Index

"binary.hpp", 1
::type, 17, 31, 33, 59
::value, 4, 17, 24, 30, 33, 39, 61
::value_type, 13, 16
<algorithm>, 115
<empty.hpp>, 298
<enum_params.hpp>, 282
<equal.hpp>, 296
<functional>, 17
<if.hpp>, 296
<iostream>, 1
<is_reference.hpp>, 22
<is_same.hpp>, 22
<iterate.hpp>, 290
<iterator>, 22
<local.hpp>, 289
<repetition.hpp>, 286
<utility>, 22
<vector20.hpp>, 92

A

abs(), 206
abstract machine, 323
abstraction, 113, 127
abstractions of the preprocessor, 283
abstractions of the problem domain, 8
abstractions, preprocessor library, 286
access adaptor, 138
access iterator
  need for random, 159
  random, 82, 92, 99, 115
  requirements, random, 83
  to the sequence element, 79
access sequence
  lazy random, 93
  limited-capacity random, 97
  random, 85, 92, 109
accumulate, 127
action, semantic, 222
adaptor
  access, 138
  iterator, 138
  traversal, 138
  views and iterator, 131
add_pointer, 49
adding extensibility, 106–108
addition and subtraction, implementing, 41
additional tools, 173
ADL (argument-dependent lookup), 206, 207
advance algorithm, 180
advance(), 182
Alexandrescu, Andrei, 194
algorithm, 113
  abstraction, 113
  advance, 180
  binary searching, 132
  compile time, 77
  counterpart, 124
  equal, 90
  filter, 137
  fold, 190
  functional, 126
  fundamental sequence, 119
  idioms, reuse and abstractions, 113
  iteration, 121
  linear traversal, 127
  MPL, 115
  querying, 122, 128
  reusability, 127
  reuse, 58

algorithm (continued)
  re-use the MPL, 127
  screensaver, 197
  sequence, 78, 109
  sequence building, 119, 123, 125, 126, 128
  sequence traversal, 120
  writing your own, 127
`<algorithm>`, 115
alias, 39
`always_int`, `struct`, 29, 57
analysis
  dimensional, 37, 38, 165
  DSEL, 276
  tools for diagnostic, 155
  user, 7
angle, 38
anti-pattern, 16
application
  context, 313
application, partial function, 240
`apply_fg()`, 17
`apply_fg()`, `template`, 16
`apply` metafunction, 52, 59
`apply`, `mpl::`, 52, 56, 59
argument
  complexity, 339
  empty, 297
  list, 284
  macro, 284, 301, 303
  metafunctions as, 16, 139
  selection, 296
  separators, macro, 297
  structural complexity of metafunction, 338
  types, 17
argument-dependent lookup (ADL), 206–207
arithmetic, logical, and comparison operations, 293
arithmetic operations, preprocessor library, 293
arithmetic operator, 72
arity, 338
array initialization, 236
arrays, 304
asserting numerical relationships, 164
assertion
  likely, 163
  messages, 165
  MPL static, 161
  negative, 163
  static, 160, 165
associated types, 78
associations, type, 11
Associative Sequence, 86, 87, 109
Associative Sequence, extensible, 88, 89, 94
automate wrapper-building, 200
automatic type erasure, 200
auxiliary object generator function, 185
avoiding unnecessary computation, 137

B

backtrace, instantiation, 144, 145, 173
Backus Naur Form, *see* BNF
Barton and Nackman trick, 205
base class, 316
`begin`, 103
`begin_impl`, `struct`, 85
Bentley, Jon, 217
bidirectional iterator, 81–82
  requirements, 82
  sequence, 84
binary
  function, 127, 296
  implementation, 6
  meta, 5
  metafunction, 42, 53, 128
  numerals, 6
  operation, 42
  recursion, 4, 5
  runtime version, 5
  search, 115
  searching algorithm, 132
  `struct`, 4
  template, 4, 15
`"binary.hpp"`, 1
`binary()`, 5, 6
`binary<>`, 1

binary<N>::value, 4, 7
BinaryOperation, 42, 44, 47
bit_iterator, 22
bit_iterator, struct, 21
bitwise operator, 72
Blitz++, 242, 264
  and expression templates, 231, 232
  array initialization, 236
  compile-time parse tree, 233
  domain, 235
  evaluation engine, 233
  library, 231
  magic, 236
  range objects, 237
  subarray syntax, 237
  syntactic innovations, 236
blob, 15, 30, 32
BNF (Backus Naur Form), 220
  context-free grammar, 220
  definition, 220
  efficiencty, 222
  extended, 222
  grammar, 225
  productions, 220
  rules, 220
  symbols, 220
boilerplate code repetition, 281
boilerplate implementation code, 8
bool constants, 30
bool valued nullary metafunction, 162
bool values, 61
Boolean
  conditions, inverting, 69
  valued metafunctions, 34
  valued operator, 71
  wrappers and operations, 61
Boost
  ::iterator_facade, 214
  ::iterator_value, 209
  ::mpl::, 39
  ::mpl::vector, 39
  Bind library, 185, 240, 244, 264
  compressed_pair, 190
  Concept Checking library, 173
  DSEL libraries, 255
  enable_if, 209
  Function library, 203, 295
  Graph library, 238
  integral metafunction, 65
  Iterator Adaptor library, 141
  Lambda library, 114, 242–244
  libraries, convention used by, 17
  metafunctions, 66
  Metaprogramming Library, 9, 15, 31, 57, 281
  namespace, 24, 72
  Preprocessor library, 283, 285
  Python library, 96
  Spirit library, 6, 243, 247
  Type Traits, 30, 64
  Type Traits library, 24, 30, 81, 212, 229, 301
BOOST_MPL_ASSERT, 162, 165
BOOST_MPL_ASSERT_MSG, 169
BOOST_PP_CAT, 300
BOOST_PP_EMPTY, 297
BOOST_PP_IDENTITY, 299
BOOST_PP_IF, 296
BOOST_PP_ITERATE, 293
BOOST_STATIC_ASSERT, 43, 50, 51, 79, 160
boundary, crossing the compile-time runtime, 175
bug, 144, 155
building anonymous functions, 239

C

_c integral shorthand, 73
C++
  classes in runtime, 127
  code, 2
  compile time, 5
  compiler, 2, 3, 7, 330
  compiler diagnostics, 143
  Generic Programming in, 8
  host language, 229
  iterators in, 12
  language for building DSELs, 277
  language note, 12, 49

