

Index

Symbols

_1, 240, 248, 273
_2, 248, 273
_3, 248, 273
_9, 240
***** (repeat), 147
... construct, 233
+ (repeat), 147
? (repeat), 149
^ metacharacter, 152

A

Abrahams, David, xxv, xxviii, xxxiii, xxxvi, xxxviii
accessing
 elements by index, 223
 stored values, 163
 tuple elements, 217, 220
adapters, Standard Library, 186-188
addable, 108
addable classes, 106
addition, 126
addressof, 92-95
add_prev, 292
add_ref, 34
ADL (argument dependent lookup), 36
Adler, Darin, xxxviii
advantages
 function objects, 261-262
 smart pointers, 176
algorithms, customizing, 237-238
Allison, Chuck, xxiv
andable, 109
antisymmetry, 119
any
 accessing stored values, 163
 empty instances, 161-162

 functions, 170-172
 predicates, 181-182
 storing in pointers, 172-176
 swapping values, 170-172
 testing, 184-185
 testing for empty values, 170-172
Any library, xxiii, 159
 storing types, 159-161, 164-167
 usage, 163-164
any_cast, 163
any_out class storage, 177-181
applications
 creating, 246
 protecting, 153
apply_visitor, 200
argument dependent lookup (ADL), 36
arguments
 bind, 247-249
 binding, xxvi, 328-334
 placeholders for in bind, 239
 signals, 351-356
arithmetic operations, 285-286
arithmetic types, 126
Array, xxiii
arrays
 dynamically allocated, 14
 shared ownership, 29
assertions, static, 80-82
Assign, xxxiv
assignments, 65-68
associative containers, 114
Austern, Matt, xxiv
auto_ptr, 5, 9
auto_ptr const, 13

B

back references, 142
bad regular expressions, 152-154
bad_any_cast exception, 163
bad_numeric_cast, 67
Bandela, John, xxiii
Barton, John, 117
Barton-Nackmann trick, 117
base class chaining, 120
base&, 219
basic_regex, 136-139
behavior, 66
Big Three, 90-92
binary visitors, 203, 205
binary_function, 292
binary_search, 357
bind
arguments, 247-249
implementing, 239
placeholders, 239-240, 247-249
semantics, 257-259
Standard Library, 238
Bind library, xxvii, 237-239, 275
combining with Function library, 328-334
creating slots, 367-370
how it works, 245-247
usage, 240
virtual functions, 259
bind1st, 238
bind2nd, 238
binders
function objects, 239
generalized binders, 237
state, 263-266
binding, 275
functions, 239, 259-260
to member variables, 260-261
versus not binding, 261-263
binds
nested binds, 254
virtual functions, 259
bloating, 120
blocks, try/catch, 167
Boost Utility, 79, 86
addressof, 92-95
BOOST_STATIC_ASSERT, 80-82
checked_delete, 82-86
disable_if, 98-102
enable_if, 95-103
libraries, list of, xxii-xxxvii
noncopyable, 87-92
BOOST_STATIC_ASSERT, 80-82
BOOST_VARIANT_ENUM_PARAMS, 206
bounded variant types, 191-195
bounds operators, 140
Brönniman, Hervé, xxxii

C

C++
language shortcomings, 235
threading, xxxix
C++ Standard Library, 136
C++ Template Metaprogramming, xxx
Cacciola, Fernando, xxx, xxxiii, xxxvii
callback functions, 235
callbacks, 319-321, 341
calling functions
member functions, 241-244
multiple functions, 255-256
Call_traits, xxix
case_statement, 298
cast functions (Conversion library), 53
lexical_cast, 73-77
numeric_cast, 64-72
polymorphic_cast, 54-61
polymorphic_downcast, 61-64
casting lambda expressions, 300-303
casts, optimizations, 62
catch_all, 309
catch_exception, 309
character class, 141
character classes, negated, 152
checked_array_delete, 86
checked_delete, 82-86
checking range, 71
Cheshire Cat idiom, 11
circular dependencies, 346
class template variant, 193
class templates, xxvi, 337
classes, 257
addable, 106
any_out storage, 177-181
creating, 246
enabling lexical_cast, 76-77
flexibility, 332
implementing pimpl idiom, 11
property classes, 167-170
shiftable, 106
sig, 292
sig_helper, 294
clear code, 255
Cleary, Steve, xxiv, xxix, xxxvii
code, 255
Colvin, Greg, xxxviii
Combiner, 344, 360-363
Command pattern, 329
comparing tuples, 221-223
compatible syntax, 315
compile time, computation at, 82
compliance (Standard Library adapters), 186-188
compose1, 238, 255
compose2, 238

