACLIB_STR(GINACLIB_MINOR_VERSION) "." \\  
GINACLIB_STR(GINACLIB_MICRO_VERSION)
```

## 7.92 wildcard.cpp File Reference

Implementation of [GiNaC](#)'s wildcard objects.

```
#include "wildcard.h"  
#include "archive.h"  
#include "utils.h"  
#include "hash_seed.h"  
#include <iostream>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (wildcard, basic, print\_func< print\_context >(&wildcard::do\_print). print\_func< print\_tree >(&wildcard::do\_print\_tree). print\_func< print\_python\_repr >(&wildcard::do\_print\_python\_repr)) wildcard
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (wildcard)
- bool [GiNaC::haswild](#) (const ex &x)

*Check whether x has a wildcard anywhere as a subexpression.*

### 7.92.1 Detailed Description

Implementation of [GiNaC](#)'s wildcard objects.

## 7.93 wildcard.h File Reference

Interface to [GiNaC](#)'s wildcard objects.

```
#include "ex.h"
#include "archive.h"
```

## Classes

- class [GiNaC::wildcard](#)  
*This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (wildcard)
- ex [GiNaC::wild](#) (unsigned label=0)  
*Create a wildcard object with the specified label.*
- bool [GiNaC::haswild](#) (const ex &x)  
*Check whether x has a wildcard anywhere as a subexpression.*

### 7.93.1 Detailed Description

Interface to [GiNaC](#)'s wildcard objects.



# Index

[\\_ex0](#)  
    [GiNaC, 300](#)

[\\_ex1](#)  
    [GiNaC, 302](#)

[\\_ex10](#)  
    [GiNaC, 306](#)

[\\_ex11](#)  
    [GiNaC, 306](#)

[\\_ex12](#)  
    [GiNaC, 307](#)

[\\_ex120](#)  
    [GiNaC, 310](#)

[\\_ex15](#)  
    [GiNaC, 307](#)

[\\_ex18](#)  
    [GiNaC, 307](#)

[\\_ex1\\_2](#)  
    [GiNaC, 302](#)

[\\_ex1\\_3](#)  
    [GiNaC, 301](#)

[\\_ex1\\_4](#)  
    [GiNaC, 301](#)

[\\_ex2](#)  
    [GiNaC, 303](#)

[\\_ex20](#)  
    [GiNaC, 308](#)

[\\_ex24](#)  
    [GiNaC, 308](#)

[\\_ex25](#)  
    [GiNaC, 308](#)

[\\_ex3](#)  
    [GiNaC, 304](#)

[\\_ex30](#)  
    [GiNaC, 309](#)

[\\_ex4](#)  
    [GiNaC, 304](#)

[\\_ex48](#)  
    [GiNaC, 309](#)

[\\_ex5](#)  
    [GiNaC, 304](#)

[\\_ex6](#)  
    [GiNaC, 305](#)

[\\_ex60](#)  
    [GiNaC, 309](#)

[\\_ex7](#)  
    [GiNaC, 305](#)

[\\_ex8](#)  
    [GiNaC, 305](#)

[\\_ex9](#)  
    [GiNaC, 306](#)

[\\_ex\\_1](#)  
    [GiNaC, 299](#)

[\\_ex\\_10](#)  
    [GiNaC, 296](#)

[\\_ex\\_11](#)  
    [GiNaC, 295](#)

[\\_ex\\_12](#)  
    [GiNaC, 295](#)

[\\_ex\\_120](#)  
    [GiNaC, 292](#)

[\\_ex\\_15](#)  
    [GiNaC, 295](#)

[\\_ex\\_18](#)  
    [GiNaC, 294](#)

[\\_ex\\_1\\_2](#)  
    [GiNaC, 299](#)

[\\_ex\\_1\\_3](#)  
    [GiNaC, 300](#)

[\\_ex\\_1\\_4](#)  
    [GiNaC, 300](#)

[\\_ex\\_2](#)  
    [GiNaC, 298](#)

[\\_ex\\_20](#)  
    [GiNaC, 294](#)

[\\_ex\\_24](#)  
    [GiNaC, 294](#)

[\\_ex\\_25](#)  
    [GiNaC, 293](#)

[\\_ex\\_3](#)  
    [GiNaC, 298](#)

[\\_ex\\_30](#)  
    [GiNaC, 293](#)

[\\_ex\\_4](#)  
    [GiNaC, 298](#)

[\\_ex\\_48](#)  
    [GiNaC, 293](#)

[\\_ex\\_5](#)  
    [GiNaC, 297](#)

[\\_ex\\_6](#)  
    [GiNaC, 297](#)

[\\_ex\\_60](#)  
    [GiNaC, 292](#)

[\\_ex\\_7](#)  
    [GiNaC, 297](#)

[\\_ex\\_8](#)  
    [GiNaC, 296](#)

[\\_ex\\_9](#)  
    [GiNaC, 296](#)

\_iter\_rep  
     GiNaC::internal::\_iter\_rep, 312  
 \_num0\_bp  
     GiNaC, 290  
 \_num0\_p  
     GiNaC, 300  
 \_num10\_p  
     GiNaC, 306  
 \_num11\_p  
     GiNaC, 306  
 \_num120\_p  
     GiNaC, 309  
 \_num12\_p  
     GiNaC, 306  
 \_num15\_p  
     GiNaC, 307  
 \_num18\_p  
     GiNaC, 307  
 \_num1\_2\_p  
     GiNaC, 302  
 \_num1\_3\_p  
     GiNaC, 301  
 \_num1\_4\_p  
     GiNaC, 301  
 \_num1\_p  
     GiNaC, 302  
 \_num20\_p  
     GiNaC, 307  
 \_num24\_p  
     GiNaC, 308  
 \_num25\_p  
     GiNaC, 308  
 \_num2\_p  
     GiNaC, 303  
 \_num30\_p  
     GiNaC, 308  
 \_num3\_p  
     GiNaC, 303  
 \_num48\_p  
     GiNaC, 309  
 \_num4\_p  
     GiNaC, 304  
 \_num5\_p  
     GiNaC, 304  
 \_num60\_p  
     GiNaC, 309  
 \_num6\_p  
     GiNaC, 304  
 \_num7\_p  
     GiNaC, 305  
 \_num8\_p  
     GiNaC, 305  
 \_num9\_p  
     GiNaC, 305  
 \_num\_10\_p  
     GiNaC, 296  
 \_num\_11\_p  
     GiNaC, 295  
 \_num\_120\_p  
     GiNaC, 292  
 \_num\_12\_p  
     GiNaC, 295  
 \_num\_15\_p  
     GiNaC, 295  
 \_num\_18\_p  
     GiNaC, 294  
 \_num\_1\_2\_p  
     GiNaC, 299  
 \_num\_1\_3\_p  
     GiNaC, 299  
 \_num\_1\_4\_p  
     GiNaC, 300  
 \_num\_1\_p  
     GiNaC, 299  
 \_num\_20\_p  
     GiNaC, 294  
 \_num\_24\_p  
     GiNaC, 294  
 \_num\_25\_p  
     GiNaC, 293  
 \_num\_2\_p  
     GiNaC, 298  
 \_num\_30\_p  
     GiNaC, 293  
 \_num\_3\_p  
     GiNaC, 298  
 \_num\_48\_p  
     GiNaC, 293  
 \_num\_4\_p  
     GiNaC, 298  
 \_num\_5\_p  
     GiNaC, 297  
 \_num\_60\_p  
     GiNaC, 292  
 \_num\_6\_p  
     GiNaC, 297  
 \_num\_7\_p  
     GiNaC, 297  
 \_num\_8\_p  
     GiNaC, 296  
 \_num\_9\_p  
     GiNaC, 296  
 \_numeric\_digits  
     GiNaC::\_numeric\_digits, 314  
 ~archive  
     GiNaC::archive, 336  
 ~basic  
     GiNaC::basic, 360  
 ~basic\_multi\_iterator  
     GiNaC::basic\_multi\_iterator< T >, 392  
 ~compare\_all\_equal  
     GiNaC::compare\_all\_equal< T >, 436  
 ~compare\_bitwise  
     GiNaC::compare\_bitwise< T >, 437  
 ~compare\_std\_less  
     GiNaC::compare\_std\_less< T >, 438

- ~container\_storage
  - GiNaC::container\_storage< C >, 487
- ~coolmulti
  - GiNaC::composition\_generator::coolmulti, 489
- ~element
  - GiNaC::composition\_generator::coolmulti::element, 553
- ~function\_options
  - GiNaC::function\_options, 687
- ~library\_init
  - GiNaC::library\_init, 816
- ~map\_function
  - GiNaC::map\_function, 821
- ~print\_context
  - GiNaC::print\_context, 1033
- ~print\_functor\_impl
  - GiNaC::print\_functor\_impl, 1048
- ~ptr
  - GiNaC::ptr< T >, 1089
- ~unarchive\_table\_t
  - GiNaC::unarchive\_table\_t, 1266
- ~visitor
  - GiNaC::visitor, 1287
- a
  - GiNaC::archive\_node, 351
  - GiNaC::Eisenstein\_kernel, 551
  - GiNaC::integral, 782
- abs
  - GiNaC, 241
- abs\_conjugate
  - GiNaC, 143
- abs\_eval
  - GiNaC, 142
- abs\_evalf
  - GiNaC, 142
- abs\_expand
  - GiNaC, 142
- abs\_expl\_derivative
  - GiNaC, 142
- abs\_imag\_part
  - GiNaC, 143
- abs\_info
  - GiNaC, 144
- abs\_power
  - GiNaC, 144
- abs\_print\_csrc\_float
  - GiNaC, 143
- abs\_print\_latex
  - GiNaC, 143
- abs\_real\_part
  - GiNaC, 143
- accept
  - GiNaC::basic, 367
  - GiNaC::ex, 585
- access\_counter
  - GiNaC::remember\_table\_entry, 1127
- acos
  - GiNaC, 232
- acos\_conjugate
  - GiNaC, 185
- acos\_deriv
  - GiNaC, 185
- acos\_eval
  - GiNaC, 184
- acos\_evalf
  - GiNaC, 184
- acosh
  - GiNaC, 235
- acosh\_conjugate
  - GiNaC, 194
- acosh\_deriv
  - GiNaC, 193
- acosh\_eval
  - GiNaC, 193
- acosh\_evalf
  - GiNaC, 193
- adaptivesimpson
  - GiNaC, 196
- add
  - GiNaC::add, 323, 324
  - GiNaC::matrix, 835
  - GiNaC::mul, 888
  - GiNaC::numeric, 961
  - GiNaC::scalar\_products, 1133
  - GiNaC::symmetry, 1229
- add.cpp, 1297
- add.h, 1298
- add\_bool
  - GiNaC::archive\_node, 345
- add\_callback
  - GiNaC::\_numeric\_digits, 315
- add\_child
  - GiNaC::class\_info< OPT >::tree\_node, 1265
- add\_dyn
  - GiNaC::numeric, 963
- add\_entry
  - GiNaC::remember\_table, 1122
  - GiNaC::remember\_table\_list, 1128
- add\_ex
  - GiNaC::archive\_node, 346
- add\_indexed
  - GiNaC::basic, 373
  - GiNaC::matrix, 832
  - GiNaC::structure< T, ComparisonPolicy >, 1174
- add\_node
  - GiNaC::archive, 338
- add\_reference
  - GiNaC::refcounted, 1101
- add\_series
  - GiNaC::pseries, 1081
- add\_string
  - GiNaC::archive\_node, 346
- add\_symbol
  - GiNaC, 212
- add\_unsigned
  - GiNaC::archive\_node, 346

- add\_vectors
  - GiNaC::scalar\_products, [1133](#)
- after\_i
  - GiNaC::composition\_generator::coolmulti, [490](#)
- algebraic
  - GiNaC::has\_options, [732](#)
  - GiNaC::subs\_options, [1201](#)
- algebraic\_match\_mul\_with\_mul
  - GiNaC, [210](#)
- algebraic\_subs\_mul
  - GiNaC::mul, [886](#)
- all
  - GiNaC::factor\_options, [638](#)
- all\_index\_values\_are
  - GiNaC::indexed, [765](#)
- antisymmetric
  - GiNaC::symmetry, [1227](#)
- antisymmetric2
  - GiNaC, [271](#)
- antisymmetric3
  - GiNaC, [271](#)
- antisymmetric4
  - GiNaC, [271](#)
- antisymmetrize
  - GiNaC, [119](#), [272](#), [276](#)
  - GiNaC::ex, [598](#), [599](#)
- append
  - GiNaC::container< C >, [481](#)
- append\_factors
  - GiNaC::ncmul, [942](#)
- archive
  - GiNaC::archive, [336](#)
  - GiNaC::basic, [377](#)
  - GiNaC::clifford, [411](#)
  - GiNaC::color, [432](#)
  - GiNaC::constant, [463](#)
  - GiNaC::container< C >, [478](#)
  - GiNaC::expairseq, [623](#)
  - GiNaC::fderivative, [654](#)
  - GiNaC::function, [675](#)
  - GiNaC::idx, [740](#)
  - GiNaC::indexed, [763](#)
  - GiNaC::integral, [780](#)
  - GiNaC::matrix, [834](#)
  - GiNaC::minkmetric, [853](#)
  - GiNaC::numeric, [960](#)
  - GiNaC::power, [1027](#)
  - GiNaC::pseries, [1078](#)
  - GiNaC::relational, [1114](#)
  - GiNaC::spinidx, [1145](#)
  - GiNaC::symbol, [1216](#)
  - GiNaC::symmetry, [1228](#)
  - GiNaC::tensepsilon, [1248](#)
  - GiNaC::varidx, [1284](#)
  - GiNaC::wildcard, [1292](#)
- archive.cpp, [1298](#)
- archive.h, [1299](#)
  - GINAC\_BIND\_UNARCHIVER, [1301](#)
  - GINAC\_DECLARE\_UNARCHIVER, [1300](#)
- archive\_atom
  - GiNaC, [58](#)
- archive\_ex
  - GiNaC::archive, [336](#)
- archive\_node
  - GiNaC::archive\_node, [345](#)
  - GiNaC::ex, [603](#)
- archive\_node\_cit
  - GiNaC::archive\_node, [344](#)
- archive\_node\_id
  - GiNaC, [58](#)
- archived\_ex
  - GiNaC::archive::archived\_ex, [354](#)
- are\_ex\_trivially\_equal
  - GiNaC, [111](#)
  - GiNaC::ex, [603](#)
- arg1
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [987](#)
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [990](#)
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [992](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [998](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [1001](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [1004](#)
- arg2
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [990](#)
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [993](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [1001](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [1004](#)
- arg3
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [993](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [1004](#)
- argument\_type
  - GiNaC::map\_function, [821](#)
- asin
  - GiNaC, [232](#)
- asin\_conjugate
  - GiNaC, [184](#)
- asin\_deriv
  - GiNaC, [183](#)
- asin\_eval
  - GiNaC, [183](#)
- asin\_evalf
  - GiNaC, [183](#)
- asin\_info
  - GiNaC, [184](#)

- asinh
  - GiNaC, [235](#)
- asinh\_conjugate
  - GiNaC, [193](#)
- asinh\_deriv
  - GiNaC, [192](#)
- asinh\_eval
  - GiNaC, [192](#)
- asinh\_evalf
  - GiNaC, [192](#)
- assertion.h, [1302](#)
  - GINAC\_ASSERT, [1302](#)
- atan
  - GiNaC, [233](#)
- atan2\_deriv
  - GiNaC, [187](#)
- atan2\_eval
  - GiNaC, [187](#)
- atan2\_evalf
  - GiNaC, [187](#)
- atan2\_info
  - GiNaC, [187](#)
- atan\_conjugate
  - GiNaC, [186](#)
- atan\_deriv
  - GiNaC, [186](#)
- atan\_eval
  - GiNaC, [185](#)
- atan\_evalf
  - GiNaC, [185](#)
- atan\_info
  - GiNaC, [186](#)
- atan\_series
  - GiNaC, [186](#)
- atanh
  - GiNaC, [235](#)
- atanh\_conjugate
  - GiNaC, [195](#)
- atanh\_deriv
  - GiNaC, [194](#)
- atanh\_eval
  - GiNaC, [194](#)
- atanh\_evalf
  - GiNaC, [194](#)
- atanh\_series
  - GiNaC, [195](#)
- atend
  - GiNaC::composition\_generator, [441](#)
- atomize
  - GiNaC::archive, [339](#)
- atoms
  - GiNaC::archive, [341](#)
- attribute\_deprecated
  - compiler.h, [1313](#)
- automatic
  - GiNaC::determinant\_algo, [494](#)
  - GiNaC::solve\_algo, [1136](#)
- B
  - GiNaC::basic\_multi\_iterator< T >, [394](#)
- b
  - GiNaC::Eisenstein\_kernel, [551](#)
  - GiNaC::integral, [783](#)
- bareiss
  - GiNaC::determinant\_algo, [494](#)
  - GiNaC::solve\_algo, [1136](#)
- base\_and\_index
  - GiNaC, [94](#)
- basic
  - GiNaC::basic, [360](#)
- basic.cpp, [1302](#)
- basic.h, [1303](#)
- basic\_multi\_iterator
  - GiNaC::basic\_multi\_iterator< T >, [391](#)
- basic\_partition\_generator
  - GiNaC::basic\_partition\_generator, [396](#)
- basis
  - GiNaC::power, [1032](#)
- begin
  - GiNaC::archive\_node::archive\_node\_cit\_range, [353](#)
  - GiNaC::container< C >, [482](#)
  - GiNaC::ex, [576](#)
- bernoulli
  - GiNaC, [240](#)
- Bernoulli\_polynomial
  - GiNaC, [198](#)
- beta\_deriv
  - GiNaC, [165](#)
- beta\_eval
  - GiNaC, [165](#)
- beta\_evalf
  - GiNaC, [165](#)
- beta\_series
  - GiNaC, [165](#)
- binomial
  - GiNaC, [240](#)
- binomial\_conjugate
  - GiNaC, [153](#)
- binomial\_eval
  - GiNaC, [153](#)
- binomial\_evalf
  - GiNaC, [152](#)
- binomial\_imag\_part
  - GiNaC, [153](#)
- binomial\_real\_part
  - GiNaC, [153](#)
- binomial\_sym
  - GiNaC, [152](#)
- bp
  - GiNaC::ex, [604](#)
- c
  - factor.cpp, [1326](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function< C >, [995](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [998](#)

- GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, 1000
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, 1003
- C\_norm
  - GiNaC::Eisenstein\_h\_kernel, 541
  - GiNaC::Eisenstein\_kernel, 552
  - GiNaC::Kronecker\_dtau\_kernel, 803
  - GiNaC::Kronecker\_dz\_kernel, 812
  - GiNaC::modular\_form\_kernel, 864
- cache
  - factor.cpp, 1328
- cache\_step\_size
  - GiNaC::integration\_kernel, 792
- cache\_vec
  - integration\_kernel.cpp, 1371
- calc\_lanczos\_A
  - GiNaC::lanczos\_coeffs, 814
- calchash
  - GiNaC::basic, 376
  - GiNaC::constant, 464
  - GiNaC::expairseq, 625
  - GiNaC::function, 673
  - GiNaC::idx, 741
  - GiNaC::numeric, 961
  - GiNaC::relational, 1116
  - GiNaC::structure< T, ComparisonPolicy >, 1177
  - GiNaC::symbol, 1217
  - GiNaC::symmetry, 1228
  - GiNaC::wildcard, 1293
- callback\_registered
  - GiNaC, 89
- callbacklist
  - GiNaC::\_numeric\_digits, 315
- can\_be\_further\_expanded
  - GiNaC::mul, 887
- can\_make\_flat
  - GiNaC::expairseq, 629
  - GiNaC::mul, 885
- canonical
  - GiNaC::relational, 1114
- canonicalize
  - GiNaC, 271
  - GiNaC::expairseq, 631
  - GiNaC::symmetry, 1231
- canonicalize\_clifford
  - GiNaC, 99
- Catalan
  - GiNaC, 289
- CatalanEvalf
  - GiNaC, 246
- charpoly
  - GiNaC, 208
  - GiNaC::matrix, 839
- children
  - GiNaC::class\_info< OPT >::tree\_node, 1265
  - GiNaC::symmetry, 1232
- cinteger
  - GiNaC::info\_flags, 770
  - cinteger\_polynomial
    - GiNaC::info\_flags, 770
  - class\_info
    - GiNaC::class\_info< OPT >, 398
  - class\_info.h, 1304
  - clear
    - GiNaC::archive, 338
    - GiNaC::scalar\_products, 1133
  - clear\_all\_entries
    - GiNaC::remember\_table, 1122
  - clearflag
    - GiNaC::basic, 381
  - clifford
    - GiNaC::clifford, 410, 411
  - clifford.cpp, 1305
  - clifford.h, 1307
  - clifford\_bar
    - GiNaC, 104
  - clifford\_inverse
    - GiNaC, 100
  - clifford\_max\_label
    - GiNaC, 100
  - clifford\_moebius\_map
    - GiNaC, 102
  - clifford\_norm
    - GiNaC, 100
  - clifford\_prime
    - GiNaC, 99
  - clifford\_star
    - GiNaC, 104
  - clifford\_star\_bar
    - GiNaC, 99
  - clifford\_to\_lst
    - GiNaC, 101
  - clifford\_unit
    - GiNaC, 95
  - cmgen
    - GiNaC::composition\_generator, 441
  - CMPINDICES
    - color.cpp, 1311
  - coeff
    - GiNaC, 114
    - GiNaC::add, 326
    - GiNaC::basic, 368
    - GiNaC::ex, 587
    - GiNaC::expair, 610
    - GiNaC::mul, 877
    - GiNaC::ncmul, 938
    - GiNaC::numeric, 956
    - GiNaC::power, 1022
    - GiNaC::pseries, 1074
    - GiNaC::structure< T, ComparisonPolicy >, 1170
    - GiNaC::symminfo, 1234
  - coefficient\_a0
    - GiNaC::Eisenstein\_h\_kernel, 539
  - coefficient\_an
    - GiNaC::Eisenstein\_h\_kernel, 539

- coeffop
  - GiNaC::pseries, 1080
- coeffs
  - GiNaC::lanczos\_coeffs, 814
- coerce
  - GiNaC, 229
- coerce< int, cln::cl\_I >
  - GiNaC, 229
- coerce< unsigned int, cln::cl\_I >
  - GiNaC, 230
- col
  - GiNaC::matrix, 846
- collect
  - GiNaC, 116
  - GiNaC::basic, 369
  - GiNaC::ex, 588
  - GiNaC::pseries, 1075
  - GiNaC::structure< T, ComparisonPolicy >, 1171
- collect\_common\_factors
  - GiNaC, 226
- collect\_symbols
  - GiNaC, 212
- color
  - GiNaC::color, 431, 432
- color.cpp, 1309
  - CMPINDICES, 1311
  - TEST\_PERMUTATION, 1310
- color.h, 1311
- color\_d
  - GiNaC, 107
- color\_f
  - GiNaC, 107
- color\_h
  - GiNaC, 108
- color\_ONE
  - GiNaC, 106
- color\_T
  - GiNaC, 107
- color\_trace
  - GiNaC, 109
- cols
  - GiNaC, 207
  - GiNaC::matrix, 835
- combine\_ex\_with\_coeff\_to\_pair
  - GiNaC::add, 331
  - GiNaC::expairseq, 627
  - GiNaC::mul, 883
- combine\_indices
  - GiNaC::make\_flat\_inserter, 819
- combine\_overall\_coeff
  - GiNaC::expairseq, 628
  - GiNaC::mul, 885
- combine\_pair\_with\_coeff\_to\_pair
  - GiNaC::add, 331
  - GiNaC::expairseq, 627
  - GiNaC::mul, 884
- combine\_same\_terms\_sorted\_seq
  - GiNaC::expairseq, 631
- commutative
  - GiNaC::return\_types, 1131
- commutator\_sign
  - GiNaC::clifford, 416
- compare
  - GiNaC::basic, 379
  - GiNaC::ex, 597
  - GiNaC::expair, 609
  - GiNaC::numeric, 966
- compare\_pointers
  - GiNaC, 283
- compare\_same\_type
  - GiNaC::basic, 375
- compile\_ex
  - GiNaC, 122, 123
- compiler.h, 1312
  - attribute\_deprecated, 1313
  - likely, 1313
  - unlikely, 1312
- complex
  - GiNaC::domain, 522
- composition
  - GiNaC::composition\_generator, 441
- composition\_generator
  - GiNaC::composition\_generator, 440
- conjugate
  - GiNaC, 112
  - GiNaC::add, 328
  - GiNaC::basic, 375
  - GiNaC::constant, 463
  - GiNaC::container< C >, 478
  - GiNaC::diracgamma5, 505
  - GiNaC::diracgammaL, 510
  - GiNaC::diracgammaR, 515
  - GiNaC::ex, 582
  - GiNaC::expair, 609
  - GiNaC::expairseq, 623
  - GiNaC::function, 674
  - GiNaC::integral, 780
  - GiNaC::matrix, 833
  - GiNaC::mul, 881
  - GiNaC::ncmul, 940
  - GiNaC::numeric, 959
  - GiNaC::power, 1026
  - GiNaC::pseries, 1077
  - GiNaC::realsymbol, 1099
  - GiNaC::spinidx, 1145
  - GiNaC::symbol, 1215
- conjugate\_conjugate
  - GiNaC, 138
- conjugate\_eval
  - GiNaC, 137
- conjugate\_evalf
  - GiNaC, 137
- conjugate\_expl\_derivative
  - GiNaC, 138
- conjugate\_f
  - GiNaC::function\_options, 721

- conjugate\_func
  - GiNaC::function\_options, [692–694](#), [714](#)
- conjugate\_funcp
  - GiNaC, [60](#)
- conjugate\_funcp\_1
  - GiNaC, [62](#)
- conjugate\_funcp\_10
  - GiNaC, [77](#)
- conjugate\_funcp\_11
  - GiNaC, [78](#)
- conjugate\_funcp\_12
  - GiNaC, [80](#)
- conjugate\_funcp\_13
  - GiNaC, [82](#)
- conjugate\_funcp\_14
  - GiNaC, [84](#)
- conjugate\_funcp\_2
  - GiNaC, [63](#)
- conjugate\_funcp\_3
  - GiNaC, [65](#)
- conjugate\_funcp\_4
  - GiNaC, [66](#)
- conjugate\_funcp\_5
  - GiNaC, [68](#)
- conjugate\_funcp\_6
  - GiNaC, [70](#)
- conjugate\_funcp\_7
  - GiNaC, [72](#)
- conjugate\_funcp\_8
  - GiNaC, [73](#)
- conjugate\_funcp\_9
  - GiNaC, [75](#)
- conjugate\_funcp\_exvector
  - GiNaC, [86](#)
- conjugate\_imag\_part
  - GiNaC, [138](#)
- conjugate\_info
  - GiNaC, [139](#)
- conjugate\_print\_latex
  - GiNaC, [137](#)
- conjugate\_real\_part
  - GiNaC, [138](#)
- conjugate\_use\_exvector\_args
  - GiNaC::function\_options, [725](#)
- conjugateepvector
  - GiNaC, [125](#)
- const\_iterator
  - GiNaC::const\_iterator, [444](#)
  - GiNaC::container< C >, [474](#)
- const\_postorder\_iterator
  - GiNaC::const\_iterator, [448](#)
  - GiNaC::const\_postorder\_iterator, [450](#)
- const\_preorder\_iterator
  - GiNaC::const\_iterator, [447](#)
  - GiNaC::const\_preorder\_iterator, [453](#), [454](#)
- const\_reverse\_iterator
  - GiNaC::container< C >, [474](#)
- constant
  - GiNaC::constant, [461](#)
- constant.cpp, [1313](#)
- constant.h, [1314](#)
- construct\_from\_2\_ex
  - GiNaC::expairseq, [629](#)
- construct\_from\_2\_expairseq
  - GiNaC::expairseq, [629](#)
- construct\_from\_basic
  - GiNaC::ex, [600](#)
- construct\_from\_double
  - GiNaC::ex, [602](#)
- construct\_from\_epvector
  - GiNaC::expairseq, [630](#)
- construct\_from\_expairseq\_ex
  - GiNaC::expairseq, [630](#)
- construct\_from\_exvector
  - GiNaC::expairseq, [630](#)
- construct\_from\_int
  - GiNaC::ex, [600](#)
- construct\_from\_long
  - GiNaC::ex, [601](#)
- construct\_from\_longlong
  - GiNaC::ex, [601](#)
- construct\_from\_string\_and\_lst
  - GiNaC::ex, [602](#)
- construct\_from\_uint
  - GiNaC::ex, [601](#)
- construct\_from\_ulong
  - GiNaC::ex, [601](#)
- construct\_from\_ulonglong
  - GiNaC::ex, [602](#)
- cont
  - factor.cpp, [1332](#)
- container
  - GiNaC::container< C >, [474](#), [475](#)
- container.h, [1315](#)
- container\_storage
  - GiNaC::container\_storage< C >, [486](#)
- content
  - GiNaC::ex, [592](#)
- contract\_with
  - GiNaC::basic, [374](#)
  - GiNaC::cliffordunit, [421](#)
  - GiNaC::diracgamma, [500](#)
  - GiNaC::matrix, [833](#)
  - GiNaC::spinmetric, [1154](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1175](#)
  - GiNaC::su3d, [1183](#)
  - GiNaC::su3f, [1189](#)
  - GiNaC::su3t, [1199](#)
  - GiNaC::tensdelta, [1241](#)
  - GiNaC::tensepsilon, [1248](#)
  - GiNaC::tensmetric, [1255](#)
- convert\_H\_to\_Li
  - GiNaC, [159](#)
- convert\_to\_poly
  - GiNaC::pseries, [1079](#)
- coolmulti



