

# Index

## Symbols and Numbers

- ansi switch, 13–14
- pedantic switch, 14
- Wall switch, 14
- (double dash), unary decrement operator, 26, 510
- #(pound character), for preprocessor directives, 15
- #define macro, 127
- #ifndef wrapper, 579
- #include
  - customizing using inheritance, 186–189
  - finding header files, 85–86
  - overview of, 15
  - preprocessor and, 579–581
  - unnecessary dependencies produced by, 174–175
- \$ (dollar sign), anchoring characters, 312
- % (percent sign), modulus operator, 26–27
- & (ampersand)
  - reference parameters using, 118
  - type modifier, 44
  - unary address-of operator, 36–38
- \* (asterisk)
  - multiplication operator, 26
  - quantifier expressions, 311
  - unary deference operator, 37
- /\* and \*/ , in comments, 15
- \*nix platform
  - fixing linker path, 177
  - installing libraries on, 176–178
  - open source development tools, 7–9
  - USER environment variable, 280
  - viewing manual pages, 14
- (...) (ellipsis), 542
- ./., pathname formats and, 250
- :: (file scope resolution operator), 468
- [ ] (square brackets), in command-line arguments, 158
- ^ (caret), anchoring characters, 312
- + (plus sign), addition operator, 25, 510
- ++ (double plus sign), unary increment operator, 26, 510
- < (less than), 27
- < > (angle brackets), #include directive, 85–86
- ≪ (insertion operator), 16
- <= (less than or equal to), 27
- = (assignment operator). *See* Assignment operators
- != (not equal to), 27
- == (equal to), 27
- > (greater than), 27
- >= (greater than or equal to), 27
- ≧ (input operator), 16
- . (dot)
  - in bash shell, 178
  - member access and, 118
  - regular expression meta-character, 310
  - operator overloading and, 115
- ! (exclamation), unary not operator, 27
- ? (question mark), quantifier expressions, 311
- / (slash)
  - division operator, 26
  - as namespace delimiter, 567
- (subtraction operator), 25, 510
- , (comma operator), 115
- () parentheses
  - grouping and capturing characters, 312
  - macros and, 128
- 8-bit integer, 445–446

**A**

Abstract base classes, 148–152  
 Abstract Factory pattern  
     benefits of, 369–372  
     creating rules and friend functions, 366–369  
     defined, 360  
     exercises, 372–373  
     importing objects, 376–380  
     and libraries, 363–365  
     overview of, 361–363  
     qApp and Singleton pattern, 365–366  
 Abstract interfaces, multiple inheritance with, 531–532  
 Accessibility, 52  
 actionEvent( ), 265–266  
 Adaptor pattern, 386–389  
 addAction( ), 262, 264  
 Addition (+) operator, 25  
 AddLayout, widgets, 251–252  
 Address of (&), unary operators, 36–38  
 Addresses, pass-by-reference, 119  
 addSpacing( ), 254–255  
 addStretch( ), 254–255  
 addStrut( ), 254–255  
 addWidget( ), 251  
 Aggregate containers, 222–224  
 Aggregate relationships  
     defined, 100  
     pointer containers and, 221–224  
 Algorithms, generic, 225–227  
 Aliases, namespace, 475  
 amaroK, 427–428  
 American National Standards Institute.  
     *See* ANSI (American National Standards Institute)  
 Ampersand character (&), reference parameters using, 44, 118  
 Ampersand character (&), unary operator, 36–38  
 Anchoring characters, regular expressions, 312  
 Animation, QThread, 290–294  
 Anonymous namespaces, 476  
 ANSI (American National Standards Institute)  
     ANSI C89, 575  
     ANSI/ISO Draft Standard for C++, 6–7  
     new operator, 515  
     typecasts, 346, 450

Anti-patterns, 342–343  
 API (Application Programmer's Interface), 179  
 Applications, reusable components, 171  
 ArgumentList, 159–163  
 Arguments, processing command-line, 158–163  
 Arithmetic operators  
     addition, 25, 510  
     division, 26  
     modulus, 26–27  
     multiplication, 26  
     overview of, 24–25  
     pointers and, 510–511, 513–514  
     subtraction, 25  
     symbols for, 438  
 Array elements, 509  
 Arrays  
     functions and return values and, 511–512  
     kinds of, 513  
     new failures and, 515–519  
     overview of, 509–510  
     reasons to avoid using in C++, 96  
     review questions, 521  
     summary, 519–520  
 Assignable data types, 221  
 Assignment operators  
     with auto\_ptr, 385  
     copy, 64–67, 156  
     for implicitly shared classes, 224  
     pointers and, 513  
     symbols for, 438  
 Assistant, Qt, 593–594  
 Associations, 101  
 Attributes  
     Qt naming conventions, 90  
     XML tags, 324  
 auto\_ptr, 384–385, 388–389

**B**

Base classes, 136–140  
     derivation from abstract, 148–152  
     extending, 140–142  
     inheritance and, 136  
     initializing, 531  
     member initialization for, 140  
     order of initialization, 156  
     overloading, function hiding, and overriding, 154–155

- parent objects vs., 193
  - subclasses derived from, 137
  - virtual, 535–536
  - Bash scripts, 178
  - Behavioral patterns, 182
  - Bi-directional association, in QObject, 193
  - Binary operators, 115
  - Binding
    - compile-time, 144
    - run-time, 142, 144
  - Bitwise operators, 438
  - Block scope
    - vs. file scope, 468–469
    - identifier, 466
    - overview of, 52–53, 465
    - statics defined inside, 63
  - Block statements, 480
  - Boolean
    - expressions, 480–481
    - operators, 27–28
    - types, 22–24
  - boost, 179
  - break
    - from loops, 484
    - from switch, 481
  - Button widgets, 239
  - byte, arrays, 513
- C**
- C
    - C++ as extension of, 6
    - preprocessor, 579
    - standard Library, 578
  - “C with Objects,” 6
  - C++, 5–46
    - arithmetic operators, 25–29
    - brief history of, 6–7
    - const, 34–35, 40–43
    - first example, 12–16
    - identifiers, 19–22
    - input and output, 16–19
    - literals, 19–22
    - main( ) and command line
      - arguments, 24–25
    - overview of, 6
    - pointers and memory access, 36–40
    - preprocessor, 579
    - reference materials for, 601
    - reference variables, 43–44
    - reserved keywords, 575–576
    - scope options, 465–466
    - simple types, 22–24
    - standard Library, 578
    - standard library strings, 30–31
    - streams, 31–34
    - types, 19–22
    - variable initialization, 465
  - The C++ Programming Language* (Stroustrup), 6–7
  - Callbacks
    - defined, 327
    - importing objects with Abstract Factory, 380
  - case labels, 482
  - CaseIgnoreStrings, 227
  - Case-sensitivity, Qt naming conventions, 90
  - Casts, Casting. *See* Typecasting
  - catch statements
    - overview of, 490–494
    - rethrowing caught exceptions, 496
    - throw and, 502
  - Categories, QWidget, 239–240
  - Central widget, 270–272
  - cerr
    - global stream, 31–34
    - input and output, 16
  - char
    - arrays, 381, 513
    - throwing, 497
  - Character sets, regular expressions, 312
  - Character types, 22–24
  - characters( ), 329
  - Children, QObject
    - Composite pattern, 196–199
    - environment variables, 281–282
    - finding, 199
    - management of, 194–196
    - overview of, 192–193
    - QProcess, 279
    - QWidget interacting with, 238
    - widget layout, 202–204, 252
  - Children, XML elements, 324
  - cin
    - global stream, 31–34
    - input and output, 16–19
  - cinlude2dot, 175
  - Circular dependencies, 582
  - class definitions
    - friend declarations within, 56
    - overview of, 49–51