C++ (continued)
  limitation of the language, 159
  metaprogram, 5, 215
  metaprogramming, 3, 9
  metaprogramming advantages, 7
  operator, built-in, 71
  overloadable operator syntaxes, 229
  preprocessors, 282
  program, 1, 224
  runtime, 175
  standard library, 149
  template syntax, 91
  templates, 9, 270
  view concept, 141
categorization, primary type, 25
categorization, secondary type, 26
charge, 38
checking, error, 4
choosing a DSL, 262
Church, Alonzo, 51
class, 12
  base, 316
  composition, 190
  customization, 198
  eliminating storage for empty, 187
  empty, 181, 187
  `float_function`, 201
  metafunction, 43, 77
  namespace of the base, 205
  runtime polymorphic base, 199
  sequence, 91
  structural changes to the, 186
  template, 29
  template specialization, 31, 179
  templates-as-functions, 15
  vs. `typename`, 310–311
`clear`, 86, 88
`clone()`, 202
closures, 241, 247, 249
code, expressive, 7
code generation, 282
code repetition, 281
code, self-documenting, 226
combining multiple sequences, 135
Comeau C++, 155, 330, 339
commands, Make, 218
common interface, 32
common syntax, 17
comparing values computed from sequence elements, 131
comparison
  heterogeneous, 133
  homogeneous, 132
  operations, 293
  operations, preprocessor library, 294
  predicate, 132
  predicate, homogeneous, 134
  value, 71
compilation, 143
  error, 19, 46, 48, 170, 179, 188, 207
  grammar, 7
  improve, 57
  phases, template, 308
  slow, 16, 323
  speed, 32
  time and long symbols, 337
  time, argument complexity effect on, 339
  times, 324
  times, compiler and, 331
compile time, 11, 18, 33, 62, 80, 127
  constant, 7, 269
  constants for comparison, 276
  error, 93, 108, 158
  error generation, intentional, 172
  execution log, 171
  lambda expressions, 114
  managing, 326
  metaprograms, 213
  performance, 323
  programming, 330
  runtime boundary, 175, 265
  runtime differences, 92
  STL, 77
  wasting, 64

compiler, 4, 16, 32
  C++, 2, 3, 330
  C/C++, 7
  Comeau C++, 155, 330, 339
  compilation times, 331
  deep `typedef` substitution, 151
  diagnostic, 158
  diagnostic formats, 155
  diagnostic using different, 155
  diagnostics, C++, 143
  EBO, 187
  EDG-based, 173
  erratic performance, 329
  error, 22, 43, 145, 160, 195
  error from VC++ 7.1, 152
  error message, 143
  GCC, 155, 156, 164, 166, 171
  GCC-3.2, 154
  GCC-3.2.2, 148
  GCC-3.3.1, 161
  GCCs, 330, 339
  generating a warning, 171
  get a second opinion, 155
  ideal, 326
  incomplete support for templates, 343
  Intel C++ 7.1, 153
  Intel C++ 8.0, 151, 169
  Intel C++ 8.1, 161
  known NOT to work with MPL, 344
  memoization, 327
  Metrowerks CodeWarrior, 24
  Metrowerks CodeWarrior Pro 9, 155
  Microsoft Visual C++ 6, 146
  modern, 146
  more work for the, 16
  object code, 7
  optimized space, 27
  optimizing storage for empty subobjects, 190
  overload resolution capability, 269
  performance, 333
  post processing filter, 156
  requiring no user workarounds, 343
  requiring user workarounds, 343
  SGI MipsPro, 24
  support, 24
  support, without, 25, 26
  supported, 162
  test, 327
  three different, 146
  tip, 155
  traits, 24
  unable to work with MPL, 344
  values of template parameters, 32
  VC++ 7.0, 150
  VC++ 7.1, 150, 168
  Visual C++ 6 revised, 148
complexity guarantees, 78
complexity tests, structural, 338
component implementations, 8
composition, class, 190
computation
  avoiding unnecessary, 137
  invalid, 57
  naming an invalid, 57
  numeric, 3
  runtime, 4, 6
  type, 5
computational model, 323
computed by a metaprogram, 6
computing with types, 5
concept, 77
concept requirements, 77
concerns, separation of, 115
constant folding, 277
constant time specialization, 103
constant wrapper, integral, 17
constants, integral, 74
constants, named local, 244
constructs, selection, 299
context application, 313
context-free grammar, 220
control structures, 295
conventions, naming, 288
copyability, 202
cost of instantiation, 326
cost of memoized lookups, 327

counterpart algorithms, 124
Curiously Recurring Template Pattern (CRTP), 203–209, 251, 267–268
 and type safety, 205
custom source code generator, 8
`customize` function, 197
customized assertion messages, 165
customized error message, 174
customized errors, 173
customizing the predicate, 165
cv-qualification, 25, 27
cv-unqualified, 61
Czarnecki, Kristof, ix

D

data types, 301
 arrays, 304
 lists, 305
 sequences, 301
 tuples, 303
debug metaprograms, 143, 153
debugging, 155
debugging the error novel, 143
declaration, single, 314
declarative languages, 226
decrementable iterator, 81
deep `typedef` substitution, 151
deeply-nested metafunction, 333
default template arguments, 150
definition, DSL, 228
definition, metafunction, 29
definition, point of, 308
dependencies, Make, 218
dependent name, 12, 49, 310
dependent type, 310
dependent type names, identifying, 312
depth, nesting, 338
`deque`, 93
dereferenceable, 80
derivation, sequence, 96
description, grammar, 2
design, DSEL, 257
design of pointers, 12
destructor, trivial, 24
development process, DSEL, 276
diagnostic, 143, 153
 additional tools, 173
 analysis, tools for, 155
 compiler, 143, 158
 customized assertion messages, 165
 customized errors, 173
 customizing the predicate, 165
 deep `typedef` substitution, 151
 earlier, 160
 error formatting quirks, 146
 filtering tools, 172
 generation, intentional, 158
 guideline, 158
 history, 172
 inline message generation, 167
 instantiation backtrace, 144, 173
 intentionally generated, 170
 MPL static assertions, 161
 post processing filter, 156
 reserved identifiers, 149
 selecting a strategy, 170
 static assertions, 160, 173
 tip, 155
 type printing, 170, 174
 `typedef` substitution, 173
 unreadable type expansions in the, 169
 using different compilers, 155
 using filters, 158
 using navigational aid, 155
`difference_type`, 13
dimensional analysis, 37, 38
 code, 165
 generating errors, 165
 implementing addition and subtraction, 41
 implementing division, 46
 implementing multiplication, 42
 representing dimensions, 38
 representing quantities, 40
dimensional mismatch, 165
dimensions, 38, 41