- GiNaC::composition\_generator::coolmulti, [489](#)
- cos
  - GiNaC, [231](#)
- cos\_conjugate
  - GiNaC, [181](#)
- cos\_deriv
  - GiNaC, [180](#)
- cos\_eval
  - GiNaC, [180](#)
- cos\_evalf
  - GiNaC, [180](#)
- cos\_imag\_part
  - GiNaC, [181](#)
- cos\_real\_part
  - GiNaC, [181](#)
- cosh
  - GiNaC, [234](#)
- cosh\_conjugate
  - GiNaC, [190](#)
- cosh\_deriv
  - GiNaC, [189](#)
- cosh\_eval
  - GiNaC, [189](#)
- cosh\_evalf
  - GiNaC, [189](#)
- cosh\_imag\_part
  - GiNaC, [190](#)
- cosh\_real\_part
  - GiNaC, [190](#)
- count
  - GiNaC::archive\_node::property\_info, [1065](#)
  - GiNaC::library\_init, [817](#)
- count\_dummy\_indices
  - GiNaC, [132](#)
- count\_factors
  - GiNaC::ncmul, [942](#)
- count\_free\_indices
  - GiNaC, [132](#)
- covariant
  - GiNaC::varidx, [1287](#)
- crational
  - GiNaC::info\_flags, [770](#)
- crational\_polynomial
  - GiNaC::info\_flags, [770](#)
- crc32
  - GiNaC, [111](#)
- crc32.h, [1315](#)
- crctab
  - GiNaC, [290](#)
- csgn
  - GiNaC, [247](#)
  - GiNaC::numeric, [965](#)
- csgn\_conjugate
  - GiNaC, [146](#)
- csgn\_eval
  - GiNaC, [146](#)
- csgn\_evalf
  - GiNaC, [146](#)
- csgn\_imag\_part
  - GiNaC, [147](#)
- csgn\_power
  - GiNaC, [147](#)
- csgn\_real\_part
  - GiNaC, [146](#)
- csgn\_series
  - GiNaC, [146](#)
- csrc
  - GiNaC, [262](#)
- csrc\_cl\_N
  - GiNaC, [263](#)
- csrc\_double
  - GiNaC, [262](#)
- csrc\_float
  - GiNaC, [262](#)
- current\_serial
  - GiNaC::function, [680](#)
- current\_updated
  - GiNaC::composition\_generator, [441](#)
  - GiNaC::partition\_generator, [979](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [982](#)
- current\_vector
  - GiNaC::lanczos\_coeffs, [814](#)
- cyclic
  - GiNaC::symmetry, [1227](#)
- cyclic\_permutation
  - GiNaC, [284](#)
- dbgprint
  - GiNaC::basic, [363](#)
  - GiNaC::ex, [579](#)
- dbgprinttree
  - GiNaC::basic, [363](#)
  - GiNaC::ex, [579](#)
- DCOUT
  - factor.cpp, [1325](#)
- DCOUT2
  - factor.cpp, [1325](#)
- DCOUTVAR
  - factor.cpp, [1325](#)
- debugprint
  - GiNaC::scalar\_products, [1134](#)
  - GiNaC::spmapkey, [1157](#)
- DECLARE\_FUNCTION\_10P
  - function.h, [1346](#)
- DECLARE\_FUNCTION\_11P
  - function.h, [1346](#)
- DECLARE\_FUNCTION\_12P
  - function.h, [1347](#)
- DECLARE\_FUNCTION\_13P
  - function.h, [1347](#)
- DECLARE\_FUNCTION\_14P
  - function.h, [1347](#)
- DECLARE\_FUNCTION\_1P
  - function.h, [1344](#)
- DECLARE\_FUNCTION\_2P
  - function.h, [1344](#)
- DECLARE\_FUNCTION\_3P

- function.h, [1344](#)
- DECLARE\_FUNCTION\_4P
  - function.h, [1345](#)
- DECLARE\_FUNCTION\_5P
  - function.h, [1345](#)
- DECLARE\_FUNCTION\_6P
  - function.h, [1345](#)
- DECLARE\_FUNCTION\_7P
  - function.h, [1345](#)
- DECLARE\_FUNCTION\_8P
  - function.h, [1346](#)
- DECLARE\_FUNCTION\_9P
  - function.h, [1346](#)
- decomp\_rational
  - GiNaC, [215](#)
- DEFAULT\_COMPARE
  - utils.h, [1417](#)
- DEFAULT\_CTOR
  - utils.h, [1417](#)
- default\_overall\_coeff
  - GiNaC::expairseq, [628](#)
  - GiNaC::mul, [884](#)
- DEFAULT\_PRINT
  - utils.h, [1417](#)
- DEFAULT\_PRINT\_LATEX
  - utils.h, [1417](#)
- deg
  - GiNaC::pole\_error, [1006](#)
- deg\_a
  - GiNaC::sym\_desc, [1206](#)
- deg\_b
  - GiNaC::sym\_desc, [1206](#)
- degree
  - GiNaC, [114](#)
  - GiNaC::add, [325](#)
  - GiNaC::basic, [368](#)
  - GiNaC::ex, [587](#)
  - GiNaC::integral, [777](#)
  - GiNaC::mul, [876](#)
  - GiNaC::ncmul, [937](#)
  - GiNaC::numeric, [955](#)
  - GiNaC::pole\_error, [1006](#)
  - GiNaC::power, [1022](#)
  - GiNaC::pseries, [1074](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1170](#)
- delete\_cyclic
  - GiNaC::remember\_strategies, [1120](#)
- delete\_lfu
  - GiNaC::remember\_strategies, [1120](#)
- delete\_lru
  - GiNaC::remember\_strategies, [1120](#)
- delete\_never
  - GiNaC::remember\_strategies, [1120](#)
- delta\_indent
  - GiNaC::print\_tree, [1062](#)
- delta\_tensor
  - GiNaC, [278](#)
- denom
  - GiNaC, [115](#), [251](#)
  - GiNaC::ex, [591](#)
  - GiNaC::numeric, [974](#)
- derivative
  - GiNaC::add, [329](#)
  - GiNaC::basic, [370](#)
  - GiNaC::constant, [464](#)
  - GiNaC::fderivative, [655](#)
  - GiNaC::function, [675](#)
  - GiNaC::idx, [741](#)
  - GiNaC::indexed, [763](#)
  - GiNaC::integral, [781](#)
  - GiNaC::mul, [881](#)
  - GiNaC::ncmul, [940](#)
  - GiNaC::numeric, [960](#)
  - GiNaC::power, [1027](#)
  - GiNaC::pseries, [1078](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1171](#)
  - GiNaC::symbol, [1216](#)
- derivative\_f
  - GiNaC::function\_options, [722](#)
- derivative\_func
  - GiNaC::function\_options, [702–704](#), [714](#)
- derivative\_funcp
  - GiNaC, [61](#)
- derivative\_funcp\_1
  - GiNaC, [62](#)
- derivative\_funcp\_10
  - GiNaC, [77](#)
- derivative\_funcp\_11
  - GiNaC, [79](#)
- derivative\_funcp\_12
  - GiNaC, [81](#)
- derivative\_funcp\_13
  - GiNaC, [83](#)
- derivative\_funcp\_14
  - GiNaC, [85](#)
- derivative\_funcp\_2
  - GiNaC, [64](#)
- derivative\_funcp\_3
  - GiNaC, [65](#)
- derivative\_funcp\_4
  - GiNaC, [67](#)
- derivative\_funcp\_5
  - GiNaC, [69](#)
- derivative\_funcp\_6
  - GiNaC, [70](#)
- derivative\_funcp\_7
  - GiNaC, [72](#)
- derivative\_funcp\_8
  - GiNaC, [74](#)
- derivative\_funcp\_9
  - GiNaC, [76](#)
- derivative\_funcp\_exvector
  - GiNaC, [87](#)
- derivative\_map\_function
  - GiNaC::derivative\_map\_function, [492](#)
- derivative\_use\_exvector\_args

- GiNaC::function\_options, 725
- derivatives
  - GiNaC::fderivative, 656
- descend
  - GiNaC::const\_postorder\_iterator, 451
- determinant
  - GiNaC, 208
  - GiNaC::matrix, 838
- determinant\_minor
  - GiNaC::matrix, 842
- dflt
  - GiNaC, 261
- diag\_matrix
  - GiNaC, 205
- diff
  - GiNaC, 117
  - GiNaC::basic, 378
  - GiNaC::ex, 588
- difference\_type
  - GiNaC::const\_iterator, 444
  - GiNaC::const\_postorder\_iterator, 449
  - GiNaC::const\_preorder\_iterator, 453
- Digits
  - GiNaC, 291
- digits
  - GiNaC::\_numeric\_digits, 315
- digits\_changed\_callback
  - GiNaC, 89
- dim
  - GiNaC::idx, 745
  - GiNaC::spmapkey, 1158
- dirac\_gamma
  - GiNaC, 95
- dirac\_gamma5
  - GiNaC, 96
- dirac\_gammaL
  - GiNaC, 96
- dirac\_gammaR
  - GiNaC, 96
- dirac\_ONE
  - GiNaC, 94
- dirac\_slash
  - GiNaC, 97
- dirac\_trace
  - GiNaC, 97, 98
- dirichlet\_character
  - GiNaC, 198
- div
  - GiNaC::numeric, 962
- div\_dyn
  - GiNaC::numeric, 963
- divfree
  - GiNaC::determinant\_algo, 494
  - GiNaC::solve\_algo, 1136
- divide
  - GiNaC, 216
- divide\_in\_z
  - GiNaC, 217
- division\_free\_elimination
  - GiNaC::matrix, 843
- do\_not\_evalf\_params
  - GiNaC::function\_options, 718
- do\_print
  - GiNaC::add, 332
  - GiNaC::basic, 381
  - GiNaC::basic\_log\_kernel, 389
  - GiNaC::cliffordunit, 422
  - GiNaC::constant, 465
  - GiNaC::container< C >, 483
  - GiNaC::diracgamma, 500
  - GiNaC::diracgamma5, 505
  - GiNaC::diracgammaL, 511
  - GiNaC::diracgammaR, 516
  - GiNaC::diracone, 520
  - GiNaC::Ebar\_kernel, 529
  - GiNaC::Eisenstein\_h\_kernel, 540
  - GiNaC::Eisenstein\_kernel, 551
  - GiNaC::ELi\_kernel, 561
  - GiNaC::expairseq, 629
  - GiNaC::fail, 642
  - GiNaC::fderivative, 656
  - GiNaC::idx, 744
  - GiNaC::indexed, 767
  - GiNaC::integral, 781
  - GiNaC::integration\_kernel, 792
  - GiNaC::Kronecker\_dtau\_kernel, 802
  - GiNaC::Kronecker\_dz\_kernel, 811
  - GiNaC::matrix, 845
  - GiNaC::minkmetric, 854
  - GiNaC::modular\_form\_kernel, 863
  - GiNaC::mul, 886
  - GiNaC::multiple\_polylog\_kernel, 928
  - GiNaC::ncmul, 941
  - GiNaC::numeric, 975
  - GiNaC::pseries, 1083
  - GiNaC::relational, 1116
  - GiNaC::spinidx, 1147
  - GiNaC::spinmetric, 1155
  - GiNaC::su3d, 1183
  - GiNaC::su3f, 1189
  - GiNaC::su3one, 1194
  - GiNaC::su3t, 1200
  - GiNaC::symbol, 1218
  - GiNaC::symmetry, 1230
  - GiNaC::tensdelta, 1242
  - GiNaC::tensepsilon, 1249
  - GiNaC::tensmetric, 1256
  - GiNaC::user\_defined\_kernel, 1275
  - GiNaC::varidx, 1286
  - GiNaC::wildcard, 1293
- do\_print\_csrc
  - GiNaC::add, 333
  - GiNaC::fderivative, 657
  - GiNaC::idx, 744
  - GiNaC::mul, 887
  - GiNaC::ncmul, 941

- GiNaC::numeric, 975
  - GiNaC::power, 1029
- do\_print\_csrc\_cl\_N
  - GiNaC::numeric, 975
  - GiNaC::power, 1030
- do\_print\_dflt
  - GiNaC::clifford, 415
  - GiNaC::power, 1029
- do\_print\_latex
  - GiNaC::add, 333
  - GiNaC::clifford, 415
  - GiNaC::cliffordunit, 422
  - GiNaC::constant, 465
  - GiNaC::diracgamma, 500
  - GiNaC::diracgamma5, 505
  - GiNaC::diracgammaL, 511
  - GiNaC::diracgammaR, 516
  - GiNaC::diracone, 520
  - GiNaC::fderivative, 657
  - GiNaC::idx, 745
  - GiNaC::indexed, 767
  - GiNaC::integral, 782
  - GiNaC::matrix, 845
  - GiNaC::minkmetric, 854
  - GiNaC::mul, 887
  - GiNaC::numeric, 975
  - GiNaC::power, 1029
  - GiNaC::pseries, 1083
  - GiNaC::spinidx, 1147
  - GiNaC::spinmetric, 1155
  - GiNaC::su3d, 1183
  - GiNaC::su3f, 1189
  - GiNaC::su3one, 1194
  - GiNaC::su3t, 1200
  - GiNaC::symbol, 1219
  - GiNaC::tensdelta, 1242
  - GiNaC::tensepsilon, 1249
- do\_print\_python
  - GiNaC::container< C >, 483
  - GiNaC::power, 1029
  - GiNaC::pseries, 1083
- do\_print\_python\_repr
  - GiNaC::add, 333
  - GiNaC::basic, 382
  - GiNaC::constant, 465
  - GiNaC::container< C >, 484
  - GiNaC::matrix, 846
  - GiNaC::mul, 887
  - GiNaC::numeric, 976
  - GiNaC::power, 1030
  - GiNaC::pseries, 1084
  - GiNaC::relational, 1116
  - GiNaC::symbol, 1219
  - GiNaC::wildcard, 1294
- do\_print\_tree
  - GiNaC::basic, 382
  - GiNaC::clifford, 416
  - GiNaC::constant, 465
  - GiNaC::container< C >, 483
  - GiNaC::expairseq, 629
  - GiNaC::fderivative, 657
  - GiNaC::idx, 745
  - GiNaC::indexed, 767
  - GiNaC::numeric, 976
  - GiNaC::pseries, 1083
  - GiNaC::spinidx, 1147
  - GiNaC::symbol, 1219
  - GiNaC::symmetry, 1230
  - GiNaC::varidx, 1286
  - GiNaC::wildcard, 1294
- do\_renaming
  - GiNaC::make\_flat\_inserter, 819
- domain
  - GiNaC::constant, 467
- dotted
  - GiNaC::spinidx, 1148
- doublefactorial
  - GiNaC, 239
- dummy
  - GiNaC::function\_options, 687
- dump\_hierarchy
  - GiNaC::class\_info< OPT >, 399
- dump\_tree
  - GiNaC::class\_info< OPT >, 399
- duplicate
  - GiNaC::basic, 361
  - GiNaC::possymbol, 1013
  - GiNaC::print\_functor\_impl, 1048
  - GiNaC::print\_memfun\_handler< T, C >, 1052
  - GiNaC::print\_ptrfun\_handler< T, C >, 1056
  - GiNaC::realsymbol, 1099
- dynallocate
  - GiNaC, 92
- dynallocated
  - GiNaC::status\_flags, 1159
- e
  - GiNaC::archive\_node, 352
  - GiNaC::const\_iterator, 448
  - GiNaC::internal::\_iter\_rep, 312
- Ebar\_kernel
  - GiNaC::Ebar\_kernel, 528
- echelon\_form
  - GiNaC::matrix, 842
- ef
  - GiNaC::constant, 466
- Eisenstein\_h\_kernel
  - GiNaC::Eisenstein\_h\_kernel, 537
- Eisenstein\_kernel
  - GiNaC::Eisenstein\_kernel, 548
- element
  - GiNaC::composition\_generator::coolmulti::element, 553
- ELi\_kernel
  - GiNaC::ELi\_kernel, 560
- EllipticE\_deriv
  - GiNaC, 160

- EllipticE\_eval
  - GiNaC, [160](#)
- EllipticE\_evalf
  - GiNaC, [160](#)
- EllipticE\_print\_latex
  - GiNaC, [161](#)
- EllipticE\_series
  - GiNaC, [161](#)
- EllipticK\_deriv
  - GiNaC, [159](#)
- EllipticK\_eval
  - GiNaC, [159](#)
- EllipticK\_evalf
  - GiNaC, [159](#)
- EllipticK\_print\_latex
  - GiNaC, [160](#)
- EllipticK\_series
  - GiNaC, [159](#)
- end
  - GiNaC::archive\_node::archive\_node\_cit\_range, [353](#)
  - GiNaC::container< C >, [482](#)
  - GiNaC::ex, [576](#)
- ensure\_if\_modifiable
  - GiNaC::basic, [381](#)
- epp
  - GiNaC, [60](#)
- epsilon\_tensor
  - GiNaC, [280](#)
- epvector
  - GiNaC, [59](#)
- equal
  - GiNaC::relational, [1112](#)
- error
  - GiNaC::error\_and\_integral, [565](#)
- error\_and\_integral
  - GiNaC::error\_and\_integral, [564](#)
- eta\_conjugate
  - GiNaC, [148](#)
- eta\_eval
  - GiNaC, [147](#)
- eta\_evalf
  - GiNaC, [147](#)
- eta\_imag\_part
  - GiNaC, [148](#)
- eta\_real\_part
  - GiNaC, [148](#)
- eta\_series
  - GiNaC, [148](#)
- Euler
  - GiNaC, [289](#)
- EulerEvalf
  - GiNaC, [246](#)
- eval
  - GiNaC, [116](#)
  - GiNaC::add, [326](#)
  - GiNaC::basic, [361](#)
  - GiNaC::ex, [577](#)
  - GiNaC::expairseq, [621](#)
  - GiNaC::fderivative, [653](#)
  - GiNaC::function, [672](#)
  - GiNaC::indexed, [762](#)
  - GiNaC::integral, [777](#)
  - GiNaC::mul, [877](#)
  - GiNaC::ncmul, [938](#)
  - GiNaC::numeric, [956](#)
  - GiNaC::power, [1023](#)
  - GiNaC::pseries, [1075](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1165](#)
  - GiNaC::symbol, [1213](#)
- eval\_f
  - GiNaC::function\_options, [721](#)
- eval\_func
  - GiNaC::function\_options, [688–690](#), [713](#)
- eval\_funcp
  - GiNaC, [60](#)
- eval\_funcp\_1
  - GiNaC, [62](#)
- eval\_funcp\_10
  - GiNaC, [76](#)
- eval\_funcp\_11
  - GiNaC, [78](#)
- eval\_funcp\_12
  - GiNaC, [80](#)
- eval\_funcp\_13
  - GiNaC, [82](#)
- eval\_funcp\_14
  - GiNaC, [84](#)
- eval\_funcp\_2
  - GiNaC, [63](#)
- eval\_funcp\_3
  - GiNaC, [65](#)
- eval\_funcp\_4
  - GiNaC, [66](#)
- eval\_funcp\_5
  - GiNaC, [68](#)
- eval\_funcp\_6
  - GiNaC, [70](#)
- eval\_funcp\_7
  - GiNaC, [71](#)
- eval\_funcp\_8
  - GiNaC, [73](#)
- eval\_funcp\_9
  - GiNaC, [75](#)
- eval\_funcp\_exvector
  - GiNaC, [86](#)
- eval\_indexed
  - GiNaC::basic, [362](#)
  - GiNaC::matrix, [832](#)
  - GiNaC::minkmetric, [853](#)
  - GiNaC::spinmetric, [1154](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1166](#)
  - GiNaC::su3d, [1183](#)
  - GiNaC::su3f, [1189](#)
  - GiNaC::tensdelta, [1241](#)
  - GiNaC::tensepsilon, [1247](#)

- GiNaC::tensmetric, 1255
- eval\_integ
  - GiNaC, 117
  - GiNaC::basic, 362
  - GiNaC::ex, 578
  - GiNaC::integral, 780
  - GiNaC::pseries, 1077
- eval\_ncmul
  - GiNaC::add, 329
  - GiNaC::basic, 362
  - GiNaC::clifford, 412
  - GiNaC::color, 433
  - GiNaC::ex, 578
  - GiNaC::function, 672
  - GiNaC::integral, 778
  - GiNaC::mul, 882
  - GiNaC::power, 1027
  - GiNaC::relational, 1115
  - GiNaC::structure< T, ComparisonPolicy >, 1166
- eval\_use\_exvector\_args
  - GiNaC::function\_options, 724
- evalchildren
  - GiNaC::expairseq, 632
- evalf
  - GiNaC, 116, 207
  - GiNaC::basic, 361
  - GiNaC::constant, 462
  - GiNaC::ex, 577
  - GiNaC::function, 672
  - GiNaC::idx, 740
  - GiNaC::integral, 777
  - GiNaC::mul, 878
  - GiNaC::numeric, 956
  - GiNaC::power, 1023
  - GiNaC::pseries, 1075
  - GiNaC::symbol, 1213
- evalf\_f
  - GiNaC::function\_options, 721
- evalf\_func
  - GiNaC::function\_options, 690–692, 713
- evalf\_funcp
  - GiNaC, 60
- evalf\_funcp\_1
  - GiNaC, 62
- evalf\_funcp\_10
  - GiNaC, 77
- evalf\_funcp\_11
  - GiNaC, 78
- evalf\_funcp\_12
  - GiNaC, 80
- evalf\_funcp\_13
  - GiNaC, 82
- evalf\_funcp\_14
  - GiNaC, 84
- evalf\_funcp\_2
  - GiNaC, 63
- evalf\_funcp\_3
  - GiNaC, 65
- evalf\_funcp\_4
  - GiNaC, 66
- evalf\_funcp\_5
  - GiNaC, 68
- evalf\_funcp\_6
  - GiNaC, 70
- evalf\_funcp\_7
  - GiNaC, 71
- evalf\_funcp\_8
  - GiNaC, 73
- evalf\_funcp\_9
  - GiNaC, 75
- evalf\_funcp\_exvector
  - GiNaC, 86
- evalf\_params\_first
  - GiNaC::function\_options, 723
- evalf\_use\_exvector\_args
  - GiNaC::function\_options, 725
- evalffunctype
  - GiNaC, 59
- evalm
  - GiNaC, 117
  - GiNaC::add, 326
  - GiNaC::basic, 361
  - GiNaC::ex, 578
  - GiNaC::matrix, 831
  - GiNaC::mul, 879
  - GiNaC::ncmul, 939
  - GiNaC::power, 1024
  - GiNaC::pseries, 1078
  - GiNaC::structure< T, ComparisonPolicy >, 1165
- evalpoint
  - factor.cpp, 1331
- evaluate
  - GiNaC::scalar\_products, 1134
- evaluated
  - GiNaC::status\_flags, 1159
- even
  - GiNaC::info\_flags, 770
- ex
  - GiNaC::basic, 382
  - GiNaC::const\_iterator, 447
  - GiNaC::ex, 574–576
- ex.cpp, 1316
- ex.h, 1316
- ex\_to
  - GiNaC, 122
  - GiNaC::ex, 603
- exadd
  - GiNaC, 251
- excompiler.cpp, 1319
- excompiler.h, 1319
- exhashmap
  - GiNaC, 87
- exmap
  - GiNaC, 59
- exminus
  - GiNaC, 252

- exmul
  - GiNaC, [252](#)
- exp
  - GiNaC, [230](#)
- exp\_conjugate
  - GiNaC, [176](#)
- exp\_deriv
  - GiNaC, [175](#)
- exp\_eval
  - GiNaC, [175](#)
- exp\_evalf
  - GiNaC, [174](#)
- exp\_expand
  - GiNaC, [175](#)
- exp\_imag\_part
  - GiNaC, [175](#)
- exp\_info
  - GiNaC, [176](#)
- exp\_power
  - GiNaC, [176](#)
- exp\_real\_part
  - GiNaC, [175](#)
- expair
  - GiNaC::expair, [608](#)
- expair.cpp, [1320](#)
- expair.h, [1321](#)
- expair\_needs\_further\_processing
  - GiNaC::expairseq, [628](#)
  - GiNaC::mul, [884](#)
- expairseq
  - GiNaC::expairseq, [619](#), [620](#)
- expairseq.cpp, [1322](#)
- expairseq.h, [1322](#)
- expand
  - GiNaC, [112](#), [207](#)
  - GiNaC::add, [332](#)
  - GiNaC::basic, [369](#)
  - GiNaC::ex, [588](#)
  - GiNaC::expairseq, [625](#)
  - GiNaC::function, [672](#)
  - GiNaC::indexed, [765](#)
  - GiNaC::integral, [779](#)
  - GiNaC::mul, [885](#)
  - GiNaC::ncmul, [938](#)
  - GiNaC::power, [1028](#)
  - GiNaC::pseries, [1076](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1171](#)
- expand\_add
  - GiNaC::power, [1030](#)
- expand\_add\_2
  - GiNaC::power, [1030](#)
- expand\_dummy\_sum
  - GiNaC, [136](#)
- expand\_f
  - GiNaC::function\_options, [722](#)
- expand\_func
  - GiNaC::function\_options, [699–701](#), [714](#)
- expand\_funcp
  - GiNaC, [61](#)
- expand\_funcp\_1
  - GiNaC, [62](#)
- expand\_funcp\_10
  - GiNaC, [77](#)
- expand\_funcp\_11
  - GiNaC, [79](#)
- expand\_funcp\_12
  - GiNaC, [81](#)
- expand\_funcp\_13
  - GiNaC, [83](#)
- expand\_funcp\_14
  - GiNaC, [85](#)
- expand\_funcp\_2
  - GiNaC, [64](#)
- expand\_funcp\_3
  - GiNaC, [65](#)
- expand\_funcp\_4
  - GiNaC, [67](#)
- expand\_funcp\_5
  - GiNaC, [69](#)
- expand\_funcp\_6
  - GiNaC, [70](#)
- expand\_funcp\_7
  - GiNaC, [72](#)
- expand\_funcp\_8
  - GiNaC, [74](#)
- expand\_funcp\_9
  - GiNaC, [75](#)
- expand\_funcp\_exvector
  - GiNaC, [86](#)
- expand\_function\_args
  - GiNaC::expand\_options, [637](#)
- expand\_indexed
  - GiNaC::expand\_options, [637](#)
- expand\_map\_function
  - GiNaC::expand\_map\_function, [636](#)
- expand\_mul
  - GiNaC::power, [1031](#)
- expand\_rename\_idx
  - GiNaC::expand\_options, [637](#)
- expand\_transcendental
  - GiNaC::expand\_options, [637](#)
- expand\_use\_exvector\_args
  - GiNaC::function\_options, [725](#)
- expandchildren
  - GiNaC::expairseq, [632](#)
  - GiNaC::mul, [887](#)
  - GiNaC::ncmul, [942](#)
- expanded
  - GiNaC::info\_flags, [770](#)
  - GiNaC::status\_flags, [1159](#)
- expl\_derivative
  - GiNaC::function, [677](#)
- expl\_derivative\_f
  - GiNaC::function\_options, [722](#)
- expl\_derivative\_func
  - GiNaC::function\_options, [704–706](#), [714](#)

- expl\_derivative\_funcp
    - GiNaC, [61](#)
  - expl\_derivative\_funcp\_1
    - GiNaC, [62](#)
  - expl\_derivative\_funcp\_10
    - GiNaC, [77](#)
  - expl\_derivative\_funcp\_11
    - GiNaC, [79](#)
  - expl\_derivative\_funcp\_12
    - GiNaC, [81](#)
  - expl\_derivative\_funcp\_13
    - GiNaC, [83](#)
  - expl\_derivative\_funcp\_14
    - GiNaC, [85](#)
  - expl\_derivative\_funcp\_2
    - GiNaC, [64](#)
  - expl\_derivative\_funcp\_3
    - GiNaC, [65](#)
  - expl\_derivative\_funcp\_4
    - GiNaC, [67](#)
  - expl\_derivative\_funcp\_5
    - GiNaC, [69](#)
  - expl\_derivative\_funcp\_6
    - GiNaC, [71](#)
  - expl\_derivative\_funcp\_7
    - GiNaC, [72](#)
  - expl\_derivative\_funcp\_8
    - GiNaC, [74](#)
  - expl\_derivative\_funcp\_9
    - GiNaC, [76](#)
  - expl\_derivative\_funcp\_exvector
    - GiNaC, [87](#)
  - expl\_derivative\_use\_exvector\_args
    - GiNaC::function\_options, [726](#)
  - exponent
    - GiNaC::power, [1032](#)
  - exponop
    - GiNaC::pseries, [1080](#)
  - exprs
    - GiNaC::archive, [341](#)
  - exprseq
    - GiNaC, [60](#)
    - GiNaC::info\_flags, [770](#)
  - exprseq.cpp, [1323](#)
  - exprseq.h, [1324](#)
  - exprtable
    - GiNaC::archive, [341](#)
  - exset
    - GiNaC, [59](#)
  - exvector
    - GiNaC, [58](#)
  - exvectorvector
    - GiNaC, [88](#)
  - F
    - GiNaC::print\_memfun\_handler< T, C >, [1052](#)
    - GiNaC::print\_ptrfun\_handler< T, C >, [1056](#)
  - f
    - GiNaC::integral, [783](#)
    - GiNaC::print\_memfun\_handler< T, C >, [1053](#)
    - GiNaC::print\_ptrfun\_handler< T, C >, [1057](#)
    - GiNaC::user\_defined\_kernel, [1276](#)
  - factor
    - GiNaC, [126](#)
  - factor.cpp, [1324](#)
  - c, [1326](#)
  - cache, [1328](#)
  - cont, [1332](#)
  - DCOUT, [1325](#)
  - DCOUT2, [1325](#)
  - DCOUTVAR, [1325](#)
  - evalpoint, [1331](#)
  - factors, [1328](#)
  - k, [1329](#)
  - last, [1329](#)
  - len, [1329](#)
  - lr, [1328](#)
  - m, [1327](#)
  - modulus, [1332](#)
  - n, [1328](#)
  - one, [1328](#)
  - options, [1332](#)
  - poly, [1330](#)
  - pp, [1332](#)
  - R, [1331](#)
  - r, [1326](#)
  - syms, [1332](#)
  - syms\_wox, [1331](#)
  - unit, [1331](#)
  - USE\_SAME\_DEGREE\_FACTOR, [1326](#)
  - value, [1326](#)
  - vn, [1332](#)
  - vnlst, [1332](#)
  - x, [1330](#)
- factor.h, [1333](#)
- factorial
  - GiNaC, [239](#)
- factorial\_conjugate
  - GiNaC, [151](#)
- factorial\_eval
  - GiNaC, [151](#)
- factorial\_evalf
  - GiNaC, [151](#)
- factorial\_imag\_part
  - GiNaC, [152](#)
- factorial\_print\_dfft\_latex
  - GiNaC, [151](#)
- factorial\_real\_part
  - GiNaC, [152](#)
- factors
  - factor.cpp, [1328](#)
- fail.cpp, [1333](#)
- fail.h, [1334](#)
- FAST\_COMPARE
  - normal.cpp, [1381](#)
- fderivative
  - GiNaC::fderivative, [652](#), [653](#)



- GiNaC::function\_options, 720
- fderivative.cpp, 1335
- fderivative.h, 1335
- fibonacci
  - GiNaC, 241
- find
  - GiNaC, 113
  - GiNaC::class\_info< OPT >, 399
  - GiNaC::ex, 583
  - GiNaC::unarchive\_table\_t, 1266
- find\_bool
  - GiNaC::archive\_node, 346
- find\_common\_factor
  - GiNaC, 226
- find\_dummy\_indices
  - GiNaC, 131
- find\_ex
  - GiNaC::archive\_node, 348
- find\_ex\_by\_loc
  - GiNaC::archive\_node, 348
- find\_ex\_node
  - GiNaC::archive\_node, 349
- find\_factory\_fcn
  - GiNaC, 91
- find\_first
  - GiNaC::archive\_node, 347
- find\_free\_and\_dummy
  - GiNaC, 130, 131
- find\_function
  - GiNaC::function, 678
- find\_last
  - GiNaC::archive\_node, 348
- find\_property\_range
  - GiNaC::archive\_node, 348
- find\_real\_imag
  - GiNaC::mul, 886
- find\_string
  - GiNaC::archive\_node, 347
- find\_unsigned
  - GiNaC::archive\_node, 347
- find\_variant\_indices
  - GiNaC, 133
- finished
  - GiNaC::composition\_generator::coolmulti, 490
- first
  - GiNaC::class\_info< OPT >, 400
- flag\_overflow
  - GiNaC::basic\_multi\_iterator< T >, 395
- flags
  - GiNaC::basic, 382
- flags.h, 1336
- force\_include\_tgamma
  - GiNaC, 290
- force\_include\_zeta1
  - GiNaC, 290
- forget
  - GiNaC::archive, 339
  - GiNaC::archive\_node, 350
- format\_index\_value
  - GiNaC, 284, 285
- frac\_cancel
  - GiNaC, 225
- fraction\_free\_elimination
  - GiNaC::matrix, 844
- fsolve
  - GiNaC, 155
- func\_arg\_info
  - GiNaC, 138
- FUNCP\_1P
  - GiNaC, 59
- FUNCP\_2P
  - GiNaC, 59
- FUNCP\_CUBA
  - GiNaC, 59
- function
  - GiNaC::function, 666–671
  - GiNaC::function\_options, 720
- function.cpp, 1337
- function.h, 1337
  - DECLARE\_FUNCTION\_10P, 1346
  - DECLARE\_FUNCTION\_11P, 1346
  - DECLARE\_FUNCTION\_12P, 1347
  - DECLARE\_FUNCTION\_13P, 1347
  - DECLARE\_FUNCTION\_14P, 1347
  - DECLARE\_FUNCTION\_1P, 1344
  - DECLARE\_FUNCTION\_2P, 1344
  - DECLARE\_FUNCTION\_3P, 1344
  - DECLARE\_FUNCTION\_4P, 1345
  - DECLARE\_FUNCTION\_5P, 1345
  - DECLARE\_FUNCTION\_6P, 1345
  - DECLARE\_FUNCTION\_7P, 1345
  - DECLARE\_FUNCTION\_8P, 1346
  - DECLARE\_FUNCTION\_9P, 1346
  - is\_ex\_the\_function, 1348
  - REGISTER\_FUNCTION, 1348
- function\_options
  - GiNaC::function\_options, 686
- functions\_with\_same\_name
  - GiNaC::function\_options, 726
- G
  - GiNaC, 156, 157
- G2\_eval
  - GiNaC, 168
- G2\_evalf
  - GiNaC, 168
- G3\_eval
  - GiNaC, 169
- G3\_evalf
  - GiNaC, 168
- gauss
  - GiNaC::determinant\_algo, 494
  - GiNaC::solve\_algo, 1136
- gauss\_elimination
  - GiNaC::matrix, 843
- gcd
  - GiNaC, 221, 244