- Class scope
  - defined, 51
  - identifier, 467
  - overview of, 466
- Class templates, generating generic containers, 216–219
- Classes, 47–79
  - const member functions, 68–78
  - constructors, 56–58
  - conversions, 67–68
  - copy constructors and assignment operators and, 64–67
  - definitions, 49–51, 464
  - destructors, 60
  - encapsulation, 54
  - form views, 400
  - friends of, 55–56
  - member access specifiers, 51–53
  - Qt naming conventions, 90
  - reusable components, 171
  - static keyword, 61–64
  - structs, 48–49
  - subobjects, 58–59
  - templates, 216–219
  - UML, 54–55
- className( ), 344
- Client code, 51–53
- Code containers, 170–171
- Code reuse, 579
- CodeVisitor
  - customizing using inheritance, 186–189
  - decoupling, 188–189
- Comma operator (,), 115
- Command line arguments
  - main( ) function and, 24–25
  - processing, 158–163
- Command pattern, 262–267
- Comments, 15
- Comparison, pointer operations, 514
- Compilers
  - GNU C compiler (Gcc), 13–15
  - moc (Meta Object Compiler), 209–210
  - switches, 13–14
  - syntax errors, 587
- Compile-time
  - binding, 144
  - dependency, 173
- Complex numbers, 112–114
- Components
  - Composite pattern, 196–197
  - frameworks with reusable, 179
  - library, 179
- Composite pattern
  - DOM as application of, 330
  - managed containers and, 221–224
  - overview of, 196–197
  - QTreeWidgetItem as implementation of, 417–418
- Composition relationships, UML
  - defined, 55, 99
  - pointer containers and, 221–224
- Compound statements, 480
- Concrete class, 148
- Concurrency, 277–305
  - QProcess. *See* QProcess
  - QThread. *See* QThread
- Conditional
  - statements, 481–482
  - expressions, 28
- Conflicts, resolving multiple inheritance conflicts, 532–534
- Connect to slots, 203–204, 292
- const
  - const\* and \*const, 40–43
  - declaring reference parameter to be, 121–122
  - and globals, 471–472
  - implicitly shared classes *vs.*, 225
  - members, 68–78
  - overloading on const-ness, 124–126
  - overview of, 34–35
  - pointers, 40–43, 513
- const\_cast, 450–453
- Constructors (ctor)
  - conversion, 67–68
  - copy constructors, 64–67
  - exceptions and, 488
  - inheritance and, 155–157
  - overview of, 56–58
  - polymorphism from, 370–372
- Container widgets, 240
- Containers
  - arrays and, 513
  - class templates generating, 216–219
  - code, 170–171
  - defined, 96, 219
  - exercises and review questions, 233–235

- generics and, 219–221
  - implicitly shared, 224–225
  - managed, 221–224
  - overview of, 96–97
  - property, 355–356
  - Qt, 504
  - Serializer pattern, 227–229
  - sorted map example, 229–232
  - Context menus, 261
  - continue, loops, 484
  - Control, inversion of, 325
  - Control structures
    - defined, 479
    - exception expressions, 497–501
    - exception handling, 486
    - exceptions, 485
    - iteration structures, 483–485
    - rethrown exceptions and, 496–497
    - review questions, 502
    - throw statements, 486–488
    - try and catch statements, 490–494
  - Controller classes
    - defined, 284, 393
    - GUI development, 240
    - MP3 player, 553
  - Controller code, 392, 394–395
  - Controlling actions, 404–405
  - Convenience functions, ID3Lib, 384
  - Conversions
    - expressions, 447–449
    - overview of, 67–68
  - Copy assignment operators,
    - 65–67, 156
  - Copy constructors
    - assignment operators and, 64–67
    - for implicitly shared classes, 224
    - never inherited, 156–157
    - not public in QObject, 192
  - Core module, Qt, 91
  - cout
    - global stream, 31–34
    - input and output, 16
  - .cpp extension, class definitions, 50
  - CPPLIBS
    - as environment variable, 280
    - reusing other libraries, 171–172
  - Creational patterns, 360–372
    - applying, 360–361
    - benefits of, 369–372
    - defined, 182
    - exercises, 372–373
    - libraries and, 363–365
    - overview of, 361–363
    - qApp and Singleton pattern, 365–366
    - review questions, 390
    - rules and friend functions, 366–369
  - Cross-language mechanism, 280–281
  - ctor. *See* Constructors (ctor)
  - CustomerFactory, Abstract Factories,
    - 363–365
  - Cycle, 175
  - Cygwin, 12
- D**
- Data members, Qt naming conventions, 90
  - Data model, Mp3File, 553–555
  - Data types
    - assignable, 221
    - GUI development and, 240
    - literals of, 20
  - Database models
    - GUI development and, 240
    - Qt SQL, 429–432
  - Database view, MP3 player, 569–571
  - DataObject
    - encoding/decoding as XML, 373–375
    - form model, 405–409
    - overview of, 353–354
  - DataObjectReader, 377–380
  - DataObjectTableModel, 412–417
  - Debugging
    - building debuggable target, 588–589
    - GNU debugger, 589–590
    - with loggers, 296–297
    - memory errors, 591–593
    - overview of, 587–588
  - Declarations
    - applying, 475
    - definitions compared with, 465
    - names, 464–465
  - Decoupling, 188–189
  - Decrement (—), unary operators, 26
  - Default arguments, 109
  - Default constructors, 57–58
  - Default labels, 482
  - Deference (\*), unary operators, 37
  - Definitions
    - class, 49–51, 56, 464
    - declarations compared with, 465
    - environment variables on \*nix, 178

- Definitions (*continued*)
  - object, function, and class, 464–465
  - polymorphic types, 524
  - private, protected, and public members, 52
  - Serializer pattern, 227–229
  - tables in MySQL, 425–426
  - template definitions in header files, 217
  - undefined pointers, 508
  - undefined reference to [identifier], 586–587
- delegates 360, 395, 405, 406
- delete operator
  - applying to pointers, 506–507
  - heap objects and, 470
  - overview of, 39
- Dependencies
  - circular, 582
  - compile-time, 173
  - customizing using inheritance, 187–188
  - defined, 173
  - managing library, 173–175
- Derivation
  - from abstract base class, 148–152
  - from ArgumentList, 160–163
  - kinds of, 138
  - polymorphism and, 142–147
  - public, protected, and private, 536–538
  - simple, 136–140
- Derived classes
  - employing inheritance using, 137–138
  - order of initialization, 156
  - overloading, function hiding, and overriding, 154–155
- Deserialization, playlists, 560
- Design, inheritance, 152–153
- Design patterns, 182–190
  - Abstract Factory pattern, 360, 361–363
  - Adaptor pattern, 386–389
  - anti-patterns, 342–343
  - Behavioral patterns, 182
  - Command pattern, 262–267
  - Composite pattern, 196–199, 330
  - Creational pattern, 189–190
  - Façade pattern. *See* Façade patterns implementing frameworks with, 179
  - Interpreter pattern, 524
  - Iteration and Visitor pattern, customizing, 184–189
  - MetaObject pattern, 344–345
  - Model-View-Controller (MVC), 392–393
  - Monostate pattern, 242
  - Observer (publish-subscribe) pattern, 200
  - overview of, 182
  - Reflection pattern, 344
  - Serializer pattern, 227–229, 373–380
  - Strategy pattern, 396
  - Visitor pattern, 182–189, 331–334
  - Wrapper pattern, 386
- Designer, Qt, 593–594
- DESTDIR variable, 176, 249
- Destructors (dtor)
  - exceptions and, 488
  - never inherited, 158
  - overview of, 60
  - static keyword and, 61–64
  - virtual, 526–528
- DevC++, 595
- Devel package, reusable components, 171
- Development environment, 579–599
  - building debuggable target, 588–589
  - debugging, 587–588
  - GNU debugger and, 588–590
  - jEdit, 598–599
  - linker, 582–584
  - linker error messages, 584–587
  - open source IDEs and development tools, 594–597
  - preprocessor, 579–581
  - Qt assistant and designer, 593–594
  - UML modeling tools, 597
- Development tools, open source, 594–597
- Dia, UML modeling tools, 597
- Dialogs
  - exercise, 248
  - input dialogs and widgets, 246–247
  - overview of, 244–246
- Directives, preprocessor, 475
- Directories
  - installing libraries in, 176
  - visiting code for, 183
- Display widgets, 240
- distort( ), 300
- Division (/) operator, 26
- .dll file, 176
- do
  - loop, 484
- Docbook, 323, 602
- DocbookDoc class, 335–339

