

Index

• Symbols •

- & (ampersand) operator (pointer-related), 113–114
 - && (AND) operator
 - bitwise, 58
 - short-circuit evaluation and, 54–55
 - simple, 49, 51
 - truth table for, 59
 - > (arrow pointer), 184
 - = (assignment) operator
 - description of, 23, 46
 - overloading, 285–291
 - * (asterisk) operator (pointer-related), 113, 116
 - \ (backslash)
 - in filename, 34
 - in preprocessor command, 157
 - \\ (backslash character), 36
 - # (begin of preprocessor command), 146, 149
 - : (colon) in constructor, 228
 - /* */ comment form, 21
 - (decrement operator)
 - description of, 45
 - loops and, 67–69
 - #define command
 - constants of user-defined type, 153–154
 - overview of, 146, 149–151
 - when not to use, 152
 - "" (double quotes) string, 105
 - // (double slash), 20
 - '\0' (end of character array), 103–105
 - == (equality operator), 50
 - #error command, 156–157
 - >> (extraction operator), 284, 294
 - __FILE__ (intrinsic #define), 323
 - __func__ (intrinsic #define), 323
 - > (greater-than) operator, 51
 - <= (greater-than-or-equal-to) operator, 51
 - #if command, 146, 154–155
 - #ifdef conditional compilation, 155
 - #ifndef conditional compilation, 155
 - #include command
 - defining member function in class and, 175–176
 - preprocessor and, 146–149
 - #include <string> statement, 108
 - ++ (increment operator)
 - description of, 45
 - loops and, 67–69
 - << (insertion operator), 284, 294
 - < (less-than) operator, 51
 - >= (less-than-or-equal-to) operator, 51
 - __LINE__ (intrinsic #define), 323
 - (minus sign) operator, 23, 44
 - % (modulus operator), 42
 - \n (newline character)
 - binary and text modes and, 297
 - endl object and, 307
 - in strings, 35
 - || (OR) operator
 - bitwise, 58
 - short-circuit evaluation and, 54–55
 - simple, 49, 51
 - truth table for, 59
 - :: (scope resolution) operator, 174
 - \t (tab character), 35
 - ~ (tilde) destructor symbol, 212
-
- **A** •
 - absolute address of program variable, 115
 - abstract class, implementing, 277–280
 - abstraction, 161–163

- accessing
 - arguments to `main()` function, 140–143
 - arrays, 101
 - members of class, 167
 - members of class from member function, 172–173
 - Account class, 276–277
 - accuracy, and floating-point variable, 30
 - `add()` function, 342
 - address
 - of array, applying operators to, 130–131
 - assigned to variable, 113, 115
 - of current object, 253–254
 - allocating heap of objects, 190
 - ampersand (&) operator (pointer-related), 113–114
 - AND operator (&&)
 - bitwise, 58
 - short-circuit evaluation and, 54–55
 - simple, 49, 51
 - truth table for, 59
 - angle brackets, include file in, 147
 - argument
 - defining constructor with, 217–219
 - to function, description of, 86
 - functions with, 87–89
 - functions with multiple, 89
 - to `main()`, accessing, 140–143
 - passing to functions, 117–120
 - array. *See also* array of pointers
 - accessing, 101
 - applying operators to address of, 130–131
 - of arrays, defining and using, 101–102
 - benefits of, 95–97, 101
 - of characters, 102–103
 - data storage and, 345
 - description of, 95
 - initializing, 100
 - limitations of, 165
 - of objects, declaring, 181–182
 - pointer compared to, 134–135
 - pointer variables and, 128–130
 - using, 97–100
 - vectors and, 340
 - array declaration, 96
 - array of pointers
 - accessing arguments to `main()`, 140–143
 - character strings, 138–139
 - declaring and using, 137–138
 - ArrayDemo program, 97–98
 - ArrayOfStudents program, 181–182
 - arrow pointer (`->`), 184
 - `assert()` function, 157
 - assignment (=) operator
 - description of, 23, 46–47
 - overloading, 285–291
 - asterisk (*) operator (pointer-related), 113, 116
 - author, Web site of, 10
 - auto-decrement feature, loops and, 67–69
 - auto-increment feature, loops and, 67–69
 - automatic (`auto`) declaration, 38–39, 354
 - avoiding temporary object, 246
- B •**
- backslash character (`\`)
 - in filename, 34
 - in preprocessor command, 157
 - backslash character (`\\`), 36
 - `bad()` member function, 298
 - base class
 - abstract classes as, 279
 - description of, 259
 - propagation of changes in, 276
 - base class function, overriding, 267
 - binary mode, 297
 - binary number, storing, 55
 - binary number system, 56–57
 - binary operator, 41–42
 - bits, 55, 56
 - BitTest program, 53, 60, 306
 - bitwise copy, 238–239