- gcd\_pf\_mul
  - GiNaC, [221](#)
- gcd\_pf\_pow
  - GiNaC, [220](#)
- gcd\_pf\_pow\_pow
  - GiNaC, [222](#)
- generalised\_Bernoulli\_number
  - GiNaC, [198](#)
- get
  - GiNaC::composition\_generator, [440](#)
  - GiNaC::partition\_generator, [979](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [981](#)
- get\_all\_dummy\_indices
  - GiNaC, [135](#)
- get\_all\_dummy\_indices\_safely
  - GiNaC, [135](#)
- get\_cache\_size
  - GiNaC::integration\_kernel, [790](#)
- get\_class\_name
  - GiNaC::structure< T, ComparisonPolicy >, [1165](#)
- get\_clifford\_comp
  - GiNaC, [101](#)
- get\_close\_delim
  - GiNaC::container< C >, [475](#)
- get\_commutator\_sign
  - GiNaC::clifford, [414](#)
- get\_default\_flags
  - GiNaC::container< C >, [475](#)
- get\_default\_TeX\_name
  - GiNaC, [268](#)
- get\_dim
  - GiNaC::idx, [743](#)
- get\_dim\_uint
  - GiNaC, [95](#)
- get\_domain
  - GiNaC::possymbol, [1013](#)
  - GiNaC::realsymbol, [1099](#)
  - GiNaC::symbol, [1217](#)
- get\_dummy\_indices
  - GiNaC::indexed, [765](#), [766](#)
- get\_ex
  - GiNaC::archive\_node, [350](#)
- get\_factors
  - GiNaC::ncmul, [942](#)
- get\_first\_symbol
  - GiNaC, [211](#)
- get\_free\_indices
  - GiNaC::add, [329](#)
  - GiNaC::basic, [372](#)
  - GiNaC::ex, [595](#)
  - GiNaC::indexed, [763](#)
  - GiNaC::integral, [779](#)
  - GiNaC::mul, [881](#)
  - GiNaC::ncmul, [939](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1174](#)
- get\_id
  - GiNaC::print\_context\_options, [1035](#)
  - GiNaC::registered\_class\_options, [1103](#)
- get\_indices
  - GiNaC::indexed, [765](#)
- get\_label
  - GiNaC::wildcard, [1293](#)
- get\_last\_access
  - GiNaC::remember\_table\_entry, [1125](#)
- get\_metric
  - GiNaC::clifford, [413](#)
- get\_name
  - GiNaC::function, [679](#)
  - GiNaC::function\_options, [719](#)
  - GiNaC::print\_context\_options, [1035](#)
  - GiNaC::registered\_class\_options, [1103](#)
  - GiNaC::symbol, [1218](#)
- get\_node
  - GiNaC::archive, [339](#)
- get\_nparams
  - GiNaC::function\_options, [719](#)
- get\_numerical\_value
  - GiNaC::Ebar\_kernel, [529](#)
  - GiNaC::Eisenstein\_h\_kernel, [539](#)
  - GiNaC::Eisenstein\_kernel, [550](#)
  - GiNaC::ELi\_kernel, [561](#)
  - GiNaC::integration\_kernel, [790](#)
  - GiNaC::Kronecker\_dtau\_kernel, [802](#)
  - GiNaC::Kronecker\_dz\_kernel, [811](#)
  - GiNaC::modular\_form\_kernel, [862](#)
- get\_numerical\_value\_impl
  - GiNaC::integration\_kernel, [791](#)
- get\_open\_delim
  - GiNaC::container< C >, [475](#)
- get\_order
  - GiNaC::lanczos\_coeffs, [813](#)
- get\_parent
  - GiNaC::class\_info< OPT >, [398](#)
- get\_parent\_name
  - GiNaC::print\_context\_options, [1035](#)
  - GiNaC::registered\_class\_options, [1103](#)
- get\_point
  - GiNaC::pseries, [1079](#)
- get\_pointer
  - GiNaC::ptr< T >, [1091](#)
- get\_print\_context
  - GiNaC, [260](#)
- get\_print\_dispatch\_table
  - GiNaC::registered\_class\_options, [1104](#)
- get\_print\_options
  - GiNaC, [260](#)
- get\_properties
  - GiNaC::archive\_node, [349](#)
- get\_refcount
  - GiNaC::refcounted, [1101](#)
- get\_registered\_functions
  - GiNaC::function, [679](#)
- get\_representation\_label
  - GiNaC, [97](#), [108](#)
  - GiNaC::clifford, [413](#)
  - GiNaC::color, [435](#)

get\_result  
     GiNaC::remember\_table\_entry, 1125  
 get\_serial  
     GiNaC::function, 679  
 get\_series\_coeff  
     GiNaC::integration\_kernel, 791  
 get\_sign  
     GiNaC::multi\_iterator\_permutation< T >, 911  
 get\_struct  
     GiNaC::structure< T, ComparisonPolicy >, 1177  
 get\_successful\_hits  
     GiNaC::remember\_table\_entry, 1126  
 get\_symbol\_stats  
     GiNaC, 212  
 get\_symmetry  
     GiNaC::indexed, 766  
 get\_TeX\_name  
     GiNaC::symbol, 1218  
 get\_top\_node  
     GiNaC::archive, 338  
 get\_type  
     GiNaC::symmetry, 1228  
 get\_value  
     GiNaC::idx, 742  
 get\_var  
     GiNaC::pseries, 1079  
 get\_vector  
     GiNaC::basic\_multi\_iterator< T >, 392  
 gethash  
     GiNaC::basic, 380  
     GiNaC::ex, 600  
 GiNaC, 19  
     \_ex0, 300  
     \_ex1, 302  
     \_ex10, 306  
     \_ex11, 306  
     \_ex12, 307  
     \_ex120, 310  
     \_ex15, 307  
     \_ex18, 307  
     \_ex1\_2, 302  
     \_ex1\_3, 301  
     \_ex1\_4, 301  
     \_ex2, 303  
     \_ex20, 308  
     \_ex24, 308  
     \_ex25, 308  
     \_ex3, 304  
     \_ex30, 309  
     \_ex4, 304  
     \_ex48, 309  
     \_ex5, 304  
     \_ex6, 305  
     \_ex60, 309  
     \_ex7, 305  
     \_ex8, 305  
     \_ex9, 306  
     \_ex\_1, 299  
     \_ex\_10, 296  
     \_ex\_11, 295  
     \_ex\_12, 295  
     \_ex\_120, 292  
     \_ex\_15, 295  
     \_ex\_18, 294  
     \_ex\_1\_2, 299  
     \_ex\_1\_3, 300  
     \_ex\_1\_4, 300  
     \_ex\_2, 298  
     \_ex\_20, 294  
     \_ex\_24, 294  
     \_ex\_25, 293  
     \_ex\_3, 298  
     \_ex\_30, 293  
     \_ex\_4, 298  
     \_ex\_48, 293  
     \_ex\_5, 297  
     \_ex\_6, 297  
     \_ex\_60, 292  
     \_ex\_7, 297  
     \_ex\_8, 296  
     \_ex\_9, 296  
     \_num0\_bp, 290  
     \_num0\_p, 300  
     \_num10\_p, 306  
     \_num11\_p, 306  
     \_num120\_p, 309  
     \_num12\_p, 306  
     \_num15\_p, 307  
     \_num18\_p, 307  
     \_num1\_2\_p, 302  
     \_num1\_3\_p, 301  
     \_num1\_4\_p, 301  
     \_num1\_p, 302  
     \_num20\_p, 307  
     \_num24\_p, 308  
     \_num25\_p, 308  
     \_num2\_p, 303  
     \_num30\_p, 308  
     \_num3\_p, 303  
     \_num48\_p, 309  
     \_num4\_p, 304  
     \_num5\_p, 304  
     \_num60\_p, 309  
     \_num6\_p, 304  
     \_num7\_p, 305  
     \_num8\_p, 305  
     \_num9\_p, 305  
     \_num\_10\_p, 296  
     \_num\_11\_p, 295  
     \_num\_120\_p, 292  
     \_num\_12\_p, 295  
     \_num\_15\_p, 295  
     \_num\_18\_p, 294  
     \_num\_1\_2\_p, 299  
     \_num\_1\_3\_p, 299  
     \_num\_1\_4\_p, 300

- [\\_num\\_1\\_p](#), 299
- [\\_num\\_20\\_p](#), 294
- [\\_num\\_24\\_p](#), 294
- [\\_num\\_25\\_p](#), 293
- [\\_num\\_2\\_p](#), 298
- [\\_num\\_30\\_p](#), 293
- [\\_num\\_3\\_p](#), 298
- [\\_num\\_48\\_p](#), 293
- [\\_num\\_4\\_p](#), 298
- [\\_num\\_5\\_p](#), 297
- [\\_num\\_60\\_p](#), 292
- [\\_num\\_6\\_p](#), 297
- [\\_num\\_7\\_p](#), 297
- [\\_num\\_8\\_p](#), 296
- [\\_num\\_9\\_p](#), 296
- [abs](#), 241
- [abs\\_conjugate](#), 143
- [abs\\_eval](#), 142
- [abs\\_evalf](#), 142
- [abs\\_expand](#), 142
- [abs\\_expl\\_derivative](#), 142
- [abs\\_imag\\_part](#), 143
- [abs\\_info](#), 144
- [abs\\_power](#), 144
- [abs\\_print\\_csrc\\_float](#), 143
- [abs\\_print\\_latex](#), 143
- [abs\\_real\\_part](#), 143
- [acos](#), 232
- [acos\\_conjugate](#), 185
- [acos\\_deriv](#), 185
- [acos\\_eval](#), 184
- [acos\\_evalf](#), 184
- [acosh](#), 235
- [acosh\\_conjugate](#), 194
- [acosh\\_deriv](#), 193
- [acosh\\_eval](#), 193
- [acosh\\_evalf](#), 193
- [adaptivesimpson](#), 196
- [add\\_symbol](#), 212
- [algebraic\\_match\\_mul\\_with\\_mul](#), 210
- [antisymmetric2](#), 271
- [antisymmetric3](#), 271
- [antisymmetric4](#), 271
- [antisymmetrize](#), 119, 272, 276
- [archive\\_atom](#), 58
- [archive\\_node\\_id](#), 58
- [are\\_ex\\_trivially\\_equal](#), 111
- [asin](#), 232
- [asin\\_conjugate](#), 184
- [asin\\_deriv](#), 183
- [asin\\_eval](#), 183
- [asin\\_evalf](#), 183
- [asin\\_info](#), 184
- [asinh](#), 235
- [asinh\\_conjugate](#), 193
- [asinh\\_deriv](#), 192
- [asinh\\_eval](#), 192
- [asinh\\_evalf](#), 192
- [atan](#), 233
- [atan2\\_deriv](#), 187
- [atan2\\_eval](#), 187
- [atan2\\_evalf](#), 187
- [atan2\\_info](#), 187
- [atan\\_conjugate](#), 186
- [atan\\_deriv](#), 186
- [atan\\_eval](#), 185
- [atan\\_evalf](#), 185
- [atan\\_info](#), 186
- [atan\\_series](#), 186
- [atanh](#), 235
- [atanh\\_conjugate](#), 195
- [atanh\\_deriv](#), 194
- [atanh\\_eval](#), 194
- [atanh\\_evalf](#), 194
- [atanh\\_series](#), 195
- [base\\_and\\_index](#), 94
- [bernoulli](#), 240
- [Bernoulli\\_polynomial](#), 198
- [beta\\_deriv](#), 165
- [beta\\_eval](#), 165
- [beta\\_evalf](#), 165
- [beta\\_series](#), 165
- [binomial](#), 240
- [binomial\\_conjugate](#), 153
- [binomial\\_eval](#), 153
- [binomial\\_evalf](#), 152
- [binomial\\_imag\\_part](#), 153
- [binomial\\_real\\_part](#), 153
- [binomial\\_sym](#), 152
- [callback\\_registered](#), 89
- [canonicalize](#), 271
- [canonicalize\\_clifford](#), 99
- [Catalan](#), 289
- [CatalanEvalf](#), 246
- [charpoly](#), 208
- [clifford\\_bar](#), 104
- [clifford\\_inverse](#), 100
- [clifford\\_max\\_label](#), 100
- [clifford\\_moebius\\_map](#), 102
- [clifford\\_norm](#), 100
- [clifford\\_prime](#), 99
- [clifford\\_star](#), 104
- [clifford\\_star\\_bar](#), 99
- [clifford\\_to\\_lst](#), 101
- [clifford\\_unit](#), 95
- [coeff](#), 114
- [coerce](#), 229
- [coerce< int, cln::cl\\_I >](#), 229
- [coerce< unsigned int, cln::cl\\_I >](#), 230
- [collect](#), 116
- [collect\\_common\\_factors](#), 226
- [collect\\_symbols](#), 212
- [color\\_d](#), 107
- [color\\_f](#), 107
- [color\\_h](#), 108
- [color\\_ONE](#), 106

color\_T, 107  
color\_trace, 109  
cols, 207  
compare\_pointers, 283  
compile\_ex, 122, 123  
conjugate, 112  
conjugate\_conjugate, 138  
conjugate\_eval, 137  
conjugate\_evalf, 137  
conjugate\_expl\_derivative, 138  
conjugate\_funcp, 60  
conjugate\_funcp\_1, 62  
conjugate\_funcp\_10, 77  
conjugate\_funcp\_11, 78  
conjugate\_funcp\_12, 80  
conjugate\_funcp\_13, 82  
conjugate\_funcp\_14, 84  
conjugate\_funcp\_2, 63  
conjugate\_funcp\_3, 65  
conjugate\_funcp\_4, 66  
conjugate\_funcp\_5, 68  
conjugate\_funcp\_6, 70  
conjugate\_funcp\_7, 72  
conjugate\_funcp\_8, 73  
conjugate\_funcp\_9, 75  
conjugate\_funcp\_exvector, 86  
conjugate\_imag\_part, 138  
conjugate\_info, 139  
conjugate\_print\_latex, 137  
conjugate\_real\_part, 138  
conjugateepvector, 125  
convert\_H\_to\_Li, 159  
cos, 231  
cos\_conjugate, 181  
cos\_deriv, 180  
cos\_eval, 180  
cos\_evalf, 180  
cos\_imag\_part, 181  
cos\_real\_part, 181  
cosh, 234  
cosh\_conjugate, 190  
cosh\_deriv, 189  
cosh\_eval, 189  
cosh\_evalf, 189  
cosh\_imag\_part, 190  
cosh\_real\_part, 190  
count\_dummy\_indices, 132  
count\_free\_indices, 132  
crc32, 111  
crctab, 290  
csgn, 247  
csgn\_conjugate, 146  
csgn\_eval, 146  
csgn\_evalf, 146  
csgn\_imag\_part, 147  
csgn\_power, 147  
csgn\_real\_part, 146  
csgn\_series, 146  
csrc, 262  
csrc\_cl\_N, 263  
csrc\_double, 262  
csrc\_float, 262  
cyclic\_permutation, 284  
decomp\_rational, 215  
degree, 114  
delta\_tensor, 278  
denom, 115, 251  
derivative\_funcp, 61  
derivative\_funcp\_1, 62  
derivative\_funcp\_10, 77  
derivative\_funcp\_11, 79  
derivative\_funcp\_12, 81  
derivative\_funcp\_13, 83  
derivative\_funcp\_14, 85  
derivative\_funcp\_2, 64  
derivative\_funcp\_3, 65  
derivative\_funcp\_4, 67  
derivative\_funcp\_5, 69  
derivative\_funcp\_6, 70  
derivative\_funcp\_7, 72  
derivative\_funcp\_8, 74  
derivative\_funcp\_9, 76  
derivative\_funcp\_exvector, 87  
determinant, 208  
dfft, 261  
diag\_matrix, 205  
diff, 117  
Digits, 291  
digits\_changed\_callback, 89  
dirac\_gamma, 95  
dirac\_gamma5, 96  
dirac\_gammaL, 96  
dirac\_gammaR, 96  
dirac\_ONE, 94  
dirac\_slash, 97  
dirac\_trace, 97, 98  
dirichlet\_character, 198  
divide, 216  
divide\_in\_z, 217  
doublefactorial, 239  
dynallocate, 92  
EllipticE\_deriv, 160  
EllipticE\_eval, 160  
EllipticE\_evalf, 160  
EllipticE\_print\_latex, 161  
EllipticE\_series, 161  
EllipticK\_deriv, 159  
EllipticK\_eval, 159  
EllipticK\_evalf, 159  
EllipticK\_print\_latex, 160  
EllipticK\_series, 159  
epp, 60  
epsilon\_tensor, 280  
epvector, 59  
eta\_conjugate, 148  
eta\_eval, 147

eta\_evalf, 147  
eta\_imag\_part, 148  
eta\_real\_part, 148  
eta\_series, 148  
Euler, 289  
EulerEvalf, 246  
eval, 116  
eval\_funcp, 60  
eval\_funcp\_1, 62  
eval\_funcp\_10, 76  
eval\_funcp\_11, 78  
eval\_funcp\_12, 80  
eval\_funcp\_13, 82  
eval\_funcp\_14, 84  
eval\_funcp\_2, 63  
eval\_funcp\_3, 65  
eval\_funcp\_4, 66  
eval\_funcp\_5, 68  
eval\_funcp\_6, 70  
eval\_funcp\_7, 71  
eval\_funcp\_8, 73  
eval\_funcp\_9, 75  
eval\_funcp\_exvector, 86  
eval\_integ, 117  
evalf, 116, 207  
evalf\_funcp, 60  
evalf\_funcp\_1, 62  
evalf\_funcp\_10, 77  
evalf\_funcp\_11, 78  
evalf\_funcp\_12, 80  
evalf\_funcp\_13, 82  
evalf\_funcp\_14, 84  
evalf\_funcp\_2, 63  
evalf\_funcp\_3, 65  
evalf\_funcp\_4, 66  
evalf\_funcp\_5, 68  
evalf\_funcp\_6, 70  
evalf\_funcp\_7, 71  
evalf\_funcp\_8, 73  
evalf\_funcp\_9, 75  
evalf\_funcp\_exvector, 86  
evalffunctype, 59  
evalm, 117  
ex\_to, 122  
exadd, 251  
exhashmap, 87  
exmap, 59  
exminus, 252  
exmul, 252  
exp, 230  
exp\_conjugate, 176  
exp\_deriv, 175  
exp\_eval, 175  
exp\_evalf, 174  
exp\_expand, 175  
exp\_imag\_part, 175  
exp\_info, 176  
exp\_power, 176  
exp\_real\_part, 175  
expand, 112, 207  
expand\_dummy\_sum, 136  
expand\_funcp, 61  
expand\_funcp\_1, 62  
expand\_funcp\_10, 77  
expand\_funcp\_11, 79  
expand\_funcp\_12, 81  
expand\_funcp\_13, 83  
expand\_funcp\_14, 85  
expand\_funcp\_2, 64  
expand\_funcp\_3, 65  
expand\_funcp\_4, 67  
expand\_funcp\_5, 69  
expand\_funcp\_6, 70  
expand\_funcp\_7, 72  
expand\_funcp\_8, 74  
expand\_funcp\_9, 75  
expand\_funcp\_exvector, 86  
expl\_derivative\_funcp, 61  
expl\_derivative\_funcp\_1, 62  
expl\_derivative\_funcp\_10, 77  
expl\_derivative\_funcp\_11, 79  
expl\_derivative\_funcp\_12, 81  
expl\_derivative\_funcp\_13, 83  
expl\_derivative\_funcp\_14, 85  
expl\_derivative\_funcp\_2, 64  
expl\_derivative\_funcp\_3, 65  
expl\_derivative\_funcp\_4, 67  
expl\_derivative\_funcp\_5, 69  
expl\_derivative\_funcp\_6, 71  
expl\_derivative\_funcp\_7, 72  
expl\_derivative\_funcp\_8, 74  
expl\_derivative\_funcp\_9, 76  
expl\_derivative\_funcp\_exvector, 87  
exprseq, 60  
exset, 59  
exvector, 58  
exvectorvector, 88  
factor, 126  
factorial, 239  
factorial\_conjugate, 151  
factorial\_eval, 151  
factorial\_evalf, 151  
factorial\_imag\_part, 152  
factorial\_print\_dflt\_latex, 151  
factorial\_real\_part, 152  
fibonacci, 241  
find, 113  
find\_common\_factor, 226  
find\_dummy\_indices, 131  
find\_factory\_fcn, 91  
find\_free\_and\_dummy, 130, 131  
find\_variant\_indices, 133  
force\_include\_tgamma, 290  
force\_include\_zeta1, 290  
format\_index\_value, 284, 285  
frac\_cancel, 225

- fsolve, 155
- func\_arg\_info, 138
- FUNCP\_1P, 59
- FUNCP\_2P, 59
- FUNCP\_CUBA, 59
- G, 156, 157
- G2\_eval, 168
- G2\_evalf, 168
- G3\_eval, 169
- G3\_evalf, 168
- gcd, 221, 244
- gcd\_pf\_mul, 221
- gcd\_pf\_pow, 220
- gcd\_pf\_pow\_pow, 222
- generalised\_Bernoulli\_number, 198
- get\_all\_dummy\_indices, 135
- get\_all\_dummy\_indices\_safely, 135
- get\_clifford\_comp, 101
- get\_default\_TeX\_name, 268
- get\_dim\_uint, 95
- get\_first\_symbol, 211
- get\_print\_context, 260
- get\_print\_options, 260
- get\_representation\_label, 97, 108
- get\_symbol\_stats, 212
- GINAC\_BIND\_UNARCHIVER, 90, 93, 94, 105, 111, 127, 129, 132, 196, 199–202, 204, 210, 211, 228, 264, 265, 267–269, 277, 278, 287, 291
- GINAC\_DECLARE\_UNARCHIVER, 90, 103, 104, 110, 111, 127, 130, 131, 137, 196, 202–204, 206, 210, 211, 246, 264, 265, 267–269, 273, 281, 282, 288
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT, 89, 91, 92, 104, 110, 125–128, 132, 195, 199–202, 204, 209, 211, 227, 263, 265, 267, 269, 277, 287
- golden\_ratio\_hash, 283
- guess\_precision, 237
- H\_deriv, 172
- H\_eval, 172
- H\_evalf, 172
- H\_print\_latex, 172
- H\_series, 172
- has, 113
- hasindex, 135
- haswild, 288
- heur\_gcd, 220
- heur\_gcd\_z, 219
- hold\_ncmul, 211
- I, 291
- idx, 290
- idx\_symmetrization, 134
- ifactor, 196
- imag, 251
- imag\_part, 113
- imag\_part\_conjugate, 141
- imag\_part\_eval, 141
- imag\_part\_evalf, 140
- imag\_part\_expl\_derivative, 141
- imag\_part\_funcp, 61
- imag\_part\_funcp\_1, 62
- imag\_part\_funcp\_10, 77
- imag\_part\_funcp\_11, 79
- imag\_part\_funcp\_12, 80
- imag\_part\_funcp\_13, 82
- imag\_part\_funcp\_14, 84
- imag\_part\_funcp\_2, 64
- imag\_part\_funcp\_3, 65
- imag\_part\_funcp\_4, 67
- imag\_part\_funcp\_5, 68
- imag\_part\_funcp\_6, 70
- imag\_part\_funcp\_7, 72
- imag\_part\_funcp\_8, 74
- imag\_part\_funcp\_9, 75
- imag\_part\_funcp\_exvector, 86
- imag\_part\_imag\_part, 141
- imag\_part\_print\_latex, 141
- imag\_part\_real\_part, 141
- index0, 269
- index1, 269
- index2, 270
- index3, 270
- index\_dimensions, 263
- indices\_consistent, 132
- info\_funcp, 61
- info\_funcp\_1, 63
- info\_funcp\_10, 78
- info\_funcp\_11, 80
- info\_funcp\_12, 82
- info\_funcp\_13, 84
- info\_funcp\_14, 86
- info\_funcp\_2, 64
- info\_funcp\_3, 66
- info\_funcp\_4, 68
- info\_funcp\_5, 69
- info\_funcp\_6, 71
- info\_funcp\_7, 73
- info\_funcp\_8, 75
- info\_funcp\_9, 76
- info\_funcp\_exvector, 87
- interpolate, 218
- inverse, 208, 247
- iquo, 243, 244
- irem, 242, 243
- is\_a, 91, 121, 265
- is\_cinteger, 250
- is\_clifford\_tinfo, 104
- is\_color\_tinfo, 108
- is\_crational, 250
- is\_dirac\_slash, 94
- is\_discriminant\_of\_quadratic\_number\_field, 197
- is\_dummy\_pair, 129
- is\_even, 249
- is\_exactly\_a, 92, 122
- is\_integer, 248

- is\_negative, 248
- is\_nonneg\_integer, 248
- is\_odd, 249
- is\_order\_function, 158
- is\_polynomial, 114
- is\_pos\_integer, 248
- is\_positive, 248
- is\_prime, 249
- is\_rational, 249
- is\_real, 249
- is\_terminating, 266
- is\_the\_function, 128
- is\_the\_function< G\_SERIAL >, 157
- is\_the\_function< iterated\_integral\_SERIAL >, 158
- is\_the\_function< psi\_SERIAL >, 158
- is\_the\_function< zeta\_SERIAL >, 156
- is\_zero, 120, 247
- isqrt, 245
- iterated\_integral, 158
- iterated\_integral2\_eval, 162
- iterated\_integral2\_evalf, 161
- iterated\_integral3\_eval, 162
- iterated\_integral3\_evalf, 162
- iterated\_integral\_evalf\_impl, 161
- kronecker\_symbol, 197
- latex, 261
- lcm, 222, 244
- lcm\_of\_coefficients\_denominators, 213
- lcmcoeff, 213
- ldegree, 114
- lgamma, 237, 238
- lgamma\_conjugate, 163
- lgamma\_deriv, 163
- lgamma\_eval, 162
- lgamma\_evalf, 162
- lgamma\_series, 163
- lhs, 120
- Li2, 237
- Li2\_, 236
- Li2\_conjugate, 149
- Li2\_deriv, 149
- Li2\_eval, 149
- Li2\_evalf, 149
- Li2\_projection, 236
- Li2\_series, 149, 236
- Li3\_eval, 150
- Li\_deriv, 170
- Li\_eval, 169
- Li\_evalf, 169
- Li\_print\_latex, 170
- Li\_series, 169
- library\_initializer, 290
- link\_ex, 124
- log, 230
- log2, 282
- log\_conjugate, 178
- log\_deriv, 177
- log\_eval, 177
- log\_evalf, 176
- log\_expand, 178
- log\_imag\_part, 177
- log\_info, 178
- log\_real\_part, 177
- log\_series, 177
- lookup\_map, 88
- lorentz\_eps, 281
- lorentz\_g, 279
- lsolve, 155
- lst, 88
- lst\_to\_clifford, 100, 101
- lst\_to\_matrix, 205
- make\_hash\_seed, 128
- make\_real\_float, 227
- make\_return\_type\_t, 266
- map\_eval\_integ, 289
- map\_evalm, 288
- match, 118
- metric\_tensor, 278
- minimal\_dim, 130
- mod, 241
- multinomial\_coefficient, 282
- multiply\_lcm, 213
- my\_ios\_callback, 260
- my\_ios\_index, 259
- next\_print\_context\_id, 291
- no\_index\_dimensions, 263
- nops, 112, 206
- normal, 115
- not\_symmetric, 270
- number\_of\_type, 133
- numer, 115, 251
- numer\_denom, 115
- op, 120
- operator!=, 258
- operator<, 259
- operator<<, 90, 111, 112, 246, 261, 285–287
- operator<=, 259
- operator>, 259
- operator>>, 91, 261
- operator>=, 259
- operator\*, 253
- operator\*==, 254, 255
- operator+, 252, 253, 255, 256
- operator++, 256–258
- operator+=, 254, 255
- operator-, 252, 253, 256
- operator--, 256–258
- operator-=, 254, 255
- operator/, 253, 254
- operator/==, 254, 255
- operator==, 258
- Order\_conjugate, 154
- Order\_eval, 154
- Order\_expl\_derivative, 155
- Order\_imag\_part, 154
- Order\_real\_part, 154