dimensions, representing, 38
disambiguating templates, 311
disambiguating types, 310
disambiguation, syntax, 311
dispatching, tag, 180
domain abstraction of FSMs, 257
domain language, 3, 8
domain-specific embedded language, 215, 276
domain-specific language, 215, 216, 218, 220, 225, 228, 241, 245, 246, 254
DSEL, 215, 229, 236, 266
  analysis, 276
  design, 267
  design walkthrough, 257
  development process, 276
  finite state machines, 257
  framework design goals, 260
  highly efficient, 277
  notations, 258
DSL, 228–229, 235, 238, 242
  Boost Spirit, 247
  choosing a, 262
  closures, 249
  declarative language, 217
  declarativeness, 277
  definition, 228
  design, 230
  embedded, 261
  FC++, 245
  framework interface basics, 261
  function object construction, 239
  inside out, 226–229
  language, 216
  library, 276
  Make, 218
  properties, 216
  summary, 225
  syntax, 231, 238
dynamic polymorphism, 17
dynamic scoping, 250

E

EBNF, 222
EDG-based compilers, 173
effectiveness of memoization, 326
efficiency, FSM, 264
efficiency, metaprogram, 323
efficiency, metaprogramming, 330
efficiency problem, 186
Eisenecker, Ullrich, ix
eliminating default template arguments, 150
eliminating storage for empty classes, 187
embedded DSL, 261
emergent property, 138
empty argument to the preprocessor, 297
Empty Base Optimization (EBO), 187
empty class, 187
`<empty.hpp>`, 298
`enable_if`, `struct`, 211
`end`, 103
`end_impl`, `struct`, 85
`enum`, 11
`<enum_params.hpp>`, 282
`equal`, 45
`equal` algorithm, 90
`<equal.hpp>`, 296
`equal_to`, 70
`equal_to`, `struct`, 70
equality, sequence, 89
equivalence of iterators, 81
`erase`, 86, 88
erasure, automatic type, 200
erasure, manual type, 199
erasure, type, 196, 251, 264
error, 101
  checking, 4, 32
  compilation, 19, 46, 48, 170, 179, 188, 207
  compiler, 43, 145, 195
  during overload resolution, 211
  formatting quirks, 146
  guideline, 158
  ignoring the, 145
  `iter_swap()`, 12

error (continued)
    message, 3, 144, 148
    message, customized, 174
    message reordering, GCC, 156
    messages examples, 143
    messages, STL, 156
    novel, debugging the, 143
    programming, 159
    realistic, 146
    reporting, advanced, 146
    strategy to customize, 170
    substitution failure is not an, 211
    `template`, 320
    `typename`, 316
    VC++ 7.1, 152
`eval_if`, 65
`eval::`
    `eval`, 67
evaluation, lazy, 59, 64
evaluation, semantic, 222
example, 197
explicit specialization, 31
explicitly managing the overload set, 209
`expr`, 6
expression
    compile-time lambda, 114
    evaluation, lazy, 234
    lambda, 51, 52, 56, 136
    placeholder, 47, 52
    regular, 215
    templates, Blitz++ and, 231, 232
    templates, drawback of, 236
    valid, 78
    wrapping and indenting, 157
expressive code, 7
Extended BNF, 222
extensibility, 86
extensibility, adding, 106
extensible associative sequence, 88, 89, 94
extensible sequence, 86
extra level of indirection, 15

F

`f()`, 12
`factor`, 6
`factorial`, 161, 168
factorial metafunction, 160
faster programs, 7
FC++, 244
FC++ language design, 246
Fibonacci function, 324
Fibonacci test, 327
file, index.html, 285
file iteration, 289, 290, 293, 298
file, numbered header, 91
filter, 126
    algorithm, 137
    function, 137
    post processing, 156
    STLFilt, 156
    STLFilt options, 157
    TextFilt, 156
`find`, 78
finite state machine construction framework, 257
finite state machines (FSM), *see* FSM
`five`, `struct`, 18
fixed part, 31
`float`, 196, 201
`float_function`, 201
flyswapper, 22
`fold`, 127
`fold` algorithm, 190
folding, constant, 277
`for_each`, 175, 176
`for()`, 5
force, 38
Form, Backus Naur, 220
formal language, 216
formatting quirks, error, 146
FORTRAN, 217, 237
forward iterators, 80
forward iterators requirements, 81
forward sequence, 92
forward sequences, 84

friend functions property, 206
FSM, 257
  class name, 268
  classes, 261
  construction framework, 257
  declaration, 277
  declarativeness, 260, 276
  description, 266
  design, 260
  domain abstraction of, 257
  efficiency, 260, 264, 276
  events, 258
  expressiveness, 260, 276
  implementation, 269
  interoperability, 260, 276
  maintainability, 260, 277
  scalability, 260, 277
  states, 257
  static type safety, 260, 277
  transitions, 258
FTSE, 13, 19, 192, 263
full template specializations, 317
function, 33
  `abs`, 206
  `advance`, 182
  application, partial, 53
  auxiliary object generator, 185
  binary, 6, 127, 296
  building anonymous, 239
  call, 270
  call operator, 186
  chaining, member, 238
  `clone`, 202
  `customize`, 197
  Fibonacci, 324
  filter, 137
  generating, 204
  generic, 14, 159
  higher order, 48, 58
  like macros, 283
  member, 32
  meta, 5
  names, member, 16
  non-member friend, 205
  object, 114, 249, 299
  object's signature, 177
  object, stored, 6
  object template, 194
  objects, runtime, 175
  ordinary, 15
  overloads, 210
  parameters, 63
  pointer to a transformation, 197
  pointer type, 97
  pointers, 25
  pointers as template arguments, 194
  property of friend, 206
  recursive, 4
  references to, 11
  runtime, 16
  source code, 290
  static member, 23, 179
  static visit member, 178
  `swap`, 19
  templates, 313
  templates and polymorphism, 196
  types returning pointers, 153
  unary, 296
  `yyparse`, 2
function composition, 240
`function`, `struct`, 296
`<functional>`, 17
functional algorithms, 126
Functional FC++, 244
fundamental abstractions of the preprocessor, 283
fundamental sequence algorithms, 119
fundamental theorem of software engineering (FTSE), 13, 19, 192, 263

G

GCC, 148, 155, 156, 164, 166, 167, 171
GCC-3.2, 154
GCC-3.2.2, 148
GCC-3.3.1, 161
GCC error messages, 157