DockWindows, 270–272  
 DOM (Document Object Model)  
     classes, 330  
     defined, 329  
     SAX *vs.*, 330  
 DomWalker, 332  
 Dot (.)  
     in bash shell, 178  
     operator overloading and, 115  
 do . . . while, iteration structures, 484  
 Downcasting. *See* RTTI  
 dtor. *See* Destructors (dtor)  
 Dynamic form models, 393–397  
 Dynamic memory, 511–512  
 Dynamic run-time binding, 144  
 dynamic\_cast. *See also* Typecasting  
     defined, 345  
     qobject\_cast similar to, 346  
     typecasting, 454–456

**E**

Eclipse, 595  
 Editing, with macro expansion, 128  
 Editors, XML, 324  
 Elements, array, 509  
 Ellipsis ( . . . ), 542  
 else, conditional statement, 481  
 emit, 201, 205  
 Encapsulation, 54  
 Encryption, 130–132  
 endElement( ), 378–379  
 endl, as manipulator, 17  
 Entries (array elements), 509  
 enum  
     converting strings to, 350  
     keyword, 443–445  
 Enumerations, 443–445  
 enumerator( ), 350  
 Env command, 10  
 Environment variables  
     on \*nix platform, 178  
     processes and, 280–281  
 Equivalence relation, 233–234  
 Errors  
     liability of macro expansion, 128–129  
     linker error messages, 584–587  
 Event loop, 201. *See also* QApplication, and  
     event loop  
 Event-driven parsing, XML, 325–329  
 eventfilter( ), Qonsole, 286–288

Events  
     Qonsole with keyboard, 286–288  
     QWidgets handling of, 238  
 Exception  
     expressions, 497–501  
     handling, 486  
     overview of, 485  
     rethrown, 496–497  
     safety, 302  
     throw( ) in function signature, 488–489  
     throw statements, 486–488  
     try statements, 490–494  
 Explicit conversions (casts), 449  
 explicit keyword, 68  
 Exporting, to XML, 375–376  
 Expressions  
     evaluating logical expressions, 443  
     explicit conversions (typecasts), 449  
     standard conversions, 447–449  
 Extended regular expressions,  
     Perl-style, 310–311  
 Extending, 140–141  
 eXtensible Markup Language. *See* XML  
     (eXtensible Markup Language)  
 extern keyword  
     declaring static objects, 476–477  
     file scope and, 467  
     global scope and, 466

**F**

Façade patterns  
     exercises, 389  
     Filetagger example, 385–389  
     functional, 384  
     overview of, 381–383  
     review questions, 390  
     smart pointers, 384–385  
 Factories  
     creating questions for forms with,  
         398–399  
     defined, 360  
 Factory method, 360  
 fifo (incoming message queue), 298  
 File formats, MP3 player, 560  
 File scope  
     vs. block scope, 468–469  
     vs. global scope, 466  
     overview of, 466  
 Filenames, finding header files, 86  
 Files, visiting code for, 183

## INDEX

610

- FileTagger
    - auto-generated form, 407
    - façade example, 386–389
    - MP3 player, 553, 568
    - SQL table, 426
  - FileVisitor
    - customizing using inheritance, 186–189
    - making into reusable tool, 184–186
  - Filters, MP3 player, 561–563
  - findChildren( ), 199
  - Floating point numbers, 22–24
  - flush, as manipulator, 17
  - for loops, iteration structures, 484
  - Form views
    - dynamic form models, 395
    - for MP3 player, 568–569
    - overview of, 400–402
  - FormDialog, 400
  - FormFactory, 399
  - FormModel, 397–399, 405–409
  - Forms
    - defined, 393
    - dynamic model, 393–397
  - FormView, 395, 400–402
  - Forward declarations, 175, 580–582
  - Frameworks, library, 178–179
  - friend
    - keyword, 55–56
    - functions, 366–369
  - Functions, 105–133
    - declaring, 106–107
    - declaring inline, 126–127
    - defining, 464
    - ellipsis ( . . . ) and, 542
    - exceptions, 488–489
    - exercises and review questions, 130–133
    - global, 114
    - hiding, 154–155
    - inline vs. macro expansion, 127–130
    - invoking with QMetaObject, 344
    - main( ), 24–25
    - operator overloading as, 111–116
    - with optional arguments, 109–111
    - overloading, 107–109, 154
    - overloading on const-ness, 124–126
    - overriding, 154
    - overview of, 105
    - passing parameters by reference, 118–121
    - passing parameters by value, 116–117
    - prototypes, 106–107
    - public, 54
    - QObjects can never be passed by value to any, 192
    - Qt naming conventions, 90
    - references to const, 121–122
    - return values, 122
    - returning references from, 122–124
    - scope, 465, 467
    - templates, 214–216
    - with variable-length argument lists, 542–543
    - virtual, 414
- G**
- Garbage collection, 543
  - Gcc (GNU C compiler), 13–15
  - gdb (GNU debugger), 588–590
  - Generalization, 137
  - Generic containers, 96
  - Generics. *See also* Templates
    - algorithms and operators, 225–227
    - defined, 96
    - exercises and review questions, 233–235
    - templates, 214–219
  - getChar( ), 279
  - getClassName( ), 138–139
  - getline( ) function, 31
  - getSwitch( ), 161
  - Global functions, 114
  - Global scope, 471
    - vs. file scope, 466
    - identifier, 466
    - partitioning into sub-scopes, 473
  - GNU C compiler (Gcc), 13–15
  - GNU debugger (gdb), 588–590
  - goto
    - avoiding in code, 468
    - switch statement and, 482
  - Graphic images, 248–251
  - Grouping characters, regular expressions, 312
  - Gui module, Qt, 91
- H**
- handler, invoking parser, 325–326
  - Handler classes, 545
  - Header files
    - class definition defined in, 49–50
    - finding with #include, 85–86
    - libraries packaged as lib+, 170



reusable components, 171  
 template definitions in, 217  
 Heap arrays, 96  
 Heap memory  
   benefits of factories, 369  
   corruption, 504  
   garbage collection and, 543  
   new operator allocating storage  
     from, 38  
   pointer problems and, 506–508  
   storage class and, 470  
 Heavyweight objects, 355  
 Hiding functions, 154–155  
 Hierarchy, types, 22, 447  
 HOME, environment variable, 280  
 Host object, 68  
 HOSTNAME, environment variable, 280  
 HTML (HyperText Markup Language)  
   converting XML into, 335–336  
   uses of, 323  
   XML vs., 322–323