- Order\_series, 154
- paramset, 60
- permutation\_sign, 283, 284
- permute\_free\_index\_to\_front, 106
- Pi, 289
- PiEvalf, 245
- pow, 246, 264
- power\_funcp, 61
- power\_funcp\_1, 63
- power\_funcp\_10, 78
- power\_funcp\_11, 79
- power\_funcp\_12, 81
- power\_funcp\_13, 83
- power\_funcp\_14, 85
- power\_funcp\_2, 64
- power\_funcp\_3, 66
- power\_funcp\_4, 67
- power\_funcp\_5, 69
- power\_funcp\_6, 71
- power\_funcp\_7, 72
- power\_funcp\_8, 74
- power\_funcp\_9, 76
- power\_funcp\_exvector, 87
- prem, 215
- primitive\_dirichlet\_character, 197
- print\_context\_class\_info, 89
- print\_func< print\_context >, 128
- print\_func< print\_dflt >, 93, 105, 277
- print\_funcp, 61
- print\_funcp\_1, 63
- print\_funcp\_10, 78
- print\_funcp\_11, 80
- print\_funcp\_12, 81
- print\_funcp\_13, 83
- print\_funcp\_14, 85
- print\_funcp\_2, 64
- print\_funcp\_3, 66
- print\_funcp\_4, 68
- print\_funcp\_5, 69
- print\_funcp\_6, 71
- print\_funcp\_7, 73
- print\_funcp\_8, 74
- print\_funcp\_9, 76
- print\_funcp\_exvector, 87
- print\_integer\_csrc, 228
- print\_operator, 267
- print\_real\_cl\_N, 230
- print\_real\_csrc, 229
- print\_real\_number, 228
- print\_sym\_pow, 263
- product\_to\_exvector, 134
- psi, 157, 238, 239
- psi1\_deriv, 166
- psi1\_eval, 166
- psi1\_evalf, 166
- psi1\_series, 167
- psi2\_deriv, 167
- psi2\_eval, 167
- psi2\_evalf, 167
- psi2\_series, 168
- python, 261
- python\_repr, 262
- quo, 214
- rank, 209
- read\_real\_float, 227
- read\_unsigned, 90
- real, 251
- real\_part, 113
- real\_part\_conjugate, 140
- real\_part\_eval, 139
- real\_part\_evalf, 139
- real\_part\_expl\_derivative, 140
- real\_part\_funcp, 60
- real\_part\_funcp\_1, 62
- real\_part\_funcp\_10, 77
- real\_part\_funcp\_11, 79
- real\_part\_funcp\_12, 80
- real\_part\_funcp\_13, 82
- real\_part\_funcp\_14, 84
- real\_part\_funcp\_2, 63
- real\_part\_funcp\_3, 65
- real\_part\_funcp\_4, 67
- real\_part\_funcp\_5, 68
- real\_part\_funcp\_6, 70
- real\_part\_funcp\_7, 72
- real\_part\_funcp\_8, 73
- real\_part\_funcp\_9, 75
- real\_part\_funcp\_exvector, 86
- real\_part\_imag\_part, 140
- real\_part\_print\_latex, 139
- real\_part\_real\_part, 140
- reduced\_matrix, 206
- reeval\_ncmul, 211
- REGISTER\_FUNCTION, 139, 140, 142, 144, 145, 147, 148, 150–153, 155, 160, 161, 163, 165, 166, 170, 171, 173, 176, 178, 180, 181, 183–187, 189, 190, 192–195
- registered\_class\_info, 89
- rem, 214
- remove\_dirac\_ONE, 99
- rename\_dummy\_indices, 133
- rename\_dummy\_indices\_uniquely, 135, 136
- replace\_with\_symbol, 224, 225
- reposition\_dummy\_indices, 133
- resultant, 226
- rhs, 120
- rotate\_left, 282
- rows, 207
- S\_deriv, 171
- S\_eval, 170
- S\_evalf, 170
- S\_print\_latex, 171
- S\_series, 171
- series, 117
- series\_funcp, 61
- series\_funcp\_1, 63

series\_funcp\_10, 78  
 series\_funcp\_11, 79  
 series\_funcp\_12, 81  
 series\_funcp\_13, 83  
 series\_funcp\_14, 85  
 series\_funcp\_2, 64  
 series\_funcp\_3, 66  
 series\_funcp\_4, 67  
 series\_funcp\_5, 69  
 series\_funcp\_6, 71  
 series\_funcp\_7, 73  
 series\_funcp\_8, 74  
 series\_funcp\_9, 76  
 series\_funcp\_exvector, 87  
 series\_to\_poly, 265  
 set\_print\_context, 260  
 set\_print\_func, 266  
 set\_print\_options, 260  
 shaker\_sort, 284  
 simplify\_indexed, 118, 134  
 simplify\_indexed\_product, 134  
 sin, 231  
 sin\_conjugate, 179  
 sin\_deriv, 179  
 sin\_eval, 179  
 sin\_evalf, 178  
 sin\_imag\_part, 179  
 sin\_real\_part, 179  
 sinh, 234  
 sinh\_conjugate, 189  
 sinh\_deriv, 188  
 sinh\_eval, 188  
 sinh\_evalf, 188  
 sinh\_imag\_part, 188  
 sinh\_real\_part, 188  
 smod, 242  
 spinor\_metric, 279  
 spmap, 88  
 sprem, 216  
 sqrfree, 223  
 sqrfree\_parfrac, 224  
 sqrfree\_yun, 222  
 sqrt, 245, 265  
 sr\_gcd, 218  
 step, 247  
 step\_conjugate, 145  
 step\_eval, 144  
 step\_evalf, 144  
 step\_imag\_part, 145  
 step\_real\_part, 145  
 step\_series, 145  
 sub\_matrix, 206  
 subs, 121  
 subsvalue, 195  
 swap, 120, 125  
 sy\_anti, 274, 275  
 sy\_cycl, 275, 276  
 sy\_none, 273  
 sy\_symm, 274  
 sym\_desc\_vec, 88  
 symbolic\_matrix, 205, 209  
 symm, 272  
 symmetric2, 270  
 symmetric3, 270  
 symmetric4, 270  
 symmetrize, 118, 119, 272, 276  
 symmetrize\_cyclic, 119, 272, 276  
 synthesize\_func, 58  
 tan, 232  
 tan\_conjugate, 183  
 tan\_deriv, 182  
 tan\_eval, 182  
 tan\_evalf, 181  
 tan\_imag\_part, 182  
 tan\_real\_part, 182  
 tan\_series, 182  
 tanh, 234  
 tanh\_conjugate, 192  
 tanh\_deriv, 191  
 tanh\_eval, 191  
 tanh\_evalf, 190  
 tanh\_imag\_part, 191  
 tanh\_real\_part, 191  
 tanh\_series, 191  
 tensor, 289  
 tgamma, 238  
 tgamma\_conjugate, 164  
 tgamma\_deriv, 164  
 tgamma\_eval, 164  
 tgamma\_evalf, 163  
 tgamma\_series, 164  
 to\_double, 250  
 to\_int, 250  
 to\_long, 250  
 to\_polynomial, 116  
 to\_rational, 116  
 trace, 208  
 trace\_string, 97  
 transpose, 207  
 tree, 262  
 trig\_info, 180  
 tryfactsubs, 210  
 uintvector, 88  
 unarch\_table\_instance, 288  
 unarchive\_map\_t, 58  
 unit\_matrix, 205, 209  
 unlink\_ex, 125  
 unsignedvector, 88  
 version\_major, 292  
 version\_micro, 292  
 version\_minor, 292  
 wild, 288  
 write\_real\_float, 228  
 write\_unsigned, 90  
 zeta, 156, 237  
 zeta1\_deriv, 173

- zeta1\_eval, 173
- zeta1\_evalf, 173
- zeta1\_print\_latex, 173
- zeta2\_deriv, 174
- zeta2\_eval, 174
- zeta2\_evalf, 174
- zeta2\_print\_latex, 174
- zetaderiv\_deriv, 150
- zetaderiv\_eval, 150
- ginac.h, 1348
- GiNaC::numeric\_digits, 313
  - \_numeric\_digits, 314
  - add\_callback, 315
  - callbacklist, 315
  - digits, 315
  - operator long, 314
  - operator=, 314
  - print, 315
  - too\_late, 315
- GiNaC::add, 316
  - add, 323, 324
  - coeff, 326
  - combine\_ex\_with\_coeff\_to\_pair, 331
  - combine\_pair\_with\_coeff\_to\_pair, 331
  - conjugate, 328
  - degree, 325
  - derivative, 329
  - do\_print, 332
  - do\_print\_csrc, 333
  - do\_print\_latex, 333
  - do\_print\_python\_repr, 333
  - eval, 326
  - eval\_ncmul, 329
  - evalm, 326
  - expand, 332
  - get\_free\_indices, 329
  - imag\_part, 329
  - info, 325
  - integer\_content, 327
  - is\_polynomial, 325
  - ldegree, 326
  - max\_coefficient, 328
  - mul, 333
  - normal, 327
  - power, 333
  - precedence, 325
  - print\_add, 332
  - real\_part, 329
  - recombine\_pair\_to\_ex, 331
  - return\_type, 330
  - return\_type\_tinfo, 330
  - series, 327
  - smod, 328
  - split\_ex\_to\_pair, 331
  - thisexpairseq, 330
- GiNaC::archive, 334
  - ~archive, 336
  - add\_node, 338
  - archive, 336
  - archive\_ex, 336
  - atomize, 339
  - atoms, 341
  - clear, 338
  - exprs, 341
  - exprtable, 341
  - forget, 339
  - get\_node, 339
  - get\_top\_node, 338
  - inv\_at\_cit, 335
  - inverse\_atoms, 341
  - nodes, 341
  - num\_expressions, 338
  - operator<<, 340
  - operator>>, 340
  - printraw, 339
  - unarchive\_ex, 337
  - unatomize, 340
- GiNaC::archive::archived\_ex, 354
  - archived\_ex, 354
  - name, 355
  - root, 355
- GiNaC::archive\_node, 342
  - a, 351
  - add\_bool, 345
  - add\_ex, 346
  - add\_string, 346
  - add\_unsigned, 346
  - archive\_node, 345
  - archive\_node\_cit, 344
  - e, 352
  - find\_bool, 346
  - find\_ex, 348
  - find\_ex\_by\_loc, 348
  - find\_ex\_node, 349
  - find\_first, 347
  - find\_last, 348
  - find\_property\_range, 348
  - find\_string, 347
  - find\_unsigned, 347
  - forget, 350
  - get\_ex, 350
  - get\_properties, 349
  - has\_ex, 350
  - has\_expression, 351
  - has\_same\_ex\_as, 349
  - operator<<, 351
  - operator>>, 351
  - operator=, 345
  - printraw, 350
  - property\_type, 344
  - propinfovector, 344
  - props, 351
  - PTYPE\_BOOL, 345
  - PTYPE\_NODE, 345
  - PTYPE\_STRING, 345
  - PTYPE\_UNSIGNED, 345

- unarchive, [349](#)
- GiNaC::archive\_node::archive\_node\_cit\_range, [352](#)
- begin, [353](#)
- end, [353](#)
- GiNaC::archive\_node::property, [1063](#)
- name, [1064](#)
- property, [1063](#)
- type, [1063](#)
- value, [1064](#)
- GiNaC::archive\_node::property\_info, [1064](#)
- count, [1065](#)
- name, [1065](#)
- property\_info, [1065](#)
- type, [1065](#)
- GiNaC::basic, [355](#)
- ~basic, [360](#)
- accept, [367](#)
- add\_indexed, [373](#)
- archive, [377](#)
- basic, [360](#)
- calchash, [376](#)
- clearflag, [381](#)
- coeff, [368](#)
- collect, [369](#)
- compare, [379](#)
- compare\_same\_type, [375](#)
- conjugate, [375](#)
- contract\_with, [374](#)
- dbgprint, [363](#)
- dbgprinttree, [363](#)
- degree, [368](#)
- derivative, [370](#)
- diff, [378](#)
- do\_print, [381](#)
- do\_print\_python\_repr, [382](#)
- do\_print\_tree, [382](#)
- duplicate, [361](#)
- ensure\_if\_modifiable, [381](#)
- eval, [361](#)
- eval\_indexed, [362](#)
- eval\_integ, [362](#)
- eval\_ncmul, [362](#)
- evalf, [361](#)
- evalm, [361](#)
- ex, [382](#)
- expand, [369](#)
- flags, [382](#)
- get\_free\_indices, [372](#)
- gethash, [380](#)
- has, [366](#)
- hashvalue, [383](#)
- hold, [379](#)
- imag\_part, [375](#)
- info, [363](#)
- integer\_content, [371](#)
- is\_equal, [379](#)
- is\_equal\_same\_type, [376](#)
- is\_polynomial, [368](#)
- ldegree, [368](#)
- let\_op, [365](#)
- map, [367](#)
- match, [366](#)
- match\_same\_type, [366](#)
- max\_coefficient, [372](#)
- nops, [364](#)
- normal, [370](#)
- op, [364](#)
- operator=, [360](#)
- operator[], [364](#), [365](#)
- precedence, [363](#)
- print, [362](#)
- print\_dispatch, [376](#), [377](#)
- read\_archive, [377](#)
- real\_part, [375](#)
- return\_type, [374](#)
- return\_type\_tinfo, [375](#)
- scalar\_mul\_indexed, [373](#)
- series, [370](#)
- setflag, [380](#)
- smod, [372](#)
- subs, [367](#)
- subs\_one\_level, [378](#)
- to\_polynomial, [371](#)
- to\_rational, [371](#)
- GiNaC::basic\_log\_kernel, [383](#)
- do\_print, [389](#)
- series\_coeff\_impl, [389](#)
- GiNaC::basic\_multi\_iterator< T >, [389](#)
- ~basic\_multi\_iterator, [392](#)
- B, [394](#)
- basic\_multi\_iterator, [391](#)
- flag\_overflow, [395](#)
- get\_vector, [392](#)
- init, [393](#)
- N, [394](#)
- operator<<, [394](#)
- operator(), [393](#)
- operator++, [394](#)
- operator[], [393](#)
- overflow, [392](#)
- size, [392](#)
- v, [395](#)
- GiNaC::basic\_partition\_generator, [395](#)
- basic\_partition\_generator, [396](#)
- mpgen, [397](#)
- GiNaC::basic\_partition\_generator::mpartition2, [864](#)
- m, [865](#)
- mpartition2, [864](#)
- n, [865](#)
- next\_partition, [865](#)
- x, [865](#)
- GiNaC::class\_info< OPT >, [397](#)
- class\_info, [398](#)
- dump\_hierarchy, [399](#)
- dump\_tree, [399](#)
- find, [399](#)

- first, 400
- get\_parent, 398
- identify\_parents, 399
- next, 400
- options, 400
- parent, 400
- parents\_identified, 400
- GiNaC::class\_info< OPT >::tree\_node, 1264
  - add\_child, 1265
  - children, 1265
  - info, 1265
  - tree\_node, 1264
- GiNaC::clifford, 401
  - archive, 411
  - clifford, 410, 411
  - commutator\_sign, 416
  - do\_print\_dflt, 415
  - do\_print\_latex, 415
  - do\_print\_tree, 416
  - eval\_ncmul, 412
  - get\_commutator\_sign, 414
  - get\_metric, 413
  - get\_representation\_label, 413
  - let\_op, 415
  - match\_same\_type, 412
  - metric, 416
  - nops, 414
  - op, 414
  - precedence, 411
  - read\_archive, 411
  - representation\_label, 416
  - return\_type, 413
  - return\_type\_tinfo, 413
  - same\_metric, 414
  - subs, 415
  - thiscontainer, 412, 413
- GiNaC::cliffordunit, 417
  - contract\_with, 421
  - do\_print, 422
  - do\_print\_latex, 422
- GiNaC::color, 422
  - archive, 432
  - color, 431, 432
  - eval\_ncmul, 433
  - get\_representation\_label, 435
  - match\_same\_type, 433
  - read\_archive, 433
  - representation\_label, 435
  - return\_type, 434
  - return\_type\_tinfo, 434
  - thiscontainer, 434
- GiNaC::compare\_all\_equal< T >, 435
  - ~compare\_all\_equal, 436
  - struct\_compare, 436
  - struct\_is\_equal, 436
- GiNaC::compare\_bitwise< T >, 437
  - ~compare\_bitwise, 437
  - struct\_compare, 437
  - struct\_is\_equal, 437
- GiNaC::compare\_std\_less< T >, 438
  - ~compare\_std\_less, 438
  - struct\_compare, 439
  - struct\_is\_equal, 438
- GiNaC::composition\_generator, 439
  - atend, 441
  - cmgen, 441
  - composition, 441
  - composition\_generator, 440
  - current\_updated, 441
  - get, 440
  - next, 440
  - trivial, 441
- GiNaC::composition\_generator::coolmulti, 489
  - ~coolmulti, 489
  - after\_i, 490
  - coolmulti, 489
  - finished, 490
  - head, 490
  - i, 490
  - next\_permutation, 490
- GiNaC::composition\_generator::coolmulti::element, 552
  - ~element, 553
  - element, 553
  - next, 553
  - value, 553
- GiNaC::const\_iterator, 442
  - const\_iterator, 444
  - const\_postorder\_iterator, 448
  - const\_preorder\_iterator, 447
  - difference\_type, 444
  - e, 448
  - ex, 447
  - i, 448
  - iterator\_category, 443
  - operator!=, 446
  - operator<, 447
  - operator<=, 447
  - operator>, 447
  - operator>=, 447
  - operator\*, 444
  - operator+, 445, 448
  - operator++, 445
  - operator+=, 445
  - operator-, 446, 448
  - operator->, 445
  - operator--, 446
  - operator-=, 446
  - operator==, 446
  - operator[], 445
  - pointer, 444
  - reference, 444
  - value\_type, 443
- GiNaC::const\_postorder\_iterator, 449
  - const\_postorder\_iterator, 450
  - descend, 451
  - difference\_type, 449

- increment, [451](#)
- iterator\_category, [449](#)
- operator!=, [451](#)
- operator\*, [450](#)
- operator++, [451](#)
- operator->, [450](#)
- operator==, [451](#)
- pointer, [450](#)
- reference, [450](#)
- s, [452](#)
- value\_type, [449](#)
- GiNaC::const\_preorder\_iterator, [452](#)
- const\_preorder\_iterator, [453](#), [454](#)
- difference\_type, [453](#)
- increment, [455](#)
- iterator\_category, [453](#)
- operator!=, [455](#)
- operator\*, [454](#)
- operator++, [454](#)
- operator->, [454](#)
- operator==, [454](#)
- pointer, [453](#)
- reference, [453](#)
- s, [455](#)
- value\_type, [453](#)
- GiNaC::constant, [456](#)
- archive, [463](#)
- calchash, [464](#)
- conjugate, [463](#)
- constant, [461](#)
- derivative, [464](#)
- do\_print, [465](#)
- do\_print\_latex, [465](#)
- do\_print\_python\_repr, [465](#)
- do\_print\_tree, [465](#)
- domain, [467](#)
- ef, [466](#)
- evalf, [462](#)
- imag\_part, [463](#)
- info, [462](#)
- is\_equal\_same\_type, [464](#)
- is\_polynomial, [462](#)
- name, [466](#)
- next\_serial, [466](#)
- number, [466](#)
- read\_archive, [463](#)
- real\_part, [463](#)
- serial, [466](#)
- TeX\_name, [466](#)
- GiNaC::container< C >, [467](#)
- append, [481](#)
- archive, [478](#)
- begin, [482](#)
- conjugate, [478](#)
- const\_iterator, [474](#)
- const\_reverse\_iterator, [474](#)
- container, [474](#), [475](#)
- do\_print, [483](#)
- do\_print\_python, [483](#)
- do\_print\_python\_repr, [484](#)
- do\_print\_tree, [483](#)
- end, [482](#)
- get\_close\_delim, [475](#)
- get\_default\_flags, [475](#)
- get\_open\_delim, [475](#)
- imag\_part, [479](#)
- info, [475](#)
- is\_equal\_same\_type, [479](#)
- let\_op, [477](#)
- nops, [476](#)
- op, [476](#)
- precedence, [476](#)
- prepend, [481](#)
- printseq, [480](#)
- rbegin, [483](#)
- read\_archive, [477](#)
- real\_part, [478](#)
- remove\_all, [482](#)
- remove\_first, [481](#)
- remove\_last, [481](#)
- rend, [483](#)
- sort, [482](#)
- sort\_, [480](#)
- STLT, [473](#)
- subs, [477](#)
- subchildren, [484](#)
- thiscontainer, [479](#), [480](#)
- unique, [482](#)
- unique\_, [481](#), [484](#)
- GiNaC::container\_storage< C >, [485](#)
- ~container\_storage, [487](#)
- container\_storage, [486](#)
- reserve, [487](#)
- seq, [488](#)
- STLT, [486](#)
- GiNaC::derivative\_map\_function, [491](#)
- derivative\_map\_function, [492](#)
- operator(), [492](#)
- s, [493](#)
- GiNaC::determinant\_algo, [493](#)
- automatic, [494](#)
- bareiss, [494](#)
- divfree, [494](#)
- gauss, [494](#)
- laplace, [494](#)
- GiNaC::diracgamma, [494](#)
- contract\_with, [500](#)
- do\_print, [500](#)
- do\_print\_latex, [500](#)
- GiNaC::diracgamma5, [501](#)
- conjugate, [505](#)
- do\_print, [505](#)
- do\_print\_latex, [505](#)
- GiNaC::diracgammaL, [506](#)
- conjugate, [510](#)
- do\_print, [511](#)

- do\_print\_latex, 511
- GiNaC::diracgammaR, 511
  - conjugate, 515
  - do\_print, 516
  - do\_print\_latex, 516
- GiNaC::diracone, 516
  - do\_print, 520
  - do\_print\_latex, 520
- GiNaC::do\_taylor, 521
- GiNaC::domain, 521
  - complex, 522
  - positive, 522
  - real, 522
- GiNaC::dunno, 522
- GiNaC::Ebar\_kernel, 522
  - do\_print, 529
  - Ebar\_kernel, 528
  - get\_numerical\_value, 529
  - is\_numeric, 528
  - let\_op, 528
  - m, 530
  - n, 530
  - nops, 528
  - op, 528
  - series\_coeff\_impl, 529
  - x, 530
  - y, 530
- GiNaC::Eisenstein\_h\_kernel, 531
  - C\_norm, 541
  - coefficient\_a0, 539
  - coefficient\_an, 539
  - do\_print, 540
  - Eisenstein\_h\_kernel, 537
  - get\_numerical\_value, 539
  - is\_numeric, 538
  - k, 540
  - Laurent\_series, 538
  - let\_op, 538
  - N, 540
  - nops, 537
  - op, 538
  - q\_expansion\_modular\_form, 540
  - r, 541
  - s, 541
  - series, 537
  - uses\_Laurent\_series, 539
- GiNaC::Eisenstein\_kernel, 542
  - a, 551
  - b, 551
  - C\_norm, 552
  - do\_print, 551
  - Eisenstein\_kernel, 548
  - get\_numerical\_value, 550
  - is\_numeric, 549
  - K, 552
  - k, 551
  - Laurent\_series, 550
  - let\_op, 549
  - N, 551
  - nops, 549
  - op, 549
  - q\_expansion\_modular\_form, 550
  - series, 548
  - uses\_Laurent\_series, 550
- GiNaC::ELi\_kernel, 554
  - do\_print, 561
  - ELi\_kernel, 560
  - get\_numerical\_value, 561
  - is\_numeric, 560
  - let\_op, 560
  - m, 562
  - n, 562
  - nops, 560
  - op, 560
  - series\_coeff\_impl, 561
  - x, 562
  - y, 562
- GiNaC::error\_and\_integral, 564
  - error, 565
  - error\_and\_integral, 564
  - integral, 565
- GiNaC::error\_and\_integral\_is\_less, 565
  - operator(), 565
- GiNaC::eval\_integ\_map\_function, 566
  - operator(), 567
- GiNaC::evalf\_map\_function, 567
  - operator(), 568
- GiNaC::evalm\_map\_function, 569
  - operator(), 570
- GiNaC::ex, 570
  - accept, 585
  - antisymmetrize, 598, 599
  - archive\_node, 603
  - are\_ex\_trivially\_equal, 603
  - begin, 576
  - bp, 604
  - coeff, 587
  - collect, 588
  - compare, 597
  - conjugate, 582
  - construct\_from\_basic, 600
  - construct\_from\_double, 602
  - construct\_from\_int, 600
  - construct\_from\_long, 601
  - construct\_from\_longlong, 601
  - construct\_from\_string\_and\_lst, 602
  - construct\_from\_uint, 601
  - construct\_from\_ulong, 601
  - construct\_from\_ulonglong, 602
  - content, 592
  - dbgprint, 579
  - dbgprinttree, 579
  - degree, 587
  - denom, 591
  - diff, 588
  - end, 576

- eval, 577
- eval\_integ, 578
- eval\_ncmul, 578
- evalf, 577
- evalm, 578
- ex, 574–576
- ex\_to, 603
- expand, 588
- find, 583
- get\_free\_indices, 595
- gethash, 600
- has, 583
- imag\_part, 583
- info, 579
- integer\_content, 593
- is\_a, 604
- is\_equal, 597
- is\_exactly\_a, 604
- is\_polynomial, 586
- is\_zero, 597
- is\_zero\_matrix, 598
- lcoeff, 587
- ldegree, 587
- let\_op, 581
- lhs, 582
- makewritable, 602
- map, 585
- match, 583, 584
- max\_coefficient, 595
- nops, 580
- normal, 589
- numer, 591
- numer\_denom, 591
- op, 580
- operator[], 581, 582
- postorder\_begin, 577
- postorder\_end, 577
- preorder\_begin, 577
- preorder\_end, 577
- primpart, 593, 594
- print, 578
- real\_part, 582
- return\_type, 599
- return\_type\_tinfo, 600
- rhs, 582
- series, 589
- share, 602
- simplify\_indexed, 596
- smod, 595
- subs, 584, 585
- swap, 576
- symmetrize, 598
- symmetrize\_cyclic, 599
- tcoeff, 588
- to\_polynomial, 590
- to\_rational, 590
- traverse, 586
- traverse\_postorder, 586
- traverse\_preorder, 586
- unit, 592
- unitcontprim, 594
- GiNaC::ex\_base\_is\_less, 605
  - operator(), 605
- GiNaC::ex\_is\_equal, 605
  - operator(), 605
- GiNaC::ex\_is\_less, 606
  - operator(), 606
- GiNaC::ex\_swap, 606
  - operator(), 606
- GiNaC::expair, 607
  - coeff, 610
  - compare, 609
  - conjugate, 609
  - expair, 608
  - is\_canonical\_numeric, 609
  - is\_equal, 608
  - is\_less, 608
  - print, 609
  - rest, 610
  - swap, 609
- GiNaC::expair\_is\_less, 610
  - operator(), 611
- GiNaC::expair\_rest\_is\_less, 611
  - operator(), 611
- GiNaC::expair\_swap, 612
  - operator(), 612
- GiNaC::expairseq, 613
  - archive, 623
  - calchash, 625
  - can\_make\_flat, 629
  - canonicalize, 631
  - combine\_ex\_with\_coeff\_to\_pair, 627
  - combine\_overall\_coeff, 628
  - combine\_pair\_with\_coeff\_to\_pair, 627
  - combine\_same\_terms\_sorted\_seq, 631
  - conjugate, 623
  - construct\_from\_2\_ex, 629
  - construct\_from\_2\_expairseq, 629
  - construct\_from\_epvector, 630
  - construct\_from\_expairseq\_ex, 630
  - construct\_from\_exvector, 630
  - default\_overall\_coeff, 628
  - do\_print, 629
  - do\_print\_tree, 629
  - eval, 621
  - evalchildren, 632
  - expair\_needs\_further\_processing, 628
  - expairseq, 619, 620
  - expand, 625
  - expandchildren, 632
  - info, 620
  - is\_canonical, 632
  - is\_equal\_same\_type, 624
  - make\_flat, 631
  - map, 621
  - match, 622



- nops, 620
- op, 621
- overall\_coeff, 634
- precedence, 620
- printpair, 626
- printseq, 626
- read\_archive, 624
- recombine\_pair\_to\_ex, 627
- return\_type, 624
- seq, 633
- split\_ex\_to\_pair, 626
- subs, 623
- subchildren, 633
- thisexpairseq, 625, 626
- to\_polynomial, 622
- to\_rational, 622
- GiNaC::expand\_map\_function, 634
  - expand\_map\_function, 636
  - operator(), 636
  - options, 636
- GiNaC::expand\_options, 636
  - expand\_function\_args, 637
  - expand\_indexed, 637
  - expand\_rename\_idx, 637
  - expand\_transcendental, 637
- GiNaC::factor\_options, 637
  - all, 638
  - polynomial, 638
- GiNaC::fail, 638
  - do\_print, 642
  - return\_type, 642
- GiNaC::fderivative, 643
  - archive, 654
  - derivative, 655
  - derivatives, 656
  - do\_print, 656
  - do\_print\_csrc, 657
  - do\_print\_latex, 657
  - do\_print\_tree, 657
  - eval, 653
  - fderivative, 652, 653
  - is\_equal\_same\_type, 655
  - match\_same\_type, 656
  - parameter\_set, 657
  - print, 653
  - read\_archive, 655
  - series, 654
  - thiscontainer, 654
- GiNaC::function, 658
  - archive, 675
  - calchash, 673
  - conjugate, 674
  - current\_serial, 680
  - derivative, 675
  - eval, 672
  - eval\_ncmul, 672
  - evalf, 672
  - expand, 672
  - expl\_derivative, 677
  - find\_function, 678
  - function, 666–671
  - get\_name, 679
  - get\_registered\_functions, 679
  - get\_serial, 679
  - imag\_part, 674
  - info, 675
  - is\_equal\_same\_type, 676
  - lookup\_remember\_table, 678
  - match\_same\_type, 676
  - pderivative, 677
  - power, 678
  - precedence, 671
  - print, 671
  - read\_archive, 675
  - real\_part, 674
  - register\_new, 678
  - registered\_functions, 677
  - remember\_table\_entry, 679
  - return\_type, 676
  - return\_type\_tinfo, 677
  - serial, 680
  - series, 673
  - store\_remember\_table, 678
  - thiscontainer, 673, 674
- GiNaC::function\_options, 680
  - ~function\_options, 687
  - conjugate\_f, 721
  - conjugate\_func, 692–694, 714
  - conjugate\_use\_exvector\_args, 725
  - derivative\_f, 722
  - derivative\_func, 702–704, 714
  - derivative\_use\_exvector\_args, 725
  - do\_not\_evalf\_params, 718
  - dummy, 687
  - eval\_f, 721
  - eval\_func, 688–690, 713
  - eval\_use\_exvector\_args, 724
  - evalf\_f, 721
  - evalf\_func, 690–692, 713
  - evalf\_params\_first, 723
  - evalf\_use\_exvector\_args, 725
  - expand\_f, 722
  - expand\_func, 699–701, 714
  - expand\_use\_exvector\_args, 725
  - expl\_derivative\_f, 722
  - expl\_derivative\_func, 704–706, 714
  - expl\_derivative\_use\_exvector\_args, 726
  - fderivative, 720
  - function, 720
  - function\_options, 686
  - functions\_with\_same\_name, 726
  - get\_name, 719
  - get\_nparams, 719
  - has\_derivative, 719
  - has\_power, 720
  - imag\_part\_f, 722

- imag\_part\_func, 697–699, 714
- imag\_part\_use\_exvector\_args, 725
- info\_f, 723
- info\_func, 711–713, 715
- info\_use\_exvector\_args, 726
- initialize, 687
- latex\_name, 687
- name, 721
- nparams, 721
- overloaded, 719
- power\_f, 722
- power\_func, 706–708, 715
- power\_use\_exvector\_args, 726
- print\_dispatch\_table, 723
- print\_func, 715–718
- print\_use\_exvector\_args, 726
- real\_part\_f, 722
- real\_part\_func, 695–697, 714
- real\_part\_use\_exvector\_args, 725
- remember, 719
- remember\_assoc\_size, 724
- remember\_size, 724
- remember\_strategy, 724
- return\_type, 723
- return\_type\_tinfo, 724
- series\_f, 723
- series\_func, 709–711, 715
- series\_use\_exvector\_args, 726
- set\_name, 687
- set\_print\_func, 720
- set\_return\_type, 718
- set\_symmetry, 719
- symtree, 727
- test\_and\_set\_nparams, 720
- TeX\_name, 721
- use\_remember, 724
- use\_return\_type, 723
- GiNaC::G2\_SERIAL, 727
  - serial, 727
- GiNaC::G3\_SERIAL, 728
  - serial, 728
- GiNaC::gcd\_options, 728
  - no\_heur\_gcd, 729
  - no\_part\_factored, 729
  - use\_sr\_gcd, 729
- GiNaC::gcdheu\_failed, 729
- GiNaC::has\_distance < T >, 729
  - no\_type, 730
  - test, 731
  - value, 730
  - yes\_type, 730
- GiNaC::has\_options, 731
  - algebraic, 732
- GiNaC::idx, 733
  - archive, 740
  - calchash, 741
  - derivative, 741
  - dim, 745
  - do\_print, 744
  - do\_print\_csrc, 744
  - do\_print\_latex, 745
  - do\_print\_tree, 745
  - evalf, 740
  - get\_dim, 743
  - get\_value, 742
  - idx, 738
  - info, 739
  - is\_dim\_numeric, 743
  - is\_dim\_symbolic, 743
  - is\_dummy\_pair\_same\_type, 742
  - is\_numeric, 742
  - is\_symbolic, 742
  - map, 739
  - match\_same\_type, 741
  - minimal\_dim, 744
  - nops, 739
  - op, 739
  - print\_index, 744
  - read\_archive, 740
  - replace\_dim, 743
  - subs, 740
  - value, 745
- GiNaC::idx\_is\_equal\_ignore\_dim, 746
  - operator(), 746
- GiNaC::indexed, 747
  - all\_index\_values\_are, 765
  - archive, 763
  - derivative, 763
  - do\_print, 767
  - do\_print\_latex, 767
  - do\_print\_tree, 767
  - eval, 762
  - expand, 765
  - get\_dummy\_indices, 765, 766
  - get\_free\_indices, 763
  - get\_indices, 765
  - get\_symmetry, 766
  - has\_dummy\_index\_for, 766
  - imag\_part, 762
  - indexed, 755–757, 759–761
  - info, 762
  - precedence, 761
  - print\_indexed, 767
  - printindices, 766
  - read\_archive, 763
  - real\_part, 762
  - reposition\_dummy\_indices, 768
  - return\_type, 764
  - return\_type\_tinfo, 764
  - simplify\_indexed, 768
  - simplify\_indexed\_product, 768
  - symtree, 769
  - thiscontainer, 764
  - validate, 767
- GiNaC::info\_flags, 769
  - cinteger, 770

