

# Index

## Symbols and Numerics

---

- & (ampersand), encoding
  - required for Web use, 337
- \* (asterisk)
  - with `Match` class, 333
  - as wildcard, 330, 333
- `#define` statements
  - `const` statement compared to, 45
  - constants for directly replacing values, 77–78
  - using `const` instead of, 45–47
- > (greater-than sign), encoding
  - required for Web use, 337
- `#include` statements for header files, 39, 40
- < (less-than sign), encoding
  - required for Web use, 337
- % (percent sign) as wildcard, 331
- + (plus operator), overloading, 120–122
- += (plus-equal operator), plus operator implementation and, 121
- ? (question mark)
  - with `Match` class, 333
  - as wildcard, 330–331, 333
- 32-bit operating systems, 54

## A

---

### abstraction

- common base class for, 18
- defined, 12
- encapsulation for, 12
- for extending functionality, 12–18

- mailing-list application
  - example, 12–18
- virtual methods and, 12
- accessor functions or methods
  - making inline, 407–408
  - for `MyString` class, 123
- `AgeProperty` class, 139–140
- algorithms
  - choosing the most efficient, 350
  - `Date` class, 153, 154, 159
  - discrete pieces in classes
    - for, 159
  - encapsulating, 7–10, 36
  - for encryption, 343
  - for encryption method,
    - encapsulating, 7–10
  - hiding from developers, 7–8
  - `Rot13` encryption algorithm, 343, 344–346
  - STL advantages for, 200
  - for STL container classes, 200, 349–350
  - `transform` function, 349–353
  - updating encapsulated algorithms, 10–11
  - vector algorithms, 200–203
  - for virtual files, 280, 290
  - XOR encryption algorithm, 343, 346–348
- `allocate` method, overriding, 299, 300
- American Heritage Dictionary*, 12
- ampersand (&), encoding
  - required for Web use, 337
- `anewfile.out` file, 431
- `anoldfile.out` file, 431
- application development
  - breaking classes into discrete pieces, 159
  - breaking complex system into components, 217
  - building tracing into applications, 375–380

- code updated by another source and, 7
  - configuration capability as hallmark, 251
  - creating general classes and, 175
  - eliminating the source of failures, 323
  - encapsulation and, 7–8, 11
  - enforcing return codes, 323–329
  - exception handling and, 319
  - exchanging data with Web-based applications, 342
  - factory pattern for, 162
  - getting the design right, 440
  - hiding algorithms from developers, 7–8
  - inserting tracing into an existing file, 380–386
  - internationalization and, 265–266
  - keeping a library of utility classes, 353
  - logging and, 389
  - memory trackers and, 135
  - optimizing code, 407–415
  - planning for Web-enabled code, 337
  - providing test suite with application file, 436
  - reducing the complexity of code, 447–453
  - stepping back from problems, 440
  - storing literal string information in classes and, 161
  - validation classes and, 142, 149
- ### arguments
- class template arguments, 179, 182, 183
  - customizing `MessageBox` function for, 102

- arguments (*continued*)
    - default, defining for functions and methods, 101–106
    - immutable, functions with, 78–79
    - non-class template arguments, 184–185
    - in signatures for methods, 399
    - types for values passed to functions, 90–91, 93–94
  - arrays
    - allocations and de-allocations, 204–208
    - Buffer class versus, 361, 363–364
    - compiler errors for, 47
    - delete operator with, 204
    - FileChunkManager class for managing, 288
    - of heterogeneous objects, 213–215
    - iterating over, 291, 292–293
    - Matrix class allowing queries for, 63–69
    - multiple array classes and the STL, 196
    - MyStringArray class, 196–199
    - new operator and, 134–135
    - of object pointers, 213–215
    - of objects, 209–212
    - overriding operators and, 63
    - pre-processor and, 47
    - printing using streams, 226–227
    - sizeof function and, 55
    - for spreadsheets, 216–222
    - static, 209, 211, 361
    - two-dimensional, 222
    - vector algorithms for, 200–203
    - vector class (STL) for, 25, 192–195, 200–203, 209–212, 226–227
    - with and without pointers, 204–208
  - assert macro
    - “debug mode” for, 388
    - debugging and, 42, 44, 387–389
    - error handling with, 44
    - for exiting programs,
      - avoiding, 43
      - never counting on, 389
      - not defined in optimized mode, 387, 389
      - output from, 388
      - purpose of, 42, 44, 387
      - “release mode” and, 388
      - run-time and, 42, 389
      - testing in optimized environment, 44
      - turning asserts on and off, 387–388
      - using, 42–43, 388–389
  - assignment
    - initialization versus, 413–415
    - operators, extensibility and, 61
    - properties and invalid assignments, 136–137
  - asterisk (\*)
    - with Match class, 333
    - as wildcard, 330, 333
  - auto\_ptr class
    - benefits of, 303
    - copying an auto\_ptr, 306
    - rules for using, 306
    - STL collections and, 306
    - testing, 305–306
    - using with functions, 303–305
- 
- ## B
- 
- Bar class, 53, 54
  - Base class
    - array of object pointers for, 213–215
    - template class, 180–183
  - base classes
    - array of object pointers for, 213–215
    - for casting examples, 91–92, 94
    - common base class for
      - abstraction, 18
    - conversion into derived class
      - by compiler, 24
    - for converting numbers to words, 440–444
    - defined, 12
    - factory class, 163–167
    - implementing a common base class, 162
    - for interfaces, 354
    - for mailing-list application, 13–14
    - mix-in classes for limiting functionality, 168
    - object pools of, 162
    - pure virtual base classes for
      - interfaces, 354
    - pure virtual method in, 12
    - saving functionality in, 24
    - serialization interface, 354–359
    - simple template, 176–178
    - stepping back from problems and, 440
    - storing derived objects in an array, 213
    - using templates as, 179
    - virtual destructors for, 22, 25
    - for virtual inheritance, 117–119
    - for virtual methods, 20
    - virtual table for, 21, 22
  - BaseMailingListEntry class, 13–14
  - Base1 and Base2 classes for
    - casting, 91–92, 94
  - basic types
    - defined, 59
    - extending, 59–62
  - Blowfish encryption algorithm, 343
  - Borland’s C++ Builder, 2
  - Buffer class (example 1), 308–311
  - Buffer class (example 2)
    - BufferException class for, 362, 364
    - character arrays versus, 363–364
    - creating, 361–364
    - returned value, 364
    - static arrays versus, 361
    - testing, 364–365
  - buffer overflows
    - Buffer class for, 361
    - defined, 360
    - prevalence of, 360
    - reasons for not fixing, 361
    - security issues, 360, 361

BufferException class, 362, 364  
 business rules  
   defined, 30  
   reusability of code and, 30, 31  
   separating from code, 30–36

## C

C++ Builder (Borland), 2  
 C++ *Timesaving Techniques For Dummies* (Telles, Matthew)  
   companion Web site, 2  
   conventions, 2–3  
   focus on saving time, 1, 2  
   goal of, 1  
   icons in margins, 4  
   organization, 3–4  
   using, 1–3  
 C++ versus C  
   error handling and, 319  
   file-handling functions and, 425–426  
   name resolution and, 85–86  
   pointers and, 175  
   reusability and, 85  
   struct construct and, 74  
   structures and, 73–74  
 calculation, discrete pieces in  
   classes for, 159  
 call stack from debuggers, 375  
 case  
   c\_str method for converting, 349  
   implementing the transform  
     function to convert strings,  
     350–351  
   strup function in C for convert-  
     ing, 349  
   testing the string conversion,  
     351–353  
 casts  
   addressing compiler problems,  
     93–94  
   base classes for, 91–92, 94  
   casting away const-ness, 81  
   derived classes for, 92, 94  
   need for, 90–91

scoping member functions  
   versus, 93–95  
 temporary objects and, 408  
 test drivers, 93, 95  
 testing, 93, 94–95  
 using, 91–93  
 cDate class  
   described, 31  
   non-inline methods, 34–35  
   source-code listing, 32–34  
   testing, 35–36  
 ch01 through ch71 files. *See* com-  
   panion Web site for this book  
 chaining errors, 319–322  
 chaining return codes, 328–329  
 ChangeManagement class, 115  
 character pointers, new operator  
   and, 134–135  
 checked member variable, 328  
 class examples. *See also*  
   examples in this book  
   AgeProperty, 139–140  
   auto\_ptr, 303–306  
   Bar, 53, 54  
   Base1, for casting examples,  
     91–92, 94  
   Base2, for casting examples,  
     91–92, 94  
   BaseMailingListEntry, 13–14  
   Buffer (example 1), 308–311  
   Buffer (example 2), 361–365  
   cDate, 31–36  
   ChangeManagement, 115  
   class with methods containing  
     default values, 103–106  
   CommandLineMailingListEntry,  
     16–17  
   Complete, 109–115  
   Complex, 433–438  
   ConfigurationFile  
     (example 1), 26–27  
   ConfigurationFile  
     (example 2), 251  
   for converting numbers to  
     words, 440–444  
   Date, 149–161  
   DBCObject, 393–398  
   debugFlowTracer, 376–377

debugFlowTracerManager,  
   377–379  
 DelimitedFileParser, 237–238  
 DelimitedRow, 236–237  
 Delimiters, 235–236  
 Derived, for casting examples,  
   92, 94  
 Derived, for sizeof function  
   examples, 52, 54  
 DirectoryList, 367–368, 369  
 enumeration class, 71–72  
 ExceptionCatcher class,  
   314–315, 316–317  
 ExceptionClass, 313–314,  
   315–316, 317  
 factory class, 163–167  
 FileChunk, 284–286  
 FileChunkManager, 286–288  
 FileHandler, 103–106  
 FileMailingListEntry, 14–15  
 FileWrapper guardian class,  
   426–431  
 Fruit, 20–22  
 Full, 52, 53  
 HundredsRangeEntry, 443–444  
 Integer, 411–412  
 IntProperty, 137–140  
 Lock, 421–424  
 LockException, 421–424  
 Logger, 389–392  
 Match, 331–334  
 Matrix, 63–69  
 MultiPathFile, 367–371  
 for multiple inheritance,  
   116–117  
 MyAlloc, 298–302  
 MyBuffer, 301–302  
 MyReturnValue, 324–329  
 MyString, 122–126  
 MyStringArray, 196–199  
 NoPointer, 205–206  
 NumberToWords, 440–446  
 OnesRangeEntry, 441–442  
 for overloaded methods,  
   400–401  
 Parser, 450–451  
 ParserFile, 450, 451  
 for passing objects by  
   reference, 411–412

## class examples (continued)

- Point, 413–415
- PointerClass, 205–206
- Properties class for
  - ConfigurationFile class, 24–25
- Properties class for documenting data flow, 417–419
- Range, 60–62
- RangeEntry, 440–441
- reading delimited files, 235–238
- return code class, 324–329
- RetVal, 324–329
- SavePairs, 25–26, 28–29
- for scope illustration, 83
- SSNValidator, 142–148
- StringCoding, 7–11
- StringConvertToLowerCase, 350–353
- StringEntry, 266–268, 271
- StringReader, 273–277
- StringUtil, 252–255, 259
- StringWriter, 268–272
- TensRangeEntry, 443
- TestIntValue, 140–141
- ThousandsRangeEntry, 444, 445
- Tracker, 304–306
- Translator, 279–282
- URLCodec, 338–341
- using namespaces, 87–88
- vector (STL), 25, 192–195, 200–203, 209–212, 226–227
- for virtual files, 283–290
- for virtual inheritance, 117–119
- XMLElement, 241–242
- XMLSuperClass, 244, 245
- XMLWriter, 241–245
- XOREncryption, 346–348

classes. *See also* class examples; templates; *specific kinds*

- for arrays, with and without pointers, 204–208
- base, defined, 12
- breaking into discrete pieces, 159
- complete class, 109–115
- for configuration information, 24–29
- constants in, 79–80