**I**

ID3 tags, 381–383  
   reusing, 559–560  
 ID3Lib  
   convenience functions, 384  
   façade example, 385–389  
   overview of, 381–383  
 Identifiers  
   overview of, 19–22  
   scope of, 51, 465  
 Identity, QObject, 192–193  
 IDEs (integrated development environments)  
   finding header files within, 86  
   open source, 594–597  
 if statement, 481  
 Images, QWidget, 248–251  
 Implementation  
   class definitions, 50–51  
   of encapsulation, 54  
   relationships, 537  
 Implicitly shared containers, 224–225  
 Importing objects, with Abstract  
   Factory, 376–380  
 Importing objects with Abstract Factory,  
   SAX parser, 377  
 Include path, files, 85–86  
 Incomplete types, 581  
 Increment ( ++ ), unary operators, 26

Indexing, pointer operations, 514  
 indexOfProperty( ), 350  
 Indirection  
   defined, 38  
   pointer operations, 514  
 Info command, 14  
 Inheritance, 135–165, 523–539. *See also*  
   Multiple inheritance  
     base classes and, 136  
     client code example, 141–142  
     command-line arguments, processing,  
       158–160  
     constructors and, 155–157  
     copy assignment operators and, 156  
     copy constructors and, 156–157  
     defined, 136  
     derivation and ArgumentList, 160–163  
     derivation from abstract base  
       class, 148–152  
     derivation with polymorphism,  
       142–147  
     design, 152–153  
     destructors and, 158  
     exercises and review questions, 163–165  
     function hiding, 154–155  
     member initialization and, 140, 531  
     multiple, 528–532  
     order of initialization, 156  
     overloading, 154  
     overriding, 154  
     polymorphism and virtual destructors,  
       526–528  
     public, protected, and private derivation,  
       536–538  
   QStringList and, 97–99  
   resolving multiple inheritance conflicts,  
     532–534  
   review questions, 539  
   simple derivation, 136–140  
   virtual base classes and, 535–536  
   virtual inheritance, 534–535  
   virtual pointers and virtual tables and,  
     524–526  
   visitor customization with, 186–189  
 inherits( ), 347  
 Initialization  
   base class members, 140  
   class members, 531  
   static, 63–64  
   validators, 309

- Inline functions
    - #define macro *vs.*, 127
    - macro expansion *vs.*, 127–129
    - overview of, 126–127
  - Input and output, 16–19
  - Input dialogs
    - exercise, 248
    - and widgets, 246–247
  - Input widgets
    - defined, 239
    - dynamic form models, 396
    - form views, 402
    - overview of, 308–309
    - unforeseen types, 403–404
  - InputField
    - dynamic forms, 396–397
    - form views, 400–402
  - Insertion operator ( $\ll$ ), 16
  - installEventFilter( ), 288
  - instance( )
    - AbstractFactory and, 361
    - Singleton pattern, 365
  - Instances, class definitions, 49
  - Instantiated, template, 215
  - int, Integer Types
    - arrays and, 512
    - enumerating, 443
    - overview of, 22–24
    - promotion, 447
    - signed and unsigned, 445–446
    - throwing, 497
  - Integrated development
    - environments (IDEs)
    - finding header files within, 86
    - open source, 594–597
  - Interface
    - generic containers, 96
    - relationships between classes, 536–537
  - Internationalization, QObject and, 211
  - Inversion of control, 325
  - iostream, 31
  - is-a relationships, 537
  - ISO, ANSI/ISO Draft Standard
    - for C++, 7
  - istream, 16–19
  - Item models, Qt 4, 409
  - Iteration
    - defined, 16
    - exercises, 101–103, 485
    - overview of, 97
  - QStringList and, 97–99
  - structures, 483–484
  - Iteration, and Visitor pattern, 182–190
    - customizing with inheritance, 186–189
    - exercises and review questions, 189–190
    - overview of, 184–186
    - QDir and QFileInfo (directories and files), 183
- ## J
- JDBC classes, 429
  - JEdit, 598–599
  - join( ), 97–99
- ## K
- kdbg, 271
  - KDE 3.x (K Desktop Environment), 7
  - KDE debugger, 271
  - KDevelop, 595–596
  - Keyboard events, Qonsole with, 286–288
  - keyToValue( ), 350
  - Keywords
    - C++ reserved, 575–576
    - const, 34–35
    - enum, 443–445
    - explicit, 68
    - extern, 467, 476–477
    - friend, 55–56
    - modifying simple types, 22
    - static, 61–64, 467
    - using, 475
    - virtual, 142–147
- ## L
- Late run-time binding, 144
  - Layouts
    - GUI development, 240
    - QObject, 202–203
    - QWidgets. *See* QLayout, widgets
  - LD\_LIBRARY\_PATH, 176–177
  - Leaf nodes, Composite pattern, 197
  - lib files, 170
  - libcustomer, 363–365
  - libdataobjects, 363–365
  - libgtk++, 179
  - Libraries, 169–180
    - Abstract Factories and, 363–365
    - code containers, 170–171

- components, 179
- defined, 169
- dependency management, 173–175
- finding header files within, 86
- frameworks, 178–179
- graphic image, 248–251
- ID3Lib, 381–383
- installing, 176–178, 585
- overview of, 170
- and plugins, 370
- QWidget and, 239
- reusing, 171–172
- review questions, 180
- LIBS variable, 172
- libutils, 171
- Linker
  - arguments to, 583
  - error messages, 584–587
  - linking process, 584
  - overview of, 582–584
  - path, 177
  - switches, 172
- Link-time dependency, 173
- The Linux Development Platform*  
(Rehman and Paul), 84
- List view, media player, 552
- Lists, 95–103
  - containers, 96–97
  - exercises and review questions,  
101–103
  - iterators, 97–99
  - overview of, 95
  - relationships, 99–101
- Literals, 19–22
- Local variables, 350
- Loggers
  - debugging with, 296–297
  - defined, 296
- Logical expressions, evaluating, 443
- Logical operators, 438
- LogWindow, 296
- loops
  - break and continue, 484
  - for, 484
- lupdate tool, 211
- Lvalue, 43
- M**
- M3U file format, 560
- Macro expansion, 127–129
- main()
  - overview of, 24–25
  - QObject child management, 194–196
  - QSettings, 243
- make command
  - cleaning up files, 88–89
  - handling project files with, 84–85
  - overview of, 86–88
- make dist command, 89
- makedep dependency generator, 175
- Makefile
  - cleaning up files, 89
  - example of qmake building, 86–88
  - overview of, 84
  - replaced in Qt by qmake, 85
- man command, 14
- Managed containers
  - implicitly shared, 224–225
  - overview of, 221–224
- Manipulators
  - defined, 17
  - stream, 31–32
- Manual pages, viewing on nix system, 14
- Mapping layer, 415
- Media players
  - components, 552–553
  - MP3 player view features, 563–564
- Member access specifiers, 51–53
- Member functions, 114
- Member initialization, 57–58, 531
- Member selection operators, 457–458
- Memory access, 503–509
  - arrays and. *See* Arrays
  - overview of, 504
  - pointer problems and, 504–506
  - pointer problems with heap  
memory, 506–508
  - pointers, 36–40
  - review questions, 521
  - summary, 509
- Memory allocation, thrashing and, 515
- Memory corruption, 506
- Memory heap. *See* Heap memory
- Memory leaks, 506–507
- Memory management operators, 438
- Meta Object Compiler (moc), 209–210
- Meta-characters, regular expression, 310–312
- Metadata, MP3 songs, 381, 559
- MetaObject pattern, 344–345, 373–375
- methodCount(), 344