- compose_f_gx**, 255
 - compose_f_gx_hx**, 255
 - composing objects**, 253-255
 - composition**, 255-256
 - Compressed_pair**, xxiv
 - computation at compile time**, 82
 - concepts, definitions**, xxxi
 - Concept_check**, xxix
 - connecting slots to signals**, 344
 - constant**, 280
 - constructing**
 - in lambda expressions, 303-307
 - tuples, 215-216
 - constructors**, 303
 - copy, 91-92
 - shared_ptr, 18
 - simple_bind, 245
 - Tuple library, 212
 - const_parameters**, 297
 - containers**, 157
 - sorting, 249-253
 - types, 183
 - contains function**, 182
 - contains predicate**, 186-188
 - control structures**, 294-298
 - Conversion library**, xxxvi, 53
 - cast functions, 53
 - lexical_cast*, 73-77
 - numeric_cast*, 64-72
 - polymorphic_cast*, 54-55, 61
 - polymorphic_downcast*, 61-64
 - conversions**
 - integers, 64
 - integral types, 64
 - pointer types, 54
 - Coplien, James**, 117
 - copy assignment operations**, 89
 - copy assignment operator**, 89, 316
 - copy construction operations**, 89
 - copy constructors**, 89-92, 363
 - copying objects**, 258-259
 - counting non-empty values**, 182-183
 - Crc**, xxxvi
 - CRC (cyclic redundancy codes)**, xxxvi
 - creating**
 - lambda expressions, 273
 - named variables, 281
 - shared_ptr from a weak_ptr, 49-50
 - shared_ptr from this, 27
 - slots using Bind and Lambda, 367-370
 - cref**, 217
 - criteria, sorting**, 249-253
 - crosscasting**, 55
 - Curiously Recurring Template Pattern**, 117
 - custom deleters**, 24-27
 - customizing algorithms**, 237-238
 - cyclic redundancy codes (CRC)**, xxxvi
- D**
- \d shortcut**, 140
 - data structures**, 157
 - Date_time**, xxxvi
 - Dawes, Beman**, xxiv, xxix, xxxii-xxxiv, xxxviii
 - de Guzman, Joel**, xxii
 - declarations using preferred syntax**, 314
 - declaring**
 - signals, 346
 - variables, 139
 - decoupling**, 346
 - decrementable**, 110
 - default_statement**, 298
 - defining**
 - classes, 167-170
 - free functions, 174
 - functions, 240
 - sorting criteria, 249-253
 - unnamed functions, 270
 - definitions, concepts**, xxxi
 - deleters**, 16, 24-27
 - deleting**
 - objects, 83
 - dynamically allocated*, 86
 - through pointers*, 82
 - pointers, 84-85
 - dereferenceable**, 111
 - dereferencing regex_token_iterator**, 150
 - dereferencing operators**, 110
 - destinations unsigned integral types**, 68-69
 - destroying pointers**, 172-173
 - destructing in lambda expressions**, 303-307
 - destructors**
 - shared_ptr, 19
 - weak_ptr, 44
 - determining types**, 245-246
 - Dijkstra's shortest path**, xxv
 - Dimov, Peter**, xxvii-xxviii, xxxviii
 - disable_if**, 98-102
 - discarding overload**, 233
 - disconnecting slots**, 345
 - discriminated types**, 188
 - discriminated unions**, 191
 - divides**, 285
 - downcast**, 55-57
 - downcasting**, 55
 - downcasts, static_cast for**, 61-64
 - do_**, 299
 - do_while_loop**, 299
 - Droba, Pavol**, xxii
 - dynamically allocated arrays**, 14