- container classes in the STL, 179, 200
- for copyright information, 175
- customizing with polymorphism, 20
- customizing with virtual functions, 19–22
- for data validation, 142–148
- date class, 149–161
- for encoding strings, 7–11
- enumeration class, 71–72
- as extensions of structure component, 73
- external operators for, 61
- factory class, 162
- friend class, 167
- generic buffer class, 360–365
- guardian classes, 425–431
- for implementing properties, 137–141
- inherited, defined, 12
- initializing versus assigning data for, 413–415
- iterating over STL collection classes, 292–296
- memory safe buffer class, 307–311
- with methods containing default values, 103–106
- minimizing code for, 31
- mix-in classes, 24–26, 168–171
- multiple inheritance, 23
- name resolution problems in libraries, 86
- with overloaded methods, 400–401
- overloading operators, 120–127
- overriding functionality with virtual methods, 162–167
- passing objects by reference, 410–412
- placing reusable classes in namespaces, 89
- properties, 136–141
- return code class, 324–329
- saving functionality in, 24
- scope handled automatically for, 82–83
- sections of, 23

- simple template, 176–178
- storing literal string information in, 161
- string array class, 196–199
- struct construct as, 73
- structures versus, 76
- with templated method, 189–191
- templating a single function, 186–189
- templating a single method, 189–191
- testing, recommendations for, 161
- for throwing and logging exceptions, 312–317
- to-do list for improvements, 290
- for tracing flow, 376–380
- URLCodec class for, 338–342
- usefulness and number of classes included, 438
- virtual inheritance, 116–119
- v-table for virtual methods, 21, 22, 23
- XML structure compared to, 240–241

clone method, 109, 113

code. *See also* source-code listings

- minimizing for classes, 31
- reducing complexity of, 447–453
- separating rules and data from, 30–36

collections

- algorithms for, 350
- arrays, 291, 292–293
- auto\_ptrs in STL collections, 306
- avoiding assuming contiguous order for strings, 349
- benefits of, 291
- constant, 296
- generic STL iterator for, 291
- iterating over STL collection classes, 292–296
- iterator needed for, 291
- linked lists, 292, 293
- maps, 292, 293, 295
- non-contiguous elements in, 291

- overriding the allocator for, 297–302
- removing items using iterators, 294, 296
- reusability and, 291
- reverse iteration, 294, 295
- STL and, 291, 349
- streams, 294, 296
- swapping elements using iterators, 294
- testing iterators, 295–296
- Column class for spreadsheet creating, 217–218
  - methods in, 216, 218
  - stored data and, 221
  - virtual method in, 217, 218
- comma separated values (CSV) files, 234. *See also* delimited files
- command line input, handling, 12, 13
- command parser, hash table for, 279
- command processor class, 99
- command processor class test driver, 99
- CommandLineMailingListEntry class, 16–17
- companion Web site for this book
  - ch1\_1a.cpp file on, 11
  - ch01.cpp file on, 10
  - ch02.cpp file on, 13
  - ch02.cpp file on, 17
  - ch03.cpp file on, 20
  - ch03.cpp file on, 21
  - ch04.cpp file on, 24
  - ch04.cpp file on, 27
  - ch05.cpp file on, 32
  - ch6\_12.cpp file on, 333
  - ch06.cpp file on, 40
  - ch07.cpp file on, 42
  - ch07.cpp file on, 44
  - ch8\_2c.cpp file on, 398
  - ch08.cpp file on, 46
  - ch09.cpp file on, 49
  - ch10.cpp file on, 52
  - ch11\_1.cpp file on, 62
  - ch11.cpp file on, 60
  - ch12.cpp file on, 64, 68
  - ch13.cpp file on, 71
  - ch14.cpp file on, 74
  - ch15.cpp file on, 77
  - ch16.cpp file on, 83
  - ch17.cpp file on, 87, 88
  - ch18.cpp file on, 91, 93
  - ch19.cpp file on, 97, 99
  - ch20.cpp file on, 103
  - ch21.cpp file on, 110, 114
  - ch22.cpp file on, 118, 119
  - ch23.cpp file on, 122
  - ch24.cpp file on, 129, 133
  - ch25.cpp file on, 137, 140
  - ch26.cpp file on, 142, 146
  - ch27.cpp file on, 151, 152, 159
  - ch27.h file on, 150
  - ch28.cpp file on, 163, 166
  - ch29.cpp file on, 169
  - ch30.cpp file on, 176
  - ch31a.cpp file on, 184
  - ch31.cpp file on, 180, 183
  - ch32.cpp file on, 186, 189
  - ch33.cpp file on, 192
  - ch34.cpp file on, 196
  - ch35.cpp file on, 201
  - ch36.cpp file on, 204
  - ch37.cpp file on, 210
  - ch38.cpp file on, 213
  - ch39\_4.cpp file on, 221
  - ch39.cpp file on, 217
  - ch40.cpp file on, 226
  - ch41.cpp file on, 228, 232
  - ch42.cpp file on, 235, 239
  - ch43.cpp file on, 241, 243
  - ch44\_6.cpp file on, 246
  - ch45\_7.cpp file on, 261
  - ch45.cpp file on, 260
  - ch46\_1.cpp file on, 273
  - ch46a.cpp file on, 277
  - ch46.cpp file on, 266
  - ch47.cpp file on, 280, 281
  - ch48.cpp file on, 284, 289
  - ch49.cpp file on, 292
  - ch50.cpp file on, 300
  - ch50.h file on, 298
  - ch51.cpp file on, 304
  - ch52.cpp file on, 308
  - ch53.cpp file on, 313, 318, 320
  - ch54.cpp file on, 324
  - ch55.cpp file on, 331
  - ch56.cpp file on, 338, 340
  - ch57.cpp file on, 344, 345, 346, 347
  - ch58.cpp file on, 350, 351
  - ch59.cpp file on, 358
  - ch59.h file on, 355
  - ch60.cpp file on, 361, 364
  - ch61.cpp file on, 367, 369
  - ch62a.cpp file on, 380
  - ch62.cpp file on, 376
  - ch63a.cpp file on, 389, 390
  - ch63b.cpp file on, 393, 397
  - ch63.cpp file on, 388
  - ch64.cpp file on, 398, 401
  - ch65a.cpp file on, 411
  - ch65b.cpp file on, 413
  - ch65.cpp file on, 408
  - ch66.cpp file on, 417, 418
  - ch67.cpp file on, 421, 422
  - ch68.cpp file on, 426, 430
  - ch69.cpp file on, 433, 436
  - ch70.cpp file on, 440, 446
  - ch71.cpp file on, 450
  - ConfigurationFile.cpp file on, 251, 261
  - ConfigurationFile.h header file on, 251
  - osdefines.h header file on, 40
  - sizeof program on, 53
  - URL for, 2
- comparison operators, overriding, 328
- compiler errors and warnings addressing immediately, 91 debugging aided by eliminating, 91
  - #define versus const statement and, 46–47
  - indicating casting is needed, 93
  - Matrix class operators and, 67
  - multiple inheritance error, 117
  - for namespace problems, 89
- compilers. *See also* pre-processor
  - assert macro and optimized mode, 387, 389
  - base class/derived class conversion by, 24

compilers (*continued*)

- const keyword as indicator to, 78–79
- default constructor called by, 26
- #define versus const statement and, 45–47
- examples in this book and, 1, 3
- exception handling and, 322
- GNU C++ compiler, 2
- inheritance implemented by, 23–24
- inline functions and, 407, 408
- instantiation for templates by, 178
- strup function with strings and, 349
- template keyword and, 178
- typeface in this book for output, 2
- types for values passed to functions and, 90–91, 93–94
- warning level setting for, 47

## Complete class

- creating a template, 110–113
- dirty flag, 110, 114
- need for, 109
- output from, 114
- rules for, 109
- source-code listing, 110–113
- testing, 113–115

## Complex class

- defining, 433–434
- implementing, 434–435
- testing, 436–438
- utility functions, 436

## complex numbers

- challenges for expertise in, 432
- defined, 432
- implementing a class for, 433–436
- template for, 433
- testing the class, 436–438
- uses for, 432
- written form for, 432

## componentizing, 449–451

## configuration files

- basic functionality, 24
- class for storing information, 24–29

## configuration-file class

- creation, 251–259
  - “endian” concerns for, 250
  - finding all files, 367
  - as hallmark of professional programs, 251
  - header file for, 251
  - text format for, 250
  - typical entry for, 250
- ConfigurationFile class
- (example 1)
  - constructor issues for, 27–29
  - implementing, 24–26
  - Properties class for, 24–25
  - SavePairs class for, 25–26, 28–29
  - source code for, 26
  - testing, 27
- ConfigurationFile class
- (example 2)
  - defined in header file, 251
  - header file for, 251
  - read function, 256, 259
  - source code for, 251, 256
  - storage functions, 259
  - StringUtil utility class for, 252–255, 259
  - test file for, 260
  - testing, 260–261
- ConfigurationFile.cpp file, 251–259, 261
- ConfigurationFile.h header file, 251
- const iterators, 296
  - const keyword
    - casting away, 81
    - in classes, 79–80
    - with Copy constructor, 80
    - for differentiating methods, 81
    - as indication to compiler, 78–79
    - for methods and functions, 77
    - versatility of, 80
  - const statements
    - #define statement compared to, 45
    - defining constants, 77–78
    - implementing constant variables, 78–79