- MinGW (Minimalist Gnu for Windows), 12
  - Mixed expressions, 27
  - moc (Meta Object Compiler), 209–210
  - modal attribute, 244
  - Models and views, 391–421
    - controller code, 392
    - controlling actions, 404–405
    - DataObject form model, 405–409
    - dynamic form models, 393–397
    - form models, 397–399
    - form views, 400–402
    - GUI development, 240
    - Model-View-Controller (MVC), 392–393
    - Qt 4, 409–411
    - review questions, 421
    - separating models from views, 392
    - table models, 411–417
    - tree models, 417–420
    - unforeseen types, 403–404
  - Model-View-Controller (MVC), 392–393
  - Modules, Qt 4, 91
  - Modulus (%) operator, 26–27
  - mono, 179
  - Monostate pattern, 242
  - Movie player
    - QPixmap and animation, 290–294
    - with QTimer, 294–295
  - MovieThread, QPixmap and animation, 290–294
  - MP3 files, 381, 553–555
  - MP3 jukebox assignments
    - data model: Mp3File, 553–555
    - database view, 569–571
    - form view for FileTagger, 568–569
    - ID3 tags, reusing, 559–560
    - media player, 552–553
    - MP3 player view features, 563–564
    - persistent settings, 567–568
    - play list models, 565
    - play list serialization, 560
    - Preference class, enumerating, 556–559
    - queries and filters, 561–563
    - source selector, 566–567
    - testing Mp3File related classes, 561
    - visitor generating playlists, 555–556
  - MSYS (from Minimalist Gnu for Windows), 12
  - Multiple inheritance
    - with abstract interfaces, 531–532
    - overview of, 528–529
  - QWidgets using, 238
    - resolving conflicts, 532–534
    - syntax, 529–531
  - Multiple threads, 296–302
  - Multiplication (\*) operator, 26
  - Multithreaded environments, 369
  - MVC (Model-View-Controller), 392–393
  - MySQL, 424–427
    - connecting from Qt, 425
    - overview of, 424–425
    - row insertion, 426–427
    - table definition, 425–426
- ## N
- Namespaces
    - aliases, 475
    - anonymous, 476
    - delimiter for, 567
    - open, 476
    - overview of, 15
    - partitioning global scope into sub-scopes, 473
    - reusable components, 171
    - scope identifier, 467
    - static objects and extern keyword and, 476–477
    - using keyword and, 475
  - Naming conventions
    - destructors, 60
    - Qt guidelines, 90–91
  - Net module, Qt, 91
  - new operator
    - failures, 515–519
    - heap objects and, 470
    - memory leaks and, 507
    - overview of, 38–39
  - newObject( ), 361–365, 380
  - nix platform. *See* \*nix platform
  - Nodes, XML, 324, 330
  - Non-const reference parameters, 118–119
  - Not (!), unary operator, 27
  - nothrow, 544
  - NULL
    - new failures and, 518–519
    - pointers, 36, 506
- ## O
- Object files, 170
  - Object module, 171

- Object oriented programming (OOP), 601–602
  - ObjectFactory
    - Abstract Factories and libraries and, 362–365
    - managing singleton instance of, 365–366
    - in multithreaded applications, 369
  - Objects
    - changes to, 544–547
    - class definitions, 49, 464
    - defined, 36
    - global, 471
    - resource sharing and, 543
    - subobjects, 58–59
  - objectToXML ( ), 375–376
  - Observer (publish-subscribe) pattern, 200
  - Observer pattern, 200
  - OkAction, 404–405
  - One-to-many relationship, 99
  - One-to-one relationship, 99
  - Online resources
    - ANSI/ISO Draft Standard for C++, 7
    - downloading open source tarball, 9–11
    - gcc documentation, 14
    - qmake, 89
    - Qt, 89
    - Qt 4 Thread Support, 302
    - shell scripting, 178
  - OOP (object oriented programming), 601–602
  - Open namespaces, 476
  - Open source
    - defining, 7
    - downloading from source, 9–11
    - IDEs and development tools, 594–597
    - requiring Qt 4, 7–9
  - Operations, with pointers, 513–514
  - Operators, 438–442
    - arithmetic. *See* Arithmetic operators
    - assignment. *See* Assignment operators
    - binary, 115
    - boolean, 27–28
    - characteristics of, 439
    - classified by use, 438
    - delete, 39, 470, 506–507
    - generic, 225–227
    - insertion, 16
    - list of C++ operators, 440–442
    - member selection, 457–458
    - modulus, 26–27
    - new, 38–39, 470, 507, 515–519
    - overloading, 111–116
    - Run-Time Type Identification, 345–347
    - scope resolution, 50, 468
    - Serializer pattern and overloaded i/o, 227–229
    - shortcut, 26
    - sizeof( ), 23–24
    - typecast, 346
    - typeid, 345
    - unary, 26–27, 36–38, 115
  - Optional arguments
    - enclosing in square brackets, 158
    - functions with, 109–111
  - Ostream, input and output, 16–19
  - Output. *See* Input and output
  - Overloading
    - on const-ness, 124–126
    - functions, 107–109, 154
    - operators, 111–116
    - unary operators and, 115
  - Overriding functions, 154
- P**
- Parameters
    - command-line arguments, 158
    - function prototypes using, 106–107
    - optional arguments and, 109–111
    - QSettings string, 242
    - reference, declaring to be const, 121–122
    - reference, overview of, 118–121
    - template *vs.* function, 214
    - value, 116–117
  - Parents, QObject
    - base classes *vs.*, 193
    - Composite pattern, 196–199
    - layout of widgets, 251
    - overview of, 192–193
    - QProcess, 279
    - QWidgets interacting with, 238
  - Parents, XML elements, 324
  - parse( ), 325–326, 329
  - Parse event handler, 327
  - Parsers
    - event-driven, 325–329
    - SAX, 330–334, 377
    - XML, 327
  - Partitioning, global scope into
    - sub-scopes, 473