GCCs, 330, 339
general-purpose DSEL, 237
general purpose sequence, 93
`generate`, 192
generating custom messages, 167
generating function, 204
generation, code, 282
generator, object, 183
generic function, 159
generic loop termination, 115
generic programming, 17
generic programming in C++, 8
global objects, 11
GNU Make, 220
grammar
  BNF, 225
  compilation, 7
  context-free, 220
  description, 2
  rules, 2
  specifications, 6
  YACC, 7
Guzman, Joel de, 252

H

handling placeholders, 50
Haskell, 5, 64, 119, 244
heterogeneous comparisons, 133
hierarchy, refinement, 181
high-level parser, 2
higher order function, 48, 58
higher-order macro, 287
higher order metafunction, 48
homogeneous comparison, 132
homogeneous comparison predicate, 134
horizontal repetition, 286
host language, 3, 229
host language translators, 3

I

IDE, 173
ideal compiler, 326
identifier, 149, 283
identifying dependent type names, 312
identity, type, 89
idiomatic abstraction, 113
`<if.hpp>`, 296
`if` statements, 178
implementation of a runtime function template, 178
implementation of placeholders, 54
implementation selection, 178
implementing
  addition and subtraction, 41
  `at` for `tiny`, 100
  division, 46
  multiplication, 42
  sequence, 138
  view, 139
implicit pattern rules, 219
incrementable, 80
independent metafunctions, 32
index.html file, 285
`inherit_linearly`, 193
inheritance, layers of, 191
inline message generation, 167
`insert`, 86, 88
inserter, optional, 124
inserters, 117, 118, 125, 128
instantiation, 32
  backtrace, 144, 145, 173
  backtrace, GCC, 148
  cost of, 326
  depth, reducing, 336
  forwarding, nested, 333
  nested template, 330
  points of, 308
  required, template, 324
  stack, 151
  template, 155, 324, 330
`int_`, `struct`, 69
`int_<N>`, 39

int dimension, 38
int*, 20
integer
  constants, 32
  large sequences of, 94
  values, 11, 61
  wrappers and operations, 69
integral
  _c, 73
  constant, 74
  constant wrapper, 17, 39, 66, 176
  operator, 71
  sequence wrapper, 40, 70, 95
  type, 70
  type wrapper operation, 61
  valued operator, 72
  valued type traits, 183
integral_c, struct, 70
Intel C++ 7.1, 153
Intel C++ 8.0, 151, 169
Intel C++ 8.1, 161
intensity, 38
intentional diagnostic generation, 158
interface basics, framework, 261
interface, common, 32
interface, preserving the, 201
internal pointers, 19
interoperability increased, 117
interoperability of the program, 16
intrinsic sequence operation, 90, 109
invalid computation, 57
invariant, 78
inverting Boolean conditions, 69
<iostream>, 1
<is_reference.hpp>, 22
<is_same.hpp>, 22
is_scalar, 66
iter_fold, 127
iter_swap, 62–63
iter_swap_impl, struct, 23
iter_swap_impl, template, 23
iter_swap(), 15, 18, 22
iter_swap(), error, 12
iter_swap(), template, 11–13, 19, 22
<iterate.hpp>, 290
iteration algorithms, 121
iteration, file, 289, 290, 293, 298
iteration, local, 289
iterator, 19, 79
  access, 79
  adaptor, 138
  Adaptor library, 141
  adaptors, views and, 131
  associated types, 13
  bidirectional, 81
  C++, 12
  categories, 109
  concept, 80, 109
  decrementable, 81
  dereferenceable, 80
  different types, 19
  equivalence, 81
  forward, 80
  handling, 114
  incrementable, 80
  large sequences of integers, 94
  operate on, 127
  output, 117
  past-the-end, 80
  random access, 82, 92, 159
  reachable, 81
  representation, 99
  sequence, 77
  struct bit, 21
  tiny, 102
  type, 9, 12
  valid, 12
  value type, 12
  values, 121
  vector<bool>, 21, 22
  zip, 139
<iterator>, 22
iterator_category, 13
iterator_range, 95
iterator_traits, 14–16
iterator_traits, partial specialization of, 14

iterator_traits, struct, 13
iterator_traits<int*>, 31
Iterator::, 15

J

joint_view, 137

K

keywords, typename and template, 307
Koenig, Andrew, 13

L

lambda
  calculus, 51
  capabilities, 53
  details, 53
  expression, 51, 52, 56, 58, 67, 68, 136
  metafunction, 51, 59
  non-metafunction template, 56
Lampson, Butler, 13
language
  C++, 277
  C++ as the, 229
  declarative, 226
  design, FC++, 246
  directions, 277
  domain, 3, 8
  domain-specific embedded, 215, 276
  DSELs, 215
  DSL declarative, 217
  formal, 216
  FORTRAN, 217
  Haskell, 5
  host, 3
  Make utility, 218
  metaprogramming in the host, 3
  metaprogramming, native, 3
  note, C++, 12, 49
  pure functional, 5, 32
  Scheme, 3
  syntax of formal, 220
  target, interaction, 7
  translators, host, 3
large sequences of integers, 94
late-binding, 17
layer of indirection, 192
layers of inheritance, 191
lazy, 211
  adaptor, 131
  evaluation, 57, 59, 64
  expression evaluation, 234
  random access sequence, 93
  sequence, 135, 138
  techniques, 137
  type selection, 64
legal nullary metafunction, 33
length, 38
level of indirection, extra, 15
library
  abstractions, 158
  abstractions, preprocessor, 286
  arithmetic operations, preprocessor, 293
  Blitz++, 231
  Boost.Bind, 240, 264
  Boost.Function, 203
  Boost.Graph, 238
  Boost.Lambda, 114, 242
  Boost.Metaprogramming, 9, 15, 31
  Boost.Preprocessor, 283
  Boost.Python, 96
  Boost.Spirit, 6, 247
  Boost.Type Traits, 24, 30, 33
  C++ standard, 149
  logical operations, preprocessor, 294
  convention used by Boost, 17
  data structures, 302
  headers, 92
  integer logical operations, preprocessor, 294
  interface boundary, 158
  Iterator Adaptor, 141
  Math.h++, 237

metafunctions, 22
metaprogramming, 5, 58, 106
Phoenix, 243
preprocessor, 289
standard, 14
structure, preprocessor, 285
type traits, 27
View Template, 141
limiting nesting depth, 334
linear traversal algorithms, 127
list, replacement, 283, 284, 287
lists, 92, 305
`<local.hpp>`, 289
local iteration, 289
`log2()`, 17
logical
coherence, 293
comparison operations, 293
operations, preprocessor library integer, 294
operator, 66, 71
operator metafunction, 67
`long_` and numeric wrappers, 70
long symbols, 337
`long*`, 20
lookup, argument dependent, 206
loop termination, generic, 115
low-level template metafunctions, 212