replacing #define values using, 77–78

as type-safe, 46

using instead of #define, 45–47

constants. *See also* const keyword; const statements

basic integer variables

versus, 59

casting away const-ness, 81

in classes, 79–80

collections, 296

const keyword for, 77, 81

#define versus const statement and, 46–47

defining, 77–78

for function with immutable argument, 78–79

implementing constant variables, 78–80

returning a const reference, 80

for SSN length and delimiter, 143, 146

testing the constant application, 80–81

uses for, 77

construct method, overriding, 299, 300

## constructors

array of object pointers and, 215

for Complete class, 109

for ConfigurationFile class, 26, 28–29

const keyword with Copy constructor, 80

copy constructor for auto\_ptr, 306

copy constructor for RetValue, 329

default called by compiler, 26

delayed construction, 27–29

error handling for, 27–28

exception types and copy constructor, 322

invoking, 26

for MyString class, 123

object pools and, 162

planning for disasters, 29

Point class, 414, 415

- print statements in, for
    - debugging, 208
  - required for Complete class, 111
  - scope and, 82
  - for structures, 76
  - templates and, 178
  - two pointers for same memory
    - block and, 28
  - type and destructor calls, 176
  - virtual inheritance and, 119
  - container classes (STL). *See also*
    - collections
    - algorithms with, 200, 349–350
    - creating, 210
    - creating arrays of objects
      - using, 209–212
    - overhead from using, 209–210
    - as templates, 179
    - transform function for modifying elements, 349–350
    - uses for, 200
  - container collections. *See* collections
  - control characters, string classes and, 348
  - conventions in this book, 2–3
  - conversion. *See also* translation
    - base class/derived class, by compiler, 24
    - of case for strings, 349–353
    - casts for, 81, 90–95
    - c\_str method for converting case, 349
    - hash tables for, 279
    - implementing operators for, 122
    - implementing the transform function to convert strings, 350–351
    - of numbers to words, 439–446
    - strup function for converting case, 349
    - testing the string case conversion, 351–353
    - of types with casts, 90–95
  - converting numbers to words
    - common set of things to look at, 439
    - components for, 439
    - creating the base classes, 440–444
    - HundredsRangeEntry class for, 443–444
    - need for, 439
    - NumberToWords class for, 445–446
    - OnesRangeEntry class for, 441–442
    - RangeEntry class for, 440–441
    - steps for, 439–440
    - TensRangeEntry class for, 443
    - testing the code, 446
    - ThousandsRangeEntry class for, 444, 445
  - copy constructor
    - for auto\_ptr, 306
    - const keyword with, 80
    - exception types and, 322
    - for RetValue class, 329
  - copying
    - an auto\_ptr, 306
    - strings, memory overwrites from, 307–308
  - copyright information class, 175
  - crashing programs
    - assert macro and, 43, 44
    - from buffer overflows, 360
    - C-style file-handling functions and, 425–426
    - eliminating the source of failures, 323
    - intentionally, avoiding, 43
  - credit card numbers, encryption for, 343
  - critical-section handlers in operating systems, 420
  - c\_str method, 349
  - CSV (comma separated values) files, 234. *See also* delimited files
  - cumbersomeness, avoiding, 438
  - customizing. *See also* templates
    - built-in functions, 102
    - classes with polymorphism, 20
    - classes with virtual functions, 19–22
  - memory allocation, 297–302
  - MessageBox function, 102
  - user-defined functions, 103–106
- ## D
- 
- data. *See also* input and output
    - documenting flow of, 416–419
    - encoding and decoding for the Web, 337–342
    - information-specific, classes for handling, 169
    - inheriting functionality and, 23–29
    - initializing versus assigning, 413–415
    - protecting from memory overwrites, 307–311
    - protecting with encapsulation, 7–11
    - separating from code, 30–36
    - undo functionality and, 419
  - data storage. *See also* storage
    - allocation
    - encapsulation benefits for, 11
    - hash tables for, 279
    - literal string information in classes, 161
    - for matrix in Matrix class, 65
    - in XML format, 241–245
  - data types. *See* types
  - Date class
    - algorithmic code, 153, 154, 159
    - basic functionality, 149
    - creating the class, 150–152
    - defining, 150–151
    - enhancements recommended for, 161
    - implementing date functionality, 152–159
    - initialization code, 152, 159
    - need for, 149
    - source file, 151–152
    - testing, 159–161
    - validation code, 153, 159
  - date.cpp file, 34

- dates. *See also* Date class
  - cDate class for, 31–36
  - checks scattered throughout a program, 30
  - Date class for, 149–161
  - hard-coded, 30
  - IsLeapYear method, 35
  - IsValidDate method, 35
  - leap year computations, 30–31
  - limitations of standard routines for, 149
  - portability and, 31
  - rejecting confusing formats, 453
  - reusability of code and, 30, 31
- DBC methodology. *See* Design by Contract methodology
- DBCObject class
  - implementing, 393–397
  - testing, 397–398
- deallocate method
  - MyBuffer class call to, 302
  - overriding, 300
- “debug mode” for assert macro, 388
- debugFlowTracer class
  - creating, 376–377
  - debugFlowTracerManager class for, 377–379
  - testing, 379–380
- debugFlowTracer objects, 385, 386
- debugFlowTracerManager class
  - creating, 377–379
  - Instance method, 379
  - purpose of, 379
- debugging. *See also* compiler errors and warnings; testing; tracing
  - assert macro definition and, 43
  - assert macro for, 42, 44, 387–389
  - avoiding versus fixing problems and, 303
  - building tracing into applications, 375–380
  - call stack from debuggers and, 375
  - categories of techniques for, 387
  - chaining errors and, 319
  - challenges for, 375
  - checking size of values during, 54
  - choosing techniques for, 387
  - creating macros and classes for, 387–398
  - date code and, 31
  - debugFlowTracer class for, 376–377
  - debugFlowTracerManager class for, 377–379
  - Design by Contract for, 392–398
  - documenting data flow and, 419
  - eliminating compiler warnings and, 91
  - encapsulation benefits for, 11
  - failing to handle errors and, 323
  - generalization of code and, 452
  - inserting tracing into an existing file, 380–386
  - logging data for, 389–392
  - logging errors and, 312
  - macro side effects and, 48, 49
  - no “right” or “wrong” way for, 387
  - overloaded methods, 399–403
  - overloaded operators and, 121
  - physical errors versus logical errors and, 31
  - print statements in constructors and destructor for, 208
  - separating business rules from code and, 31
  - specialization and, 452, 453
  - system flow and, 375
  - testing the flow trace system, 379–380
  - validation and time saved in, 142
- decode method, 339–340, 341
- decoding. *See also* encoding; encryption
  - decode method, 339–340, 341
  - library methods lacking for, 337
  - reusable class helpful for, 337
  - URLConnection class for, 338–342
  - for the Web, 337–342
  - when exchanging data with Web-based applications, 342
- defaults
  - arguments for methods and functions, defining, 101–106
  - class with methods containing default values, 103–106
  - constructor called by compiler, 26
- #define statements
  - const statement compared to, 45
  - constants for directly replacing values, 77–78
  - using const instead of, 45–47
- delayed construction, 27–29
- delete operator
  - with arrays, 204
  - calling correct operator for delete, 135
  - handler for, 129–131
  - matching up with invocation method, 84, 204
  - memory allocation problems and, 128–129
  - output from memory tracking program, 133–134
  - overloaded handler for, 131–132
  - overloading to track memory allocation, 129–132
  - rules for handler implementation, 129
  - uses for, 128
- delimited files
  - assumptions for examples, 234
  - defined, 234
  - generic method for reading, 234–238
  - output from reading, 239
  - testing the code for reading, 238–239
- DelimitedFileParser class, 237–238
- DelimitedRow class, 236–237
- Delimiters class, 235–236
- Derived class
  - array of object pointers for, 213–215
  - for casting examples, 92, 94
  - for sizeof function examples, 52, 54

- derived classes
    - array of object pointers for, 213–215
    - for casting examples, 92, 94
    - conversion into base class by compiler, 24
    - factory pattern and, 162
    - for interfaces, 355
    - for `sizeof` function examples, 52, 54
    - storing derived objects in an array, 213
    - test drivers for casting, 93, 95
    - virtual destructors for, 22, 25
    - virtual methods and, 162
    - v-table for, 23
  - derived structures, 75, 76
  - Design by Contract (DBC)
    - methodology
      - creation by Eiffel programming language, 392
    - documenting assumptions made by code, 393
    - implementing, 393–397
    - parts of code according to, 392–393
    - post-conditions for code, 393
    - preconditions for code, 392, 397
    - purpose of, 393
    - test program for, 397–398
    - validity checks for code, 392–393
  - destroy method, overriding, 299, 300
  - destructors. *See also* virtual destructors
    - array of object pointers and, 215
    - arrays of objects and, 211–212
    - clearing pointers for exception object, 322
    - for `Complete` class, 109
    - deleting array elements and, 204
    - object pools and, 162
    - planning for disasters, 29
    - `print` statements in, for debugging, 208
    - required for `Complete` class, 111
    - scope and, 82
    - storing derived objects in an array and, 213
    - templates and, 178
    - two pointers for same memory block and, 28
    - type and, 176
    - virtual, 22, 25
  - diagnostics, `dump` method and, 278
  - dictionary, hash table for, 279
  - `DirectoryList` class
    - described, 369
    - path delimiter for, 369
    - source-code listing, 367–368
  - dirty flag
    - `ChangeManagement` class, 115
    - `Complete` class, 110, 114
    - uses for, 115
  - divide-by-zero error, 317–319
  - document class, 87–88
  - document concept, 86
  - documentation
    - for assumptions made by code, 393
    - constants as self-documentation, 77
    - of data flow, 416–419
    - STL, 199
  - documenting data flow
    - importance of, 419
    - learning how code operates, 416–418
    - need for, 416
    - `Properties` class for, 417–418
    - testing the `Properties` class, 418–419
  - `do_xor` method, 346, 347
  - `dump` method, 278
- 
- ## E
- 
- EBCDIC systems, `ROT13Encryption` class and, 345
  - Eiffel programming language, DBC created by, 392
  - embedded processors, `new` in place operator and, 135
  - encapsulation
    - abstraction using, 12
    - for algorithms, 7–10, 36
    - benefits of, 7–8, 10, 11
    - creating and implementing and encapsulated class, 7–10
    - defined, 7
    - encapsulated code as black box, 11
    - for encryption method, 7–11
    - information-specific data and, 169
    - for `Matrix` class array row, 64
    - protecting data with, 7–11
    - reusability and, 30, 31
    - for separating rules and data from code, 30–36
    - in `Spreadsheet` class, 216
    - `struct` construct as beginning of, 73
    - type validation and, 142
    - updates to an encapsulated class, 10–11
  - `Encode` method of `StringCoding` class, 9, 10–11
  - `encode` method of `URLConnection` class, 339, 340, 341
  - encoding. *See also* decoding; encryption
    - defined, 337
    - `Encode` method of `StringCoding` class, 9, 10–11
    - `encode` method of `URLConnection` class, 339, 340
    - library methods lacking for, 337
    - required for special characters on the Web, 337
    - reusable class helpful for, 337
    - `URLConnection` class for, 338–342
    - for the Web, 337–342
    - when exchanging data with Web-based applications, 342
  - encryption. *See also* encoding
    - Blowfish encryption algorithm, 343
    - defined, 343
    - encapsulated method for, 7–10
    - encrypting and decrypting strings, 343–348

- encryption (*continued*)
  - information requiring, 343
  - Rot13 encryption algorithm, 343, 344–346
  - RSS encryption algorithm, 343
  - for text string files, 273
  - updating encapsulated method, 10–11
  - XOR encryption algorithm, 343, 344–346
- “endian” concerns for configuration files, 250
- enhancing
  - Date class, 161
  - keeping a to-do list for classes, 290
  - manager class, 167
  - MultiPathFile class, 371
  - virtual file class, 290
- enumerations
  - basic form, 70
  - defined, 70
  - implementing a class for, 71–72
  - for readability, 70, 71
  - as syntactical sugar, 70
  - testing the class, 72
- error handling. *See also* error messages; exception handling
  - for `assert` statements, 44
  - in C++ versus C, 319
  - chaining errors, 319–322
  - for construction, 27–28
  - enforcing return codes, 323–329
  - importance of, 323
  - inheritance from base class, 15
  - reusability and, 31
  - virtual methods for, 19
- error messages. *See also* compiler errors and warnings; error handling; exception handling; return codes or status codes
  - `assert` macro for, 42, 43
  - descriptive and informative, 265, 329
  - for `MessageBox` function, 102
  - exceptions or logging errors versus, 31
- internationalization and, 265, 266
  - security issues, 273
- `MessageBox` function, 102
- `errors.log` file, 317
- examples in this book. *See also* class examples; function examples; method examples
  - compilers and, 1, 3
  - entering code by hand, 3
  - operating systems and, 1
  - typeface conventions, 2–3
  - using code from, 1
- exception handling. *See also* error handling; error messages; exceptions
  - aborting the application versus, 43
  - application development and, 319
  - `BufferException` class for, 362, 364
  - caveats for, 322
  - chaining errors, 319–322
  - clearing all pointers in the destructor, 322
  - dealing with un-handled exceptions, 317–319
  - defined, 312
  - memory leaks from, 322
  - performance and, 322
  - restructuring, 452
  - re-throwing exceptions, 319–322
  - in `Row` class for spreadsheet, 218, 219
  - throwing and logging exceptions, 312–317
- `ExceptionCatcher` class
  - output from, 317
  - passing exceptions to a higher level, 320–322
  - purpose of, 317
  - source-code listing, 314–315, 316
- `ExceptionClass` class
  - dealing with un-handled exceptions, 318–319
  - output from, 317
- passing exceptions to a higher level, 320–322
  - purpose of, 317
  - source-code listing, 313–314, 315–316
- exceptions. *See also* exception handling
  - custom forms versus basic types, 364
  - defined, 312
  - error messages versus, 31
  - passing to a higher level, 319–322
  - re-throwing, 319–322
  - throwing and logging, 312–317
  - unhandled, dealing with, 317–319
- exiting programs abnormally, avoiding, 43
- expertise, challenges of seeking, 432
- eXtended Markup Language. *See* XML
- extending
  - assignment operators and extensibility, 61
  - basic types, 59–62
  - classes, pure virtual methods and, 12
  - functionality using abstraction, 12–18
  - `int` type by `Range` class, 60–62
  - `IntProperty` class, 139–140
  - suitability of extension for template classes, 183
  - template class, 179–185
- external operators for classes, 61