## INDEX

616

- Pass-by-pointer, 120–121
  - Pass-by-reference, 120–121
  - Passive interface, 201, 326–327
  - PATH
    - as environment variable, 280
    - fixing linker path in Windows, 176–177
  - Paths, finding header files, 85–86
  - Patterns. *See* Design patterns
  - Performance, inline functions and, 126–127
  - Perl, regular expressions, 311
  - Persistent settings, MP3 player, 567–568
  - Play lists, MP3 player, 555–556, 561
  - Player view, MP3 player, 552, 563–564
  - Plug-ins
    - and libraries, 370
    - parsing XML with, 327
  - Pointers
    - arithmetic operators and, 510–511
    - const, 513
    - containers, 221
    - heap memory problems and, 506–508
    - memory access and, 36–40
    - operations with, 513–514
    - overview of, 22–24
    - problems due to improper handling of, 504–506
    - to QObject children, 192
    - smart, 39, 457
    - symbols for, 438
  - Polymorphism
    - from constructors, 370–372
    - defining polymorphic type, 524
    - derivation with, 142–147
    - exercises and review questions, 163–165
    - virtual destructors and, 526–528
  - POSIX (Portable Operating System Interface for UNIX), 7–9
  - Preference class, enumerating for MP3 player, 556–559
  - Prepared statements, 427
  - Preprocessor
    - development environment, 579–581
    - directives, 15
    - macros, 35
  - Primitives, 350
  - private derivation, 530, 536–538
  - private member, 52, 55
  - .pro file, 89
  - process( ), 300
  - Process control. *See* QProcess
  - processDir( ), 186
  - processFile( ), 184–186, 187
  - Profiler, finding memory errors, 591–593
  - Program stack, storage class and, 470
  - Programming style, Qt guidelines, 90–91
  - Project files
    - cleaning up, 88–89
    - defined, 83
    - finding header files, 85–86
    - handling with make command, 83–85, 86–88
  - Promotion, expression conversion, 447
  - Properties
    - accessing, 350–352
    - containers (PropsMap), 355–356
    - describing QObject, 347–350
  - property( ), 352
  - PropQuestion, 406, 408
  - PropsMap, 355–356, 362
  - protected derivation, 530, 536–538
  - protected member, 52, 137
  - public derivation, 530, 536–538
  - public functions, 54
  - Public interface, 54
  - public member, 52, 55
  - Pure virtual functions, in abstract base classes, 149–152
  - push( ), 218
- ## Q
- Q\_ENUM macro, 350
  - Q\_PROPERTY macro, 347–350, 351
  - QAbstractItemModel, 417
  - QAbstractTableModel, 411–412, 414, 429
  - QAbstractxxxModel, 411
  - QActionGroups, 262–267
  - QActions
    - exercises, 267–269
    - implementing Command pattern, 262–267
    - QMenu, QMenuBar and, 260–262
    - Qtoolbars, QActionGroups and, 262–270
    - synchronizing data between model and view, 403–404
  - qApp
    - defined, 203
    - signals and slots, 204–209
    - Singleton pattern and, 365–366

- QApplication, and event loop, 200–209
  - connecting to slots, 203–204
  - layouts, 202–203
  - overview of, 200–202
  - signals and slots, 204–209
- QApplication, example creating, 82–83
- QBoxLayout, 251
- QByteArray, QSettings, 242–243
- QCache<Key,T>, 220
- QCoreApplication functions, 242
- QDate member functions, 92–93
- QDefaultxxxModel, 411
- qDeleteAll ( ), 222
- QDialog, 244–247
- QDir, 183
- QDockWidgets, 270–272
- QDomDocument, 330, 339
- QDomElement, 330, 333, 335–336
- QDomNode, 330–331, 333, 339
- QDoubleValidator, 308–309
- QEvents, 200–202, 286
- QFileInfo, 183
- QGridLayout, 251–260
- QHash<Key,T>, 220
- QHBoxLayout, 251
- QImage, 249
- QIntValidator, 308–309
- qjots application, 418–420
- QLabel, 82–83, 253
- QLayout, widgets, 251–260
  - exercises, 258–260
  - moving widgets across layouts, 256–258
  - overview of, 202–203, 251–254
  - spacing, stretching and struts, 254–255
- QLineEdit, 402
- QLinkedList<T>, 220
- QList, 96–97, 102
- QList<QString>, 220
- QList<T>, 219
- QListView, 417
- QMainWindow
  - managing dock window regions, 270–272
  - overview of, 240–241
  - QSettings and, 242–243
  - restoreState( ), 243
  - saveState( ), 242
- qmake
  - cleaning up files, 89
  - example of, 86–88
  - installing libraries, 176–177
  - online guide to, 89
  - overview of, 85
  - reusing other libraries, 172
  - downloading from source, 10
  - Win32 setup, 12
- QMap
  - example of, 229–234
  - implementing property containers, 355–356
- QMap<Key,T>, 220
- QMenu, 260–262, 267–270
- QMenuBar, 260–262, 267–270
- QMessageBox, 244–246
- QMetaObject, 344–345
- QMetaProperty
  - accessing properties, 352
  - describing QObject properties, 349–350
  - overview of, 344–345
- QModelIndex, 409, 411
- QMultiMap<Key,T>, 220
- QMutex, 302
- qobject\_cast, 345–347
- QObject::inherits( ), 345
- QObjectList, 192
- QObjects, 191–212
  - child management in, 194–196
  - Composite pattern, 196–199
  - connecting to slots, 203–204
  - DataObject extension of, 353–354
  - defined, 192
  - layouts, 202–203
  - moc and, 209–210
  - overview of, 192–193
  - QApplication and event loop, 200–209
  - QWidgets as, 238
  - review questions, 212
  - signals and slots, 204–209
  - storage class, 471
  - thread safety and, 302
  - tr( ) and internationalization, 211
  - values and objects, 210
- Qonsole
  - with keyboard events, 286–288
  - writing Xterm in Qt, 284–286
- QPaintDevice, 238
- QPicture, 249
- QPixmap
  - animation, 290–294
  - handling images, 249–251

INDEX

618

- QProcess, 278–289
  - exercises, 288–289
  - overview of, 278–280
  - processes and environment, 280–283
  - Qonsole, 284–288
  - QThread *vs.*, 304
  - review questions, 305
- .qrc resource files, 248
- QRegExp
  - overview of, 310–312
  - phone number recognition, 313–316
  - regular expression validation, 316–317
- QRegExpValidator, 316–317
- QSemaphore, 302
- QSet<T>, 220
- QSettings, 242–243, 405
- QSlider widget, 293
- qSort( ), 225–227
- QSplitter, 296
- QSqlQuery, 429
- QStack<T>, 220
- QStackedLayout, 251
- QStackedWidget, 251
- QString::arg( ), 211
- QStringList
  - adding CaseIgnoreStrings to, 227
  - defined, 220
  - derivation and ArgumentList, 160–163
  - as implicitly shared classes, 225
  - input dialogs and widgets, 246–247
  - and iteration, 97–99, 101–102
  - processing command-line arguments, 160
  - views of, 272–274
- QStrings
  - debugging and, 590
  - example using, 82–83
  - as implicitly shared classes, 225
  - input dialogs and widgets, 246–247
  - processing command-line arguments, 160
- Qt, 233–235
  - assistant and designer, 593–594
  - building with debugging symbols, 588
  - connecting to MySQL, 425
  - containers, 504
  - core modules, 91
  - dates, 91–93
  - defined, 81
  - exercises and review questions, 93–94
  - getting help online, 89
  - heap memory cleanup, 39
  - lists, 96
  - Makefile, 82–85
  - namespace delimiter, 567
  - project files, 83–89
  - QApplication and QLabel, 82–83
  - QDir and QFileInfo for visiting files, 183
  - qmake, 85
  - reference material, 601
  - setup, open source platforms, 7–11
  - setup, Win32, 12
  - streams, 91–93
  - style guides and naming conventions, 90
  - widgets. *See* QWidgets
  - XML Module, 325
- Qt 3, 7
- Qt 4
  - installing from packages, 8–9
  - models and views, 409–411
  - modules, 91
  - nix open source platform requiring, 7
  - reusable components of, 179
  - viewing version installed on your system, 8
- Qt Interest Mailing List, 89
- Qt source tarball, 10
- Qt SQL, 424–433
  - database models, 429–432
  - introduction to MySQL, 424–427
  - queries and result sets, 427–429
  - review questions, 433
- QTableView, 411–412, 414, 417, 429–431, 565, 569
- QtCentre, 89
- QTextStream, 31, 82–83, 91–93
- QThread, 290–304
  - exercises and review questions, 303–305
  - movie player with QTimer, 294–295
  - multiple threads, queues and loggers, 296–302
  - overview of, 290
  - QPixmap and animation, 290–294
  - QProcess *vs.*, 304
  - thread safety and QObjects, 302
  - using QTimer *vs.*, 295
- QTimer, 294–295
- QToolBar, 262–267
- QTreeView, 417
- QTreeWidgetItem, 417, 419
- Quantifiers, regular expressions, 311