- cinteger\_polynomial, 770
- crational, 770
- crational\_polynomial, 770
- even, 770
- expanded, 770
- exprseq, 770
- has\_indices, 770
- idx, 770
- indefinite, 770
- indexed, 770
- integer, 770
- integer\_polynomial, 770
- list, 770
- negative, 770
- negint, 770
- nonnegative, 770
- nonnegint, 770
- numeric, 770
- odd, 770
- polynomial, 770
- posint, 770
- positive, 770
- prime, 770
- rational, 770
- rational\_function, 770
- rational\_polynomial, 770
- real, 770
- relation, 770
- relation\_equal, 770
- relation\_greater, 770
- relation\_greater\_or\_equal, 770
- relation\_less, 770
- relation\_less\_or\_equal, 770
- relation\_not\_equal, 770
- symbol, 770
- GiNaC::integral, 771
  - a, 782
  - archive, 780
  - b, 783
  - conjugate, 780
  - degree, 777
  - derivative, 781
  - do\_print, 781
  - do\_print\_latex, 782
  - eval, 777
  - eval\_integ, 780
  - eval\_ncmul, 778
  - evalf, 777
  - expand, 779
  - f, 783
  - get\_free\_indices, 779
  - integral, 776
  - ldegree, 778
  - let\_op, 779
  - max\_integration\_level, 782
  - nops, 778
  - op, 778
  - precedence, 777
  - read\_archive, 780
  - relative\_integration\_error, 782
  - return\_type, 779
  - return\_type\_tinfo, 780
  - series, 781
  - x, 782
- GiNaC::integration\_kernel, 783
  - cache\_step\_size, 792
  - do\_print, 792
  - get\_cache\_size, 790
  - get\_numerical\_value, 790
  - get\_numerical\_value\_impl, 791
  - get\_series\_coeff, 791
  - has\_trailing\_zero, 789
  - is\_numeric, 789
  - Laurent\_series, 789
  - series, 788
  - series\_coeff, 791
  - series\_coeff\_impl, 790
  - series\_vec, 792
  - set\_cache\_step, 791
  - uses\_Laurent\_series, 790
- GiNaC::internal, 310
- GiNaC::internal::iter\_rep, 311
  - \_iter\_rep, 312
  - e, 312
  - i, 313
  - i\_end, 313
  - operator!=, 312
  - operator==, 312
- GiNaC::is\_not\_a\_clifford, 792
  - operator(), 793
- GiNaC::is\_summation\_idx, 793
  - operator(), 793
- GiNaC::iterated\_integral2\_SERIAL, 794
  - serial, 794
- GiNaC::iterated\_integral3\_SERIAL, 794
  - serial, 795
- GiNaC::Kronecker\_dtau\_kernel, 795
  - C\_norm, 803
  - do\_print, 802
  - get\_numerical\_value, 802
  - is\_numeric, 801
  - K, 803
  - Kronecker\_dtau\_kernel, 801
  - let\_op, 801
  - n, 803
  - nops, 801
  - op, 801
  - series\_coeff\_impl, 802
  - z, 803
- GiNaC::Kronecker\_dz\_kernel, 804
  - C\_norm, 812
  - do\_print, 811
  - get\_numerical\_value, 811
  - is\_numeric, 810
  - K, 812
  - Kronecker\_dz\_kernel, 810

- let\_op, 810
- n, 812
- nops, 810
- op, 810
- series\_coeff\_impl, 811
- tau, 812
- z\_j, 812
- GiNaC::lanczos\_coeffs, 813
  - calc\_lanczos\_A, 814
  - coeffs, 814
  - current\_vector, 814
  - get\_order, 813
  - lanczos\_coeffs, 813
  - sufficiently\_accurate, 813
- GiNaC::library\_init, 815
  - ~library\_init, 816
  - count, 817
  - init\_unarchivers, 817
  - library\_init, 816
- GiNaC::make\_flat\_inserter, 817
  - combine\_indices, 819
  - do\_renaming, 819
  - handle\_factor, 818
  - make\_flat\_inserter, 818
  - used\_indices, 819
- GiNaC::map\_function, 820
  - ~map\_function, 821
  - argument\_type, 821
  - operator(), 821
  - result\_type, 821
- GiNaC::matrix, 822
  - add, 835
  - add\_indexed, 832
  - archive, 834
  - charpoly, 839
  - col, 846
  - cols, 835
  - conjugate, 833
  - contract\_with, 833
  - determinant, 838
  - determinant\_minor, 842
  - division\_free\_elimination, 843
  - do\_print, 845
  - do\_print\_latex, 845
  - do\_print\_python\_repr, 846
  - echelon\_form, 842
  - eval\_indexed, 832
  - evalm, 831
  - fraction\_free\_elimination, 844
  - gauss\_elimination, 843
  - imag\_part, 833
  - inverse, 840
  - is\_zero\_matrix, 842
  - let\_op, 831
  - m, 846
  - markowitz\_elimination, 844
  - match\_same\_type, 834
  - matrix, 828, 830
  - mul, 836
  - mul\_scalar, 836
  - nops, 831
  - op, 831
  - operator(), 837
  - pivot, 844
  - pow, 836
  - print\_elements, 845
  - rank, 841, 842
  - read\_archive, 834
  - real\_part, 833
  - return\_type, 834
  - row, 846
  - rows, 835
  - scalar\_mul\_indexed, 832
  - set, 838
  - solve, 841
  - sub, 835
  - subs, 832
  - trace, 839
  - transpose, 838
- GiNaC::minkmetric, 847
  - archive, 853
  - do\_print, 854
  - do\_print\_latex, 854
  - eval\_indexed, 853
  - info, 852
  - minkmetric, 852
  - pos\_sig, 854
  - read\_archive, 853
  - return\_type, 854
- GiNaC::modular\_form\_kernel, 855
  - C\_norm, 864
  - do\_print, 863
  - get\_numerical\_value, 862
  - is\_numeric, 862
  - k, 863
  - Laurent\_series, 862
  - let\_op, 862
  - modular\_form\_kernel, 861
  - nops, 861
  - op, 861
  - P, 864
  - q\_expansion\_modular\_form, 863
  - series, 861
  - uses\_Laurent\_series, 863
- GiNaC::mul, 866
  - add, 888
  - algebraic\_subs\_mul, 886
  - can\_be\_further\_expanded, 887
  - can\_make\_flat, 885
  - coeff, 877
  - combine\_ex\_with\_coeff\_to\_pair, 883
  - combine\_overall\_coeff, 885
  - combine\_pair\_with\_coeff\_to\_pair, 884
  - conjugate, 881
  - default\_overall\_coeff, 884
  - degree, 876

- derivative, 881
- do\_print, 886
- do\_print\_csrc, 887
- do\_print\_latex, 887
- do\_print\_python\_repr, 887
- eval, 877
- eval\_ncmul, 882
- evalf, 878
- evalm, 879
- expair\_needs\_further\_processing, 884
- expand, 885
- expandchildren, 887
- find\_real\_imag, 886
- get\_free\_indices, 881
- has, 877
- imag\_part, 878
- info, 876
- integer\_content, 880
- is\_polynomial, 876
- ldegree, 877
- max\_coefficient, 881
- mul, 874, 875
- ncmul, 888
- normal, 879
- power, 888
- precedence, 875
- print\_overall\_coeff, 886
- real\_part, 878
- recombine\_pair\_to\_ex, 884
- return\_type, 882
- return\_type\_tinfo, 882
- series, 879
- smod, 880
- split\_ex\_to\_pair, 883
- thisexpairseq, 882, 883
- GiNaC::multi\_iterator\_counter< T >, 889
  - init, 891
  - multi\_iterator\_counter, 891
  - operator<<, 892
  - operator++, 891
- GiNaC::multi\_iterator\_counter\_indv< T >, 892
  - init, 895
  - multi\_iterator\_counter\_indv, 894, 895
  - Nv, 896
  - operator<<, 896
  - operator++, 895
- GiNaC::multi\_iterator\_ordered< T >, 896
  - init, 899
  - multi\_iterator\_ordered, 898, 899
  - operator<<, 900
  - operator++, 899
- GiNaC::multi\_iterator\_ordered\_eq< T >, 900
  - init, 903
  - multi\_iterator\_ordered\_eq, 902
  - operator<<, 903
  - operator++, 903
- GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 904
  - init, 907
  - multi\_iterator\_ordered\_eq\_indv, 906
  - Nv, 907
  - operator<<, 907
  - operator++, 907
- GiNaC::multi\_iterator\_permutation< T >, 908
  - get\_sign, 911
  - init, 911
  - multi\_iterator\_permutation, 910
  - operator<<, 912
  - operator++, 911
- GiNaC::multi\_iterator\_shuffle< T >, 912
  - init, 915
  - multi\_iterator\_shuffle, 915
  - N\_internal, 916
  - operator<<, 916
  - operator++, 915
  - v\_internal, 916
  - v\_orig, 916
- GiNaC::multi\_iterator\_shuffle\_prime< T >, 917
  - init, 920
  - multi\_iterator\_shuffle\_prime, 920
  - operator<<, 920
- GiNaC::multiple\_polylog\_kernel, 921
  - do\_print, 928
  - is\_numeric, 927
  - let\_op, 927
  - multiple\_polylog\_kernel, 927
  - nops, 927
  - op, 927
  - series\_coeff\_impl, 928
  - z, 928
- GiNaC::ncmul, 929
  - append\_factors, 942
  - coeff, 938
  - conjugate, 940
  - count\_factors, 942
  - degree, 937
  - derivative, 940
  - do\_print, 941
  - do\_print\_csrc, 941
  - eval, 938
  - evalm, 939
  - expand, 938
  - expandchildren, 942
  - get\_factors, 942
  - get\_free\_indices, 939
  - hold\_ncmul, 943
  - imag\_part, 940
  - info, 937
  - ldegree, 938
  - ncmul, 936, 937
  - power, 943
  - precedence, 937
  - real\_part, 940
  - reeval\_ncmul, 943
  - return\_type, 941
  - return\_type\_tinfo, 941
  - thiscontainer, 939, 940

- GiNaC::normal\_map\_function, 943
  - operator(), 944
- GiNaC::numeric, 945
  - add, 961
  - add\_dyn, 963
  - archive, 960
  - calchash, 961
  - coeff, 956
  - compare, 966
  - conjugate, 959
  - csgn, 965
  - degree, 955
  - denom, 974
  - derivative, 960
  - div, 962
  - div\_dyn, 963
  - do\_print, 975
  - do\_print\_csrc, 975
  - do\_print\_csrc\_cl\_N, 975
  - do\_print\_latex, 975
  - do\_print\_python\_repr, 976
  - do\_print\_tree, 976
  - eval, 956
  - evalf, 956
  - has, 956
  - imag, 973
  - imag\_part, 959
  - info, 954
  - int\_length, 974
  - integer\_content, 958
  - inverse, 965
  - is\_cinteger, 969
  - is\_crational, 969
  - is\_equal, 966
  - is\_equal\_same\_type, 960
  - is\_even, 968
  - is\_integer, 967
  - is\_negative, 967
  - is\_nonneg\_integer, 968
  - is\_odd, 968
  - is\_polynomial, 955
  - is\_pos\_integer, 967
  - is\_positive, 967
  - is\_prime, 968
  - is\_rational, 969
  - is\_real, 969
  - is\_zero, 966
  - ldegree, 955
  - max\_coefficient, 959
  - mul, 962
  - mul\_dyn, 963
  - normal, 957
  - numer, 974
  - numeric, 952–954
  - operator!=, 970
  - operator<, 970
  - operator<=, 970
  - operator>, 972
  - operator>=, 972
  - operator=, 964, 965
  - operator==, 970
  - power, 962
  - power\_dyn, 964
  - precedence, 954
  - print\_numeric, 974
  - read\_archive, 960
  - real, 973
  - real\_part, 959
  - smod, 958
  - step, 965
  - sub, 961
  - sub\_dyn, 963
  - subs, 957
  - to\_cl\_N, 973
  - to\_double, 973
  - to\_int, 972
  - to\_long, 972
  - to\_polynomial, 958
  - to\_rational, 957
  - value, 976
- GiNaC::op0\_is\_equal, 977
  - operator(), 977
- GiNaC::partition\_generator, 977
  - current\_updated, 979
  - get, 979
  - next, 979
  - partition, 979
  - partition\_generator, 979
- GiNaC::partition\_with\_zero\_parts\_generator, 980
  - current\_updated, 982
  - get, 981
  - m, 982
  - next, 982
  - partition, 982
  - partition\_with\_zero\_parts\_generator, 981
- GiNaC::pointer\_to\_map\_function, 983
  - operator(), 984
  - pointer\_to\_map\_function, 984
  - ptr, 985
- GiNaC::pointer\_to\_map\_function\_1arg< T1 >, 985
  - arg1, 987
  - operator(), 987
  - pointer\_to\_map\_function\_1arg, 987
  - ptr, 987
- GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, 988
  - arg1, 990
  - arg2, 990
  - operator(), 989
  - pointer\_to\_map\_function\_2args, 989
  - ptr, 990
- GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, 990
  - arg1, 992
  - arg2, 993
  - arg3, 993

- operator(), [992](#)
- pointer\_to\_map\_function\_3args, [992](#)
- ptr, [992](#)
- GiNaC::pointer\_to\_member\_to\_map\_function< C >, [993](#)
- c, [995](#)
- operator(), [995](#)
- pointer\_to\_member\_to\_map\_function, [995](#)
- ptr, [995](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [996](#)
- arg1, [998](#)
- c, [998](#)
- operator(), [997](#)
- pointer\_to\_member\_to\_map\_function\_1arg, [997](#)
- ptr, [998](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [998](#)
- arg1, [1001](#)
- arg2, [1001](#)
- c, [1000](#)
- operator(), [1000](#)
- pointer\_to\_member\_to\_map\_function\_2args, [1000](#)
- ptr, [1000](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [1001](#)
- arg1, [1004](#)
- arg2, [1004](#)
- arg3, [1004](#)
- c, [1003](#)
- operator(), [1003](#)
- pointer\_to\_member\_to\_map\_function\_3args, [1003](#)
- ptr, [1003](#)
- GiNaC::pole\_error, [1005](#)
- deg, [1006](#)
- degree, [1006](#)
- pole\_error, [1006](#)
- GiNaC::possymbol, [1007](#)
- duplicate, [1013](#)
- get\_domain, [1013](#)
- possymbol, [1013](#)
- GiNaC::power, [1014](#)
- archive, [1027](#)
- basis, [1032](#)
- coeff, [1022](#)
- conjugate, [1026](#)
- degree, [1022](#)
- derivative, [1027](#)
- do\_print\_csrc, [1029](#)
- do\_print\_csrc\_cl\_N, [1030](#)
- do\_print\_dflt, [1029](#)
- do\_print\_latex, [1029](#)
- do\_print\_python, [1029](#)
- do\_print\_python\_repr, [1030](#)
- eval, [1023](#)
- eval\_ncmul, [1027](#)
- evalf, [1023](#)
- evalm, [1024](#)
- expand, [1028](#)
- expand\_add, [1030](#)
- expand\_add\_2, [1030](#)
- expand\_mul, [1031](#)
- exponent, [1032](#)
- has, [1025](#)
- imag\_part, [1026](#)
- info, [1021](#)
- is\_polynomial, [1022](#)
- ldegree, [1022](#)
- map, [1021](#)
- mul, [1031](#)
- nops, [1021](#)
- normal, [1025](#)
- op, [1021](#)
- power, [1020](#)
- precedence, [1020](#)
- print\_power, [1028](#)
- read\_archive, [1027](#)
- real\_part, [1026](#)
- return\_type, [1028](#)
- return\_type\_tinfo, [1028](#)
- series, [1024](#)
- subs, [1024](#)
- to\_polynomial, [1026](#)
- to\_rational, [1025](#)
- GiNaC::print\_context, [1032](#)
- ~print\_context, [1033](#)
- options, [1033](#)
- print\_context, [1033](#)
- s, [1033](#)
- GiNaC::print\_context\_options, [1034](#)
- get\_id, [1035](#)
- get\_name, [1035](#)
- get\_parent\_name, [1035](#)
- id, [1036](#)
- name, [1035](#)
- parent\_name, [1035](#)
- print\_context\_options, [1034](#)
- GiNaC::print\_csrc, [1036](#)
- print\_csrc, [1037](#)
- GiNaC::print\_csrc\_cl\_N, [1038](#)
- print\_csrc\_cl\_N, [1039](#)
- GiNaC::print\_csrc\_double, [1040](#)
- print\_csrc\_double, [1041](#)
- GiNaC::print\_csrc\_float, [1042](#)
- print\_csrc\_float, [1043](#)
- GiNaC::print\_dflt, [1044](#)
- print\_dflt, [1045](#)
- GiNaC::print\_functor, [1045](#)
- impl, [1047](#)
- is\_valid, [1047](#)
- operator(), [1047](#)
- operator=, [1046](#)
- print\_functor, [1046](#)
- GiNaC::print\_functor\_impl, [1048](#)
- ~print\_functor\_impl, [1048](#)
- duplicate, [1048](#)

- operator(), 1049
- GiNaC::print\_latex, 1049
- print\_latex, 1050
- GiNaC::print\_memfun\_handler< T, C >, 1051
- duplicate, 1052
- F, 1052
- f, 1053
- operator(), 1053
- print\_memfun\_handler, 1052
- GiNaC::print\_options, 1053
- print\_index\_dimensions, 1055
- GiNaC::print\_ptrfun\_handler< T, C >, 1055
- duplicate, 1056
- F, 1056
- f, 1057
- operator(), 1057
- print\_ptrfun\_handler, 1056
- GiNaC::print\_python, 1058
- print\_python, 1059
- GiNaC::print\_python\_repr, 1059
- print\_python\_repr, 1060
- GiNaC::print\_tree, 1061
- delta\_indent, 1062
- print\_tree, 1062
- GiNaC::pseries, 1066
- add\_series, 1081
- archive, 1078
- coeff, 1074
- coeffop, 1080
- collect, 1075
- conjugate, 1077
- convert\_to\_poly, 1079
- degree, 1074
- derivative, 1078
- do\_print, 1083
- do\_print\_latex, 1083
- do\_print\_python, 1083
- do\_print\_python\_repr, 1084
- do\_print\_tree, 1083
- eval, 1075
- eval\_integ, 1077
- evalf, 1075
- evalm, 1078
- expand, 1076
- exponop, 1080
- get\_point, 1079
- get\_var, 1079
- imag\_part, 1077
- is\_compatible\_to, 1080
- is\_terminating, 1080
- is\_zero, 1080
- ldegree, 1074
- mul\_const, 1081
- mul\_series, 1081
- nops, 1073
- normal, 1076
- op, 1073
- point, 1084
- power\_const, 1082
- precedence, 1073
- print\_series, 1082
- pseries, 1072, 1073
- read\_archive, 1078
- real\_part, 1077
- seq, 1084
- series, 1075
- shift\_exponents, 1082
- subs, 1076
- var, 1084
- GiNaC::psi1\_SERIAL, 1085
- serial, 1085
- GiNaC::psi2\_SERIAL, 1086
- serial, 1086
- GiNaC::ptr< T >, 1087
- ~ptr, 1089
- get\_pointer, 1091
- makewritable, 1089
- operator!=, 1090, 1091
- operator<<, 1092
- operator\*, 1089
- operator->, 1089
- operator=, 1089
- operator==, 1090, 1091
- p, 1092
- ptr, 1088
- std::less< ptr< T > >, 1090
- swap, 1090
- GiNaC::realsymbol, 1092
- conjugate, 1099
- duplicate, 1099
- get\_domain, 1099
- imag\_part, 1099
- real\_part, 1099
- realsymbol, 1098
- GiNaC::refcounted, 1100
- add\_reference, 1101
- get\_refcount, 1101
- refcount, 1102
- refcounted, 1101
- remove\_reference, 1101
- set\_refcount, 1102
- GiNaC::registered\_class\_options, 1102
- get\_id, 1103
- get\_name, 1103
- get\_parent\_name, 1103
- get\_print\_dispatch\_table, 1104
- name, 1105
- parent\_name, 1105
- print\_dispatch\_table, 1105
- print\_func, 1104
- registered\_class\_options, 1103
- set\_print\_func, 1104
- tinfo\_key, 1105
- GiNaC::relational, 1106
- archive, 1114
- calchash, 1116



- canonical, 1114
- do\_print, 1116
- do\_print\_python\_repr, 1116
- equal, 1112
- eval\_ncmul, 1115
- greater, 1112
- greater\_or\_equal, 1112
- info, 1113
- less, 1112
- less\_or\_equal, 1112
- lh, 1118
- lhs, 1116
- make\_safe\_bool, 1117
- map, 1113
- match\_same\_type, 1115
- nops, 1113
- not\_equal, 1112
- o, 1118
- op, 1113
- operator safe\_bool, 1117
- operator!, 1117
- operators, 1112
- precedence, 1112
- read\_archive, 1114
- relational, 1112
- return\_type, 1115
- return\_type\_tinfo, 1115
- rh, 1118
- rhs, 1117
- safe\_bool, 1112
- subs, 1114
- GiNaC::relational::safe\_bool\_helper, 1131
- nonnull, 1132
- GiNaC::remember\_strategies, 1118
  - delete\_cyclic, 1120
  - delete\_lfu, 1120
  - delete\_lru, 1120
  - delete\_never, 1120
- GiNaC::remember\_table, 1120
  - add\_entry, 1122
  - clear\_all\_entries, 1122
  - init\_table, 1123
  - lookup\_entry, 1122
  - max\_assoc\_size, 1123
  - remember\_strategy, 1123
  - remember\_table, 1121
  - remember\_tables, 1122
  - show\_statistics, 1122
  - table\_size, 1123
- GiNaC::remember\_table\_entry, 1124
  - access\_counter, 1127
  - get\_last\_access, 1125
  - get\_result, 1125
  - get\_successful\_hits, 1126
  - hashvalue, 1126
  - is\_equal, 1125
  - last\_access, 1126
  - remember\_table\_entry, 1125
  - result, 1126
  - seq, 1126
  - successful\_hits, 1126
- GiNaC::remember\_table\_list, 1127
  - add\_entry, 1128
  - lookup\_entry, 1128
  - max\_assoc\_size, 1129
  - remember\_strategy, 1129
  - remember\_table\_list, 1128
- GiNaC::return\_type\_t, 1129
  - operator!=, 1130
  - operator<, 1130
  - operator==, 1130
  - rl, 1130
  - tinfo, 1130
- GiNaC::return\_types, 1131
  - commutative, 1131
  - noncommutative, 1131
  - noncommutative\_composite, 1131
- GiNaC::scalar\_products, 1132
  - add, 1133
  - add\_vectors, 1133
  - clear, 1133
  - debugprint, 1134
  - evaluate, 1134
  - is\_defined, 1133
  - spm, 1134
- GiNaC::series\_options, 1135
  - suppress\_branchcut, 1135
- GiNaC::solve\_algo, 1135
  - automatic, 1136
  - bareiss, 1136
  - divfree, 1136
  - gauss, 1136
  - markowitz, 1136
- GiNaC::spinidx, 1137
  - archive, 1145
  - conjugate, 1145
  - do\_print, 1147
  - do\_print\_latex, 1147
  - do\_print\_tree, 1147
  - dotted, 1148
  - is\_dotted, 1146
  - is\_dummy\_pair\_same\_type, 1145
  - is\_undotted, 1146
  - match\_same\_type, 1146
  - read\_archive, 1145
  - spinidx, 1144
  - toggle\_dot, 1147
  - toggle\_variance\_dot, 1147
- GiNaC::spinmetric, 1148
  - contract\_with, 1154
  - do\_print, 1155
  - do\_print\_latex, 1155
  - eval\_indexed, 1154
  - info, 1154
- GiNaC::spmapkey, 1155
  - debugprint, 1157

- dim, 1158
- operator<, 1157
- operator==, 1157
- spmapkey, 1156, 1157
- v1, 1157
- v2, 1158
- GiNaC::status\_flags, 1158
  - dynallocated, 1159
  - evaluated, 1159
  - expanded, 1159
  - has\_indices, 1159
  - has\_no\_indices, 1159
  - hash\_calculated, 1159
  - is\_negative, 1159
  - is\_positive, 1159
  - not\_shareable, 1159
  - purely\_indefinite, 1159
- GiNaC::structure< T, ComparisonPolicy >, 1159
  - add\_indexed, 1174
  - calchash, 1177
  - coeff, 1170
  - collect, 1171
  - contract\_with, 1175
  - degree, 1170
  - derivative, 1171
  - eval, 1165
  - eval\_indexed, 1166
  - eval\_ncmul, 1166
  - evalm, 1165
  - expand, 1171
  - get\_class\_name, 1165
  - get\_free\_indices, 1174
  - get\_struct, 1177
  - has, 1168
  - info, 1167
  - integer\_content, 1173
  - is\_equal\_same\_type, 1176
  - ldegree, 1170
  - let\_op, 1168
  - map, 1170
  - match, 1169
  - match\_same\_type, 1169
  - max\_coefficient, 1174
  - nops, 1167
  - normal, 1172
  - obj, 1178
  - op, 1167
  - operator->, 1177
  - operator[], 1167, 1168
  - precedence, 1167
  - print, 1166
  - return\_type, 1176
  - return\_type\_tinfo, 1176
  - scalar\_mul\_indexed, 1175
  - series, 1172
  - smod, 1173
  - structure, 1165
  - subs, 1169
    - to\_polynomial, 1173
    - to\_rational, 1172
- GiNaC::su3d, 1178
  - contract\_with, 1183
  - do\_print, 1183
  - do\_print\_latex, 1183
  - eval\_indexed, 1183
  - return\_type, 1183
- GiNaC::su3f, 1184
  - contract\_with, 1189
  - do\_print, 1189
  - do\_print\_latex, 1189
  - eval\_indexed, 1189
  - return\_type, 1189
- GiNaC::su3one, 1190
  - do\_print, 1194
  - do\_print\_latex, 1194
- GiNaC::su3t, 1195
  - contract\_with, 1199
  - do\_print, 1200
  - do\_print\_latex, 1200
- GiNaC::subs\_options, 1200
  - algebraic, 1201
  - no\_index\_renaming, 1201
  - no\_pattern, 1201
  - pattern\_is\_not\_product, 1201
  - pattern\_is\_product, 1201
  - really\_subs\_idx, 1201
  - subs\_algebraic, 1201
  - subs\_no\_pattern, 1201
- GiNaC::sy\_is\_less, 1201
  - operator(), 1201
  - sy\_is\_less, 1201
  - v, 1202
- GiNaC::sy\_swap, 1202
  - operator(), 1202
  - swapped, 1203
  - sy\_swap, 1202
  - v, 1203
- GiNaC::sym\_desc, 1203
  - deg\_a, 1206
  - deg\_b, 1206
  - ldeg\_a, 1206
  - ldeg\_b, 1206
  - max\_deg, 1206
  - max\_lcnops, 1206
  - operator<, 1205
  - sym, 1205
  - sym\_desc, 1205
- GiNaC::symbol, 1207
  - archive, 1216
  - calchash, 1217
  - conjugate, 1215
  - derivative, 1216
  - do\_print, 1218
  - do\_print\_latex, 1219
  - do\_print\_python\_repr, 1219
  - do\_print\_tree, 1219

- eval, 1213
- evalf, 1213
- get\_domain, 1217
- get\_name, 1218
- get\_TeX\_name, 1218
- imag\_part, 1215
- info, 1213
- is\_equal\_same\_type, 1217
- is\_polynomial, 1215
- name, 1219
- next\_serial, 1220
- normal, 1214
- read\_archive, 1216
- real\_part, 1215
- serial, 1219
- series, 1213
- set\_name, 1218
- set\_TeX\_name, 1218
- subs, 1214
- symbol, 1212
- TeX\_name, 1220
- to\_polynomial, 1215
- to\_rational, 1214
- GiNaC::symbolset, 1220
  - has, 1221
  - insert\_symbols, 1221
  - s, 1221
  - symbolset, 1221
- GiNaC::symmetry, 1222
  - add, 1229
  - antisymmetric, 1227
  - archive, 1228
  - calchash, 1228
  - canonicalize, 1231
  - children, 1232
  - cyclic, 1227
  - do\_print, 1230
  - do\_print\_tree, 1230
  - get\_type, 1228
  - has\_cyclic, 1230
  - has\_nonsymmetric, 1230
  - has\_symmetry, 1229
  - indices, 1231
  - none, 1227
  - read\_archive, 1228
  - set\_type, 1229
  - sy\_is\_less, 1231
  - sy\_swap, 1231
  - symmetric, 1227
  - symmetry, 1227
  - symmetry\_type, 1227
  - type, 1231
  - validate, 1229
- GiNaC::symminfo, 1232
  - coeff, 1234
  - num, 1235
  - orig, 1234
  - symminfo, 1234
  - symmterm, 1234
- GiNaC::symminfo\_is\_less\_by\_orig, 1235
  - operator(), 1235
- GiNaC::symminfo\_is\_less\_by\_symmterm, 1235
  - operator(), 1236
- GiNaC::tensdelta, 1236
  - contract\_with, 1241
  - do\_print, 1242
  - do\_print\_latex, 1242
  - eval\_indexed, 1241
  - info, 1241
  - return\_type, 1241
- GiNaC::tensepsilon, 1242
  - archive, 1248
  - contract\_with, 1248
  - do\_print, 1249
  - do\_print\_latex, 1249
  - eval\_indexed, 1247
  - info, 1247
  - minkowski, 1249
  - pos\_sig, 1249
  - read\_archive, 1248
  - return\_type, 1248
  - tensepsilon, 1247
- GiNaC::tensmetric, 1250
  - contract\_with, 1255
  - do\_print, 1256
  - eval\_indexed, 1255
  - info, 1255
  - return\_type, 1256
- GiNaC::tensor, 1257
  - replace\_contr\_index, 1261
  - return\_type, 1261
- GiNaC::terminfo, 1262
  - orig, 1263
  - symm, 1263
  - terminfo, 1263
- GiNaC::terminfo\_is\_less, 1263
  - operator(), 1263
- GiNaC::unarchive\_table\_t, 1265
  - ~unarchive\_table\_t, 1266
  - find, 1266
  - insert, 1266
  - unarch\_map, 1267
  - unarchive\_table\_t, 1266
  - usecount, 1267
- GiNaC::user\_defined\_kernel, 1267
  - do\_print, 1275
  - f, 1276
  - is\_numeric, 1275
  - Laurent\_series, 1275
  - let\_op, 1274
  - nops, 1274
  - op, 1274
  - user\_defined\_kernel, 1274
  - uses\_Laurent\_series, 1275
  - x, 1276
- GiNaC::varidx, 1276