---

## F

---

- factory class
  - creating, 163–166
  - defined, 162
  - derived classes, 162
  - enhancing the manager class, 167
  - memory dumps and, 166

- methods reporting object state for, 166
  - Report method, 165–166, 167
  - testing, 166–167
- factory pattern, 162
- fclose function, problems from, 426, 431
- file processing. *See* processing files
- FileChunk class
  - data management by, 288
  - source-code listing, 284–286
  - testing, 289–290
- FileChunkManager class
  - array management by, 288
  - source-code listing, 286–288
  - testing, 289–290
- file-guardian class. *See* FileWrapper guardian class
- FileHandler class
  - creating, 103–105
  - fixing, 106
  - open methods, 104–105, 106
  - testing, 105
  - writing files, 105
- FileMailingListEntry class, 14–15
- files. *See also* processing files; reading files
  - delimited, reading, 234–239
  - handling input from, 12–13
  - opening using multiple paths, 366–371
  - virtual files, 283–290
- files, source-code. *See* companion Web site for this book
- FileWrapper guardian class
  - creating, 426–430
  - open function, 428–429, 430
  - puts function, 429, 430
  - testing, 430–431
- First method, 13–14, 15
- fixed-length input, checking, 452–453
- fixed-size records, 234. *See also* delimited files
- floating-point values. *See* complex numbers; numbers
- flow
  - classes for tracing system flow, 376–380
  - documenting data flow, 416–419
- fopen function, 425–426, 430
- fprintf function
  - problems from, 426
  - stream components versus, 225
- free function, 132
- friend class, 167
- Fruit class, 20–22
- Full class, 52, 53
- function examples
  - complex variable utility functions, 435–436
  - ErrorBox, 102
  - fopen, 425–426, 430
  - Load, 270, 271
  - Matrix class manipulation functions, 67–68
  - open, 428–429, 430
  - ProcessEntries, 17, 18
  - puts, 429, 430
  - read, 256, 259
  - set\_terminate, 318–319
  - storage functions for
    - ConfigurationFile class, 259
    - strip\_leading, 247, 248–249
    - strip\_trailing, 247, 248–249
    - strup function (C language), 349
    - term\_func, 318–319
    - transform, for converting strings, 349–353
- function pointers (C language), 96–97
- function templates
  - automatic generation of, 189
  - defined, 186
  - implementing, 186–189
  - types and, 189
- functionality
  - exercising all when testing classes, 161
  - extending using abstraction, 12–18
  - inheriting data and, 23–29
- mix-in classes for limiting, 168
- saving in base classes, 24
- functions. *See also* function examples; member functions; methods; *specific kinds*
  - auto\_ptr class with, 303–306
  - chaining errors from, 319–322
  - complex variable utility functions, 435–436
  - const keyword for, 77
  - for copying strings, memory overwrites from, 307–308
  - customizing built-in functions, 102
  - customizing user-defined functions, 103–106
  - defining default arguments for, 101–106
  - enforcing return codes, 323–329
  - enumerations for integer value input to, 72
  - free, 132
  - with immutable argument, 78–79
  - inline, 407–408
  - instantiation of variables and, 412–413
  - macro side effects and calls to, 49, 50
  - macros versus, 49–51
  - malloc, 132
  - memory safe buffer class for, 307–311
  - MessageBox, 101–102
  - passing objects by reference, 411–412
  - pure virtual functions, 12–15, 19
  - sizeof function, 52–55
  - STL algorithm functions, 200
  - templates, 186–189
  - values passed to, types and, 90–91, 93–94
  - virtual functions, 19–22, 23
  - for white space removal, 247, 248–249

## G

- generalization of code, debugging and, 452
- generic method for reading delimited files, 234–238
- generic pointers, 175–176
- get methods
  - for `Complete` class, 109, 112–113
  - for `IntProperty` class, 138, 139–140
  - for properties, 136
- `getClassname` virtual method, 357
- `getElement` virtual method, 357
- global scope, 83
- GNU C++ compiler, 2. *See also* compilers
- GNU organization Web site, 2
- greater-than sign (>), encoding required for Web use, 337
- guardian classes
  - creating the file-guardian class, 426–430
  - defined, 425
  - testing the file-guardian class, 430–431
  - uses for, 425, 426
  - wrapping potentially unsafe code in, 426

## H

- hackers, strings and, 273
- hash tables
  - defined, 279
  - in the STL, 279, 281
  - `Translator` class using, 279–282
  - uses for, 279
- header files
  - for `Complex` class definition, 433, 435
  - complex number template in the STL, 433
  - for configuration-file class, 251
  - for `MyAlloc` class, 298–300
  - `osdefines.h` header file, 39–40

- standard C++ files, problems with, 39
- template class implementation and, 178, 179
- test program, 40–41
- verifying that OS must be defined for, 41
- heap, creating arrays using, 209–212
- hiding. *See also* encapsulation algorithms from developers, 7–8
  - implementation from users, 11
- `HundredsRangeEntry` class, 443–444

## I

- icons in margins of this book, 4
- identifying program-specific input, 452
- imaginary numbers, 432. *See also* complex numbers
- improving. *See* enhancing
- `#include` statements for header files, 39, 40
- inheritance
  - compiler implementation of, 23–24
  - of data and functionality, 23–29
  - defined, 23
  - of error handling from base class, 15
  - inherited classes defined, 12
  - interfaces and, 354
  - levels of, 23–24
  - mix-in classes and, 168
  - multiple inheritance, 23, 116–117, 176
  - of storage allocation from base class, 15
  - virtual, 117–119
- inherited classes, 12
- initialization
  - assignment versus, 413–415
  - `Date` class code for, 152, 159
  - discrete pieces in classes for, 159
  - of values for classes or structures, 76
- inline functions
  - defined, 407
  - for optimizing code, 407–408
  - overhead from, 407
  - rules for, 408
  - speed and, 407
- input and output
  - creating a configuration file, 250–261
  - of data formats, extracting into separate classes, 234
  - fixed-length, checking, 452–453
  - function input enumerations for integer values, 72
  - identifying program-specific input, 452
  - input text file for internationalization, 272
  - reading delimited files, 234–239
  - reading in and processing files, 228–233
  - reading internationalization files, 272–277
  - removing white space from input, 246–249
  - using `stream` components to format data, 225–227
  - writing objects as XML, 240–245
- `input.cfg` file, 260
- inserting tracing into an existing file
  - caveats for insertion programs, 380
  - `debugFlowTracer` objects, 385, 386
  - functionality, 385
  - need for, 380
  - source-code listing, 381–384
  - `temp.cpp.tmp` file as output, 385–386
  - temporary program to illustrate, 385–386
  - testing the program, 385–386
- `Instance` method, 379

instantiation  
 of templates, header files  
 and, 178  
 templates versus macros  
 and, 178  
 testing for templated classes,  
 182–183  
 of variables, optimizing,  
 412–413

`int` type, Range class extending,  
 60–62

`Integer` class, 411–412

interfaces  
 base classes for, 354  
 defined, 354  
 serialization interface,  
 355–359  
 steps for implementing,  
 354–355  
 uses for, 354

internationalization  
 application development and,  
 265–266  
 building language files for,  
 266–272  
 creating input text file for, 272  
 defined, 265  
 need for, 265, 266  
 reading the international file,  
 272–277

`StringEntry` class for,  
 266–268, 271

`StringReader` class for, 273–277

`StringWriter` class for, 268–272

testing the string reader,  
 277–278

threefold process of, 265–266

Internet, the. *See also* companion  
 Web site for this book;  
`URLConnection` class  
 encoding and decoding for the  
 Web, 337–342  
 GNU C++ compiler Web site, 2  
 planning for Web-enabled  
 code, 337  
 rules for URLs, 337

`IntProperty` class  
 extending, 139–140  
 get methods, 138, 139–140  
 implementing, 137–139  
 set methods, 138, 139–140  
 source-code listings, 137–139  
 testing, 140–141  
 using in another class, 139–140

`IsLeapYear` method, 35

`IsValidDate` method, 35

iterators  
 for arrays, 291, 292–293  
 caveats for, 296  
`const` versus `non-const`, 296  
 defined, 291  
 generic STL iterator, 291  
 iterating over STL collection  
 classes, 292–296  
 for linked lists, 292, 293  
 for maps, 292, 293, 295  
 need for, 291  
 output from test, 295–296  
 power of, 296  
 removing items using, 294, 296  
 reverse iteration, 294, 295  
 for streams, 294, 296  
 swapping elements using, 294

---

## J

jump tables, 23

---

## K

KISS (Keep It Simple, Stupid)  
 principle, 31, 447

---

## L

language files for  
 internationalization, 266–272

leading spaces. *See* white space

leap year computations, 30–31, 35

less-than sign (<), encoding  
 required for Web use, 337

libraries. *See also* STL (Standard  
 Template Library)  
 avoiding clumsiness, 438  
 encoding and decoding meth-  
 ods lacking in, 337  
 memory leaks in low-level  
 libraries, 131  
 name resolution problems in C,  
 85–86  
 name resolution problems with  
 classes, 86  
 of utility classes, 353

lifetime. *See* scope

linked lists, 292, 293

linking  
 method functions and, 189  
 to STL, `vector` class and, 192

`Load` function, 270, 271

loading  
 inline functions and, 407  
 pre-loading virtual file  
 chunks, 290  
 virtual files and speed for, 283

local scope, 83

`localtime` function, 149

`Lock` class  
 creating, 421–422  
`setLock` and `unlock`  
 methods, 422  
 testing, 422–424

`LockException` class  
 creating, 421–422  
 testing, 422–424

locking a program  
 creating the locking mecha-  
 nism, 421–422  
 custom versus operating sys-  
 tem implementation, 420  
 defined, 420  
 need for, 420  
 portability and, 420  
 testing the locking mechanism,  
 422–424  
 ways of implementing, 420