M

machine, abstract, 323
machines, finite state, 257
macro
argument separators, 297
arguments, 284, 301, 303
function-like, 283
higher-order, 287
naming conventions, 288
nullary, 299
object-like, 283
parameter, 284
preprocessor, 283
Make, 227, 228, 261
commands, 218
dependencies, 218
GNU, 220
language construct, 218
manual, GNU, 219
rule, 218
system, 219
targets, 218
utility language, 218
makefile, 218, 219
managing compilation time, 326
managing overload resolution, 207
managing the overload set, 209
manipulation, type, 11
manual type erasure, 199
map, 126
`map`, 94
mass, 38
Math.h++ library, 237
maximum MPL interoperability, 107
member function bodies, 32
member function chaining, 238
member function names, 16
memoization, 324
effectiveness of, 326
record, 330
memoized lookups, cost of, 327
mental model, reusable, 9
mentioning specialization, 329
message
compiler error, 143
customized, 165
customized assertion, 165
customized error, 174
error, 3, 144, 146, 148
examples, error, 143
formatting, 170
generating custom, 167
generation, inline, 167
reordering, GCC error, 156
STL error, 156
template error, 155, 158

metadata, 32, 40
  non-type, 11
  numerical, 33
  polymorphic, 61
  pure, 277
  traits, 33
  type, 11
  type wrappers, 33
metafunction, 15, 24, 25, 28, 33, 37, 47, 77, 122
  add_pointer, 49
  application, partial, 53
metafunction (continued)
  apply, 52, 55, 59
  arguments, structural complexity of, 338
  as arguments, 16, 139
  begin, 79
  binary, 42, 53
  blob, 16
  bool-valued, 24
  bool-valued nullary, 162
  Boolean-valued, 34
  Boost integral, 65
  Boost's numerical, 24
  call, 145
  class, 43, 50, 51, 55, 58, 77
  composition, 53, 58
  composition of three, 53
  deeply-nested, 333
  definition, 29
  deref, 79
  efficiency issue, 16
  equal_to, 70
  eval_if, 65
  factorial, 160
  forwarding, 57, 107
  higher-order, 48
  implementing a, 127
  independent, 32
  inherit_linearly, 193
  insert, 88
  integral constants passed to, 18
  integral-valued, 24
  invoked lazily, 57
  lambda, 51, 59
  legal nullary, 33
  library, 22, 33
  low-level template, 212
  MPL, 31, 33, 62
  MPL logical operator, 67
  mpl::advance, 82
  mpl::apply, 52, 56, 59
  mpl::end, 79
  mpl::find, 79
  mpl::identity, 65
  mpl::prior, 81
  multiple return values, 15
  name, 17
  next, 72
  nullary, 29, 33, 57, 61, 64, 211
  numerical, 17, 33, 39
  numerical result, 18
  operating on another metafunction, 48
  order, 87
  padded_size, 132
  param_type<T>, 63
  polymorphic, 18
  polymorphism among, 17
  preprocessing phase, 283
  prior, 72
  protocol, 9
  returning integral constants, 61
  returning itself, 107
  reverse_fold, 120
  self returning, 98
  sequence, 90
  single-valued, 30
  specialization, 15
  transform, 42
  type categorization, 25
  type manipulation, 28, 33
  types of individual class members, 185
  unary, 25
  zero-argument, 29
metaprogram, 56
  C++, 5, 215
  complexity, 324

computed by a, 6
correct and maintainable, 7
debug, 143
debugging, 156
efficiency, 97, 323
execution, 143
implementation, 326
interfacing, 8
misbehavior, 170
more expressive code, 7
preprocessor, 288
Scheme, 3
template, 1, 24
testing the, 282
what is it?, 2
metaprogramming
benefits, 6
C++, 3
C++, advantages of, 7
class generation, 193
compile time, 8
conditions, 8
efficiency, 330
in the host language, 3
introduction to preprocessor, 281
library, 5, 58, 106
library, why a, 9
native language, 3
techniques, 205
template, 156
type computations, 11
when to use, 8
metasyntax, 220
Metrowerks CodeWarrior Pro 9, 155
Microsoft Visual C++ 6, 146
`minus_f`, 46
`minus_f`, `struct`, 46
model, computational, 323
model, reusable mental, 9
model the concept, 77
MPL (Boost Metaprogramming Library), 9, 31, 33, 39, 58
adaptor, 139
algorithms in the, 115
benefits, 9
class generation, 193
known NOT to work with, 344
compilers requiring no user workarounds, 343
compilers that require user workarounds, 343
deque, 93
forward iterator requirements, 81
fun, 9
generating custom messages, 167
int wrapper, 69
integral sequence wrappers, 40
interoperability, maximum, 107
iterator, 79
iterator concepts, 80
iterator range, 95
lambda, 53
lambda function, 51
logical operator metafunction, 67
map, 94
metafunction, 33, 62
metafunction equal to, 70
placeholders, 47
portability, 9, 343
productivity, 9
quality, 9
reuse, 9
sequence, 86, 91
sequence building algorithms, 123
sequence querying algorithms, 122
set, 95
static assertion macros, 162
static assertions, 161
transform, 42
type sequence, 39, 97
`mpl::`, 39
`advance`, 82, 85, 103, 142
`and`, 58, 69, 71, 74
`apply`, 52, 56, 59, 60
`arg`, 54
`at`, 85, 95, 101, 103, 110, 136
`back`, 85, 118, 124
`begin`, 84–86, 103, 117