- Queries
    - MP3 player, 561–563
    - Qt SQL, 427–429
  - Questions
    - dynamic form models, 395
    - form models, 397–399
    - rephrasing, 406
  - Queues, 296–302
  - QValidator, 308
  - QVariant, 350–352
  - QVBoxLayout, 251
  - QVector<T>, 220
  - QWaitCondition, 302
  - QWidget::addLayout, 251
  - QWidgets
    - categories, 239–240
    - defined, 238
    - dialogs, 244–248
    - images and resources, 248–251
    - layouts, 202–203, 251–260
    - overview of, 238–239
    - QActions, QMenu and QMenuBar, 260–262
    - QActions, Qtoolbars, and QActionGroups, 262–270
    - QMainWindow, 240–241
    - QSettings, 242–243
    - regions and QDockWidgets, 270–272
    - review questions, 275
    - sending QEvents, 201
    - signals and slots, 205–209
    - views of QList, 272–274
  - QXmlContentHandler, 326–327, 328
  - QXmlDefaultHandler, 328, 378
  - QXmlSimpleReader, 326–327
  - QXMLSimpleReader, 378
- R**
- rcc, resource compiler, 249
  - read( ), 279
  - readAllStandardOutput( ), 279
  - reader, invoking parser, 325–326
  - Reading strings, 33–34
  - readLine( ), 279
  - readyReadStandardOutput( ), 279
  - Refactoring, 136
  - Reference counting, 543
  - Reference parameters
    - declaring to be const, 121–122
    - overview of, 43–44, 118–121
  - Reference returns, from functions, 122–124
  - Reflective, defined, 343
  - Reflective programming, 341–358
    - anti-patterns, 342–343
    - DataObject, 353–355
    - exercises, 354–355
    - PropsMap, 355
    - Q\_PROPERTY macro, 347–350
    - QMetaObject, 344–345
    - QVariant class, 350–352
    - review questions, 357
    - RTTI and qobject\_cast, 345–347
  - Reflection pattern, 344
  - regex. *See* Regular expressions (regex)
  - Regions, QDockWidgets and, 270–272
  - Register, storage class, 470
  - Regular expressions (regex)
    - exercises and review questions, 318–319
    - overview of, 310–311
    - phone number recognition, 313–316
    - syntax, 311–312
    - validation, 316–317
  - reinterpret\_cast, 453–454
  - Relational operators, 438
  - Relationships
    - defined, 55
    - exercises, 101
    - overview of, 99–101
    - review questions, 103
  - Reparenting, in QObject, 193
  - Required arguments, 158
  - Reserved keywords, C++, 575–576
  - Resource Collection File, 248
  - Resources
    - QWidgets, 248–251
    - sharing, 543, 547
  - restoreState( ), QMainWindow, 243
  - Result sets, Qt SQL, 427–429
  - Rethrown, exceptions, 496–497
  - Return values
    - arrays, 511–512
    - functions, 122
  - Returning references, from functions, 122
  - Reusing, other libraries, 171–172
  - Root, of tree, 197
  - Row insertion, MySQL, 426–427
  - RTTI (run-time type identification), 454–458
    - dynamic\_cast, 454–456
    - qobject\_cast and, 345–347
    - typeid( ), 456

- run( ),
  - QProcess, 292
  - QThread, 300
- Run-time binding
  - dynamic or late, 144
  - enabling with virtual keyword, 142
- Run-time errors, debugging, 588
- run-time type identification. *See*
  - RTTI (run-time type identification)
- S**
- saveState( ), QMainWindow, 242
- SAX parser
  - DOM vs., 330
  - importing objects with Abstract Factory, 377
  - overview of, 331–334
- Scope
  - block, 52–53
  - class, 51
  - declaration determining, 464
  - file scope vs. block scope, 468–469
  - function, 465, 467
  - global, 471
  - identifier, 51, 465, 467
  - resolution operators, 50, 438
  - review questions, 478
  - storage class compared with, 470
  - types of, 465–468
- Searches, finding header files, 85–86
- Selection models, Qt 4, 409
- Selection statements
  - defined, 15
  - exercise, 483
  - overview of, 480–482
- Selector, media player, 552
- Serialization, playlist, 560
- Serializer pattern, 373–380
  - defining, 227–229
  - exporting to XML, 375–376
  - importing objects with Abstract Factory, 376–380
  - overview of, 373–375
  - review questions, 390
- setApplicationName( ), 242
- setContent( ), 330
- setGeometry( ) function, 250
- setOrganizationName( ), 242
- setReadChannel( ), 279
- Setup
  - open source platforms, 7–11
  - Win32, 12
- setUpForm( ), 309
- setValue( )
  - form models, 397–398
  - QSettings, 242
- Sharing resources, 543, 547
- Shell scripting, 178
- Shortcut operators, 26
- Shout button, 202
- Signals
  - defined, 204
  - making concurrent code easier to read, 279
  - QMetaObject, 344
  - QObject, 204–209
  - slots and, 204–209
  - speaking with, 297–302
  - synchronous or asynchronous, 208
  - transmitting data to objects across threads, 292
- Signatures, function, 107
- Signed integral types, 445–446
- Simple statements, 480
- Simple types, 22–24, 29–30
- SimpleListApp, 272–274
- Singleton pattern
  - defined, 360–361
  - qApp and, 365–366
- Size, pointers, 24
- sizeof( ) operator, 23–24
- Slacker class, DOM tree walking, 332–334
- Slacker's DocBook, 323
- Slash (/), as namespace delimiter, 567
- Slots
  - connecting signals with, 301–302
  - connections to, 203–204
  - defined, 204
  - making concurrent code easier to read, 279
  - QMetaObject, 344
  - signals and, 204–209
- Smart pointers, 39
  - auto\_ptr, 384–385
  - member selection, 457
- Sorting, qSort( ), 225–227
- Source code
  - libraries packaged as, 170
  - reusable components, 171

- Source selector, MP3 player, 553, 566–567
- Spacing, widget layout, 254–255
- Special characters, regular expressions, 310
- split( ), QList and iteration, 97–99
- SQL, Qt, 424–433
  - database models, 429–432
  - introduction to MySQL, 424–427
  - overview of, 91
  - queries and result sets, 427–429
  - review questions, 433
- Standard headers, 577–578
- Standard Library (STL)
  - dynamic memory and, 504
  - finding header files within, 86
  - heap memory cleanup, 39
  - lists, 96
  - standard headers, 577–578
  - strings, 30–31
- Standards, C++, 6–7
- start( )
  - processes, 278–279
  - threads, 291
- startElement( ), 378–379
- State of the object, 49
- Statements
  - block, 480
  - compound, 480
  - conditional, 481–482
  - connect( ), 292
  - defined, 479
  - overview of, 480
  - prepared, 427
  - review questions, 502
  - selection, 15, 480–483
  - simple, 480
  - switch, 481–482
  - throw, 486–488, 497–498
  - try and catch, 490–494
- static
  - binding, 144
  - block-scope, 63
  - keyword, 61–64, 467
  - local variables, 350
  - storage area, 470
  - using in Singleton pattern, 365
  - declaring, 476
  - namespaces and, 476–477
  - static\_cast, 450
- stderr, 279
- std::list, 96
- stdout, 279
- STL (Standard Template Library). *See* Standard Library (STL)
- Storage class
  - const, 471–472
  - exercise, 472–473
  - globals, statics, and QObjects, 471
  - overview of, 470
  - register, 470
  - review questions, 478
- Strategy pattern, 396
- Streams, 31–34
- Stretching, widget layout, 254–255
- String literals, 20–21
- StringInputField, 402
- Strings
  - converting to enums, 350
  - QList and iteration, 97–99
  - reading, 33–34
  - Standard Library (STL), 30–31
  - writing, 32–33
- Stroustrup, Bjarne, 6–7
- struct
  - arrays of, 513
  - classes *vs.*, 53
  - overview of, 48–49
- Structural patterns, 182
- Struts, widget layout, 254–255
- Subclasses
  - defined, 137
  - QLayout, 251
- SubObjects, 58–59
- Subtraction (–) operator, 25
- Suffolk University, 197–198
- superClass( ), 344
- switch statement, 481–482
- Switched parameters, command-line arguments, 158
- Switches
  - command-line arguments, 158–163
  - compiler, 13–14
- Symbols, enclosing, 15
- Syntax
  - compiler errors, 587
  - multiple inheritance, 529–531
  - regular expressions (regex), 310–312
  - throw, 494