`Log` method, 170

Logger class  
 implementing, 389–390  
 output from, 390, 392  
 testing, 389–390  
 turning logging on and off, 390  
 uses for, 390

logging  
 actions, by mix-in classes,  
 170, 171  
 data, for debugging, 389–392  
 defined, 389  
 error messages versus logging  
 errors, 31  
 errors, debugging and, 312  
 exceptions, 312–317  
 implementing a logging class,  
 389–390  
 for overloaded methods,  
 401–403  
 testing the logging class,  
 390–392  
 turning on and off, 389, 390

logical errors, 31

log.txt and log2.txt files, 105

loop scope, 83

lowercase. *See* case

## M

macros  
 assert macro, 42–44, 387–389  
 avoiding, reasons for, 48–49  
 code size increased by, 48  
 debugging functionality lack-  
 ing for side effects, 48, 49  
 determining errors when  
 using, 50–51  
 function calls and side effects  
 of, 49, 50  
 functions versus, 49–51  
 string macros, avoiding prob-  
 lems with, 49–51  
 as syntactical sugar, 51  
 templates as giant macros, 178  
 templates versus, 178  
 using appropriately, 51

mailing-list application  
 base class, 13–14  
 BaseMailingListEntry class,  
 13–14  
 CommandLineMailingListEntry  
 class, 16–17  
 FileMailingListEntry class,  
 14–15  
 handling input from a file, 12–13  
 handling input from the  
 command line, 12, 13  
 mailing-list entries, 12  
 in operation, 18  
 overview, 12–13  
 ProcessEntries function, 17, 18  
 steps for creating, 13–15  
 testing, 17–18

maintaining code  
 documenting data flow and, 419  
 failing to handle errors and, 323  
 hiding algorithms and, 8  
 separating rules and data from  
 code and, 30

malloc function, 132

Manager class, 176–178

managers. *See also* factory class  
 enhancing the manager  
 class, 167  
 friend class for, 167  
 for virtual files, 283–290

map class (STL), 281

maps, iterating over, 292, 293, 295

Match class  
 creating, 331–333  
 matches method, 334  
 purpose of, 333  
 testing, 333–334  
 wildcards, 333

matches method, 334

Matrix class  
 creating, 64–65  
 manipulation functions, 67–68  
 multidimensional array classes  
 modeled on, 65  
 operators, 65–66, 67–68  
 output from, 69  
 overriding operators and, 63

scalar multiplication, 66–68  
 source-code listing, 64–65  
 testing, 68–69

member functions  
 pointers to, 96–100  
 scoping versus casting, 93–95

member variables  
 checked, 328  
 using templates as, 179

member-function pointers  
 defined, 96–97  
 function pointers (C language)  
 versus, 96–97  
 implementing, 97–98  
 power of, 96  
 testing member pointer code,  
 99–100  
 updating code with, 99

memcpy function, memory over-  
 writes from, 307–308

memory  
 STL use and, 192, 195, 196, 199  
 virtual files for conserving, 283

memory allocation. *See also*  
 memory leaks; memory  
 tracking program  
 for arrays of object  
 pointers, 214  
 arrays of objects and, 209  
 for arrays, with and without  
 pointers, 204–208  
 avoiding assuming contiguous  
 order for strings, 349  
 avoiding overwrites, 307–311  
 customizing, 297–302  
 delete operator problems for,  
 128–129  
 deleting array elements  
 and, 204  
 embedded processors and, 135  
 free function for de-allocation,  
 132  
 guardian classes and, 425  
 malloc function for, 132  
 new operator problems for,  
 128–129

- overloading `new` and `delete` operators to track, 129–135
- `set_terminate` function and, 318
- testing production code with memory-leak tool, 133
- two pointers for same block, 28
- using `auto_ptr` class, 303–306
- memory allocators
  - creating a custom allocator, 298–300
  - methods overridden by, 299–300
  - for `new` and `delete` operators, 129–135
  - output from test driver, 302
  - overriding for STL collections, 297–302
  - purpose of, 297
  - STL complications for, 297
  - test driver for custom allocator, 301–302
- memory leaks. *See also* memory allocation
  - `auto_ptr` class for avoiding, 303–306
  - avoiding versus fixing problems, 303
  - deleting all manager class objects at shutdown and, 167
  - duration of, 82
  - exception handling and, 322
  - from failure to delete arrays properly, 204, 208, 209, 212
  - in low-level libraries, 131
  - from macro side effects, 51
  - memory allocation by functions and, 129
  - from not matching deletion method with invocation method, 84, 204, 212
  - object allocation by manager class and, 167
  - from pointers, 212, 303
  - set methods for `NULL` pointers and, 109
  - STL container class implementation and, 210
  - storing derived objects in an array and, 213
  - storing pointers to objects in arrays and, 212
  - testing production code with memory-leak tool, 133
  - tracking memory allocation to avoid, 128, 131, 134
- memory overwrites
  - Buffer class for avoiding (example 1), 308–311
  - Buffer class for avoiding (example 2), 361–365
  - buffer overflows, 360
  - from copying strings, 307–308
  - defined, 307
  - difficulties tracking, 307
  - memory safe buffer class, 307–311
  - problems from, 307
- memory safe buffer class, 307–311
- memory tracking program
  - allocation report, 131
  - `new` and `delete` handlers, 129–131
  - output from, 133–134
  - overloaded `new` and `delete` handlers, 131–132
  - rules for `new` and `delete` handlers, 129
  - testing, 133–135
- `MessageBox` function, 101–102
- method examples. *See also* examples in this book
  - accessor methods for `MyString` class, 123
  - `allocate`, overriding, 299, 300
  - complex variable utility methods, 435–436
  - `construct`, overriding, 299, 300
  - `c_str`, for converting strings, 349
  - `deallocate`, overriding, 300
  - `decode`, 339–340, 341
  - `destroy`, overriding, 299, 300
  - `do_xor`, 346, 347
  - `Encode` (`StringCoding` class), 9, 10–11
  - `encode` (`URLConnection` class), 339, 340, 341
  - `First`, 13–14, 15
  - `getClassName`, 357
  - `getElements`, 357
  - `Instance`, 379
  - `IsLeapYear`, 35
  - `IsValidDate`, 35
  - `Log`, 170
  - `matches`, 334
  - `Multiply` method template, 189–191
  - `myVector`, 301, 302
  - `Next`, 13–14, 15
  - non-inline methods for `cDate` class, 34–35
  - `open` (`FileHandler` class), 104–105, 106
  - `OpenFile`, 28
  - operator, 351
  - `operator=`, 402, 403
  - `ProcessEntries`, 17, 18
  - `puts`, 429, 430
  - `Report`, 165–166, 167
  - `Save`, 270, 272
  - `setLock`, 422
  - `setX`, 402, 403
  - `undo`, 418, 419
  - `unlock`, 422
  - `write` member method, 356–357
  - writing files, 105
- method templates
  - creating, 189–190
  - defined, 186
  - testing, 190–191
  - uses for, 189
- methods. *See also* functions; member functions; method examples; *specific kinds*
  - `allocate`, overriding, 299, 300
  - class with methods containing default values, 103–106
  - complex variable utility methods, 435–436

## methods (continued)

- const keyword for, 77
- construct, overriding, 299, 300
- deallocate, overriding, 300
- debugging overloaded methods, 399–403
- defining default arguments for, 101–106
- destroy, overriding, 299, 300
- differentiating with const keyword, 81
- encapsulating, for encrypting strings, 7–11
- enforcing return codes, 323–329
- inline, 407–408
- minimizing in classes, 31
- in mix-in classes, controlling availability, 168–169
- MyString class accessor methods, 123
- non-static, working with, 104
- for opening a file using multiple paths, 366–371
- overloaded, defined, 399
- overridden by memory allocator, 299–300
- pure virtual methods, 12–15, 19
- for reading delimited files, generic, 234–238
- reporting object state for factory class, 166
- required for Complete class, 109
- scoping member functions versus casting, 93–95
- self-cloning, 167
- signatures for, 399
- static, jump tables for, 23
- templates, 186, 189–191
- virtual methods, 19–22, 23
- for writing files, 105
- min macro side effects, 48–49
- mix-in classes
  - compiling, 170
  - controlling available methods, 168–169
  - creating a single functional class from, 24–26
  - implementing, 169–170
  - inheritance and, 168
  - logging by, 170, 171
  - overview, 168–169
  - testing, 170–171
  - using, 168–171
- monospaced font in this book, 2–3
- MultiPathFile class
  - creating, 367–369
  - described, 369
  - improving, 371
  - path delimiter for, 369
  - responsibilities of, 366
  - saving and reading paths, 371
  - testing, 369–371
- multiple inheritance. *See also* virtual inheritance
  - compiler errors for, 117
  - defined, 23
  - deriving all objects from common base class and, 176
  - example, 116–117
  - multiple operating systems, handling, 39–41
  - multiple paths, opening a file using, 366–371
- Multiply method template
  - creating, 189–190
  - testing, 190–191
- MyAlloc class
  - creating, 298–300
  - methods overridden by, 299–300
  - test driver, 301–302
- MyBuffer class, 301–302
- MyClass for overloaded methods
  - adding logging, 401–402
  - initial implementation, 399–401
  - operator= method, 402, 403
  - output from, 401, 402–403
  - setX method, 402, 403
- MyClass test class for serialization interface, 358–359
- MyClass.xml file, 359

- my.eng file, 272
- my.eng.idx file, 272
- my\_min template function, 186–189
- MyReturnValue class
  - output from, 328
  - source-code listing, 326–327
- MyString class
  - accessor methods, 123
  - as complete class, 123
  - constructors, 123
  - implementing, 122–123
  - operators, 124–125
  - testing, 125–127
- MyStringArray class
  - creating, 196–198
  - memory and speed and, 199
  - output from, 198
- myVector method, 301, 302

## N

- name resolution. *See also* namespaces
  - C++ versus C and, 85–86
  - namespaces and, 86
  - problems for classes in libraries, 86
- namespaces
  - basic format for defining, 86
  - class name collision avoided by, 86
  - creating a namespace application, 86–88
  - document class, 87–88
  - placing reusable classes in, 89
  - testing the application, 88–89
  - using in an application, 88
  - using namespace statement for, 87
- new in place operator, 135
- new operator
  - for array allocation, 211
  - array operator, 134–135
  - calling correct delete operator for, 135
  - character pointers and, 134–135

handler for, 129–131  
 memory allocation problems  
   and, 128–129  
 new in place operator  
   versus, 135  
 output from memory tracking  
   program, 133–134  
 overloaded handler for,  
   131–132  
 overloading to track memory  
   allocation, 129–132  
 rules for handler  
   implementation, 129  
   uses for, 128  
 Next method, 13–14, 15  
 non-class template arguments,  
   184–185  
 non-const iterators, 296  
 non-static methods, 104  
 NoPointer class  
   creating, 204–206  
   output from, 207–208  
 nulls  
   memory leaks and set meth-  
   ods for NULL pointers, 109  
   string classes and, 348  
 numbers  
   complex, working with,  
   432–438  
   converting to words, 438–446  
   encrypting credit card  
   numbers, 343  
   enumerations, 70–72  
   Social Security Number  
   validation, 142–148  
 NumberToWords class  
   base classes for, 440–444  
   implementing, 445–446  
   testing, 446