mpl:: (continued)
  bind1, 154
  bool, 58, 70, 212
  contains, 137
  copy, 118, 128, 129
  deref, 79–81, 84, 85, 99, 116, 133–135, 139
  distance, 82
  empty, 193, 299
  empty_base, 193
  end, 79, 84, 85, 103, 117, 141
  equal, 45, 46, 71, 90, 109, 126, 129, 165
  erase, 86, 91
  eval, 65, 67–69, 73, 98, 161, 297
  false_, 183
  filter, 274
  filter_view, 138
  find, 58, 78, 79, 335
  fold, 120, 191, 274
  for_each, 175, 177
  forward, 139
  front, 84, 124
  greater, 162, 164
  has, 95
  identity, 65, 68, 69, 73, 317
  if, 62–65, 68, 74, 75, 180, 295
  inherit, 193
  insert, 86, 88, 89
  inserter, 117
  int, 39, 40, 69, 119, 144, 161, 171, 281, 287
  integral, 70, 154
  iterator, 141
  joint, 141
  lambda, 51, 55, 60
  lambda1, 154
  less, 116, 122, 133, 163
  list, 84, 86, 92, 118, 124, 142, 326
  long, 70
  lower, 110, 132, 134, 137, 326
  map, 87, 94
  minus, 46, 47, 53, 103
  multiplies, 53, 56, 161
  next, 72, 79–81, 83, 86, 99, 100, 139
  not, 58
  not_, 163
  or, 67–69, 73, 74
  pair, 94
  placeholders, 47, 153
  plus, 43, 44, 52, 53, 56, 69, 72, 75, 103, 119, 136, 171
  plus_dbg, 171
  pop, 89
  print, 171
  prior, 73, 81, 83, 85
  push, 89, 92, 117, 119
  quote1, 154
  random, 99
  range, 93, 142, 171
  replace, 117
  reverse, 124, 125
  set, 87
  shift, 126
  size, 106, 110
  sizeof, 117, 132, 133, 135
  transform, 42–44, 47, 50, 67, 68, 119, 124, 136, 137, 153, 177
  transform_view, 135, 138, 141
  true_, 183
  unpack_args, 136
  vector, 40, 78, 93, 119, 129, 142, 326
  void, 97, 154
  zip, 136
multiple return values, metafunctions, 15
multiple return values of traits templates, 15
multiple sequences, 135
multiplication, implementing, 42

N

name, dependent, 12, 310
named class template parameters, 239
named local constants, 244
named parameters, 238
names, namespace, 231
namespace aliases, 39
namespace boost, 24
namespace names, 39, 231

namespace std, 13
naming an invalid computation, 57
naming conventions, 288
native language metaprogramming, 3
negative assertions, 163
nested instantiations without forwarding, 333
nested template instantiations, 330
nested types, 15, 30
nesting depth, 338
nodes, number of, 338
noise, syntactic, 263
non-empty sequence, 284
non-member friend functions, 205
non-qualified names, 316
non-types, metadata, 11
nullary macro, 299
nullary metafunction, 29, 33, 57, 61, 64, 211
number of nodes, 338
number of partial specializations, 336
numbered header file, 91
numeric computations, 3
numeric relation, 174
numeric wrappers
    long_, 70
    size_t, 70
numerical
    comparison, 164
    metadata, 33
    metafunction, 17, 33, 39
    relationships, asserting, 164

O

object
    Blitz++ range, 237
    different types, 17
    function, 299
    generator, 183
    generator function, 185
    global, 11
    like macros, 283
    oriented programming, 17, 199
    polymorphic, 34
    polymorphic class type, 182
    runtime function, 175
    signature, function, 177
    template, function, 194
    types of the resulting function, 203
one definition rule, 207
operations
    arithmetic, logical and comparison, 293
    Boolean-valued operators, 71
    Boolean wrappers, 61
    comparison, 293
    integer wrappers and, 69
    integral operator, 71
    integral type wrappers, 61
    intrinsic sequence, 90, 109
    logical, 293
    logical operators, 66
    preprocessor array, 304
    preprocessor library arithmetic, 293
    preprocessor library comparison, 294
    preprocessor library logical, 294
    preprocessor sequence, 302
operator
    arithmetic, 72
    bitwise, 72
    Boolean-valued, 71
    function-call, 186
    integral, 71
    integral-valued, 72
    logical, 66, 71
    syntaxes, C++ overloadable, 229
    token-pasting, 300
operator*, 21, 22, 43, 44
operator=(), 21
optimization, 20, 24, 28, 115
optimization, empty base, 187
optional inserter, 124
ordering, strict weak, 122
ordinary functions, 15
output iterator, 117
overload resolution, managing, 207
overload set, 209

P

param_type, 66
param_type, struct, 64, 68
param_types, 67
parameter, macro, 284
parameter, template, 272
parameters, named, 238
parametric polymorphism, 17
parse tables, 225
parser construction, 6
parser generators, 2
parser, high-level, 2
partial
  function application, 53, 240
  metafunction application, 53, 58
  specialization, 31, 100, 105
  specialization of iterator_traits, 14
pasting, token, 299, 300
performance, compile time, 323
Perl, 156
Phoenix library, 243
placeholder, 53–54, 244
  expression, 52, 58
  expression definition, 56
  handling, 50
  implementation of, 54
  unnamed, 55
plus, 53
point of definition, 308
pointer, 11, 13–15
  data members, 25
  design of, 12
  function, 25
  internal, 19
  member functions, 25
  members, 11
  pointers, 50
  single base class, 17
  template arguments, function, 194
  transformation function, 197
points of instantiation, 308
polymorphic metadata, 61
polymorphism, 30–32
  definition of, 17
  example, 39
  function templates and, 196
  parametric, 17
  static, 17, 196
portability, MPL, 343
position, 38
post processing filter, 156
predicate, comparison, 132
predicate, customizing the, 165
preprocessing phase, metafunction of the, 283
preprocessing tokens, 283
preprocessor
  array operations, 304
  data types, 301
  empty argument to the, 297
  file iteration, 290
  fundamental abstractions of the, 283
  fundamental unit of data, 283
  horizontal repetition, 286
  library abstractions, 286
  library arithmetic operations, 293
  library comparison operations, 294
  library integer logical operations, 294
  library structure, 285
  local iteration, 289
  macro, 283
  metaprogram, 282, 288
  metaprogramming, 281
  repetition, 286
  self-iteration, 292
  sequence operations, 302
  vertical repetition, 288, 289
library, 289
preserving the interface, 201
primary
  template, 31
  traits, 25
  type categorization, 25
print_type, struct, 176, 177
printing, type, 176
problem domain, abstractions of the, 8

processing, selective element, 137
productions, BNF, 220
program
  C++, 1, 224
  faster, 7
  interoperability, 16
  test, 326
programming
  compile time, 330
  error, 159
  generic, 17
  higher-order functional, 48
  language, FORTRAN, 217
  object-oriented, 17, 199
properties, DSL, 216
properties, type, 27
property, emergent, 138
proxy reference, 21
`proxy`, `struct`, 21
pseudo-English, 35
pure functional language, 5, 32
pure, metadata, 277

Q

quantities, representing, 40
`quantity`, 41
`quantity`, `struct`, 41, 45
`quantity<float, force>`, 45
querying algorithm, 122, 128