dynamically allocated objects, 86

Dynamic_bitset, xxiv

dynamic_cast, 54, 61

 null pointers, 54

 reference types, 55

 safety, 59

 versus `polymorphic_cast`, 58-59

E

elements

 accessing by index, 223

 storing, 241

element_less, 223

ellipsis (...) construct, 233

empty base optimization, 120

empty instances, 161-162

empty values, testing for, 170-172

Enable_if, xxx, 95-103

enabling classes, 76-77

enclosing subexpressions, 139

equality vs. equivalence, 124-126

equality_comparable, 107

equivalence relation, 116

equivalent, 110

error handling, 59

exception handling, 308-311

exception safety, 4

exceptions

bad_any_cast, 163

std::bad_cast, 54

expanding matches, 146

expression templates, 270

expressions

 lambda expressions, 290

casting, 300-303

constructing and destructing, 303-307

control structures, 294-298

 placeholders, 274

extracting types from containers, 183

extractor functions, 184

F

failing polymorphic_cast, 57-60

fclose, 25

Filesystem, xxxiv

find_if, 288

flexibility (Signals library), 346

floating point integral types, 70-72

Ford, Eric, xxxii

Format, xxxv

for_each, 231

for_each_element, 232

for_loop, 299

free functions

 defining, 174

 intrusive_ptr, 32

 predicates, 287

 scoped_ptr, 8

 shared_ptr, 20

 versus member functions, 247

Free functions Tuple library, 213

Friedman, Eric, xxvi

Function library, xxvii, 274, 313-314

 argument binding, 323

 combining with Bind library, 328-334

 combining with Lambda library, 335

 cost, 335

 declarations using compatible syntax, 315

 declarations using preferred syntax, 314

 function objects, 337-340

 invoking a pointer to a member function, 337

 members, 316

 storing and invoking a function pointer, 336

 usage, 317

callbacks, 319-321

functions that are class members, 323-324

stateful function objects, 324-327

function objects, 245, 269, 316, 322

 advantages, 261-262

 combining Function and Lambda libraries, 335

 composing, 253-255

 creating, 239

 Lambda library, 290

function pointers, 320, 335

function scope, 81-82

Functional, xxvii

functional composition, 255-256

functional programming, 270

functions

 binding, 239, 275-278

 contains, 182

 defining, 240

 extractor, 184

 free versus member, 247

 lambda function, 270

 calling

member functions, 241-244

multiple functions, 255-256

 nullary functions, 275

 of any, 170-172

 placeholders, 248-249

 stateful function objects, 324-327

 swap, 180

 that are class members, 323-324

 Tuple library, 211

 virtual, 178, 259-260

G

Garcia, Ronald, xxv
 Garland, Jeff, xxxvi
 general libraries, 1
 generalized binder, 237, 267
 generic constructs, tuples, 227
 generic visitors, 202-203
 get, 217
 get (variant), 197
 Graph, xxiv
 greater, 253
 greed versus repeats, 147-149
 Gregor, Douglas, xxvii-xxviii, xxxix
 Group parameter, 347
 grouping slots, 347-350
 Gurtovoy, Aleksey, xxx, xxxiii

H - I

handling exceptions, 308-311
 Henney, Kevlin, xxiii, xxxvi
 Hinnant, Howard, xxiv, xxix
 Holin, Huberty, xxxii
 if_, 297
 if_then, 296
 if_then_else, 297
 if_then_else_return, 296
 illustration, polymorphic_cast, 57
 implementation-defined behavior, 66
 implementing

- Bind, 239
- callbacks, 319
- functions, virtual, 178
- operator <, 130
- operators (Operators library), 128
- pimpl idiom, 11-12

 incomplete type, 82
 incrementable, 109
 indexable, 111
 indiscriminate types, xxiii, 160
 input, validating, 140-142
 input operators, 177
 input streaming, 225
 InputStreamable, 73
 Integer, xxxii
 integers, conversions, 64
 integral types