## O

Object base class  
   for factory class, 163–166  
   Report method, 165–166, 167  
   testing the factory, 166–167

object pools  
   for common base class, 162  
   new in place operator and, 135  
 object-array allocation  
   creating an array of objects,  
   210–211  
   output from the array alloca-  
   tion program, 211–212  
   ways of creating arrays of  
   objects, 209–210  
 objects  
   arrays of object pointers,  
   213–215  
   arrays of objects, 209–212  
   deriving all from common base  
   class, 176  
   methods reporting state of, 166  
   passing by reference, 410–412  
   passing by value, 408  
   scope handled automatically  
   for, 82–83  
   temporary, avoiding, 408–410  
   XMLElement objects, 243  
 OnesRangeEntry class, 441–442  
 open function, 428–429, 430  
 open methods (FileHandler  
   class), 104–105, 106  
 OpenAndReadFileNew method,  
   231, 232  
 OpenAndReadFileOld method,  
   230, 232  
 OpenFile method, 28  
 opening a file using multiple paths  
   configuration files and, 367  
   DirectoryList class for,  
   367–368, 369  
   improving the class for, 371  
   MultiPathFile class for,  
   367–371  
   operating systems and, 366  
   path delimiter for, 369  
   responsibilities of utility class  
   for, 366  
 operating systems  
   critical-section handlers, 420  
   examples in this book and, 1  
   locking by, 420  
   multiple, handling, 39–41

  opening files and, 366  
   32-bit, 54  
 operator keyword, 122  
 operator method, 351  
 operator= method, 402, 403  
 operators  
   assignment operators and  
   extensibility, 61  
   complex variable utility  
   methods, 435–436  
   conversion operators, 122  
   enumerations and, 70  
   external operators for  
   classes, 61  
   Matrix class, 65–66, 67–68  
   overloaded, 120–127  
   overriding, arrays and, 63  
   polymorphism, 19  
   Range class, 61  
   for retrieving line of file into  
   string object, 231, 232  
   strings and xor operation, 348  
 optimization. *See also* overhead;  
   speed  
   in application development  
   process, 407  
   asserts not defined in opti-  
   mized mode, 387, 389  
   avoiding temporary objects,  
   408–410  
   initialization versus assign-  
   ment and, 413–415  
   inline functions for, 407–408  
   passing objects by reference,  
   410–412  
   post-development, 407  
   postponing variable  
   declarations, 412–413  
   testing asserts in optimized  
   environment, 44  
   of variable instantiation,  
   412–413  
 osdefines.h header file  
   creating, 39–40  
   source-code listing, 40  
   test program, 40–41  
   verifying that OS must be  
   defined for, 41

output. *See* input and output overhead. *See also* optimization; speed

- exception handling and, 322
- from inline functions, 407
- from STL use, 192, 195, 196, 199, 209
- structures for minimizing, 74
- for templates, 179
- from temporary objects, 408–410
- usefulness of classes and, 438
- from `vector` class, 192, 195
- for virtual versus non-virtual methods, 21

overloaded methods

- class with, 400–401
- debugging, 399–403
- defined, 399
- logging, 401–403
- signatures for, 399

overloaded operators

- associated operators and, 121–122
- benefits of, 120–121, 127
- conversion operators, 122
- creating only when necessary, 121
- debugging complicated by, 121
- for memory allocation tracking, 129–135
- `MyString` class for, 122–127
- `new` and `delete` operators, 129–135
- power of, 127
- rules for creating, 121–122
- side effects, avoiding, 121
- standard usage and, 121
- using, 122–125
- for `vector` objects, 227

overriding classes

- allocator for collections, 297–302
- pure virtual methods and, 14
- virtual methods and, 19, 21, 162–167

overriding methods for memory allocator, 299–300

overriding operators

- arrays and, 63
- comparison operators, 328

overwriting memory. *See* memory overwrites

## P

`Parser` class, 450–451

`ParserFile` class, 450, 451, 452–453

passing

- exceptions to a higher level, 320–322

- objects by reference, 410–412

- objects by value, 408

- values to functions, types and, 90–91, 93–94

passwords

- encryption for, 343

- white space and, 246

path delimiter, 369

paths, opening a file using multiple, 366–371

percent sign (%) as wildcard, 331

performance. *See* optimization; overhead; speed

physical errors, 31

plus operator (+), overloading, 120–122

plus-equal operator (+=), plus operator implementation and, 121

`Point` class

- constructors, 414, 415

- implementing, 413–414

- output from, 415

`PointerClass` class

- creating, 204–206

- output from, 207–208

pointers

- allocating arrays with and without, 204–208

- allocation not verified by C++ before freeing, 132

- arrays of object pointers, 213–215

- `auto_ptr` class for avoiding memory leaks, 303–306

in C++ versus C, 175

character, `new` operator and, 134–135

clearing in destructor for exception object, 322

generic, 175–176

member-function pointers, 96–100

memory leaks from, 212, 303

for same memory block, 28

size of, 54

`sizeof` function with, 55

structures and, 76

void pointers, 175, 176

polymorphism

- customizing a class with, 20
- defined, 19

portability. *See also* reusability

- custom locking mechanism and, 420

- opening files and, 366

- separating rules and data from code for, 31

post-conditions for code in DBC, 393

postponing variable declarations, 412–413

precision method for streams, 227

preconditions for code in DBC, 392, 397

pre-loading virtual file chunks, 290

pre-processor

- `assert` statements and, 42–44

- `#define` versus `const` statement and, 45, 47

- handling multiple operating systems, 39–41

- header files and, 39–41

- macro code and, 48

- macros and, 48–51

- `sizeof` function and, 52–55

- using `const` instead of `#define`, 45–47

print statements for  
     debugging, 208  
 printf function, stream components versus, 225  
 printing  
     arrays using streams, 226–227  
     virtual methods for, 19  
 ProcessEntries function, 17, 18  
 processing files  
     C++ versus C and, 425–426  
     creating the test file, 233  
     C-style file-handling function problems, 425–426  
     data processing with same code, 228  
     file-reading class, 228–230  
     opening using multiple paths, 366–371  
     reading delimited files, 234–239  
     reading internationalization files, 272–277  
     testing file-reading code, 232  
     using streams for file reading, 230–233  
     virtual files and speed for, 283  
     word-parser program, 448–451, 452–453  
 Processor class, 98–99  
 Processor2 class, 99  
 Processor2 class test driver, 99  
 processors, new in place operator and embedded, 135  
 program-specific input, identifying, 452  
 properties  
     in C# and Java, 136, 141  
     class for implementing, 137–139, 417–418  
     defined, 136  
     extending the implementation class, 139–140  
     invalid assignments and, 136–137  
     read-only, 137  
     set and get methods for, 136  
     testing the implementation class, 140–141, 418–419

Properties class (example 1)  
     for ConfigurationFile class, 24–25  
     virtual destructors in, 24, 25  
 Properties class (example 2)  
     for documenting data flow, 417–419  
     implementing, 417–418  
     testing, 418–419  
     undo method, 418, 419  
 protecting data  
     with encapsulation, 7–11  
     from memory overwrites, 307–311  
 pure virtual base classes for interfaces, 354  
 pure virtual methods. *See also* virtual methods  
     abstraction and, 12  
     code reuse through, 12  
     defined, 12  
     derived classes and, 354  
     First method, 13–14, 15  
     Next method, 13–14, 15  
     virtual methods versus, 19, 354  
 puts function, 429, 430

---

## Q

question mark (?)  
     with Match class, 333  
     as wildcard, 330–331, 333

---

## R

Range class  
     assignment operators in, 61  
     enumerations compared to, 71  
     implementing, 60–62  
     need for, 59–60  
     operators, 61  
     source-code listing, 60–61  
     testing, 62  
 RangeEntry class, 440–441  
 read function, 256, 259

readability  
     enumerations for, 70, 71  
     overloaded operators and, 121  
 read\_file function, chaining errors from, 321  
 reading files  
     creating the test file, 233  
     data processing with same code, 228  
     delimited files, 234–239  
     file-reading class, 228–230  
     internationalization file, 272–277  
     opening a file using multiple paths, 366–371  
     testing file-reading code, 232  
     using streams, 230–233  
     virtual files and speed for, 283  
     word-parser program, 448–451, 452–453  
 read-only properties, 137  
 read\_record function, chaining errors from, 321  
 reducing the complexity of code  
     basic principles, 447  
     by componentizing, 449–451  
     KISS principle for, 31, 447  
     by restructuring, 451–452  
     sample program, 447–449  
     by specialization, 452–453  
 redundant code, restructuring, 451–452  
 refactoring, 451–452  
 regression tests, 436  
 “release mode” for assert macro, 388  
 removing  
     items using iterators, 294, 296  
     white space from input, 246–249  
 Report method, 165–166, 167  
 restructuring, 451–452  
 re-throwing exceptions  
     code example, 320–321  
     output using, 321–322  
     uses for, 319

- retrieving line of file into string object, 232
- return codes or status codes
  - chaining, 328–329
  - forcing checking of, 324–329
  - return code class, 324–329
  - in signatures for methods, 399
  - status codes defined, 323
  - typical example, 323
- RetVal class
  - copy constructor, 329
  - output from, 328
  - overview, 328
  - source-code listing, 324–327
- reusability. *See also* portability; templates
  - for business rule code, 30, 31
  - C++ versus C and, 85
  - collections and, 291
  - for date code, 30, 31
  - encapsulation and, 30, 31
  - encoding and decoding class for, 337
  - error handling and, 31
  - generic method for reading delimited files and, 234
  - placing classes in namespaces for, 89
  - pure virtual methods and, 12
- reverse iteration, 294, 295
- Rot13 encryption algorithm, 343, 344–346
- ROT13Encryption class
  - algorithm described, 344
  - EBCDIC systems and, 345
  - implementing, 344–345
  - testing, 345–346
- Row class for spreadsheet
  - creating, 218–219
  - exception handling in, 218, 219
  - functionality needed for, 216
- RSS encryption algorithm, 343
- rules
  - for auto\_ptr class, 306
  - business rules, 30–36
  - for Complete class, 109
  - for delete handler implementation, 129
  - for inline functions, 408

- for new handler implementation, 129
- for overloaded operator creation, 121–122
- of specialization, 452–453
- for types, encapsulating within a class, 142
- for URLs, 337
- run-time
  - assert macro and, 42, 389
  - turning logging on and off, 389, 390

## S

- Save method, 270, 272
- Save mix-in class, 169–171
- SavePairs class, 25–26, 28–29
- scope
  - arrays of objects on the stack and, 209
  - defined, 82
  - global, local, and loop, 83
  - handled automatically for classes and objects, 82–83
  - scoping member functions versus casting, 93–95
  - scoping variables, 82–84
  - viewing visually, 83–84
- searching
  - for files across multiple paths, 366–371
  - hash table for search and replace, 279
  - wildcards for, 330–334
- security
  - buffer overflows and, 360, 361
  - hiding algorithms and, 8
  - strings and, 273
- separating rules and data from code, 30–36
- Serialization class
  - getClassname virtual method, 357
  - getElements virtual method, 357
  - source-code listing, 356–357
  - write member method, 356–357

- serialization interface
  - implementing the serialization interface, 355–358
  - interface defined, 354
  - serialization defined, 354
  - steps for implementing an interface, 354–355
  - testing the serialization interface, 358–359
- SerializeEntry class
  - source-code listing, 355–356
  - testing, 358–359
- set methods
  - for Complete class, 109, 112
  - for IntProperty class, 138, 139–140
  - for properties, 136
- setLock method, 422
- set\_terminate function
  - described, 318
  - divide-by-zero error and, 317–319
  - output from application, 319
  - using in applications, 318–319
- setX method, 402, 403
- SGML, XML as variant of, 240
- side effects
  - of macros, 48–49, 50
  - of min macro, 48–49
  - of overloaded operators, 121
- signatures for methods, 399
- simplicity. *See also* reducing the complexity of code
  - KISS principle for, 31, 447
  - usefulness of classes and, 438
- sizeof function
  - arrays and, 55
  - evaluating results of, 54–55
  - with pointers, 55
  - pre-processor and, 52
  - uses for, 52
  - using, 52–54
- Social Security Number validation
  - constants for SSN length and delimiter, 143, 146
  - defining the Validator object for, 142–143
  - testing, 146–148
  - validation module for, 143–146