R

`r1`, `typedef`, 22
`r2`, `typedef`, 23
Random Access Iterator, 82, 99, 115, 159
Random Acess Iterator requirements, 83
Random Access Sequence, 85, 92, 109
`range_c`, 93
reachable iterator, 81
realistic error, 146
recurring template pattern, curiously, 203, 208
recursion, 5
recursion unrolling to limit nesting depth, 334
recursive function, 4
recursive sequence traversal, 121
reducing instantiation depth, 336
reference, 13, 63
  bit, 21
  functions, 11
  -ness, 22
  non-const, 22
  proxy, 21
  to references, 66
  types, 22
refine, 77
refinement hierarchy, 181
regular expressions, 215
relation, numeric, 174
relationship between types, 28
repetition
  boilerplate code, 281
  horizontal, 286
  preprocessor, 286
  specialization generated by horizontal, 289
  specialization using horizontal, 286
  vertical, 288, 289
`<repetition.hpp>`, 286
replacement-list, 283, 284, 287
representation, iterator, 99
representing dimensions, 38
representing quantities, 40
reserved identifiers, 149
resolution, overload, 207
return type, 133
reusable mental model, 9
reuse and abstraction, 113
`reverse_fold`, metafunction, 120
`reverse`, `struct`, 120
`reverse_unique`, 126
rule, 207, 218
rules, BNF, 220
rules for `template` and `typename`, 312
rules, grammar, 2
rules, implicit pattern, 219

runtime, 42, 109
  boundary, 277
  boundary, crossing compile-time, 175
  C++, 175
  call stack backtrace, 145
  class template specialization, 179
  complexity, 323
  computation, 6
  constructs, 213
  data corruption, 171
  dispatch, 17
  dispatching, 196
  function, 16
  function objects, 175
  `if` statements, 178
  implementation selection, 178
  linked list, 305
  polymorphic base class, 199
  polymorphism, 252
  tag dispatching, 180

S

Scheme, 3
Scheme metaprogrammer, 3
scoping, dynamic, 250
screensaver algorithm, 197
secondary traits, 26
secondary type categorization, 26
selection
  argument, 296
  constructs, 299
  implementation, 178
  lazy type, 64
  structure, 185
  type, 62
selective element processing, 137
self-documenting code, 226
self-iteration, 292
self-returning metafunction, 98
semantic action, 222
semantic evaluation, 222
semantic value, 222
semantics, 133
separation of concerns, 115
sequence, 115
  algorithm, 78, 109
  algorithms, fundamental, 119
  associative, 86, 87, 109
  bidirectional, 84
  building a tiny, 97
  building algorithms, 119, 123, 125, 126, 128
  combining multiple, 135
  comparing, 96
  concept, 83, 109
  derivation, 96
  derivation to limit structural complexity, using, 339
  elements, 131
  equality, 89–90
  extensible, 86
  extensible associative, 88, 89, 94
  forward, 84, 92
  general purpose type, 93
  implementing a, 138
  integers, large, 94
  integral constant wrappers, 176
  iterator, 77
  lazy, 135, 138
  lazy random access, 93
  `map`, 94
  MPL, 86
  MPL type, 97
  `mpl::list`, 92
  non-empty, 284
  operation, intrinsic, 90, 109
  operations, preprocessor, 302
  querying algorithms, 122
  random access, 85, 92
  sequences, 119
  sorted, 132
  tag, 102
  `tiny`, 97
  traversal algorithms, 120
  traversal concept, 83
  traversal, recursive, 121

- vector, 92
- view, 131
- wrapper, integral, 95
- writing your own, 97

sequence classes, 91
- deque, 93
- iterator_range, 95
- list, 92
- map, 94
- range_c, 93
- set, 95
- vector, 92

set, 95
SFINAE, 211
SGI type traits, 30
signature, struct, 300
single declaration, 314
single template, 30
size_t and numeric wrappers, 70
sizeof trick, 212
slow, compilation, 323
sorted sequence, 132
source code, function, 290
specialization, 31, 89
- class template, 31, 179
- constant time, 103
- explicit, 31
- full template, 317
- generate, 292
- generated by horizontal repetition, 289
- mentioning, 329
- metafunctions, 15
- number of partial, 336
- omitted, 144
- partial, 31, 105
- pattern, 293
- terminating, 5
- tiny_size, 105
- traits template, 15
- using horizontal repetition, 286

specifications, grammar, 6
standard library, 14
state transition table, 259
state vector, 198
static
- assertions, 160, 165, 173
- assertions, MPL, 161
- condition, 178
- interfaces, 173
- member function, 23, 179
- noise, 56
- polymorphism, 17, 196
- type checking operations, 37
- type safety, 260, 277
- visit member function, 178

static_cast, 205
std, namespace, 13
std::
- abs, 15
- binary_function, 296
- for_each, 115
- iterator_traits, 15
- lower_bound, 115
- negate, 17
- reverse_iterator, 138
- stable_sort, 115
- swap(), 19, 22, 23
- unary_function, 296

STL, 58, 77, 79, 128
STL error messages, 156
STLFilt, 172
STLFilt options, 157
storage, eliminating, 187
stored function object, 6
strategy to customize error, 170
strict weak ordering, 122
strings, vectors of, 19
struct
- always_int, 29, 57
- begin_impl, 85
- binary, 4
- bit_iterator, 21
- enable_if, 211
- end_impl, 85
- equal_to, 70
- five, 18

struct (continued)
- function, 296
- int_, 69
- integral_c, 70
- iter_swap_impl, 23
- iterator_traits, 13, 14
- minus_f, 46
- padded_size, 132
- param_type, 64, 68
- print_type, 176, 177
- proxy, 21
- quantity, 40, 45
- reverse, 120
- signature, 300
- tiny_size, 281, 282, 286, 287, 290, 291
- transform, 42
- twice, 49
- type_traits, 30
- visit_type, 178
- wrap, 177

structural
- changes to the class, 186
- complexity of metafunction arguments, 338
- complexity tests, 338
- complexity, using sequence derivation to limit, 339
- variation, 188

structure, preprocessor library, 285
structure selection, 185, 188
structures, control, 295
STT, 259, 262, 264, 276
subrules, 251, 252
Substitution Failure Is Not An Error, 211
substitution, typedef, 147
subtleties, 314
subtraction, addition and, 41
Sutter, Herb, xi, 21
swap(), std, 19, 22, 23
swap(), template, 19
symbols, BNF, 220
symbols, long, 337
syntactic constructs, 229
syntactic noise, 263
syntax, common, 17
syntax disambiguation, 311
syntax of formal languages, 220