- conversions, 64
- floating point, 70-72
- mixing, 69-70

 intent of programmers, stating, 54
 Interval, xxxii
 intrusive reference-counted smart pointers, 15, 37

intrusive_ptr, 29

- free functions, 32
- members, 31
- providing reference counters, 34-36
- supporting different reference counters, 39-41
- usage, 33
- when to use, 42

invoking constructors, 91-92**In_place_factory, xxx****Io_state_savers, xxxv****irreflexivity, 119****is_int predicate (any), 181-182****is_string predicate (any), 181-182****Iterator, xxv****iterators, 110****J - K****Järvi, Jaakko, xxvi-xxx****Josuttis, Nicolai, xxiv****Karvonen, Vesa, xxxvii****Kempf, William, xxxix****key=, 277****keywords, 275****Kleene star, 139****Koch, Mathias, xxxiv****Krempp, Samuel, xxxv****Kruskal's minimum spanning tree, xxv****L****lambda calculus, 270****lambda expressions, 270**

- casting, 300-303
- constructing and destructing, 303-307
- control structures, 294-298
- creating, 273
- storing expressions, 274
- throwing and catching exceptions, 308-311

lambda function, 270**Lambda library, xxviii, 269**

- combining with Function library, 335
- creating slots, 367-370
- function objects, 290
- usage, 272
 - arithmetic operations, 285-286
 - binding to a function, 275-278
 - naming constants and variables, 280-283
 - renaming placeholders, 279-280
 - writing readable predicates, 287-293

last_value, 344**Leak detected!, 145****Lee, Lie-Quan, xxv****LessThanComparable, 124****less_equal, 253****less_than classes, 131**

less_than_comparable, 107
lexical_cast (Conversion library), 73, 77
 enabling classes, 76-77
 example, 73-75
 programming with, 75
 usage, 73