- source-code files. *See* companion Web site for this book
- source-code listings
- AgeProperty class, 139
  - allocating arrays with and without pointers, 205–206
  - array of object pointers, 214
  - auto\_ptr test program, 305
  - base class for casting, 91–92, 94
  - Base template class, 180–182
  - Base template class test driver, 183
  - base-class inheritance, 118
  - BaseMailingListEntry class, 13–14
  - Buffer class (example 1), 308–310
  - Buffer class (example 2), 361–363
  - Buffer class test driver, 364
  - BufferException class, 362
  - cDate class implementation, 32–34
  - cDate class non-inline methods, 35
  - cDate class test program, 36
  - ChangeManagement class, 115
  - Column class, 217
  - command processor class, 99
  - command processor class test driver, 99
  - CommandLineMailingListEntry class, 16–17
  - Complete class implementation, 110–113
  - Complete class test driver, 114
  - Complex class definition, 433
  - Complex class implementation, 434–435
  - Complex class test program, 437
  - Complex class variable methods, 436
  - ConfigurationFile class (example 1) implementation, 26
  - ConfigurationFile class (example 1) test program, 27
  - ConfigurationFile class (example 2) definition, 251
  - ConfigurationFile class (example 2) implementation, 256
  - ConfigurationFile class (example 2) test program, 260
  - ConfigurationFile.cpp file, 251–259
  - ConfigurationFile.h header file, 251
  - const keyword to differentiate methods, 81
  - constants and their definitions, 78
  - constants in classes, 79–80
  - conventions in this book, 2–3
  - conversion base classes, 440–444
  - conversion code for STL string class, 350–351
  - Date class definition, 150–151
  - Date class functionality, 152–158
  - Date class source file, 151–152
  - Date class test driver, 159–160
  - DBCObject class implementation, 393–397
  - DBCObject class test program, 397–398
  - debugFlowTracer class implementation, 376–377
  - debugFlowTracer class test program, 379
  - debugFlowTracerManager class, 378–379
  - decode method, 339–340
  - DelimitedFileParser class, 237–238
  - DelimitedRow class, 236–237
  - Delimiters class, 235–236
  - derived class test drivers, 93, 95
  - derived classes for casting, 92, 94
  - Design by Contract example, 393–397
  - Design by Contract test program, 397–398
  - DirectoryList class, 367–368
  - document class, 87–88
  - Encode method, 9, 10–11
  - ExceptionCatcher class, 314–315, 316
  - ExceptionClass class, 313–314, 315–316
  - factory class, 163–166
  - factory object test driver, 166–167
  - file read test driver, 232
  - FileHandler class implementation, 103–104
  - FileHandler class test driver, 105
  - FileMailingListEntry class, 14–15
  - file-reading class, 228–230
  - FileWrapper guardian class implementation, 427–429
  - FileWrapper guardian class test program, 430–431
  - First method, 13–14, 15
  - Fruit class, 20
  - function with immutable argument, 78
  - header file test program, 40–41
  - HundredsRangeEntry class, 443–444
  - inserting tracing into an existing file, 381–384
  - IntProperty class extension, 139
  - IntProperty class implementation, 137–139
  - Lock class implementation, 421
  - Lock class test driver, 422–423
  - LockException class implementation, 421
  - LockException class test driver, 422–423
  - Logger class implementation, 389–390
  - Logger class test driver, 391
  - macro file, 49–51

source-code listings (*continued*)

- mailing-list application test program, 17–18
- Match class, 331–333
- Match class test driver, 334
- Matrix class implementation, 64–65
- Matrix class manipulation functions, 67–68
- Matrix class operators, 66, 67
- Matrix class test driver, 68
- memory allocator test driver, 133
- mix-in class, 169–170
- MultiPathFile class implementation, 367–369
- MultiPathFile class test program, 370
- multiple inheritance, 116
- Multiply method template, 189–190
- Multiply method template test driver, 190
- MyAlloc class definition, 298–300
- MyAlloc class test driver, 301
- MyBuffer class, 301
- MyClass test class for serialization interface, 358–359
- my\_min template function, 187–188
- MyReturnValue class, 326–327
- MyString class accessor methods, 123
- MyString class implementation, 123
- MyString class operators, 124–125
- MyString class test driver, 126
- MyStringArray class, 197–198
- new and delete handlers, 130
- Next method, 13–14, 15
- NumberToWords class implementation, 445
- NumberToWords class test program, 446
- Object base class, 163–166

## object-array allocation program, 210

- OnesRangeEntry class, 441–442
- OpenFile method, 28
- osdefines.h header file, 40
- overloaded methods, initial implementation, 400–401
- overloaded methods, logging added, 401–402
- overloaded new and delete handlers, 132
- Parser class, 450–451
- ParserFile class, 450
- passing objects by reference, 411
- Point class, 413–414
- pointers to member functions, 97–98
- printing arrays using streams, 226–227
- ProcessEntries function, 17
- Properties class for ConfigurationFile class, 24–25
- Properties class for documenting data flow, 417–418
- Properties class test program, 418–419
- Range class implementation, 60–61
- Range class test driver, 62
- RangeEntry class, 440–441
- reading delimited files, 235–238
- RetVal class, 324–327
- ROT13Encryption class implementation, 344–345
- ROT13Encryption class test driver, 345
- Row class, 218
- Save mix-in class, 169–170
- SavePairs class, 25, 29
- for scope illustration, 83
- Serialization class, 356–357
- serialization interface test driver, 358–359
- SerializeEntry class, 355–356

## set\_terminate function, 318

- simple template, 176–177
- sizeof program, 53
- Spreadsheet class implementation, 219–220
- Spreadsheet class test driver, 221
- SSNValidator class implementation, 143, 144–145
- SSNValidator class test driver, 146–147
- StringCoding class implementation, 8–9
- StringCoding class update, 10–11
- StringConvertToLowerCase class implementation, 351
- StringConvertToLowerCase class test driver, 352
- StringEntry class, 266–268
- StringReader class, 273–277
- StringUtil utility class, 252–255
- StringWriter class, 268–271
- structure implementation, 74–75
- structure test harness, 75
- temp.cpp file, 385
- template class with non-class argument, 184
- templated class test driver, 183
- templated classes in code, 180–182
- temporary objects, 408–409
- TensRangeEntry class, 443
- test driver for constant application, 80–81
- test drivers for casting, 93, 95
- TestIntValue class implementation, 140
- TestIntValue class test driver, 149
- ThousandsRangeEntry class, 444
- Tracker class, 304–305

- Translator class
  - implementation, 280
- Translator class test driver, 281–282
- URLConnection class implementation, 338–340
- URLConnection class test driver, 340–341
- using casts, 91–92
- using code in your own programs, 1
- using constants, 46
- using namespace statement, 87
- using namespaces in an application, 88
- using streams for file reading, 230–231
- using vector class, 193–194
- using wrong namespace class, 89
- vector algorithm program, 201–202
- white space removal code, 246–248
- word-parser program
  - componentized, 450–451
- word-parser program original, 448–449
- XMLElement class, 241–242
- XMLSuperClass class, 244
- XMLWriter class, 241–243
- XMLWriter class test driver, 243–244
- XOREncryption class
  - implementation, 346–347
- XOREncryption class test driver, 348
- specialization, 452–453
- speed. *See also* optimization; overhead
  - exception handling and, 322
  - inline functions and, 407
  - STL use and, 199
  - virtual files for conserving, 283
- spelling out numbers. *See* converting numbers to words
- splitting into components. *See* componentizing
- Spreadsheet class
  - Column class for, 216, 217–218, 221
  - creating, 219–221
  - data encapsulation in, 216
  - Row class for, 216, 218–219
  - testing, 221–222
- spreadsheets
  - basic elements, 216
  - creating the column class, 217–218
  - creating the row class, 218–219
  - creating the spreadsheet class, 219–221
  - mimicking a two-dimensional array, 222
  - overview, 216
  - shell from STL, 216
  - simple arrays versus, 216
  - testing, 221–222
- sprintf function versus stream components, 225
- SSNValidator class
  - constants for SSN length and delimiter, 143, 146
  - defining the Validator object, 142–143
  - Social Security Number validation module, 143–146
  - testing, 146–148
- stack, declaring arrays on, 209–212
- Standard Template Library. *See* STL
- static arrays
  - buffer overflow and, 361
  - declaring on the stack, 209, 211
- static data members, 424
- static methods
  - calling as a default value, 104
  - jump tables for, 23
- status codes. *See* return codes or status codes
- stdio.h header file (Windows), 39, 41
- STL (Standard Template Library)
  - algorithm functions, 200
  - auto\_ptr class, 303–306
  - benefits of, 199, 200
  - collections, 291–296
  - complex number template in, 433
  - configurable classes as
    - strength of, 297
  - container classes, 179, 200, 209–212
  - creating arrays of objects
    - using, 209–212
  - documentation, 199
  - hash tables in, 279, 281
  - inserting classes in templates, 297
  - iterating over collection
    - classes, 292–296
  - map class, 281
  - memory allocation complications for, 297
  - multiple array classes and, 196
  - overhead from using, 192, 195, 196, 199, 209
  - overriding the allocator for
    - collections, 297–302
  - Properties class use of, 25
  - for spreadsheet shell, 216
  - transform function, 349–353
  - using arrays from, creating
    - your own versus, 199
  - vector class, 25, 192–195, 200–203, 209–212
- storage allocation. *See also* data storage
  - inheritance from base class, 15
  - in Matrix class, 65
  - section of classes for, 23
- storage classes. *See* container classes (STL)
- strcpy function, memory overwrites from, 307–308
- stream components
  - benefits of, 225
  - C-style functions versus, 225, 228, 231–232, 233

- stream components (*continued*)
  - file reading using, 230–231
  - file-reading class, 228–230
  - formatting data using, 225–227
  - functionality built into, 225
  - iterating over, 294, 296
  - passing objects to functions, 410–411
  - precision method, 227
  - testing file-reading code, 232
  - vectors with, 226–227
  - width method, 227
- string array class
  - creating, 196–198
  - memory and speed and, 199
  - output from, 198
- string macros, avoiding problems with, 49–51
- string objects, nulls or control characters and, 348
- StringCoding class
  - benefits of, 7–8
  - creating and implementing, 8–10
  - Encode method, 9, 10–11
  - output from, 10
  - source-code listing, 8–9
  - updating, 10–11
- StringConvertToLowerCase class
  - creating, 350–351
  - operator method, 351
  - testing, 351–353
- StringEntry class
  - creating, 266–268, 271
  - input text file for, 272
- StringEntry.cpp file, 272
- StringReader class
  - building, 271–277
  - testing, 277–278
- strings
  - associated operators for, 121–122
  - avoiding assuming contiguous order for, 349
  - as containers of characters, 349
  - control characters and string classes, 348
  - converting case, 349–353
  - converting numbers to words, 439–446
  - encapsulated encryption
    - method for, 7–10
  - encoding and decoding for the Web, 337–342
  - encrypting and decrypting, 343–348
  - encrypting text string files, 273
  - hackers and, 273
  - implementing the transform function to convert case, 350–351
  - Match class for wildcard searches, 331–334
  - memory overwrites from copying, 307–308
  - MyString class for overloaded operators, 122–127
  - MyStringArray class, 196–199
  - nulls and string classes, 348
  - retrieving line of file into string object, 232
  - security issues, 273
  - size of, 54
  - storing literal information in classes, 161
  - StringCoding class, 8–11
  - StringEntry class for internationalization, 266–268, 271
  - StringReader class for internationalization, 273–277
  - StringUtil utility class
    - for configuration files, 252–255, 259
  - StringWriter class for internationalization, 268–272
  - testing the string case conversion, 351–353
  - testing the string reader, 277–278
  - word-parser program, 447–451, 452–453
  - written by Logger class, 390
  - xor operation and, 348
- StringUtil utility class, 252–255, 259
- StringWriter class
  - building, 268–272
  - Load function, 270, 271
  - Save method, 270, 272
- strip\_leading function, 247, 248–249
- strip\_trailing function, 247, 248–249
- struct construct
  - as beginning of encapsulation, 73
  - in C++ versus C, 74
  - as a class with public members, 73
  - defining, 74–75
- structures
  - in C++ versus C, 73–74
  - classes versus, 76
  - constructors for, 76
  - derived, 75, 76
  - implementing, 74–75
  - initialization of data values
    - required for, 76
  - interpreting output, 75–76
  - limitations of, 74
  - overhead minimized by, 74
  - overriding classes, 73
  - pointers and, 76
  - test harness, 75
  - v-tables missing from, 74, 76
- strup function (C language), 349
- support, documenting data flow and, 419
- swapping elements using iterators, 294
- syntactical sugar
  - enumerations as, 70
  - macros as, 51