- archive, [1284](#)
- covariant, [1287](#)
- do\_print, [1286](#)
- do\_print\_tree, [1286](#)
- is\_contravariant, [1286](#)
- is\_covariant, [1285](#)
- is\_dummy\_pair\_same\_type, [1284](#)
- match\_same\_type, [1285](#)
- read\_archive, [1284](#)
- toggle\_variance, [1286](#)
- varidx, [1283](#)
- GiNaC::visitor, [1287](#)
- ~visitor, [1287](#)
- GiNaC::wildcard, [1288](#)
  - archive, [1292](#)
  - calchash, [1293](#)
  - do\_print, [1293](#)
  - do\_print\_python\_repr, [1294](#)
  - do\_print\_tree, [1294](#)
  - get\_label, [1293](#)
  - label, [1294](#)
  - match, [1292](#)
  - read\_archive, [1293](#)
  - wildcard, [1292](#)
- GiNaC::zeta1\_SERIAL, [1295](#)
  - serial, [1295](#)
- GiNaC::zeta2\_SERIAL, [1295](#)
  - serial, [1296](#)
- GINAC\_ASSERT
  - assertion.h, [1302](#)
- GINAC\_BIND\_UNARCHIVER
  - archive.h, [1301](#)
  - GiNaC, [90](#), [93](#), [94](#), [105](#), [111](#), [127](#), [129](#), [132](#), [196](#), [199–202](#), [204](#), [210](#), [211](#), [228](#), [264](#), [265](#), [267–269](#), [277](#), [278](#), [287](#), [291](#)
- GINAC\_DECLARE\_PRINT\_CONTEXT
  - print.h, [1397](#)
- GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE
  - print.h, [1396](#)
- GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON
  - print.h, [1396](#)
- GINAC\_DECLARE\_REGISTERED\_CLASS
  - registrar.h, [1402](#)
- GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON
  - registrar.h, [1401](#)
- GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS
  - registrar.h, [1402](#)
- GINAC\_DECLARE\_UNARCHIVER
  - archive.h, [1300](#)
  - GiNaC, [90](#), [103](#), [104](#), [110](#), [111](#), [127](#), [130](#), [131](#), [137](#), [196](#), [202–204](#), [206](#), [210](#), [211](#), [246](#), [264](#), [265](#), [267–269](#), [273](#), [281](#), [282](#), [288](#)
- GINAC\_IMPLEMENT\_PRINT\_CONTEXT
  - print.h, [1397](#)
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS
  - registrar.h, [1403](#)
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT
  - GiNaC, [89](#), [91](#), [92](#), [104](#), [110](#), [125–128](#), [132](#), [195](#), [199–202](#), [204](#), [209](#), [211](#), [227](#), [263](#), [265](#), [267](#), [269](#), [277](#), [287](#)
  - registrar.h, [1403](#)
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T
  - registrar.h, [1403](#)
- GINAC\_LT\_AGE
  - version.h, [1420](#)
- GINAC\_LT\_CURRENT
  - version.h, [1420](#)
- GINAC\_LT\_REVISION
  - version.h, [1420](#)
- GINACLIB\_ARCHIVE\_AGE
  - version.h, [1421](#)
- GINACLIB\_ARCHIVE\_VERSION
  - version.h, [1420](#)
- GINACLIB\_MAJOR\_VERSION
  - version.h, [1420](#)
- GINACLIB\_MICRO\_VERSION
  - version.h, [1420](#)
- GINACLIB\_MINOR\_VERSION
  - version.h, [1420](#)
- GINACLIB\_STR
  - version.h, [1421](#)
- GINACLIB\_STR\_HELPER
  - version.h, [1421](#)
- GINACLIB\_VERSION
  - version.h, [1421](#)
- golden\_ratio\_hash
  - GiNaC, [283](#)
- greater
  - GiNaC::relational, [1112](#)
- greater\_or\_equal
  - GiNaC::relational, [1112](#)
- guess\_precision
  - GiNaC, [237](#)
- H\_deriv
  - GiNaC, [172](#)
- H\_eval
  - GiNaC, [172](#)
- H\_evalf
  - GiNaC, [172](#)
- H\_print\_latex
  - GiNaC, [172](#)
- H\_series
  - GiNaC, [172](#)
- handle\_factor
  - GiNaC::make\_flat\_inserter, [818](#)
- has
  - GiNaC, [113](#)
  - GiNaC::basic, [366](#)
  - GiNaC::ex, [583](#)
  - GiNaC::mul, [877](#)
  - GiNaC::numeric, [956](#)
  - GiNaC::power, [1025](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1168](#)
  - GiNaC::symbolset, [1221](#)
- has\_cyclic
  - GiNaC::symmetry, [1230](#)

- has\_derivative
  - GiNaC::function\_options, 719
- has\_dummy\_index\_for
  - GiNaC::indexed, 766
- has\_ex
  - GiNaC::archive\_node, 350
- has\_expression
  - GiNaC::archive\_node, 351
- has\_indices
  - GiNaC::info\_flags, 770
  - GiNaC::status\_flags, 1159
- has\_no\_indices
  - GiNaC::status\_flags, 1159
- has\_nonsymmetric
  - GiNaC::symmetry, 1230
- has\_power
  - GiNaC::function\_options, 720
- has\_same\_ex\_as
  - GiNaC::archive\_node, 349
- has\_symmetry
  - GiNaC::symmetry, 1229
- has\_trailing\_zero
  - GiNaC::integration\_kernel, 789
- hash\_calculated
  - GiNaC::status\_flags, 1159
- hash\_map.h, 1349
- hash\_seed.h, 1350
- hashvalue
  - GiNaC::basic, 383
  - GiNaC::remember\_table\_entry, 1126
- hasindex
  - GiNaC, 135
- haswild
  - GiNaC, 288
- head
  - GiNaC::composition\_generator::coolmulti, 490
- heur\_gcd
  - GiNaC, 220
- heur\_gcd\_z
  - GiNaC, 219
- hold
  - GiNaC::basic, 379
- hold\_ncmul
  - GiNaC, 211
  - GiNaC::ncmul, 943
- I
  - GiNaC, 291
- i
  - GiNaC::composition\_generator::coolmulti, 490
  - GiNaC::const\_iterator, 448
  - GiNaC::internal::\_iter\_rep, 313
- i\_end
  - GiNaC::internal::\_iter\_rep, 313
- id
  - GiNaC::print\_context\_options, 1036
- identify\_parents
  - GiNaC::class\_info< OPT >, 399
- idx
  - GiNaC, 290
  - GiNaC::idx, 738
  - GiNaC::info\_flags, 770
- idx.cpp, 1350
- idx.h, 1351
- idx\_symmetrization
  - GiNaC, 134
- ifactor
  - GiNaC, 196
- imag
  - GiNaC, 251
  - GiNaC::numeric, 973
- imag\_part
  - GiNaC, 113
  - GiNaC::add, 329
  - GiNaC::basic, 375
  - GiNaC::constant, 463
  - GiNaC::container< C >, 479
  - GiNaC::ex, 583
  - GiNaC::function, 674
  - GiNaC::indexed, 762
  - GiNaC::matrix, 833
  - GiNaC::mul, 878
  - GiNaC::ncmul, 940
  - GiNaC::numeric, 959
  - GiNaC::power, 1026
  - GiNaC::pseries, 1077
  - GiNaC::realsymbol, 1099
  - GiNaC::symbol, 1215
- imag\_part\_conjugate
  - GiNaC, 141
- imag\_part\_eval
  - GiNaC, 141
- imag\_part\_evalf
  - GiNaC, 140
- imag\_part\_expl\_derivative
  - GiNaC, 141
- imag\_part\_f
  - GiNaC::function\_options, 722
- imag\_part\_func
  - GiNaC::function\_options, 697–699, 714
- imag\_part\_funcp
  - GiNaC, 61
- imag\_part\_funcp\_1
  - GiNaC, 62
- imag\_part\_funcp\_10
  - GiNaC, 77
- imag\_part\_funcp\_11
  - GiNaC, 79
- imag\_part\_funcp\_12
  - GiNaC, 80
- imag\_part\_funcp\_13
  - GiNaC, 82
- imag\_part\_funcp\_14
  - GiNaC, 84
- imag\_part\_funcp\_2
  - GiNaC, 64
- imag\_part\_funcp\_3

- GiNaC, [65](#)
- imag\_part\_funcp\_4
  - GiNaC, [67](#)
- imag\_part\_funcp\_5
  - GiNaC, [68](#)
- imag\_part\_funcp\_6
  - GiNaC, [70](#)
- imag\_part\_funcp\_7
  - GiNaC, [72](#)
- imag\_part\_funcp\_8
  - GiNaC, [74](#)
- imag\_part\_funcp\_9
  - GiNaC, [75](#)
- imag\_part\_funcp\_exvector
  - GiNaC, [86](#)
- imag\_part\_imag\_part
  - GiNaC, [141](#)
- imag\_part\_print\_latex
  - GiNaC, [141](#)
- imag\_part\_real\_part
  - GiNaC, [141](#)
- imag\_part\_use\_exvector\_args
  - GiNaC::function\_options, [725](#)
- impl
  - GiNaC::print\_functor, [1047](#)
- increment
  - GiNaC::const\_postorder\_iterator, [451](#)
  - GiNaC::const\_preorder\_iterator, [455](#)
- indefinite
  - GiNaC::info\_flags, [770](#)
- index0
  - GiNaC, [269](#)
- index1
  - GiNaC, [269](#)
- index2
  - GiNaC, [270](#)
- index3
  - GiNaC, [270](#)
- index\_dimensions
  - GiNaC, [263](#)
- indexed
  - GiNaC::indexed, [755–757](#), [759–761](#)
  - GiNaC::info\_flags, [770](#)
- indexed.cpp, [1352](#)
- indexed.h, [1354](#)
- indices
  - GiNaC::symmetry, [1231](#)
- indices\_consistent
  - GiNaC, [132](#)
- info
  - GiNaC::add, [325](#)
  - GiNaC::basic, [363](#)
  - GiNaC::class\_info< OPT >::tree\_node, [1265](#)
  - GiNaC::constant, [462](#)
  - GiNaC::container< C >, [475](#)
  - GiNaC::ex, [579](#)
  - GiNaC::expairseq, [620](#)
  - GiNaC::function, [675](#)
  - GiNaC::idx, [739](#)
  - GiNaC::indexed, [762](#)
  - GiNaC::minkmetric, [852](#)
  - GiNaC::mul, [876](#)
  - GiNaC::ncmul, [937](#)
  - GiNaC::numeric, [954](#)
  - GiNaC::power, [1021](#)
  - GiNaC::relational, [1113](#)
  - GiNaC::spinmetric, [1154](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1167](#)
  - GiNaC::symbol, [1213](#)
  - GiNaC::tensdelta, [1241](#)
  - GiNaC::tensepsilon, [1247](#)
  - GiNaC::tensmetric, [1255](#)
- info\_f
  - GiNaC::function\_options, [723](#)
- info\_func
  - GiNaC::function\_options, [711–713](#), [715](#)
- info\_funcp
  - GiNaC, [61](#)
- info\_funcp\_1
  - GiNaC, [63](#)
- info\_funcp\_10
  - GiNaC, [78](#)
- info\_funcp\_11
  - GiNaC, [80](#)
- info\_funcp\_12
  - GiNaC, [82](#)
- info\_funcp\_13
  - GiNaC, [84](#)
- info\_funcp\_14
  - GiNaC, [86](#)
- info\_funcp\_2
  - GiNaC, [64](#)
- info\_funcp\_3
  - GiNaC, [66](#)
- info\_funcp\_4
  - GiNaC, [68](#)
- info\_funcp\_5
  - GiNaC, [69](#)
- info\_funcp\_6
  - GiNaC, [71](#)
- info\_funcp\_7
  - GiNaC, [73](#)
- info\_funcp\_8
  - GiNaC, [75](#)
- info\_funcp\_9
  - GiNaC, [76](#)
- info\_funcp\_exvector
  - GiNaC, [87](#)
- info\_use\_exvector\_args
  - GiNaC::function\_options, [726](#)
- inifcns.cpp, [1355](#)
- inifcns.h, [1358](#)
- inifcns\_elliptic.cpp, [1360](#)
- inifcns\_gamma.cpp, [1361](#)
- inifcns\_nstdsums.cpp, [1362](#)
- inifcns\_trans.cpp, [1364](#)

- init
  - GiNaC::basic\_multi\_iterator< T >, 393
  - GiNaC::multi\_iterator\_counter< T >, 891
  - GiNaC::multi\_iterator\_counter\_indv< T >, 895
  - GiNaC::multi\_iterator\_ordered< T >, 899
  - GiNaC::multi\_iterator\_ordered\_eq< T >, 903
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 907
  - GiNaC::multi\_iterator\_permutation< T >, 911
  - GiNaC::multi\_iterator\_shuffle< T >, 915
  - GiNaC::multi\_iterator\_shuffle\_prime< T >, 920
- init\_table
  - GiNaC::remember\_table, 1123
- init\_unarchivers
  - GiNaC::library\_init, 817
- initialize
  - GiNaC::function\_options, 687
- insert
  - GiNaC::unarchive\_table\_t, 1266
- insert\_symbols
  - GiNaC::symbolset, 1221
- int\_length
  - GiNaC::numeric, 974
- integer
  - GiNaC::info\_flags, 770
- integer\_content
  - GiNaC::add, 327
  - GiNaC::basic, 371
  - GiNaC::ex, 593
  - GiNaC::mul, 880
  - GiNaC::numeric, 958
  - GiNaC::structure< T, ComparisonPolicy >, 1173
- integer\_polynomial
  - GiNaC::info\_flags, 770
- integral
  - GiNaC::error\_and\_integral, 565
  - GiNaC::integral, 776
- integral.cpp, 1367
- integral.h, 1368
- integration\_kernel.cpp, 1369
  - cache\_vec, 1371
  - order, 1370
  - qbar, 1370
  - x, 1371
- integration\_kernel.h, 1371
- interpolate
  - GiNaC, 218
- inv\_at\_cit
  - GiNaC::archive, 335
- inverse
  - GiNaC, 208, 247
  - GiNaC::matrix, 840
  - GiNaC::numeric, 965
- inverse\_atoms
  - GiNaC::archive, 341
- iquo
  - GiNaC, 243, 244
- irem
  - GiNaC, 242, 243
- is\_a
  - GiNaC, 91, 121, 265
  - GiNaC::ex, 604
- is\_canonical
  - GiNaC::expairseq, 632
- is\_canonical\_numeric
  - GiNaC::expair, 609
- is\_cinteger
  - GiNaC, 250
  - GiNaC::numeric, 969
- is\_clifford\_tinfo
  - GiNaC, 104
- is\_color\_tinfo
  - GiNaC, 108
- is\_compatible\_to
  - GiNaC::pseries, 1080
- is\_contravariant
  - GiNaC::varidx, 1286
- is\_covariant
  - GiNaC::varidx, 1285
- is\_crational
  - GiNaC, 250
  - GiNaC::numeric, 969
- is\_defined
  - GiNaC::scalar\_products, 1133
- is\_dim\_numeric
  - GiNaC::idx, 743
- is\_dim\_symbolic
  - GiNaC::idx, 743
- is\_dirac\_slash
  - GiNaC, 94
- is\_discriminant\_of\_quadratic\_number\_field
  - GiNaC, 197
- is\_dotted
  - GiNaC::spinidx, 1146
- is\_dummy\_pair
  - GiNaC, 129
- is\_dummy\_pair\_same\_type
  - GiNaC::idx, 742
  - GiNaC::spinidx, 1145
  - GiNaC::varidx, 1284
- is\_equal
  - GiNaC::basic, 379
  - GiNaC::ex, 597
  - GiNaC::expair, 608
  - GiNaC::numeric, 966
  - GiNaC::remember\_table\_entry, 1125
- is\_equal\_same\_type
  - GiNaC::basic, 376
  - GiNaC::constant, 464
  - GiNaC::container< C >, 479
  - GiNaC::expairseq, 624
  - GiNaC::fderivative, 655
  - GiNaC::function, 676
  - GiNaC::numeric, 960
  - GiNaC::structure< T, ComparisonPolicy >, 1176
  - GiNaC::symbol, 1217
- is\_even

- GiNaC, 249
  - GiNaC::numeric, 968
- is\_ex\_the\_function
  - function.h, 1348
- is\_exactly\_a
  - GiNaC, 92, 122
  - GiNaC::ex, 604
- is\_integer
  - GiNaC, 248
  - GiNaC::numeric, 967
- is\_less
  - GiNaC::expair, 608
- is\_negative
  - GiNaC, 248
  - GiNaC::numeric, 967
  - GiNaC::status\_flags, 1159
- is\_nonneg\_integer
  - GiNaC, 248
  - GiNaC::numeric, 968
- is\_numeric
  - GiNaC::Ebar\_kernel, 528
  - GiNaC::Eisenstein\_h\_kernel, 538
  - GiNaC::Eisenstein\_kernel, 549
  - GiNaC::ELi\_kernel, 560
  - GiNaC::idx, 742
  - GiNaC::integration\_kernel, 789
  - GiNaC::Kronecker\_dtau\_kernel, 801
  - GiNaC::Kronecker\_dz\_kernel, 810
  - GiNaC::modular\_form\_kernel, 862
  - GiNaC::multiple\_polylog\_kernel, 927
  - GiNaC::user\_defined\_kernel, 1275
- is\_odd
  - GiNaC, 249
  - GiNaC::numeric, 968
- is\_order\_function
  - GiNaC, 158
- is\_polynomial
  - GiNaC, 114
  - GiNaC::add, 325
  - GiNaC::basic, 368
  - GiNaC::constant, 462
  - GiNaC::ex, 586
  - GiNaC::mul, 876
  - GiNaC::numeric, 955
  - GiNaC::power, 1022
  - GiNaC::symbol, 1215
- is\_pos\_integer
  - GiNaC, 248
  - GiNaC::numeric, 967
- is\_positive
  - GiNaC, 248
  - GiNaC::numeric, 967
  - GiNaC::status\_flags, 1159
- is\_prime
  - GiNaC, 249
  - GiNaC::numeric, 968
- is\_rational
  - GiNaC, 249
- GiNaC::numeric, 969
- is\_real
  - GiNaC, 249
  - GiNaC::numeric, 969
- is\_symbolic
  - GiNaC::idx, 742
- is\_terminating
  - GiNaC, 266
  - GiNaC::pseries, 1080
- is\_the\_function
  - GiNaC, 128
- is\_the\_function< G\_SERIAL >
  - GiNaC, 157
- is\_the\_function< iterated\_integral\_SERIAL >
  - GiNaC, 158
- is\_the\_function< psi\_SERIAL >
  - GiNaC, 158
- is\_the\_function< zeta\_SERIAL >
  - GiNaC, 156
- is\_undotted
  - GiNaC::spinidx, 1146
- is\_valid
  - GiNaC::print\_functor, 1047
- is\_zero
  - GiNaC, 120, 247
  - GiNaC::ex, 597
  - GiNaC::numeric, 966
  - GiNaC::pseries, 1080
- is\_zero\_matrix
  - GiNaC::ex, 598
  - GiNaC::matrix, 842
- isqrt
  - GiNaC, 245
- iterated\_integral
  - GiNaC, 158
- iterated\_integral2\_eval
  - GiNaC, 162
- iterated\_integral2\_evalf
  - GiNaC, 161
- iterated\_integral3\_eval
  - GiNaC, 162
- iterated\_integral3\_evalf
  - GiNaC, 162
- iterated\_integral\_evalf\_impl
  - GiNaC, 161
- iterator\_category
  - GiNaC::const\_iterator, 443
  - GiNaC::const\_postorder\_iterator, 449
  - GiNaC::const\_preorder\_iterator, 453
- K
  - GiNaC::Eisenstein\_kernel, 552
  - GiNaC::Kronecker\_dtau\_kernel, 803
  - GiNaC::Kronecker\_dz\_kernel, 812
- k
  - factor.cpp, 1329
  - GiNaC::Eisenstein\_h\_kernel, 540
  - GiNaC::Eisenstein\_kernel, 551
  - GiNaC::modular\_form\_kernel, 863



- Kronecker\_dtau\_kernel
  - GiNaC::Kronecker\_dtau\_kernel, [801](#)
- Kronecker\_dz\_kernel
  - GiNaC::Kronecker\_dz\_kernel, [810](#)
- kronecker\_symbol
  - GiNaC, [197](#)
- label
  - GiNaC::wildcard, [1294](#)
- lanczos\_coeffs
  - GiNaC::lanczos\_coeffs, [813](#)
- laplace
  - GiNaC::determinant\_algo, [494](#)
- last
  - factor.cpp, [1329](#)
- last\_access
  - GiNaC::remember\_table\_entry, [1126](#)
- latex
  - GiNaC, [261](#)
- latex\_name
  - GiNaC::function\_options, [687](#)
- Laurent\_series
  - GiNaC::Eisenstein\_h\_kernel, [538](#)
  - GiNaC::Eisenstein\_kernel, [550](#)
  - GiNaC::integration\_kernel, [789](#)
  - GiNaC::modular\_form\_kernel, [862](#)
  - GiNaC::user\_defined\_kernel, [1275](#)
- lcm
  - GiNaC, [222](#), [244](#)
- lcm\_of\_coefficients\_denominators
  - GiNaC, [213](#)
- lcmcoeff
  - GiNaC, [213](#)
- lcoeff
  - GiNaC::ex, [587](#)
- ldeg\_a
  - GiNaC::sym\_desc, [1206](#)
- ldeg\_b
  - GiNaC::sym\_desc, [1206](#)
- ldegree
  - GiNaC, [114](#)
  - GiNaC::add, [326](#)
  - GiNaC::basic, [368](#)
  - GiNaC::ex, [587](#)
  - GiNaC::integral, [778](#)
  - GiNaC::mul, [877](#)
  - GiNaC::ncmul, [938](#)
  - GiNaC::numeric, [955](#)
  - GiNaC::power, [1022](#)
  - GiNaC::pseries, [1074](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1170](#)
- len
  - factor.cpp, [1329](#)
- less
  - GiNaC::relational, [1112](#)
- less\_or\_equal
  - GiNaC::relational, [1112](#)
- let\_op
  - GiNaC::basic, [365](#)
- GiNaC::clifford, [415](#)
- GiNaC::container< C >, [477](#)
- GiNaC::Ebar\_kernel, [528](#)
- GiNaC::Eisenstein\_h\_kernel, [538](#)
- GiNaC::Eisenstein\_kernel, [549](#)
- GiNaC::ELi\_kernel, [560](#)
- GiNaC::ex, [581](#)
- GiNaC::integral, [779](#)
- GiNaC::Kronecker\_dtau\_kernel, [801](#)
- GiNaC::Kronecker\_dz\_kernel, [810](#)
- GiNaC::matrix, [831](#)
- GiNaC::modular\_form\_kernel, [862](#)
- GiNaC::multiple\_polylog\_kernel, [927](#)
- GiNaC::structure< T, ComparisonPolicy >, [1168](#)
- GiNaC::user\_defined\_kernel, [1274](#)
- lgamma
  - GiNaC, [237](#), [238](#)
- lgamma\_conjugate
  - GiNaC, [163](#)
- lgamma\_deriv
  - GiNaC, [163](#)
- lgamma\_eval
  - GiNaC, [162](#)
- lgamma\_evalf
  - GiNaC, [162](#)
- lgamma\_series
  - GiNaC, [163](#)
- lh
  - GiNaC::relational, [1118](#)
- lhs
  - GiNaC, [120](#)
  - GiNaC::ex, [582](#)
  - GiNaC::relational, [1116](#)
- Li2
  - GiNaC, [237](#)
- Li2\_
  - GiNaC, [236](#)
- Li2\_conjugate
  - GiNaC, [149](#)
- Li2\_deriv
  - GiNaC, [149](#)
- Li2\_eval
  - GiNaC, [149](#)
- Li2\_evalf
  - GiNaC, [149](#)
- Li2\_projection
  - GiNaC, [236](#)
- Li2\_series
  - GiNaC, [149](#), [236](#)
- Li3\_eval
  - GiNaC, [150](#)
- Li\_deriv
  - GiNaC, [170](#)
- Li\_eval
  - GiNaC, [169](#)
- Li\_evalf
  - GiNaC, [169](#)
- Li\_print\_latex

- GiNaC, [170](#)
- Li\_series
  - GiNaC, [169](#)
- library\_init
  - GiNaC::library\_init, [816](#)
- library\_initializer
  - GiNaC, [290](#)
- likely
  - compiler.h, [1313](#)
- link\_ex
  - GiNaC, [124](#)
- list
  - GiNaC::info\_flags, [770](#)
- log
  - GiNaC, [230](#)
- log2
  - GiNaC, [282](#)
- log\_conjugate
  - GiNaC, [178](#)
- log\_deriv
  - GiNaC, [177](#)
- log\_eval
  - GiNaC, [177](#)
- log\_evalf
  - GiNaC, [176](#)
- log\_expand
  - GiNaC, [178](#)
- log\_imag\_part
  - GiNaC, [177](#)
- log\_info
  - GiNaC, [178](#)
- log\_real\_part
  - GiNaC, [177](#)
- log\_series
  - GiNaC, [177](#)
- lookup\_entry
  - GiNaC::remember\_table, [1122](#)
  - GiNaC::remember\_table\_list, [1128](#)
- lookup\_map
  - GiNaC, [88](#)
- lookup\_remember\_table
  - GiNaC::function, [678](#)
- lorentz\_eps
  - GiNaC, [281](#)
- lorentz\_g
  - GiNaC, [279](#)
- lr
  - factor.cpp, [1328](#)
- lsolve
  - GiNaC, [155](#)
- lst
  - GiNaC, [88](#)
- lst.cpp, [1373](#)
- lst.h, [1373](#)
- lst\_to\_clifford
  - GiNaC, [100](#), [101](#)
- lst\_to\_matrix
  - GiNaC, [205](#)

- m
  - factor.cpp, [1327](#)
  - GiNaC::basic\_partition\_generator::mpartition2, [865](#)
  - GiNaC::Ebar\_kernel, [530](#)
  - GiNaC::ELi\_kernel, [562](#)
  - GiNaC::matrix, [846](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [982](#)
- make\_flat
  - GiNaC::expairseq, [631](#)
- make\_flat\_inserter
  - GiNaC::make\_flat\_inserter, [818](#)
- make\_hash\_seed
  - GiNaC, [128](#)
- make\_real\_float
  - GiNaC, [227](#)
- make\_return\_type\_t
  - GiNaC, [266](#)
- make\_safe\_bool
  - GiNaC::relational, [1117](#)
- makewritable
  - GiNaC::ptr< T >, [1089](#)
- makewritable
  - GiNaC::ex, [602](#)
- map
  - GiNaC::basic, [367](#)
  - GiNaC::ex, [585](#)
  - GiNaC::expairseq, [621](#)
  - GiNaC::idx, [739](#)
  - GiNaC::power, [1021](#)
  - GiNaC::relational, [1113](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1170](#)
- map\_eval\_integ
  - GiNaC, [289](#)
- map\_evalm
  - GiNaC, [288](#)
- markowitz
  - GiNaC::solve\_algo, [1136](#)
- markowitz\_elimination
  - GiNaC::matrix, [844](#)
- match
  - GiNaC, [118](#)
  - GiNaC::basic, [366](#)
  - GiNaC::ex, [583](#), [584](#)
  - GiNaC::expairseq, [622](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1169](#)
  - GiNaC::wildcard, [1292](#)
- match\_same\_type
  - GiNaC::basic, [366](#)
  - GiNaC::clifford, [412](#)
  - GiNaC::color, [433](#)
  - GiNaC::fderivative, [656](#)
  - GiNaC::function, [676](#)
  - GiNaC::idx, [741](#)
  - GiNaC::matrix, [834](#)
  - GiNaC::relational, [1115](#)
  - GiNaC::spinidx, [1146](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1169](#)

- GiNaC::varidx, [1285](#)
- matrix
  - GiNaC::matrix, [828](#), [830](#)
- matrix.cpp, [1374](#)
- matrix.h, [1375](#)
- max\_assoc\_size
  - GiNaC::remember\_table, [1123](#)
  - GiNaC::remember\_table\_list, [1129](#)
- max\_coefficient
  - GiNaC::add, [328](#)
  - GiNaC::basic, [372](#)
  - GiNaC::ex, [595](#)
  - GiNaC::mul, [881](#)
  - GiNaC::numeric, [959](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1174](#)
- max\_deg
  - GiNaC::sym\_desc, [1206](#)
- max\_integration\_level
  - GiNaC::integral, [782](#)
- max\_lcnops
  - GiNaC::sym\_desc, [1206](#)
- metric
  - GiNaC::clifford, [416](#)
- metric\_tensor
  - GiNaC, [278](#)
- minimal\_dim
  - GiNaC, [130](#)
  - GiNaC::idx, [744](#)
- minkmetric
  - GiNaC::minkmetric, [852](#)
- minkowski
  - GiNaC::tensepsilon, [1249](#)
- mod
  - GiNaC, [241](#)
- modular\_form\_kernel
  - GiNaC::modular\_form\_kernel, [861](#)
- modulus
  - factor.cpp, [1332](#)
- mpartition2
  - GiNaC::basic\_partition\_generator::mpartition2, [864](#)
- mpgen
  - GiNaC::basic\_partition\_generator, [397](#)
- mul
  - GiNaC::add, [333](#)
  - GiNaC::matrix, [836](#)
  - GiNaC::mul, [874](#), [875](#)
  - GiNaC::numeric, [962](#)
  - GiNaC::power, [1031](#)
- mul.cpp, [1376](#)
- mul.h, [1377](#)
- mul\_const
  - GiNaC::pseries, [1081](#)
- mul\_dyn
  - GiNaC::numeric, [963](#)
- mul\_scalar
  - GiNaC::matrix, [836](#)
- mul\_series
  - GiNaC::pseries, [1081](#)
- multi\_iterator\_counter
  - GiNaC::multi\_iterator\_counter< T >, [891](#)
- multi\_iterator\_counter\_indv
  - GiNaC::multi\_iterator\_counter\_indv< T >, [894](#), [895](#)
- multi\_iterator\_ordered
  - GiNaC::multi\_iterator\_ordered< T >, [898](#), [899](#)
- multi\_iterator\_ordered\_eq
  - GiNaC::multi\_iterator\_ordered\_eq< T >, [902](#)
- multi\_iterator\_ordered\_eq\_indv
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, [906](#)
- multi\_iterator\_permutation
  - GiNaC::multi\_iterator\_permutation< T >, [910](#)
- multi\_iterator\_shuffle
  - GiNaC::multi\_iterator\_shuffle< T >, [915](#)
- multi\_iterator\_shuffle\_prime
  - GiNaC::multi\_iterator\_shuffle\_prime< T >, [920](#)
- multinomial\_coefficient
  - GiNaC, [282](#)
- multiple\_polylog\_kernel
  - GiNaC::multiple\_polylog\_kernel, [927](#)
- multiply\_lcm
  - GiNaC, [213](#)
- my\_ios\_callback
  - GiNaC, [260](#)
- my\_ios\_index
  - GiNaC, [259](#)
- N
  - GiNaC::basic\_multi\_iterator< T >, [394](#)
  - GiNaC::Eisenstein\_h\_kernel, [540](#)
  - GiNaC::Eisenstein\_kernel, [551](#)
- n
  - factor.cpp, [1328](#)
  - GiNaC::basic\_partition\_generator::mpartition2, [865](#)
  - GiNaC::Ebar\_kernel, [530](#)
  - GiNaC::ELi\_kernel, [562](#)
  - GiNaC::Kronecker\_dtau\_kernel, [803](#)
  - GiNaC::Kronecker\_dz\_kernel, [812](#)
- N\_internal
  - GiNaC::multi\_iterator\_shuffle< T >, [916](#)
- name
  - GiNaC::archive::archived\_ex, [355](#)
  - GiNaC::archive\_node::property, [1064](#)
  - GiNaC::archive\_node::property\_info, [1065](#)
  - GiNaC::constant, [466](#)
  - GiNaC::function\_options, [721](#)
  - GiNaC::print\_context\_options, [1035](#)
  - GiNaC::registered\_class\_options, [1105](#)
  - GiNaC::symbol, [1219](#)
- ncmul
  - GiNaC::mul, [888](#)
  - GiNaC::ncmul, [936](#), [937](#)
- ncmul.cpp, [1378](#)
- ncmul.h, [1379](#)
- negative
  - GiNaC::info\_flags, [770](#)