ll_const_cast, 300

ll_dynamic_cast, 300

ll_reinterpret_cast, 300

ll_static_cast, 300-303

logical_and, 253

Lumsdaine, Andrew, xxv, xxx

M

macros, 206

Maddock, Dr. John, xxii, xxiv, xxix, xxxi

make_pair, 216

make_statement, 298

Maman, Itay, xxvi

managing connections (signals), 364-367

manual delete, 9

mark_count, 137

matches, expanding subexpressions, 146

match_results, 144

Math, xxxii

Maurer, Jens, xxxiii

Melquiond, xxxii

member functions

 binding, 277

 calling, 241-244

 versus free functions, 247

member variables, binding to, 260-261

members

 Function library, 316

 intrusive_ptr, 31

 scoped_ptr, 6

 shared_ptr, 18

 Signals library, 344

 Tuple library, 212

 Variant library, 194

 weak_ptr, 44-48

mem_fun, 285

Mensonides, Paul, xxxvii

men_fun_ref, 285

metacharacters, 152

metaprogramming revolution, 82

metaprograms, 227

Minmax, xxxii

minus, 285

mixing integral types, 69-70

modulus, 285

Moore, Paul, xxxiv

Mpl, xxx, 99

Multi-index, xxv

MultiArray, xxv

multicast callbacks, 341

multiple functions, 255-256

multiple return values, 209

Muñoz, Joaquín M López, xxv

N

Nackmann, Lee, 117

naming

 constants, 280-283

 placeholders, 279

 variables, 280-283

negated character classes, 152

nested bind, 254

new_ptr, 303

non-empty values, counting, 182-183

non-greedy repeats, 148

non-intrusive reference-counted smart pointers, 15

noncopyable, 87-89, 92

 Big Three, 90-92

 classes, 88-89

 usage, 87-90

not binding versus binding, 261-263

notifier class, rewriting, 320

null pointers, dynamic_cast, 54

nullary functions, 275

Numeric conversion, xxxiii

numeric_cast, 64-72

numeric_limits, 66

O

object bloating, 120

objects

 copying, 258-259

 deleting, 82-83

 dynamically allocated, 86

 function objects

advantages, 261-262

composing, 253-255

creating, 239

observer pattern, xxviii, 235

observers, 42

operations, 89

operator&, 93-94

operator <, 114, 125, 130

operator <=, 116

operator*, 8

operator+, 121

operator+=, 121

operator-<, 8

operator-=, 129

operator<, 114

operator==, 116, 125

operators, 105

 arithmetic operators, 134

 comparison, 134

 composite arithmetic operators, 112

- different types, 121-123
- input/output, 177
- use of, 129
- Operators library, 105**
 - arithmetic types, 126
 - base classes, 107-110
 - composite arithmetic operators, 112
 - implementing operators, 128
 - supplying missing operators, 122
 - understanding how it works, 130-132
 - usage, 113-114
- optimizations, casts, 62**
- Optional, xxxvii**
- orable, 109**
- Ottosen, Thorsten, xxvi, xxxiv**
- output operators, 177**
- output streaming, 225**
- OutputStreamable, 73, 203**
- P**
- pair, 225**
- parsers, xxii**
- passing class instances to function objects, 324**
- pimpl idiom, 11-12, 22**
- Pion, Sylvain, xxxii**
- placeholders, 239, 273**
 - bind, 240, 247-249
 - creating, 246-247
 - for arguments in bind, 239
 - functions, 248-249
 - names, 279
- plus, 285**
- pointer semantics, 257-259**
- pointer types, 54**
- pointer values, 46**
- pointer-to-member, 246**
- pointers**
 - deleting, 84-85
 - deleting objects through, 82
 - destroying, 172-173
 - raw, 13
 - smart pointers, 176, 243
 - intrusive_ptr*, 29-36, 39-42
 - scoped_array*, 14
 - scoped_ptr*, 9-14
 - shared_array*, 29
 - shared_ptr*, 15, 18-28
 - weak_ptr*, 43-51
 - storing, 172-180
 - testing, 59
- polymorphic_cast (Conversion library), 54, 61**
 - error handling, 59
 - failing, 57-60
 - illustration, 57
 - usage, 55
 - versus *dynamic_cast*, 58-59
- polymorphic_downcast, 61-64**
- Pool, xxxvii**
- Powell, Gary, xxviii**
- predicates, 287**
 - any, 181-182
 - contains, 186-188
 - sorting predicates, 250
- preferred syntax, 314**
- Preprocessor, xxxvii**
- print function, 199, 233**
- print_helper, 227**
- programmers, stating intent, 54**
- programming, lexical_cast, 75**
- programs, searching in, 143-145**
- Program_options, xxxvii**
- property classes, defining, 167-170**
- Property_map, xxxi**
- protecting applications, 153**
- Prus, Vladimir, xxxvii**
- ptr_fun, 285**
- publisher-subscriber pattern, xxviii**
- Python, xxxviii**
- Q - R**
- Ramey, Robert, xxxv**
- Random, xxxiii**
- Range, xxvi**
- range checks, 71**
- Rational, xxxiii**
- Ref, xxviii, 217**
- Ref utilities, 213**
- reference counters, intrusive_ptr, 34-36, 39-41**
- reference types, 54-55**
- reference wrapper, 316**
- reference-counted smart pointers, 15, 37**
- Regex, xxii, 135**
 - troubleshooting, 147
 - usage, 139-154
- regex_iterator, 149-150**
- regex_match, 138-139, 147**
- regex_replace, 138, 145-146**
- regex_search, 138, 143-147**
- regex_token_iterator, 150-151**
- regular expressions, 136, 139**
 - bad, 152-154
 - basic_regex, 136-139
 - input, 140-142
 - regex_iterator, 149-150
 - regex_match, 138-139
 - regex_replace, 138, 145-146
 - regex_search, 138, 143-145
 - regex_token_iterator, 150-151
 - sregex_token_iterator, 151
 - syntax, 151-152
 - text-processing, 135
 - wildcards, 139