T

tables, parse, 225
tag dispatching, 180
tag dispatching technique, 106
tag type, 101, 180
target language interaction, 7
targets, Make, 218
temperature, 38
template
- allowed, 320
- and typename, rules, 312
- apply_fg(), 16
- arguments, eliminating default, 150
- arguments, function pointers as, 194
- binary(), 4, 15
- Blitz++ and expression, 231, 232
- boost::function, 203
- C++, 9
- class, 29
- compilation phases, 308
- compilers with incomplete support for, 343
- dependent names, 319
- disambiguating, 311
- drawback of expression, 236
- error, 143, 320
- error message, 155, 158
- features, traits, 15
- forbidden, 320
- function, 313
- function object, 194
- functions, class, 15
- how to apply, 307
- implementation of a runtime function, 178
- instantiated, 16
- instantiation, 32, 155, 324, 330
- instantiations, nested, 330
- instantiations required, 324
- iter_swap_impl, 23
- iter_swap(), 11–13, 19, 22

iterator_traits, 14
keywords, typename and, 307
lambda non-metafunction, 56
mechanism, 3
members, 91
metaprogram, 1, 24
metaprogram misbehavior, 170
metaprogramming, 5, 9, 57, 156
metaprograms interpretation, 323
multiple return values of traits, 15
name, 31
parameter, 16, 32, 272
parameter lists, 311, 313
parameters, named class, 239
pattern, curiously recurring, 203, 208, 251, 267
primary, 31
required, 319
single, 30
specialization, 31, 55, 338
specialization, class, 31, 179
specialization of traits, 15
specializations, full, 317
struct param_type, 64
swap(), 19, 20
syntax, C++, 91
traits, feature of, 13
type_traits, 30
when to use, 319
wrapper, 177
term, 6
terminating specializations, 5
test programs, 326
testing the metaprogram, 282
tests, structural complexity, 338
TextFilt, 156
theorem of software engineering, fundamental, 13, 19, 263
time, 38
time, compile, 18
tiny, 97
tiny_iterator implementation, 102
tiny_size, 105
tiny_size.hpp, 292
tiny_size, struct, 281, 282, 286, 287, 291
token pasting, 299, 300
token-pasting operator, 300
tokens, preprocessing, 283
tools for diagnostic analysis, 155
traits, 33
blob, 16
boost type, 64
integral valued type, 183
primary, 25
secondary, 26
SGI type, 30
templates feature, 13, 15
type, 31, 33
type manipulation, 11
traits1, typedef, 22
traits2, typedef, 22
transform, 42–44, 46, 48, 114, 119, 185
transform, struct, 42
transform_view, 135
transformations, type, 28
transition table, 262
translators, host language, 3
traversal, 79
traversal adaptor, 138
traversal, recursive sequence, 121
trivial destructor, 24
tuples, 303
twice, struct, 49
type, 17, 29, 39, 77, 168
::value, 18
arguments, 17
associated, 78
associations, 11
associations short cut, 14
categorization metafunctions, 25
categorization, primary, 25
categorization, secondary, 26
computating with, 5
computation, 15
data, 301
dependent, 310

type (continued)
    difference, 13
    different argument, 17
    disambiguating, 310
    element, 86
    erasure, 196, 201, 251, 264
    erasure, automatic, 200
    erasure example, 197
    erasure, manual, 199
    expression, 6
    float, 196
    function pointer, 97
    generate, 192
    identity, 89
    integral, 70
    integral constant wrapper, 17
    iterator, 9, 12
    iterators of different, 19
    iterator's value, 12
    key, 86
    manipulation, traits and, 11
    manipulations, 28
    nested, 15, 30
    non-intrusively, 13
    object, 17
    object of polymorphic class, 182
    of the resulting function object, 203
    parameters, 8
    printing, 170, 174, 176
    properties, 27
    relationships between, 28
    results, 28
    return, 133
    returning a type called, 33
    safety, CRTP and, 205
    selection, 62
    selection, lazy, 64
    sequence general purpose, 93
    sequences, 39
    specifier, 312
    tag, 101, 180
    traits, 30, 31, 33
    traits library, 27
    transformations, 28
    two type members, 117
    value_type, 12–14, 21
    visitation, 177
    wrapper, 33, 39
::type, 31, 59
type_traits, struct, 30
typedef
    boost::function, 203
    r1, 22
    r2, 23
    s, 91, 177
    substitution, 147, 151, 169, 173
    traits1, 22
    traits2, 22
    type, 29
    v1, 22
    v2, 23
    value_type, 14
typename, 12, 13, 310
    allowed, 315
    base class, 316
    class, 310
    error, 316
    forbidden, 316
    full template specializations, 317
    function templates, 313
    how to apply, 307
    iterator_traits, 20, 23
    non-qualified names, 316
    notes, 317
    outside of templates, 316
    required, 312
    single declaration, several, 314
    template keywords, 307
    template parameter lists, 313
    when to use, 312
typeof operator, 213

U

unary_function, 296
unary lambda expression, 53

unary metafunctions, 25
unique, 126
unit, 60
Unix tools, 172
unnamed placeholder, 55
unpack_args, 136
use_swap, 23
user analysis, 7
using recursion unrolling to limit nesting depth, 334
using sequence derivation to limit structural complexity, 339
<utility>, 22

V

v1, typedef, 22
v2, typedef, 23
valid expression, 78
valid iterators, 12
value, 32
::value, 4, 17, 24, 30, 33, 61
value comparison, 71
value, semantic, 222
::value_type, 13, 16
value_type, 12–15, 21, 22
value_type, typedef, 14
values computed from sequence elements, 131
variable part, 31
VC++ 7.0, 150
VC++ 7.1, 150, 159, 168
vector, 19, 92
vector-building inserter, 118
vector properties, 124
<vector20.hpp>, 92
vector<bool>, 21
vectors of strings, 19
Veldhuizen, Todd, 229
vertical repetition, 288, 289
view
    concept, 138
    definition, 131
    examples, 131
    history, 141
    implementing a, 139
    iterator adaptor, 131
    Template library, 141
    writing your own, 139
visit member function, 178
visit_type, struct, 178
visitation, type, 177
Visitor pattern, 177
Visitor::visit(), 178
VTL, 141

W

with clauses, 148, 149
wrap, struct, 177
wrapper, 18
    building, automate, 200
    integral constant, 17, 39, 66
    integral sequence, 40, 70, 95
    MPL Boolean constant, 67
    operations, Boolean, 61
    operations, integer, 69
    operations, integral type, 61
    sequence of integral constant, 176
    template, 177
    type, 33, 39
writing your own view, 139

Y

YACC, 2, 6, 7, 222, 226–228, 257, 261
YACC grammar, 7
yyparse(), 2

Z

zip iterator, 139
zip_view, 140
zip_with, 126