- negint
  - GiNaC::info\_flags, 770
- next
  - GiNaC::class\_info< OPT >, 400
  - GiNaC::composition\_generator, 440
  - GiNaC::composition\_generator::coolmulti::element, 553
  - GiNaC::partition\_generator, 979
  - GiNaC::partition\_with\_zero\_parts\_generator, 982
- next\_partition
  - GiNaC::basic\_partition\_generator::mpartition2, 865
- next\_permutation
  - GiNaC::composition\_generator::coolmulti, 490
- next\_print\_context\_id
  - GiNaC, 291
- next\_serial
  - GiNaC::constant, 466
  - GiNaC::symbol, 1220
- no\_heur\_gcd
  - GiNaC::gcd\_options, 729
- no\_index\_dimensions
  - GiNaC, 263
- no\_index\_renaming
  - GiNaC::subs\_options, 1201
- no\_part\_factored
  - GiNaC::gcd\_options, 729
- no\_pattern
  - GiNaC::subs\_options, 1201
- no\_type
  - GiNaC::has\_distance< T >, 730
- nodes
  - GiNaC::archive, 341
- noncommutative
  - GiNaC::return\_types, 1131
- noncommutative\_composite
  - GiNaC::return\_types, 1131
- none
  - GiNaC::symmetry, 1227
- nonnegative
  - GiNaC::info\_flags, 770
- nonnegint
  - GiNaC::info\_flags, 770
- nonnull
  - GiNaC::relational::safe\_bool\_helper, 1132
- nops
  - GiNaC, 112, 206
  - GiNaC::basic, 364
  - GiNaC::clifford, 414
  - GiNaC::container< C >, 476
  - GiNaC::Ebar\_kernel, 528
  - GiNaC::Eisenstein\_h\_kernel, 537
  - GiNaC::Eisenstein\_kernel, 549
  - GiNaC::ELi\_kernel, 560
  - GiNaC::ex, 580
  - GiNaC::expairseq, 620
  - GiNaC::idx, 739
  - GiNaC::integral, 778
  - GiNaC::Kronecker\_dtau\_kernel, 801
  - GiNaC::Kronecker\_dz\_kernel, 810
  - GiNaC::matrix, 831
  - GiNaC::modular\_form\_kernel, 861
  - GiNaC::multiple\_polylog\_kernel, 927
  - GiNaC::power, 1021
  - GiNaC::pseries, 1073
  - GiNaC::relational, 1113
  - GiNaC::structure< T, ComparisonPolicy >, 1167
  - GiNaC::user\_defined\_kernel, 1274
- normal
  - GiNaC, 115
  - GiNaC::add, 327
  - GiNaC::basic, 370
  - GiNaC::ex, 589
  - GiNaC::mul, 879
  - GiNaC::numeric, 957
  - GiNaC::power, 1025
  - GiNaC::pseries, 1076
  - GiNaC::structure< T, ComparisonPolicy >, 1172
  - GiNaC::symbol, 1214
- normal.cpp, 1379
  - FAST\_COMPARE, 1381
  - STATISTICS, 1382
  - USE\_REMEMBER, 1382
  - USE\_TRIAL\_DIVISION, 1382
- normal.h, 1382
- not\_equal
  - GiNaC::relational, 1112
- not\_shareable
  - GiNaC::status\_flags, 1159
- not\_symmetric
  - GiNaC, 270
- nparams
  - GiNaC::function\_options, 721
- num
  - GiNaC::symminfo, 1235
- num\_expressions
  - GiNaC::archive, 338
- number
  - GiNaC::constant, 466
- number\_of\_type
  - GiNaC, 133
- numer
  - GiNaC, 115, 251
  - GiNaC::ex, 591
  - GiNaC::numeric, 974
- numer\_denom
  - GiNaC, 115
  - GiNaC::ex, 591
- numeric
  - GiNaC::info\_flags, 770
  - GiNaC::numeric, 952–954
- numeric.cpp, 1383
- numeric.h, 1387
- Nv
  - GiNaC::multi\_iterator\_counter\_indv< T >, 896
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 907

- o
  - GiNaC::relational, [1118](#)
- obj
  - GiNaC::structure< T, ComparisonPolicy >, [1178](#)
- odd
  - GiNaC::info\_flags, [770](#)
- one
  - factor.cpp, [1328](#)
- op
  - GiNaC, [120](#)
  - GiNaC::basic, [364](#)
  - GiNaC::clifford, [414](#)
  - GiNaC::container< C >, [476](#)
  - GiNaC::Ebar\_kernel, [528](#)
  - GiNaC::Eisenstein\_h\_kernel, [538](#)
  - GiNaC::Eisenstein\_kernel, [549](#)
  - GiNaC::ELi\_kernel, [560](#)
  - GiNaC::ex, [580](#)
  - GiNaC::expairseq, [621](#)
  - GiNaC::idx, [739](#)
  - GiNaC::integral, [778](#)
  - GiNaC::Kronecker\_dtau\_kernel, [801](#)
  - GiNaC::Kronecker\_dz\_kernel, [810](#)
  - GiNaC::matrix, [831](#)
  - GiNaC::modular\_form\_kernel, [861](#)
  - GiNaC::multiple\_polylog\_kernel, [927](#)
  - GiNaC::power, [1021](#)
  - GiNaC::pseries, [1073](#)
  - GiNaC::relational, [1113](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1167](#)
  - GiNaC::user\_defined\_kernel, [1274](#)
- operator long
  - GiNaC::\_numeric\_digits, [314](#)
- operator safe\_bool
  - GiNaC::relational, [1117](#)
- operator!
  - GiNaC::relational, [1117](#)
- operator!=
  - GiNaC, [258](#)
  - GiNaC::const\_iterator, [446](#)
  - GiNaC::const\_postorder\_iterator, [451](#)
  - GiNaC::const\_preorder\_iterator, [455](#)
  - GiNaC::internal::\_iter\_rep, [312](#)
  - GiNaC::numeric, [970](#)
  - GiNaC::ptr< T >, [1090](#), [1091](#)
  - GiNaC::return\_type\_t, [1130](#)
- operator<
  - GiNaC, [259](#)
  - GiNaC::const\_iterator, [447](#)
  - GiNaC::numeric, [970](#)
  - GiNaC::return\_type\_t, [1130](#)
  - GiNaC::spmapkey, [1157](#)
  - GiNaC::sym\_desc, [1205](#)
- operator<<
  - GiNaC, [90](#), [111](#), [112](#), [246](#), [261](#), [285–287](#)
  - GiNaC::archive, [340](#)
  - GiNaC::archive\_node, [351](#)
  - GiNaC::basic\_multi\_iterator< T >, [394](#)
  - GiNaC::multi\_iterator\_counter< T >, [892](#)
  - GiNaC::multi\_iterator\_counter\_indv< T >, [896](#)
  - GiNaC::multi\_iterator\_ordered< T >, [900](#)
  - GiNaC::multi\_iterator\_ordered\_eq< T >, [903](#)
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, [907](#)
  - GiNaC::multi\_iterator\_permutation< T >, [912](#)
  - GiNaC::multi\_iterator\_shuffle< T >, [916](#)
  - GiNaC::multi\_iterator\_shuffle\_prime< T >, [920](#)
  - GiNaC::ptr< T >, [1092](#)
- operator<=
  - GiNaC, [259](#)
  - GiNaC::const\_iterator, [447](#)
  - GiNaC::numeric, [970](#)
- operator>
  - GiNaC, [259](#)
  - GiNaC::const\_iterator, [447](#)
  - GiNaC::numeric, [972](#)
- operator>>
  - GiNaC, [91](#), [261](#)
  - GiNaC::archive, [340](#)
  - GiNaC::archive\_node, [351](#)
- operator>=
  - GiNaC, [259](#)
  - GiNaC::const\_iterator, [447](#)
  - GiNaC::numeric, [972](#)
- operator\*
  - GiNaC, [253](#)
  - GiNaC::const\_iterator, [444](#)
  - GiNaC::const\_postorder\_iterator, [450](#)
  - GiNaC::const\_preorder\_iterator, [454](#)
  - GiNaC::ptr< T >, [1089](#)
- operator\*=
  - GiNaC, [254](#), [255](#)
- operator()
  - GiNaC::basic\_multi\_iterator< T >, [393](#)
  - GiNaC::derivative\_map\_function, [492](#)
  - GiNaC::error\_and\_integral\_is\_less, [565](#)
  - GiNaC::eval\_integ\_map\_function, [567](#)
  - GiNaC::evalf\_map\_function, [568](#)
  - GiNaC::evalm\_map\_function, [570](#)
  - GiNaC::ex\_base\_is\_less, [605](#)
  - GiNaC::ex\_is\_equal, [605](#)
  - GiNaC::ex\_is\_less, [606](#)
  - GiNaC::ex\_swap, [606](#)
  - GiNaC::expair\_is\_less, [611](#)
  - GiNaC::expair\_rest\_is\_less, [611](#)
  - GiNaC::expair\_swap, [612](#)
  - GiNaC::expand\_map\_function, [636](#)
  - GiNaC::idx\_is\_equal\_ignore\_dim, [746](#)
  - GiNaC::is\_not\_a\_clifford, [793](#)
  - GiNaC::is\_summation\_idx, [793](#)
  - GiNaC::map\_function, [821](#)
  - GiNaC::matrix, [837](#)
  - GiNaC::normal\_map\_function, [944](#)
  - GiNaC::op0\_is\_equal, [977](#)
  - GiNaC::pointer\_to\_map\_function, [984](#)
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [987](#)

GiNaC::pointer\_to\_map\_function\_2args< T1, T2  
     >, 989  
 GiNaC::pointer\_to\_map\_function\_3args< T1, T2,  
     T3 >, 992  
 GiNaC::pointer\_to\_member\_to\_map\_function< C  
     >, 995  
 GiNaC::pointer\_to\_member\_to\_map\_function\_1arg<  
     C, T1 >, 997  
 GiNaC::pointer\_to\_member\_to\_map\_function\_2args<  
     C, T1, T2 >, 1000  
 GiNaC::pointer\_to\_member\_to\_map\_function\_3args<  
     C, T1, T2, T3 >, 1003  
 GiNaC::print\_functor, 1047  
 GiNaC::print\_functor\_impl, 1049  
 GiNaC::print\_memfun\_handler< T, C >, 1053  
 GiNaC::print\_ptrfun\_handler< T, C >, 1057  
 GiNaC::sy\_is\_less, 1201  
 GiNaC::sy\_swap, 1202  
 GiNaC::symminfo\_is\_less\_by\_orig, 1235  
 GiNaC::symminfo\_is\_less\_by\_symmterm, 1236  
 GiNaC::terminfo\_is\_less, 1263  
 std::equal\_to< GiNaC::ex >, 563  
 std::hash< GiNaC::ex >, 732  
 std::less< GiNaC::ptr< T > >, 815  
 operator+  
     GiNaC, 252, 253, 255, 256  
     GiNaC::const\_iterator, 445, 448  
 operator++  
     GiNaC, 256–258  
     GiNaC::basic\_multi\_iterator< T >, 394  
     GiNaC::const\_iterator, 445  
     GiNaC::const\_postorder\_iterator, 451  
     GiNaC::const\_preorder\_iterator, 454  
     GiNaC::multi\_iterator\_counter< T >, 891  
     GiNaC::multi\_iterator\_counter\_indv< T >, 895  
     GiNaC::multi\_iterator\_ordered< T >, 899  
     GiNaC::multi\_iterator\_ordered\_eq< T >, 903  
     GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 907  
     GiNaC::multi\_iterator\_permutation< T >, 911  
     GiNaC::multi\_iterator\_shuffle< T >, 915  
 operator+=  
     GiNaC, 254, 255  
     GiNaC::const\_iterator, 445  
 operator-  
     GiNaC, 252, 253, 256  
     GiNaC::const\_iterator, 446, 448  
 operator->  
     GiNaC::const\_iterator, 445  
     GiNaC::const\_postorder\_iterator, 450  
     GiNaC::const\_preorder\_iterator, 454  
     GiNaC::ptr< T >, 1089  
     GiNaC::structure< T, ComparisonPolicy >, 1177  
 operator--  
     GiNaC, 256–258  
     GiNaC::const\_iterator, 446  
 operator==  
     GiNaC, 254, 255  
     GiNaC::const\_iterator, 446  
 operator/  
     GiNaC, 253, 254  
 operator/=  
     GiNaC, 254, 255  
 operator=  
     GiNaC::numeric\_digits, 314  
     GiNaC::archive\_node, 345  
     GiNaC::basic, 360  
     GiNaC::numeric, 964, 965  
     GiNaC::print\_functor, 1046  
     GiNaC::ptr< T >, 1089  
 operator==  
     GiNaC, 258  
     GiNaC::const\_iterator, 446  
     GiNaC::const\_postorder\_iterator, 451  
     GiNaC::const\_preorder\_iterator, 454  
     GiNaC::internal::\_iter\_rep, 312  
     GiNaC::numeric, 970  
     GiNaC::ptr< T >, 1090, 1091  
     GiNaC::return\_type\_t, 1130  
     GiNaC::spmapkey, 1157  
 operator[]  
     GiNaC::basic, 364, 365  
     GiNaC::basic\_multi\_iterator< T >, 393  
     GiNaC::const\_iterator, 445  
     GiNaC::ex, 581, 582  
     GiNaC::structure< T, ComparisonPolicy >, 1167,  
         1168  
 operators  
     GiNaC::relational, 1112  
 operators.cpp, 1389  
 operators.h, 1391  
 options  
     factor.cpp, 1332  
     GiNaC::class\_info< OPT >, 400  
     GiNaC::expand\_map\_function, 636  
     GiNaC::print\_context, 1033  
 order  
     integration\_kernel.cpp, 1370  
 Order\_conjugate  
     GiNaC, 154  
 Order\_eval  
     GiNaC, 154  
 Order\_expl\_derivative  
     GiNaC, 155  
 Order\_imag\_part  
     GiNaC, 154  
 Order\_real\_part  
     GiNaC, 154  
 Order\_series  
     GiNaC, 154  
 orig  
     GiNaC::symminfo, 1234  
     GiNaC::terminfo, 1263  
 overall\_coeff  
     GiNaC::expairseq, 634  
 overflow  
     GiNaC::basic\_multi\_iterator< T >, 392

- overloaded
  - GiNaC::function\_options, 719
- P
  - GiNaC::modular\_form\_kernel, 864
- p
  - GiNaC::ptr< T >, 1092
- parameter\_set
  - GiNaC::fderivative, 657
- paramset
  - GiNaC, 60
- parent
  - GiNaC::class\_info< OPT >, 400
- parent\_name
  - GiNaC::print\_context\_options, 1035
  - GiNaC::registered\_class\_options, 1105
- parents\_identified
  - GiNaC::class\_info< OPT >, 400
- partition
  - GiNaC::partition\_generator, 979
  - GiNaC::partition\_with\_zero\_parts\_generator, 982
- partition\_generator
  - GiNaC::partition\_generator, 979
- partition\_with\_zero\_parts\_generator
  - GiNaC::partition\_with\_zero\_parts\_generator, 981
- pattern\_is\_not\_product
  - GiNaC::subs\_options, 1201
- pattern\_is\_product
  - GiNaC::subs\_options, 1201
- pderivative
  - GiNaC::function, 677
- permutation\_sign
  - GiNaC, 283, 284
- permute\_free\_index\_to\_front
  - GiNaC, 106
- Pi
  - GiNaC, 289
- PiEvalf
  - GiNaC, 245
- pivot
  - GiNaC::matrix, 844
- point
  - GiNaC::pseries, 1084
- pointer
  - GiNaC::const\_iterator, 444
  - GiNaC::const\_postorder\_iterator, 450
  - GiNaC::const\_preorder\_iterator, 453
- pointer\_to\_map\_function
  - GiNaC::pointer\_to\_map\_function, 984
- pointer\_to\_map\_function\_1arg
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, 987
- pointer\_to\_map\_function\_2args
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, 989
- pointer\_to\_map\_function\_3args
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, 992
- pointer\_to\_member\_to\_map\_function
  - GiNaC::pointer\_to\_member\_to\_map\_function< C >, 995
- pointer\_to\_member\_to\_map\_function\_1arg
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, 997
- pointer\_to\_member\_to\_map\_function\_2args
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, 1000
- pointer\_to\_member\_to\_map\_function\_3args
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, 1003
- pole\_error
  - GiNaC::pole\_error, 1006
- poly
  - factor.cpp, 1330
- polynomial
  - GiNaC::factor\_options, 638
  - GiNaC::info\_flags, 770
- pos\_sig
  - GiNaC::minkmetric, 854
  - GiNaC::tensepsilon, 1249
- posint
  - GiNaC::info\_flags, 770
- positive
  - GiNaC::domain, 522
  - GiNaC::info\_flags, 770
- possymbol
  - GiNaC::possymbol, 1013
- postorder\_begin
  - GiNaC::ex, 577
- postorder\_end
  - GiNaC::ex, 577
- pow
  - GiNaC, 246, 264
  - GiNaC::matrix, 836
- power
  - GiNaC::add, 333
  - GiNaC::function, 678
  - GiNaC::mul, 888
  - GiNaC::ncmul, 943
  - GiNaC::numeric, 962
  - GiNaC::power, 1020
- power.cpp, 1393
- power.h, 1394
- power\_const
  - GiNaC::pseries, 1082
- power\_dyn
  - GiNaC::numeric, 964
- power\_f
  - GiNaC::function\_options, 722
- power\_func
  - GiNaC::function\_options, 706–708, 715
- power\_funcp
  - GiNaC, 61
- power\_funcp\_1
  - GiNaC, 63
- power\_funcp\_10
  - GiNaC, 78



- power\_funcp\_11
  - GiNaC, [79](#)
- power\_funcp\_12
  - GiNaC, [81](#)
- power\_funcp\_13
  - GiNaC, [83](#)
- power\_funcp\_14
  - GiNaC, [85](#)
- power\_funcp\_2
  - GiNaC, [64](#)
- power\_funcp\_3
  - GiNaC, [66](#)
- power\_funcp\_4
  - GiNaC, [67](#)
- power\_funcp\_5
  - GiNaC, [69](#)
- power\_funcp\_6
  - GiNaC, [71](#)
- power\_funcp\_7
  - GiNaC, [72](#)
- power\_funcp\_8
  - GiNaC, [74](#)
- power\_funcp\_9
  - GiNaC, [76](#)
- power\_funcp\_exvector
  - GiNaC, [87](#)
- power\_use\_exvector\_args
  - GiNaC::function\_options, [726](#)
- pp
  - factor.cpp, [1332](#)
- precedence
  - GiNaC::add, [325](#)
  - GiNaC::basic, [363](#)
  - GiNaC::clifford, [411](#)
  - GiNaC::container< C >, [476](#)
  - GiNaC::expairseq, [620](#)
  - GiNaC::function, [671](#)
  - GiNaC::indexed, [761](#)
  - GiNaC::integral, [777](#)
  - GiNaC::mul, [875](#)
  - GiNaC::ncmul, [937](#)
  - GiNaC::numeric, [954](#)
  - GiNaC::power, [1020](#)
  - GiNaC::pseries, [1073](#)
  - GiNaC::relational, [1112](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1167](#)
- prem
  - GiNaC, [215](#)
- preorder\_begin
  - GiNaC::ex, [577](#)
- preorder\_end
  - GiNaC::ex, [577](#)
- prepend
  - GiNaC::container< C >, [481](#)
- prime
  - GiNaC::info\_flags, [770](#)
- primitive\_dirichlet\_character
  - GiNaC, [197](#)
- primpart
  - GiNaC::ex, [593](#), [594](#)
- print
  - GiNaC::\_numeric\_digits, [315](#)
  - GiNaC::basic, [362](#)
  - GiNaC::ex, [578](#)
  - GiNaC::expair, [609](#)
  - GiNaC::fderivative, [653](#)
  - GiNaC::function, [671](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1166](#)
- print.cpp, [1394](#)
- print.h, [1395](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT, [1397](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE, [1396](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON, [1396](#)
  - GINAC\_IMPLEMENT\_PRINT\_CONTEXT, [1397](#)
- print\_add
  - GiNaC::add, [332](#)
- print\_context
  - GiNaC::print\_context, [1033](#)
- print\_context\_class\_info
  - GiNaC, [89](#)
- print\_context\_options
  - GiNaC::print\_context\_options, [1034](#)
- print\_csrc
  - GiNaC::print\_csrc, [1037](#)
- print\_csrc\_cl\_N
  - GiNaC::print\_csrc\_cl\_N, [1039](#)
- print\_csrc\_double
  - GiNaC::print\_csrc\_double, [1041](#)
- print\_csrc\_float
  - GiNaC::print\_csrc\_float, [1043](#)
- print\_dflt
  - GiNaC::print\_dflt, [1045](#)
- print\_dispatch
  - GiNaC::basic, [376](#), [377](#)
- print\_dispatch\_table
  - GiNaC::function\_options, [723](#)
  - GiNaC::registered\_class\_options, [1105](#)
- print\_elements
  - GiNaC::matrix, [845](#)
- print\_func
  - GiNaC::function\_options, [715–718](#)
  - GiNaC::registered\_class\_options, [1104](#)
- print\_func< print\_context >
  - GiNaC, [128](#)
- print\_func< print\_dflt >
  - GiNaC, [93](#), [105](#), [277](#)
- print\_funcp
  - GiNaC, [61](#)
- print\_funcp\_1
  - GiNaC, [63](#)
- print\_funcp\_10
  - GiNaC, [78](#)
- print\_funcp\_11
  - GiNaC, [80](#)



- print\_funcp\_12
  - GiNaC, [81](#)
- print\_funcp\_13
  - GiNaC, [83](#)
- print\_funcp\_14
  - GiNaC, [85](#)
- print\_funcp\_2
  - GiNaC, [64](#)
- print\_funcp\_3
  - GiNaC, [66](#)
- print\_funcp\_4
  - GiNaC, [68](#)
- print\_funcp\_5
  - GiNaC, [69](#)
- print\_funcp\_6
  - GiNaC, [71](#)
- print\_funcp\_7
  - GiNaC, [73](#)
- print\_funcp\_8
  - GiNaC, [74](#)
- print\_funcp\_9
  - GiNaC, [76](#)
- print\_funcp\_exvector
  - GiNaC, [87](#)
- print\_functor
  - GiNaC::print\_functor, [1046](#)
- print\_index
  - GiNaC::idx, [744](#)
- print\_index\_dimensions
  - GiNaC::print\_options, [1055](#)
- print\_indexed
  - GiNaC::indexed, [767](#)
- print\_integer\_csrc
  - GiNaC, [228](#)
- print\_latex
  - GiNaC::print\_latex, [1050](#)
- print\_memfun\_handler
  - GiNaC::print\_memfun\_handler< T, C >, [1052](#)
- print\_numeric
  - GiNaC::numeric, [974](#)
- print\_operator
  - GiNaC, [267](#)
- print\_overall\_coeff
  - GiNaC::mul, [886](#)
- print\_power
  - GiNaC::power, [1028](#)
- print\_ptrfun\_handler
  - GiNaC::print\_ptrfun\_handler< T, C >, [1056](#)
- print\_python
  - GiNaC::print\_python, [1059](#)
- print\_python\_repr
  - GiNaC::print\_python\_repr, [1060](#)
- print\_real\_cl\_N
  - GiNaC, [230](#)
- print\_real\_csrc
  - GiNaC, [229](#)
- print\_real\_number
  - GiNaC, [228](#)
- print\_series
  - GiNaC::pseries, [1082](#)
- print\_sym\_pow
  - GiNaC, [263](#)
- print\_tree
  - GiNaC::print\_tree, [1062](#)
- print\_use\_exvector\_args
  - GiNaC::function\_options, [726](#)
- printindices
  - GiNaC::indexed, [766](#)
- printpair
  - GiNaC::expairseq, [626](#)
- printraw
  - GiNaC::archive, [339](#)
  - GiNaC::archive\_node, [350](#)
- printseq
  - GiNaC::container< C >, [480](#)
  - GiNaC::expairseq, [626](#)
- product\_to\_exvector
  - GiNaC, [134](#)
- property
  - GiNaC::archive\_node::property, [1063](#)
- property\_info
  - GiNaC::archive\_node::property\_info, [1065](#)
- property\_type
  - GiNaC::archive\_node, [344](#)
- propinfovector
  - GiNaC::archive\_node, [344](#)
- props
  - GiNaC::archive\_node, [351](#)
- pseries
  - GiNaC::pseries, [1072](#), [1073](#)
- pseries.cpp, [1398](#)
- pseries.h, [1398](#)
- psi
  - GiNaC, [157](#), [238](#), [239](#)
- psi1\_deriv
  - GiNaC, [166](#)
- psi1\_eval
  - GiNaC, [166](#)
- psi1\_evalf
  - GiNaC, [166](#)
- psi1\_series
  - GiNaC, [167](#)
- psi2\_deriv
  - GiNaC, [167](#)
- psi2\_eval
  - GiNaC, [167](#)
- psi2\_evalf
  - GiNaC, [167](#)
- psi2\_series
  - GiNaC, [168](#)
- ptr
  - GiNaC::pointer\_to\_map\_function, [985](#)
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [987](#)
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [990](#)

- GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [992](#)
- GiNaC::pointer\_to\_member\_to\_map\_function< C >, [995](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [998](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [1000](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [1003](#)
- GiNaC::ptr< T >, [1088](#)
- ptr.h, [1399](#)
- PTYPE\_BOOL
  - GiNaC::archive\_node, [345](#)
- PTYPE\_NODE
  - GiNaC::archive\_node, [345](#)
- PTYPE\_STRING
  - GiNaC::archive\_node, [345](#)
- PTYPE\_UNSIGNED
  - GiNaC::archive\_node, [345](#)
- purely\_indefinite
  - GiNaC::status\_flags, [1159](#)
- python
  - GiNaC, [261](#)
- python\_repr
  - GiNaC, [262](#)
- q\_expansion\_modular\_form
  - GiNaC::Eisenstein\_h\_kernel, [540](#)
  - GiNaC::Eisenstein\_kernel, [550](#)
  - GiNaC::modular\_form\_kernel, [863](#)
- qbar
  - integration\_kernel.cpp, [1370](#)
- quo
  - GiNaC, [214](#)
- R
  - factor.cpp, [1331](#)
- r
  - factor.cpp, [1326](#)
  - GiNaC::Eisenstein\_h\_kernel, [541](#)
- rank
  - GiNaC, [209](#)
  - GiNaC::matrix, [841](#), [842](#)
- rational
  - GiNaC::info\_flags, [770](#)
- rational\_function
  - GiNaC::info\_flags, [770](#)
- rational\_polynomial
  - GiNaC::info\_flags, [770](#)
- rbegin
  - GiNaC::container< C >, [483](#)
- read\_archive
  - GiNaC::basic, [377](#)
  - GiNaC::clifford, [411](#)
  - GiNaC::color, [433](#)
  - GiNaC::constant, [463](#)
  - GiNaC::container< C >, [477](#)
  - GiNaC::expairseq, [624](#)
  - GiNaC::fderivative, [655](#)
  - GiNaC::function, [675](#)
  - GiNaC::idx, [740](#)
  - GiNaC::indexed, [763](#)
  - GiNaC::integral, [780](#)
  - GiNaC::matrix, [834](#)
  - GiNaC::minkmetric, [853](#)
  - GiNaC::numeric, [960](#)
  - GiNaC::power, [1027](#)
  - GiNaC::pseries, [1078](#)
  - GiNaC::relational, [1114](#)
  - GiNaC::spinidx, [1145](#)
  - GiNaC::symbol, [1216](#)
  - GiNaC::symmetry, [1228](#)
  - GiNaC::tensepsilon, [1248](#)
  - GiNaC::varidx, [1284](#)
  - GiNaC::wildcard, [1293](#)
- read\_real\_float
  - GiNaC, [227](#)
- read\_unsigned
  - GiNaC, [90](#)
- real
  - GiNaC, [251](#)
  - GiNaC::domain, [522](#)
  - GiNaC::info\_flags, [770](#)
  - GiNaC::numeric, [973](#)
- real\_part
  - GiNaC, [113](#)
  - GiNaC::add, [329](#)
  - GiNaC::basic, [375](#)
  - GiNaC::constant, [463](#)
  - GiNaC::container< C >, [478](#)
  - GiNaC::ex, [582](#)
  - GiNaC::function, [674](#)
  - GiNaC::indexed, [762](#)
  - GiNaC::matrix, [833](#)
  - GiNaC::mul, [878](#)
  - GiNaC::ncmul, [940](#)
  - GiNaC::numeric, [959](#)
  - GiNaC::power, [1026](#)
  - GiNaC::pseries, [1077](#)
  - GiNaC::realsymbol, [1099](#)
  - GiNaC::symbol, [1215](#)
- real\_part\_conjugate
  - GiNaC, [140](#)
- real\_part\_eval
  - GiNaC, [139](#)
- real\_part\_evalf
  - GiNaC, [139](#)
- real\_part\_expl\_derivative
  - GiNaC, [140](#)
- real\_part\_f
  - GiNaC::function\_options, [722](#)
- real\_part\_func
  - GiNaC::function\_options, [695–697](#), [714](#)
- real\_part\_funcp
  - GiNaC, [60](#)
- real\_part\_funcp\_1