relational operators (Tuple library), 214

release, 34

renaming placeholders, 279-280

repeats

*, 147

+, 147

?, 149

bounds operators, 140

non-greedy, 148

versus greed, 147-149

replacing text, 145-146

ret, 278

retrieving stored values, 166-167

rewriting notifier class, 320

Rodgers, Mark, xxvii

Rozental, Gennadiy, xxxviii

S

\s shortcut, 141

safe bool idiom, 20

safety

dynamic_cast, 59

type safety, 159

scope, limiting, xxxviii

scoped_array, 14

scoped_ptr, 5

auto_ptr const, 13

compared to scoped_ptr, 9

free functions, 8

manual delete, 9

members, 6

pimpl idiom, 11-12

use of, 8

when to use, 14

searching in programs, 143-145

security, custom deleters, 26-27

Seik, Jeremy, xxiv-xxv, xxix, xxxiii

select1st, 238

select2nd, 238

semantics, bind, 257-259

separating GUIs from details on how to handle

events from the user, 328-330, 333-334

Serialization, xxxv

set of types, 216

SFINAE (Substitution Failure is Not An Error), xxx, 97-98

shared_array, 29

shared ownership, 4

shared_ptr, 15, 172-176

creating from a weak_ptr, 49-50

creating from this, 27

custom deleters, 24-27

destructor, 19

free functions, 20

members, 18

pimpl idiom, 22

standard library containers, 22-24

usage, 21

when to use, 28

shiftable classes, 106

shortcuts, 140-141

Siek, Jeremy, xxxi

sig, 292

signal, 346

Signal library

combining results, 360-363

copy constructors, 363

grouping slots, 347-350

managing connections, 364-367

signals with arguments, 351-356

types, 343-344

signals, xxviii, 341

Combiner, 360-363

copy constructors, 363

managing connections, 364-367

with arguments, 351-356

Signals library, 341-342

members, 344

usage, 346

sig_helper class, 294

simple_bind constructor, 245

SlotFunction parameter, 344

slots, xxviii, 341, 346

Combiner, 360

connecting to signals, 344

creating using Bind and Lambda, 367-370

disconnecting, 345

grouping, 347-350

managing connections, 364-367

returning false, 362

smart pointers, xxxviii, 1-3, 243

advantages, 176

common errors, 4

intrusive_ptr, 29

free functions, 32

members, 31

providing reference counters, 34-36

supporting different reference counters, 39-41

usage, 33

when to use, 42

scoped_array, 14

scoped_ptr, 5

auto_ptr const, 13

compared to auto_ptr, 9

free functions, 8

manual delete, 9

members, 6

pimpl idiom, 11-12

use of, 8

when to use, 14

- shared_array, 29
 - shared_ptr, 15
 - creating from this, 27
 - custom deleters, 24-27
 - destructor, 19
 - free functions, 20
 - members, 18
 - pimpl idiom, 22
 - standard library containers, 22-24
 - usage, 21
 - when to use, 28
 - storage, 16
 - weak_ptr, 43
 - creating a shared_ptr, 49-50
 - members, 44
 - pointer values, 46-48
 - usage, 44
 - when to use, 51
 - when to use, 4
 - Smart_ptr, xxxviii, 3-5**
 - sorting, 114, 249-253**
 - specializations, 95**
 - Spirit, xxii**
 - splitting strings, 150-151**
 - sregex_token_iterator, 151**
 - Standard Library**
 - adapters, 186-188
 - Bind, 238
 - containers, 232
 - Function library, 314
 - Lambda library, 271
 - Operators library, 106
 - Signals library, 342
 - Smart_ptr, 5
 - Tuple library, 210
 - Variant library, 192
 - state binders, 263-266**
 - stateful function objects, 324-327**
 - static assertions, 80-82**
 - static variables, 322**
 - Static_assert, xxxi**
 - static_cast for downcasts, 61, 64**
 - static_visitor, 200**
 - std::bad_cast exception, 54**
 - storage**
 - any_out, 177-181
 - function objects, 322
 - non-intrusive smart pointers, 16
 - shared_ptr, 22-24