## T

Telles, Matthew (*C++ Timesaving Techniques For Dummies*), 1–4

temp.cpp file, 385

temp.cpp.tmp output file, 385–386

- template keyword, 178
- templated functions
  - automatic generation of, 189
  - defined, 186
  - implementing, 186–189
  - types and, 189
- templated methods
  - creating, 189–190
  - defined, 186
  - testing, 190–191
  - uses for, 189
- templates. *See also* STL (Standard Template Library)
  - auto\_ptrs in STL collections, 306
  - benefits of, 110
  - class template arguments, 179, 182, 183
  - compilers and template keyword, 178
  - Complete class, 110–113
  - for complex numbers in the STL, 433
  - constructors and destructors and, 178
  - creating a simple template class, 175–178
  - defined, 175
  - extending a template class, 179–185
  - function templates, 186–189
    - as giant macros, 178
  - header files for implementing classes, 178, 179
  - implementing classes in code, 180–182
  - macros versus, 178
  - method templates, 186, 189–191
  - non-class template arguments, 184–185
  - overhead for, 179
  - for printing arrays using streams, 227–228
  - STL classes in, 297
  - suitability for extension, 183
  - testing template classes, 182–183
  - ways of using, 179
  - wrapping pointers in auto\_ptr
    - template object, 303–305
- temporary objects
  - avoiding, 408–410
  - code example overdoing, 408–409
  - output from code example, 410
  - processes creating, 408
- TensRangeEntry class, 443
- term\_func function, 318–319
- test\_delimited.txt file, 239
- test.in.eng file, 272
- testing. *See also* debugging
  - asserts in optimized environment, 44
  - Buffer class, 364–365
  - casts, 93, 94–95
  - cDate class, 35–36
  - code for converting numbers to words, 446
  - Complete class, 113–115
  - ConfigurationFile class (example 1), 27
  - ConfigurationFile class (example 2), 260–261
  - constant application, 80–81
  - creating generic test drivers for validators, 148
  - Date class, 159–161
  - DBObject class, 397–398
  - debugFlowTracer class, 379–380
  - delimited-file-reading code, 238–239
  - Design by Contract methodology, 397–398
  - enumeration class, 72
  - exercising all functionality for classes, 161
  - FileHandler class, 105
  - file-reading code, 232
  - FileWrapper guardian class, 430–431
  - inserting tracing into an existing file, 385–386
  - IntProperty class, 140–141
  - iterators, 295–296
  - Logger class, 389–390
  - mailing-list application, 17–18
  - Match class, 333–334
  - Matrix class, 68–69
  - member pointer code, 99–100
  - memory tracking program, 133–135
  - mix-in classes, 170–171
  - Multiply method template, 190–191
  - MyAlloc class, 301–302
  - MyString class, 125–127
  - namespace application, 88–89
  - non-class template arguments, 184–185
  - osdefines.h header file, 40–41
  - Properties class, 418–419
  - providing test suite with application file, 436
  - Range class, 62
  - regression tests, 436
  - ROT13Encryption class, 345–346
  - serialization interface, 358–359
  - Spreadsheet class, 221–222
  - SSNValidator class, 146–148
  - string case conversion, 351–353
  - StringReader class, 277–278
  - structures, 75–76
  - template classes, 182–183
  - Tracker class, 305–306
  - Translator class, 281–282
  - unit tests versus complete tests, 115
  - URLCodec class, 340–341
  - validation versus, 115
  - virtual file class, 289–290
  - white space removal application, 249
  - XMLWriter class, 243–245
  - XOREncryption class, 347–348
- TestIntValue class, 140–141
- test.out file, 431
- test2.xml file, 244

test.txt file  
 for file-reading code, 233  
 for mix-in classes, 170–171

test.xml file, 244

text format for configuration files, 250

32-bit operating systems, 54

ThousandsRangeEntry class, 444, 445

throwing exceptions. *See also* exception handling  
 classes for, 312–317  
 re-throwing, 319–322

time function limitations, 149

to-do list for class improvements, 290

tracing. *See also* logging  
 adding after the fact, 380–386  
 building into applications, 375–380  
 caveats for insertion programs, 380  
 debugFlowTracer class for, 376–377  
 debugFlowTracerManager class for, 377–379  
 need for, 375  
 testing the flow trace system, 379–380

Tracker class  
 creating, 304–305  
 testing, 305–306

trailing spaces. *See* white space

transform function  
 described, 349–350  
 implementing to convert strings, 350–351  
 operator method called by, 351  
 StringConvertToLowerCase class and, 351  
 testing the string conversion, 351–353

translate.txt file, 282

translation. *See also* conversion; encryption  
 hash table uses for, 279  
 Translator class, 279–282

Translator class  
 creating, 279–281  
 testing, 281–282  
 translate.txt file for, 282

two-dimensional arrays, 222

typeface conventions in this book, 2–3

types. *See also* constants  
 casting to an object, 408  
 const constructs as  
 type-safe, 46  
 converting with casts, 90–95  
 creating new types, 63–69  
 defining default arguments and, 101–106  
 encapsulating types and rules, 142  
 enumerations, 70–72  
 extending basic types, 59–62  
 for hiding implementation from user, 11  
 identifying and validating for applications, 142  
 pointers to member functions and, 96–100  
 scoping variables, 82–84  
 sizeof function and, 52  
 stream components and, 225  
 structures, 73–76  
 as template arguments, 184–185  
 template functions and, 189  
 temporary objects and, 408  
 using namespaces, 85–89

---

## U

---

undo method, 418, 419

unistd.h header file (Unix), 39

unit tests versus complete tests, 115

unLock method of Lock class, 422

updating  
 benefits of encapsulation for, 10  
 encapsulated class, 10–11

uppercase. *See* case

URLConnection class  
 creating, 338–340  
 decode method, 339–340, 341  
 encode method, 339, 340, 341  
 testing, 340–341

URLs, rules for, 337

user name encryption, 343

using namespace statement, 87

utilities  
 caveats for insertion programs, 380  
 complex variable utility methods, 435–436  
 converting the case of a string, 349–353  
 encoding and decoding data for the Web, 337–342  
 encrypting and decrypting strings, 343–348  
 generic buffer class, 360–365  
 inserting tracing into an existing file, 380–386  
 keeping a library of utility classes, 353  
 opening a file using multiple paths, 366–371  
 serialization interface, 354–359

---

## V

---

validation  
 classes for data validation, 142–148  
 Date class code for, 153, 159  
 discrete pieces in classes for, 159  
 documenting data flow and, 419  
 for fixed-length input, 452–453  
 guardian classes and hardware inputs, 425  
 missing from standard date routines, 149  
 optimizing instantiation of variables and, 412–413  
 Range class for, 60–62  
 testing versus, 115

validity checks for code in  
     DBC, 392–393  
 of values, enumerations for, 71  
 validation classes. *See also* Date  
     class; SSNValidator class  
 application development and,  
     142, 149  
 defining the Validator object,  
     142–143  
 generic test drivers for  
     validators, 148  
 including numeric manipula-  
     tion in, 149  
 Social Security Number  
     validation module, 143–146  
 testing, 146–148  
 values. *See also* casts  
     class with methods containing  
         defaults, 103–106  
     enumerations for meaningful  
         names, 70  
     enumerations for validating, 71  
     hard-coded, 30  
     initializing for classes or  
         structures, 76  
     passed to functions, types and,  
         90–91, 93–94  
     passing by reference, 411–412  
     passing objects by, 408  
     replacing #define values using  
         constants, 77–78  
 variables  
     complex variable utility  
         methods, 435–436  
     constants versus basic integer  
         variables, 59  
     implementing constant  
         variables, 78–80  
     optimizing instantiation of,  
         412–413  
     postponing declarations,  
         412–413  
     scoping, 82–84  
     templates as member  
         variables, 179  
     verifying that value falls within  
         a range, 60–62

vector algorithms  
     benefits of, 200  
     output from the program, 203  
     program using, 201–202  
 vector class (STL)  
     algorithms, 200–203  
     arrays of objects using,  
         209–212  
     data manipulation in vector  
         by, 212  
     defining array of objects,  
         210, 211  
     overhead from, 192, 195, 209  
     overview, 25, 192  
     printing arrays using streams,  
         226–227  
     single function using multiple  
         algorithms, 203  
     using, 192–195  
 vectors. *See* arrays  
 virtual destructors. *See also*  
     destructors  
     for base classes, 22, 25  
     for Complete class, 109  
     described, 22  
     in Fruit class, 22  
     in Properties class, 24, 25  
     required for Complete class, 111  
 virtual files  
     algorithm, 280, 290  
     benefits of, 283  
     configurable chunk size for, 290  
     creating a virtual file class,  
         283–288  
     defined, 283  
     improving the virtual file  
         class, 290  
     memory and speed conserved  
         by, 283  
     pre-loading chunks, 290  
     testing the virtual file class,  
         289–290  
 virtual inheritance  
     correcting the code, 119  
     defined, 117  
     implementing, 118–119  
     virtual methods and, 119

virtual methods. *See also* pure  
     virtual methods  
     base class for, 20  
     in Column class for spread-  
         sheet, 217, 218  
     customizing a class with, 19–22  
     defined, 19  
     derived classes and, 23  
     factory pattern and, 162  
     getClassName, 357  
     getElements, 357  
     main driver for, 21  
     not allowed in structures, 74  
     overhead for, 21  
     overriding classes, 19, 21,  
         162–167  
     pure virtual methods versus,  
         19, 354  
     size of classes and, 54  
     testing, 21–22  
     using whenever possible, 19  
     virtual inheritance and, 119  
     v-table for, 21, 22, 23, 54  
 Visual Studio C++ compiler, 2  
 void pointers, 175, 176  
 v-tables  
     derived classes and, 23  
     missing from structures, 74, 76  
     section of classes for, 23  
     size of classes and, 54  
     for virtual methods, 21, 22, 23

## W

Web. *See* companion Web site for  
     this book; Internet, the;  
     URLConnection class  
 white space  
     application accountability  
         for, 246  
     code for removing, 246–248  
     output from program for  
         removing, 248  
     passwords and, 246  
     removing from input, 246–249  
     testing the application, 249  
 width method for streams, 227

**wildcards**

- asterisk (\*), 330
- defined, 330
- Match class for, 331–334
- percent sign (%), 331
- power of, 330–331
- question mark (?), 330–331
- uses for, 330–331

**word-parser program**

- componentizing, 449–451
- input file for, 449
- source-code listing, 448–449
- specialization, 452–453

**words, converting numbers to.**

*See* converting numbers to words

**World Wide Web. *See* companion**

Web site for this book;  
Internet, the; `URLConnection` class

**wrapping**

- pointers in `auto_ptr` template object, 303–305
- potentially unsafe code in guardian classes, 425–431
- `write` member method, 356–357

---

**X**

---

- XML (eXtended Markup Language).** *See also* delimited files
- class structure compared to, 240–241
  - creating the `XMLWriter` class, 241–243
  - delimited files using, 234
  - general format of structures, 240

**importance of capability for output as,** 240

testing the `XMLWriter` class, 243–245

`XMLElement` class, 241–242, 245

`XMLElement` objects, 243

`XMLSuperClass` class, 244, 245

`XMLWriter` class

creating, 241–243

testing, 243–245

`XMLElement` objects, 243

**XOR encryption algorithm,** 343, 346–348

`xor` operation, strings and, 348

`XOREncryption` class

algorithm described, 346

`do_xor` method, 346, 347

implementing, 346–347

strings and `xor` operation, 348

testing, 347–348