- GiNaC, [62](#)
- real\_part\_funcp\_10
  - GiNaC, [77](#)
- real\_part\_funcp\_11
  - GiNaC, [79](#)
- real\_part\_funcp\_12
  - GiNaC, [80](#)
- real\_part\_funcp\_13
  - GiNaC, [82](#)
- real\_part\_funcp\_14
  - GiNaC, [84](#)
- real\_part\_funcp\_2
  - GiNaC, [63](#)
- real\_part\_funcp\_3
  - GiNaC, [65](#)
- real\_part\_funcp\_4
  - GiNaC, [67](#)
- real\_part\_funcp\_5
  - GiNaC, [68](#)
- real\_part\_funcp\_6
  - GiNaC, [70](#)
- real\_part\_funcp\_7
  - GiNaC, [72](#)
- real\_part\_funcp\_8
  - GiNaC, [73](#)
- real\_part\_funcp\_9
  - GiNaC, [75](#)
- real\_part\_funcp\_exvector
  - GiNaC, [86](#)
- real\_part\_imag\_part
  - GiNaC, [140](#)
- real\_part\_print\_latex
  - GiNaC, [139](#)
- real\_part\_real\_part
  - GiNaC, [140](#)
- real\_part\_use\_exvector\_args
  - GiNaC::function\_options, [725](#)
- really\_subs\_idx
  - GiNaC::subs\_options, [1201](#)
- realsymbol
  - GiNaC::realsymbol, [1098](#)
- recombine\_pair\_to\_ex
  - GiNaC::add, [331](#)
  - GiNaC::expairseq, [627](#)
  - GiNaC::mul, [884](#)
- reduced\_matrix
  - GiNaC, [206](#)
- reeval\_ncmul
  - GiNaC, [211](#)
  - GiNaC::ncmul, [943](#)
- refcount
  - GiNaC::refcounted, [1102](#)
- refcounted
  - GiNaC::refcounted, [1101](#)
- reference
  - GiNaC::const\_iterator, [444](#)
  - GiNaC::const\_postorder\_iterator, [450](#)
  - GiNaC::const\_preorder\_iterator, [453](#)
- REGISTER\_FUNCTION
  - function.h, [1348](#)
  - GiNaC, [139](#), [140](#), [142](#), [144](#), [145](#), [147](#), [148](#), [150–153](#), [155](#), [160](#), [161](#), [163](#), [165](#), [166](#), [170](#), [171](#), [173](#), [176](#), [178](#), [180](#), [181](#), [183–187](#), [189](#), [190](#), [192–195](#)
- register\_new
  - GiNaC::function, [678](#)
- registered\_class\_info
  - GiNaC, [89](#)
- registered\_class\_options
  - GiNaC::registered\_class\_options, [1103](#)
- registered\_functions
  - GiNaC::function, [677](#)
- registrar.cpp, [1400](#)
- registrar.h, [1400](#)
  - GINAC\_DECLARE\_REGISTERED\_CLASS, [1402](#)
  - GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON, [1401](#)
  - GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS, [1402](#)
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS, [1403](#)
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT, [1403](#)
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T, [1403](#)
- relation
  - GiNaC::info\_flags, [770](#)
- relation\_equal
  - GiNaC::info\_flags, [770](#)
- relation\_greater
  - GiNaC::info\_flags, [770](#)
- relation\_greater\_or\_equal
  - GiNaC::info\_flags, [770](#)
- relation\_less
  - GiNaC::info\_flags, [770](#)
- relation\_less\_or\_equal
  - GiNaC::info\_flags, [770](#)
- relation\_not\_equal
  - GiNaC::info\_flags, [770](#)
- relational
  - GiNaC::relational, [1112](#)
- relational.cpp, [1404](#)
- relational.h, [1404](#)
- relative\_integration\_error
  - GiNaC::integral, [782](#)
- rem
  - GiNaC, [214](#)
- remember
  - GiNaC::function\_options, [719](#)
- remember.cpp, [1405](#)
- remember.h, [1405](#)
- remember\_assoc\_size
  - GiNaC::function\_options, [724](#)
- remember\_size
  - GiNaC::function\_options, [724](#)
- remember\_strategy

- GiNaC::function\_options, [724](#)
  - GiNaC::remember\_table, [1123](#)
  - GiNaC::remember\_table\_list, [1129](#)
- remember\_table
  - GiNaC::remember\_table, [1121](#)
- remember\_table\_entry
  - GiNaC::function, [679](#)
  - GiNaC::remember\_table\_entry, [1125](#)
- remember\_table\_list
  - GiNaC::remember\_table\_list, [1128](#)
- remember\_tables
  - GiNaC::remember\_table, [1122](#)
- remove\_all
  - GiNaC::container< C >, [482](#)
- remove\_dirac\_ONE
  - GiNaC, [99](#)
- remove\_first
  - GiNaC::container< C >, [481](#)
- remove\_last
  - GiNaC::container< C >, [481](#)
- remove\_reference
  - GiNaC::refcounted, [1101](#)
- rename\_dummy\_indices
  - GiNaC, [133](#)
- rename\_dummy\_indices\_uniquely
  - GiNaC, [135](#), [136](#)
- rend
  - GiNaC::container< C >, [483](#)
- replace\_contr\_index
  - GiNaC::tensor, [1261](#)
- replace\_dim
  - GiNaC::idx, [743](#)
- replace\_with\_symbol
  - GiNaC, [224](#), [225](#)
- reposition\_dummy\_indices
  - GiNaC, [133](#)
  - GiNaC::indexed, [768](#)
- representation\_label
  - GiNaC::clifford, [416](#)
  - GiNaC::color, [435](#)
- reserve
  - GiNaC::container\_storage< C >, [487](#)
- rest
  - GiNaC::expair, [610](#)
- result
  - GiNaC::remember\_table\_entry, [1126](#)
- result\_type
  - GiNaC::map\_function, [821](#)
- resultant
  - GiNaC, [226](#)
- return\_type
  - GiNaC::add, [330](#)
  - GiNaC::basic, [374](#)
  - GiNaC::clifford, [413](#)
  - GiNaC::color, [434](#)
  - GiNaC::ex, [599](#)
  - GiNaC::expairseq, [624](#)
  - GiNaC::fail, [642](#)
  - GiNaC::function, [676](#)
  - GiNaC::function\_options, [723](#)
  - GiNaC::indexed, [764](#)
  - GiNaC::integral, [779](#)
  - GiNaC::matrix, [834](#)
  - GiNaC::minkmetric, [854](#)
  - GiNaC::mul, [882](#)
  - GiNaC::ncmul, [941](#)
  - GiNaC::power, [1028](#)
  - GiNaC::relational, [1115](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1176](#)
  - GiNaC::su3d, [1183](#)
  - GiNaC::su3f, [1189](#)
  - GiNaC::tensdelta, [1241](#)
  - GiNaC::tensepsilon, [1248](#)
  - GiNaC::tensmetric, [1256](#)
  - GiNaC::tensor, [1261](#)
- return\_type\_tinfo
  - GiNaC::add, [330](#)
  - GiNaC::basic, [375](#)
  - GiNaC::clifford, [413](#)
  - GiNaC::color, [434](#)
  - GiNaC::ex, [600](#)
  - GiNaC::function, [677](#)
  - GiNaC::function\_options, [724](#)
  - GiNaC::indexed, [764](#)
  - GiNaC::integral, [780](#)
  - GiNaC::mul, [882](#)
  - GiNaC::ncmul, [941](#)
  - GiNaC::power, [1028](#)
  - GiNaC::relational, [1115](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1176](#)
- rh
  - GiNaC::relational, [1118](#)
- rhs
  - GiNaC, [120](#)
  - GiNaC::ex, [582](#)
  - GiNaC::relational, [1117](#)
- rl
  - GiNaC::return\_type\_t, [1130](#)
- root
  - GiNaC::archive::archived\_ex, [355](#)
- rotate\_left
  - GiNaC, [282](#)
- row
  - GiNaC::matrix, [846](#)
- rows
  - GiNaC, [207](#)
  - GiNaC::matrix, [835](#)
- s
  - GiNaC::const\_postorder\_iterator, [452](#)
  - GiNaC::const\_preorder\_iterator, [455](#)
  - GiNaC::derivative\_map\_function, [493](#)
  - GiNaC::Eisenstein\_h\_kernel, [541](#)
  - GiNaC::print\_context, [1033](#)
  - GiNaC::symbolset, [1221](#)
- S\_deriv
  - GiNaC, [171](#)

- S\_eval
  - GiNaC, 170
- S\_evalf
  - GiNaC, 170
- S\_print\_latex
  - GiNaC, 171
- S\_series
  - GiNaC, 171
- safe\_bool
  - GiNaC::relational, 1112
- same\_metric
  - GiNaC::clifford, 414
- scalar\_mul\_indexed
  - GiNaC::basic, 373
  - GiNaC::matrix, 832
  - GiNaC::structure< T, ComparisonPolicy >, 1175
- seq
  - GiNaC::container\_storage< C >, 488
  - GiNaC::expairseq, 633
  - GiNaC::pseries, 1084
  - GiNaC::remember\_table\_entry, 1126
- serial
  - GiNaC::constant, 466
  - GiNaC::function, 680
  - GiNaC::G2\_SERIAL, 727
  - GiNaC::G3\_SERIAL, 728
  - GiNaC::iterated\_integral2\_SERIAL, 794
  - GiNaC::iterated\_integral3\_SERIAL, 795
  - GiNaC::psi1\_SERIAL, 1085
  - GiNaC::psi2\_SERIAL, 1086
  - GiNaC::symbol, 1219
  - GiNaC::zeta1\_SERIAL, 1295
  - GiNaC::zeta2\_SERIAL, 1296
- series
  - GiNaC, 117
  - GiNaC::add, 327
  - GiNaC::basic, 370
  - GiNaC::Eisenstein\_h\_kernel, 537
  - GiNaC::Eisenstein\_kernel, 548
  - GiNaC::ex, 589
  - GiNaC::fderivative, 654
  - GiNaC::function, 673
  - GiNaC::integral, 781
  - GiNaC::integration\_kernel, 788
  - GiNaC::modular\_form\_kernel, 861
  - GiNaC::mul, 879
  - GiNaC::power, 1024
  - GiNaC::pseries, 1075
  - GiNaC::structure< T, ComparisonPolicy >, 1172
  - GiNaC::symbol, 1213
- series\_coeff
  - GiNaC::integration\_kernel, 791
- series\_coeff\_impl
  - GiNaC::basic\_log\_kernel, 389
  - GiNaC::Ebar\_kernel, 529
  - GiNaC::ELi\_kernel, 561
  - GiNaC::integration\_kernel, 790
  - GiNaC::Kronecker\_dtau\_kernel, 802
  - GiNaC::Kronecker\_dz\_kernel, 811
  - GiNaC::multiple\_polylog\_kernel, 928
- series\_f
  - GiNaC::function\_options, 723
- series\_func
  - GiNaC::function\_options, 709–711, 715
- series\_funcp
  - GiNaC, 61
- series\_funcp\_1
  - GiNaC, 63
- series\_funcp\_10
  - GiNaC, 78
- series\_funcp\_11
  - GiNaC, 79
- series\_funcp\_12
  - GiNaC, 81
- series\_funcp\_13
  - GiNaC, 83
- series\_funcp\_14
  - GiNaC, 85
- series\_funcp\_2
  - GiNaC, 64
- series\_funcp\_3
  - GiNaC, 66
- series\_funcp\_4
  - GiNaC, 67
- series\_funcp\_5
  - GiNaC, 69
- series\_funcp\_6
  - GiNaC, 71
- series\_funcp\_7
  - GiNaC, 73
- series\_funcp\_8
  - GiNaC, 74
- series\_funcp\_9
  - GiNaC, 76
- series\_funcp\_exvector
  - GiNaC, 87
- series\_to\_poly
  - GiNaC, 265
- series\_use\_exvector\_args
  - GiNaC::function\_options, 726
- series\_vec
  - GiNaC::integration\_kernel, 792
- set
  - GiNaC::matrix, 838
- set\_cache\_step
  - GiNaC::integration\_kernel, 791
- set\_name
  - GiNaC::function\_options, 687
  - GiNaC::symbol, 1218
- set\_print\_context
  - GiNaC, 260
- set\_print\_func
  - GiNaC, 266
  - GiNaC::function\_options, 720
  - GiNaC::registered\_class\_options, 1104
- set\_print\_options

- GiNaC, [260](#)
- set\_refcount
  - GiNaC::refcounted, [1102](#)
- set\_return\_type
  - GiNaC::function\_options, [718](#)
- set\_symmetry
  - GiNaC::function\_options, [719](#)
- set\_TeX\_name
  - GiNaC::symbol, [1218](#)
- set\_type
  - GiNaC::symmetry, [1229](#)
- setflag
  - GiNaC::basic, [380](#)
- shaker\_sort
  - GiNaC, [284](#)
- share
  - GiNaC::ex, [602](#)
- shift\_exponents
  - GiNaC::pseries, [1082](#)
- show\_statistics
  - GiNaC::remember\_table, [1122](#)
- simplify\_indexed
  - GiNaC, [118](#), [134](#)
  - GiNaC::ex, [596](#)
  - GiNaC::indexed, [768](#)
- simplify\_indexed\_product
  - GiNaC, [134](#)
  - GiNaC::indexed, [768](#)
- sin
  - GiNaC, [231](#)
- sin\_conjugate
  - GiNaC, [179](#)
- sin\_deriv
  - GiNaC, [179](#)
- sin\_eval
  - GiNaC, [179](#)
- sin\_evalf
  - GiNaC, [178](#)
- sin\_imag\_part
  - GiNaC, [179](#)
- sin\_real\_part
  - GiNaC, [179](#)
- sinh
  - GiNaC, [234](#)
- sinh\_conjugate
  - GiNaC, [189](#)
- sinh\_deriv
  - GiNaC, [188](#)
- sinh\_eval
  - GiNaC, [188](#)
- sinh\_evalf
  - GiNaC, [188](#)
- sinh\_imag\_part
  - GiNaC, [188](#)
- sinh\_real\_part
  - GiNaC, [188](#)
- size
  - GiNaC::basic\_multi\_iterator< T >, [392](#)
- smod
  - GiNaC, [242](#)
  - GiNaC::add, [328](#)
  - GiNaC::basic, [372](#)
  - GiNaC::ex, [595](#)
  - GiNaC::mul, [880](#)
  - GiNaC::numeric, [958](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1173](#)
- solve
  - GiNaC::matrix, [841](#)
- sort
  - GiNaC::container< C >, [482](#)
- sort\_
  - GiNaC::container< C >, [480](#)
- spinidx
  - GiNaC::spinidx, [1144](#)
- spinor\_metric
  - GiNaC, [279](#)
- split\_ex\_to\_pair
  - GiNaC::add, [331](#)
  - GiNaC::expairseq, [626](#)
  - GiNaC::mul, [883](#)
- spm
  - GiNaC::scalar\_products, [1134](#)
- spmap
  - GiNaC, [88](#)
- spmapkey
  - GiNaC::spmapkey, [1156](#), [1157](#)
- sprem
  - GiNaC, [216](#)
- sqrfree
  - GiNaC, [223](#)
- sqrfree\_parfrac
  - GiNaC, [224](#)
- sqrfree\_yun
  - GiNaC, [222](#)
- sqrt
  - GiNaC, [245](#), [265](#)
- sr\_gcd
  - GiNaC, [218](#)
- STATISTICS
  - normal.cpp, [1382](#)
- std, [310](#)
  - swap, [310](#)
- std::equal\_to< GiNaC::ex >, [562](#)
  - operator(), [563](#)
- std::hash< GiNaC::ex >, [732](#)
  - operator(), [732](#)
- std::less< GiNaC::ptr< T > >, [814](#)
  - operator(), [815](#)
- std::less< ptr< T > >
  - GiNaC::ptr< T >, [1090](#)
- step
  - GiNaC, [247](#)
  - GiNaC::numeric, [965](#)
- step\_conjugate
  - GiNaC, [145](#)
- step\_eval

- GiNaC, [144](#)
- step\_evalf
  - GiNaC, [144](#)
- step\_imag\_part
  - GiNaC, [145](#)
- step\_real\_part
  - GiNaC, [145](#)
- step\_series
  - GiNaC, [145](#)
- STLT
  - GiNaC::container< C >, [473](#)
  - GiNaC::container\_storage< C >, [486](#)
- store\_remember\_table
  - GiNaC::function, [678](#)
- struct\_compare
  - GiNaC::compare\_all\_equal< T >, [436](#)
  - GiNaC::compare\_bitwise< T >, [437](#)
  - GiNaC::compare\_std\_less< T >, [439](#)
- struct\_is\_equal
  - GiNaC::compare\_all\_equal< T >, [436](#)
  - GiNaC::compare\_bitwise< T >, [437](#)
  - GiNaC::compare\_std\_less< T >, [438](#)
- structure
  - GiNaC::structure< T, ComparisonPolicy >, [1165](#)
- structure.h, [1406](#)
- sub
  - GiNaC::matrix, [835](#)
  - GiNaC::numeric, [961](#)
- sub\_dyn
  - GiNaC::numeric, [963](#)
- sub\_matrix
  - GiNaC, [206](#)
- subs
  - GiNaC, [121](#)
  - GiNaC::basic, [367](#)
  - GiNaC::clifford, [415](#)
  - GiNaC::container< C >, [477](#)
  - GiNaC::ex, [584](#), [585](#)
  - GiNaC::expairseq, [623](#)
  - GiNaC::idx, [740](#)
  - GiNaC::matrix, [832](#)
  - GiNaC::numeric, [957](#)
  - GiNaC::power, [1024](#)
  - GiNaC::pseries, [1076](#)
  - GiNaC::relational, [1114](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1169](#)
  - GiNaC::symbol, [1214](#)
- subs\_algebraic
  - GiNaC::subs\_options, [1201](#)
- subs\_no\_pattern
  - GiNaC::subs\_options, [1201](#)
- subs\_one\_level
  - GiNaC::basic, [378](#)
- subchildren
  - GiNaC::container< C >, [484](#)
  - GiNaC::expairseq, [633](#)
- subvalue
  - GiNaC, [195](#)
- successful\_hits
  - GiNaC::remember\_table\_entry, [1126](#)
- sufficiently\_accurate
  - GiNaC::lanczos\_coeffs, [813](#)
- suppress\_branchcut
  - GiNaC::series\_options, [1135](#)
- swap
  - GiNaC, [120](#), [125](#)
  - GiNaC::ex, [576](#)
  - GiNaC::expair, [609](#)
  - GiNaC::ptr< T >, [1090](#)
  - std, [310](#)
- swapped
  - GiNaC::sy\_swap, [1203](#)
- sy\_anti
  - GiNaC, [274](#), [275](#)
- sy\_cycl
  - GiNaC, [275](#), [276](#)
- sy\_is\_less
  - GiNaC::sy\_is\_less, [1201](#)
  - GiNaC::symmetry, [1231](#)
- sy\_none
  - GiNaC, [273](#)
- sy\_swap
  - GiNaC::sy\_swap, [1202](#)
  - GiNaC::symmetry, [1231](#)
- sy\_symm
  - GiNaC, [274](#)
- sym
  - GiNaC::sym\_desc, [1205](#)
- sym\_desc
  - GiNaC::sym\_desc, [1205](#)
- sym\_desc\_vec
  - GiNaC, [88](#)
- symbol
  - GiNaC::info\_flags, [770](#)
  - GiNaC::symbol, [1212](#)
- symbol.cpp, [1407](#)
- symbol.h, [1407](#)
- symbolic\_matrix
  - GiNaC, [205](#), [209](#)
- symbolset
  - GiNaC::symbolset, [1221](#)
- symm
  - GiNaC, [272](#)
  - GiNaC::terminfo, [1263](#)
- symmetric
  - GiNaC::symmetry, [1227](#)
- symmetric2
  - GiNaC, [270](#)
- symmetric3
  - GiNaC, [270](#)
- symmetric4
  - GiNaC, [270](#)
- symmetrize
  - GiNaC, [118](#), [119](#), [272](#), [276](#)
  - GiNaC::ex, [598](#)
- symmetrize\_cyclic

- GiNaC, [119](#), [272](#), [276](#)
  - GiNaC::ex, [599](#)
- symmetry
  - GiNaC::symmetry, [1227](#)
- symmetry.cpp, [1408](#)
- symmetry.h, [1409](#)
- symmetry\_type
  - GiNaC::symmetry, [1227](#)
- symminfo
  - GiNaC::symminfo, [1234](#)
- symmterm
  - GiNaC::symminfo, [1234](#)
- syms
  - factor.cpp, [1332](#)
- syms\_wox
  - factor.cpp, [1331](#)
- symtree
  - GiNaC::function\_options, [727](#)
  - GiNaC::indexed, [769](#)
- synthesize\_func
  - GiNaC, [58](#)
- table\_size
  - GiNaC::remember\_table, [1123](#)
- tan
  - GiNaC, [232](#)
- tan\_conjugate
  - GiNaC, [183](#)
- tan\_deriv
  - GiNaC, [182](#)
- tan\_eval
  - GiNaC, [182](#)
- tan\_evalf
  - GiNaC, [181](#)
- tan\_imag\_part
  - GiNaC, [182](#)
- tan\_real\_part
  - GiNaC, [182](#)
- tan\_series
  - GiNaC, [182](#)
- tanh
  - GiNaC, [234](#)
- tanh\_conjugate
  - GiNaC, [192](#)
- tanh\_deriv
  - GiNaC, [191](#)
- tanh\_eval
  - GiNaC, [191](#)
- tanh\_evalf
  - GiNaC, [190](#)
- tanh\_imag\_part
  - GiNaC, [191](#)
- tanh\_real\_part
  - GiNaC, [191](#)
- tanh\_series
  - GiNaC, [191](#)
- tau
  - GiNaC::Kronecker\_dz\_kernel, [812](#)
- tcoeff
  - GiNaC::ex, [588](#)
- tensepsilon
  - GiNaC::tensepsilon, [1247](#)
- tensor
  - GiNaC, [289](#)
- tensor.cpp, [1411](#)
- tensor.h, [1412](#)
- terminfo
  - GiNaC::terminfo, [1263](#)
- test
  - GiNaC::has\_distance< T >, [731](#)
- test\_and\_set\_nparams
  - GiNaC::function\_options, [720](#)
- TEST\_PERMUTATION
  - color.cpp, [1310](#)
- TeX\_name
  - GiNaC::constant, [466](#)
  - GiNaC::function\_options, [721](#)
  - GiNaC::symbol, [1220](#)
- tgamma
  - GiNaC, [238](#)
- tgamma\_conjugate
  - GiNaC, [164](#)
- tgamma\_deriv
  - GiNaC, [164](#)
- tgamma\_eval
  - GiNaC, [164](#)
- tgamma\_evalf
  - GiNaC, [163](#)
- tgamma\_series
  - GiNaC, [164](#)
- thiscontainer
  - GiNaC::clifford, [412](#), [413](#)
  - GiNaC::color, [434](#)
  - GiNaC::container< C >, [479](#), [480](#)
  - GiNaC::fderivative, [654](#)
  - GiNaC::function, [673](#), [674](#)
  - GiNaC::indexed, [764](#)
  - GiNaC::ncmul, [939](#), [940](#)
- thisexpairseq
  - GiNaC::add, [330](#)
  - GiNaC::expairseq, [625](#), [626](#)
  - GiNaC::mul, [882](#), [883](#)
- tinfo
  - GiNaC::return\_type\_t, [1130](#)
- tinfo\_key
  - GiNaC::registered\_class\_options, [1105](#)
- to\_cl\_N
  - GiNaC::numeric, [973](#)
- to\_double
  - GiNaC, [250](#)
  - GiNaC::numeric, [973](#)
- to\_int
  - GiNaC, [250](#)
  - GiNaC::numeric, [972](#)
- to\_long
  - GiNaC, [250](#)
  - GiNaC::numeric, [972](#)



- to\_polynomial
  - GiNaC, [116](#)
  - GiNaC::basic, [371](#)
  - GiNaC::ex, [590](#)
  - GiNaC::expairseq, [622](#)
  - GiNaC::numeric, [958](#)
  - GiNaC::power, [1026](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1173](#)
  - GiNaC::symbol, [1215](#)
- to\_rational
  - GiNaC, [116](#)
  - GiNaC::basic, [371](#)
  - GiNaC::ex, [590](#)
  - GiNaC::expairseq, [622](#)
  - GiNaC::numeric, [957](#)
  - GiNaC::power, [1025](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1172](#)
  - GiNaC::symbol, [1214](#)
- toggle\_dot
  - GiNaC::spinidx, [1147](#)
- toggle\_variance
  - GiNaC::varidx, [1286](#)
- toggle\_variance\_dot
  - GiNaC::spinidx, [1147](#)
- too\_late
  - GiNaC::\_numeric\_digits, [315](#)
- trace
  - GiNaC, [208](#)
  - GiNaC::matrix, [839](#)
- trace\_string
  - GiNaC, [97](#)
- transpose
  - GiNaC, [207](#)
  - GiNaC::matrix, [838](#)
- traverse
  - GiNaC::ex, [586](#)
- traverse\_postorder
  - GiNaC::ex, [586](#)
- traverse\_preorder
  - GiNaC::ex, [586](#)
- tree
  - GiNaC, [262](#)
- tree\_node
  - GiNaC::class\_info< OPT >::tree\_node, [1264](#)
- trig\_info
  - GiNaC, [180](#)
- trivial
  - GiNaC::composition\_generator, [441](#)
- tryfactsubs
  - GiNaC, [210](#)
- type
  - GiNaC::archive\_node::property, [1063](#)
  - GiNaC::archive\_node::property\_info, [1065](#)
  - GiNaC::symmetry, [1231](#)
- uintvector
  - GiNaC, [88](#)
- unarch\_map
  - GiNaC::unarchive\_table\_t, [1267](#)
- unarch\_table\_instance
  - GiNaC, [288](#)
- unarchive
  - GiNaC::archive\_node, [349](#)
- unarchive\_ex
  - GiNaC::archive, [337](#)
- unarchive\_map\_t
  - GiNaC, [58](#)
- unarchive\_table\_t
  - GiNaC::unarchive\_table\_t, [1266](#)
- unatomize
  - GiNaC::archive, [340](#)
- unique
  - GiNaC::container< C >, [482](#)
- unique\_
  - GiNaC::container< C >, [481](#), [484](#)
- unit
  - factor.cpp, [1331](#)
  - GiNaC::ex, [592](#)
- unit\_matrix
  - GiNaC, [205](#), [209](#)
- unitcontprim
  - GiNaC::ex, [594](#)
- unlikely
  - compiler.h, [1312](#)
- unlink\_ex
  - GiNaC, [125](#)
- unsignedvector
  - GiNaC, [88](#)
- USE\_REMEMBER
  - normal.cpp, [1382](#)
- use\_remember
  - GiNaC::function\_options, [724](#)
- use\_return\_type
  - GiNaC::function\_options, [723](#)
- USE\_SAME\_DEGREE\_FACTOR
  - factor.cpp, [1326](#)
- use\_sr\_gcd
  - GiNaC::gcd\_options, [729](#)
- USE\_TRIAL\_DIVISION
  - normal.cpp, [1382](#)
- usecount
  - GiNaC::unarchive\_table\_t, [1267](#)
- used\_indices
  - GiNaC::make\_flat\_inserter, [819](#)
- user\_defined\_kernel
  - GiNaC::user\_defined\_kernel, [1274](#)
- uses\_Laurent\_series
  - GiNaC::Eisenstein\_h\_kernel, [539](#)
  - GiNaC::Eisenstein\_kernel, [550](#)
  - GiNaC::integration\_kernel, [790](#)
  - GiNaC::modular\_form\_kernel, [863](#)
  - GiNaC::user\_defined\_kernel, [1275](#)
- utils.cpp, [1413](#)
- utils.h, [1415](#)
  - DEFAULT\_COMPARE, [1417](#)
  - DEFAULT\_CTOR, [1417](#)
  - DEFAULT\_PRINT, [1417](#)

- DEFAULT\_PRINT\_LATEX, [1417](#)
- utils\_multi\_iterator.h, [1418](#)
- v
  - GiNaC::basic\_multi\_iterator< T >, [395](#)
  - GiNaC::sy\_is\_less, [1202](#)
  - GiNaC::sy\_swap, [1203](#)
- v1
  - GiNaC::spmapkey, [1157](#)
- v2
  - GiNaC::spmapkey, [1158](#)
- v\_internal
  - GiNaC::multi\_iterator\_shuffle< T >, [916](#)
- v\_orig
  - GiNaC::multi\_iterator\_shuffle< T >, [916](#)
- validate
  - GiNaC::indexed, [767](#)
  - GiNaC::symmetry, [1229](#)
- value
  - factor.cpp, [1326](#)
  - GiNaC::archive\_node::property, [1064](#)
  - GiNaC::composition\_generator::coolmulti::element, [553](#)
  - GiNaC::has\_distance< T >, [730](#)
  - GiNaC::idx, [745](#)
  - GiNaC::numeric, [976](#)
- value\_type
  - GiNaC::const\_iterator, [443](#)
  - GiNaC::const\_postorder\_iterator, [449](#)
  - GiNaC::const\_preorder\_iterator, [453](#)
- var
  - GiNaC::pseries, [1084](#)
- varidx
  - GiNaC::varidx, [1283](#)
- version.h, [1419](#)
  - GINAC\_LT\_AGE, [1420](#)
  - GINAC\_LT\_CURRENT, [1420](#)
  - GINAC\_LT\_REVISION, [1420](#)
  - GINACLIB\_ARCHIVE\_AGE, [1421](#)
  - GINACLIB\_ARCHIVE\_VERSION, [1420](#)
  - GINACLIB\_MAJOR\_VERSION, [1420](#)
  - GINACLIB\_MICRO\_VERSION, [1420](#)
  - GINACLIB\_MINOR\_VERSION, [1420](#)
  - GINACLIB\_STR, [1421](#)
  - GINACLIB\_STR\_HELPER, [1421](#)
  - GINACLIB\_VERSION, [1421](#)
- version\_major
  - GiNaC, [292](#)
- version\_micro
  - GiNaC, [292](#)
- version\_minor
  - GiNaC, [292](#)
- vn
  - factor.cpp, [1332](#)
- vnlst
  - factor.cpp, [1332](#)
- wild
  - GiNaC, [288](#)
- wildcard
  - GiNaC::wildcard, [1292](#)
- wildcard.cpp, [1421](#)
- wildcard.h, [1422](#)
- write\_real\_float
  - GiNaC, [228](#)
- write\_unsigned
  - GiNaC, [90](#)
- x
  - factor.cpp, [1330](#)
  - GiNaC::basic\_partition\_generator::mpartition2, [865](#)
  - GiNaC::Ebar\_kernel, [530](#)
  - GiNaC::ELi\_kernel, [562](#)
  - GiNaC::integral, [782](#)
  - GiNaC::user\_defined\_kernel, [1276](#)
  - integration\_kernel.cpp, [1371](#)
- y
  - GiNaC::Ebar\_kernel, [530](#)
  - GiNaC::ELi\_kernel, [562](#)
- yes\_type
  - GiNaC::has\_distance< T >, [730](#)
- z
  - GiNaC::Kronecker\_dtau\_kernel, [803](#)
  - GiNaC::multiple\_polylog\_kernel, [928](#)
- z\_j
  - GiNaC::Kronecker\_dz\_kernel, [812](#)
- zeta
  - GiNaC, [156](#), [237](#)
- zeta1\_deriv
  - GiNaC, [173](#)
- zeta1\_eval
  - GiNaC, [173](#)
- zeta1\_evalf
  - GiNaC, [173](#)
- zeta1\_print\_latex
  - GiNaC, [173](#)
- zeta2\_deriv
  - GiNaC, [174](#)
- zeta2\_eval
  - GiNaC, [174](#)
- zeta2\_evalf
  - GiNaC, [174](#)
- zeta2\_print\_latex
  - GiNaC, [174](#)
- zetaderiv\_deriv
  - GiNaC, [150](#)
- zetaderiv\_eval
  - GiNaC, [150](#)