 - stored values**
 - any, 163
 - retrieving, 166-167
 - storing**
 - elements, 241
 - pointers, 172-180
 - types, 159-161, 164-167
 - streaming tuples, 225-226**
 - strict weak ordering, 118**
 - strings**
 - splitting, 150-151
 - String_algo, xxii
 - stringstream, 73**
 - String_algo, xxii**
 - Stroustrup, Bjarne, xxiv**
 - structs, 210**
 - subexpressions**
 - enclosing, 139
 - expanding matches, 146
 - Substitution Failure Is Not An Error (SFINAE), xxx, 97-98**
 - subtractable, 108**
 - subtraction, 126**
 - swap function, 180**
 - swapping values, 170-172**
 - switch statement, 298**
 - switch_statement, 298**
 - syntax**
 - compatible syntax, 315
 - preferred syntax, 314
 - regular expressions, 151-152
- T**
- template parameters, 343**
 - template specialization, 95**
 - Test, xxxviii**
 - testing**
 - any, 184-185
 - binds, 259
 - for empty values, 170-172
 - pointers, 59
 - polymorphic_downcast, 61
 - text, replacing, 145-146**
 - text-processing, 135**
 - this, creating a shared_ptr, 27**
 - Thread, xxxix**
 - throw_exception, 311**
 - tie, 224**
 - Timer, xxxix**
 - Tokenizer, xxiii**
 - tools, Boost Utility, 79**
 - to_string, 75**
 - tracer class, 257**
 - transform, 291**
 - transitivity, 119**
 - transitivity of equivalence, 119**
 - Tribool, xxxix**
 - triple, 218**
 - troubleshooting**
 - bad regular expressions, 153
 - Boost.Regex, 147
 - catching exceptions (lambda expressions), 308-311
 - smart pointers, 4

try/catch blocks, 167

try_catch, 309

Tuple library, xxvi, 209-211

advanced features, 227-230

for_each, 231

Free functions, 213

Index, 213

members, 212

relational operators, 214

tuples

accessing elements, 217, 220

comparing, 221-223

constructing, 215-216

streaming, 225-226

tying elements to variables, 224

usage, 215

turning off operations, 89

tying tuple elements to variables, 224

type safety, 159

typedef, 230

types

assignments, 65-68

determining, 245-246

from containers, 183

indiscriminate, 160

lambda types, 272

Signal library, 343-344

signaling events, 357

storing, 159-167

variants, 191

Type_traits, xxxi, 80

U - V

uBLAS, xxxiv

unary_function, 292

unions, 206

unnamed functions, 270-272

unsigned integral types, 68-69

unspecified-bool-type, 20

use_count, 20

utilities, 2

Utility library, xxxix, 79

validating input, 140-142

value semantics, bind expressions, 257-259

value wrapping, 69

values

multiple return values, 209

non-empty, 182-183

stored, 166-167

swapping, 170-172

Value_initialized, xl

var, 280

variables

basic_regex, 139

member variables, 260-261

names, 280-283

tying tuple elements to, 224

variant, 159

Variant library, xxvi, 191

advanced features, 206

members, 194

usage, 196-198

binary visitors, 203-205

generic visitors, 202-203

visiting variants, 198-200

variant class template, 193

variant types, xxiii, 191

bounded, 191

unions, 206

variants

binary visitors, 203-205

generic visitors, 202-203

visiting variants, 198-200

var_type, 281

vector, 14

virtual functions

binding, 259-260

implementing, 178

testing binds, 259

visitation, 199

visiting variants, 198-200

visitors

accepting arguments by value, 201

binary, 203-205

generic, 202-203

W-Z

\w shortcut, 141

Walker, Daryle, xxxii-xxxiii, xxxv

Walter, Joerg, xxxiv

weak_ptr, 42-43

creating a shared_ptr, 49-50

members, 44

pointer values, 46-48

usage, 44

when to use, 51

while_, 299

while_loop, 298-299

wildcards, 139

Willcock, Jeremiah, xxx

Witt, Thomas, xxv

writing readable predicates, 287-293