**T**

Table definition, MySQL, 425–426  
 Table models, 411–416  
 taglib, 381  
 Tags, XML, 324  
 Tarball
 

- downloading, 9
- overview of, 9

 TARGET, make command, 88  
 target.path, make command, 88  
 Template<class T>, 217  
 Templates. *See also* Design patterns
 

- auto\_ptr, 384–385
- causing generated code for each type, 350
- class, 216–219
- container, 219–221
- declaration code for, 217
- functions vs., 214–216
- generics and, 214–219
- instantiated, 215
- parameters, 214
- QList as template class, 96–97

 Text editors, 598  
 this pointer, 68–70  
 Thrashing, memory allocation and, 515  
 Threads. *See* QThread  
 Thread-safe objects, 302  
 Throw( ), in function signature, 488–489  
 throw statements
 

- exception expressions and, 497–498
- overview of, 486–488
- uses of, 494

 Tilde (~) character, destructor
 

- names, 60

 toString( ), 138–139, 142, 351–353  
 tr( ), and internationalization, 211  
 Trailing arguments, functions with, 109  
 Tree models, 417–420
 

- exercises, 420
- extended tree widget items, 418–420
- overview of, 417–418

 triggered( ) signals, QAction, 264  
 Trolltech Online Documentation, 89  
 try statements
 

- nesting, 498
- overview of, 490–494

 type( ) member function, 200  
 Type modifier, 44  
 Typecast operators, 346

Typecasting, 449–454
 

- ANSI standards, 450
- const\_cast, 450–453
- C-style, 454
- downcasting, 333, 454
- dynamic\_cast, 454–456
- overview of, 449
- reinterpret\_cast, 453–454
- static\_cast, 450
- typeid( ), 456

 typeid operator, 345, 456–458, 524–525  
 Type-restricted, 385  
 Types, 437–461
 

- casting, 449–454
- conversion, 447
- enumeration, 443–445
- exercises, 458–460
- hierarchy of, 447
- logical expressions, 443
- member selection, 457–458
- operators, 438–442
- overview of, 19–22
- review questions, 461
- run-time type identification (RTTI), 454–456
- signed and unsigned integral types, 445–446
- simple, 22–24, 29–30
- variables, 49

**U**

Umbrello design tool, 54, 597  
*The Umbrello UML Modeller Handbook*, 54  
 UML (Unified Modeling Language)
 

- diagramming inheritance, 137
- inheritance design with, 153
- introduction to, 54–55
- modeling tools, 597
- relationships, 55

 Unable to Find libxxx.so.x, linker error
 

- messages, 585

 Unary operators
 

- address of (&), 36–38
- decrement (—), 26
- deference (\*), 37
- increment (++), 26
- not (!), 27
- overloading and, 115

 Undefined pointers, 508

- Undefined Reference to [identifier], linker error messages, 586–587
  - Undefined Reference to vtable for `ClassName`, linker error messages, 587
  - Unforeseen types, 403–404
  - Unified Modeling Language. *See* UML (Unified Modeling Language)
  - Union, 351
  - Unmanaged containers, 221–224
  - Unsigned integral types, 445–446
  - `updateCursor()`, 287–288
  - USER, as environment variable, 280
  - USERNAME, as environment variable, 280
  - using declaration, namespaces, 15, 475
  - `/usr/local`, 176
  - utils library, reusing, 171–172
- V**
- `valgrind`, profiler, 591–593
  - Validation, 307–319
    - exercises and review questions, 318–319
    - phone number recognition, 313–316
    - regular expressions, 316–317
    - regular expressions syntax, 310–312
    - using for, 310
    - validators, 308–309
  - Validators, 308–309
  - Value containers, 221
  - Value parameters, 116–117
  - Values
    - function return, 122
    - `QObject`, 210
  - Variable-length argument lists, 542–543
  - Variables
    - class types, 49
    - `const`, 471–472
    - declarations, 15
    - defined, 36
    - environment variables, 280–282
    - global, 466
    - initialization in C++, 465
    - local, 350
    - reference, 43–44
  - vector class, 488–489
  - Views. *See also* Models and views
    - form, 400–402
    - Qt 4, 409–411
    - separating models from views, 392
  - Virtual base classes, 535–536
  - Virtual destructors, 526–528
  - virtual functions
    - overriding and, 154
    - pure, 149–152
    - `QAbstractTableModel`, 414
  - Virtual inheritance, 534–535
  - virtual keyword
    - derivation with polymorphism using, 142–147
    - enabling runtime binding with, 142
  - virtual methods, parsing XML with, 327
  - Virtual pointers, 524–526
  - Virtual tables (vtables), inheritance and, 524–526
  - Visibility, 52
  - Visitor, generating playlists, 555–556
  - Visitor pattern, iteration and, 182–190
    - customizing with inheritance, 186–189
    - DOM tree walking, 331–334
    - exercises and review questions, 189–190
    - overview of, 184–186
    - `QDir` and `QFileInfo` (directories and files), 183
  - `void`, 22
  - Vtables (vital tables), 524
- W**
- `walkTree()` method, 333
  - while, iteration structures, 483–484
  - Whitespace, pointer problems and, 505
  - Widgets. *See also* `QWidget`
    - displaying current play list on MP3 player, 565
    - Qt designer, 594
    - tree, 417–420
  - Win32, setup, 12
  - Window, `QWidget`, 238
  - Windows
    - environment variables, 281
    - installing libraries in, 176–177
    - USERNAME environment variable, 280
  - Wrappers
    - `FileTagger` (façade example), 386–389
    - header files, 50
    - using `auto_ptr` in, 384–385
  - `write()`, 279
  - Writing strings, 32–33
  - Wt<sup>8</sup>, 179
  - `wxWidgets`, 179



## INDEX

624

**X**

XML (eXtensible Markup Language),  
321–340  
encoding/decoding DataObjects as,  
373–375  
event-driven parsing, 325–329  
exercises and review questions, 339  
exporting to, 375–376  
generating output with DOM,  
335–339

HTML vs., 321–323  
importing objects with Abstract Factory,  
376–380  
nodes, 324  
Qt XML Module, 325  
tree structures and DOM, 329–334  
XML editors, 324  
Xml module, Qt, 91, 325  
xmllint, 325  
Xterm, 284–286