- bitwise operation, performing, 58–62
- bitwise operator
 - decimal number system and, 55–56
 - description of, 49
 - single-bit, 58–59
 - using, 60–62
- bool variable
 - description of, 37
 - program demonstrating use of, 51–52
- braces
 - classes and, 166
 - if statement and, 64
 - while loop and, 66
- branch statement, 63–65
- BranchDemo program, 64–65
- break commands
 - loop controls and, 73–75
 - switch statement and, 78
- BreakDemo program, 73–74
- breakpoint, setting in constructor, 231
- Budget programs, on CD-ROM, 4, 280
- buffer flushing, 307
- buffer overflow, 300
- building
 - definition of, 10
 - first program, 17
- byte, 56
- byte-by-byte (shallow copy), 239–244
- C •**
- calculating expression, 23–24
- calculation speed, and floating-point variable, 30
- calling
 - function using reference operator, 188–189
 - function with object pointer, 186–187
 - function with object value, 185–186
 - member functions, 170–172
 - overloaded functions, 179
 - sumSequence(), 85
- CallMemberFunction program, 170–171
- CallStaticMember program, 251–252
- CascadingException program, 318–320
- case sensitivity, 21
- catch keyword, 315
- catching exception, 324
- CD-ROM
 - Budget programs, 4, 280
 - ConstructDataMembers program, 227–228
 - ConstructSeparateID program, 226
 - contents of, 2, 378–379
 - copying code from, 16–17
 - installing Code::Blocks from, 11
 - installing programs from, 378
 - int2Month() program, 139
 - LayoutError program, 117
 - LinkedListForward program, 196
 - MacroConfusion program, 150
 - NamespaceExample program, 149
 - system requirements for, 377–378
 - troubleshooting, 380
 - TypeConversion program, 234
 - Wiley Product Technical Support for, 381
- cerr object, 294, 295
- char variable, 35, 36
- character string
 - arrays of, 138–139
 - pointer-based manipulation of, 131–133
- character string function, 337
- character types, 36
- characters
 - creating array of, 102–103
 - creating string of, 103–105
 - manipulating string of, 105–107
- charArray type
 - description of, 130
 - pArray compared to, 134–136
- CharDisplay program, 102–103
- Checking class, 273–277, 280
- cin (input channel)
 - boilerplate code and, 89
 - description of, 22–23
 - insertion and extraction operators and, 284
 - stream I/O and, 293, 294–295
 - whitespace and, 107

- class. *See also* inheritance; subclass;
 - Student class; template class
 - abstract, 277–280
 - accessing members from member functions, 172–173
 - accessing members of, 167
 - Account, 276–277
 - adding member functions, 169–170
 - base, 259, 276, 279
 - braces and, 166
 - calling member functions, 170–172
 - Checking, 273–277, 280
 - concrete, 278
 - defining member functions inside, 175–176
 - defining member functions outside, 177–178
 - definition of, 165
 - format of, 166
 - giving nonmember functions access to protected members of, 201–204
 - graphical expression of, 274
 - ifstream, 295–296, 298–301
 - istream, 294, 301–303, 307–308
 - with limited interface, 201
 - member functions, 168–169
 - naming, 166
 - objects and, 167, 205
 - ofstream, 295–296
 - ostream, 294, 301–303
 - overloading member functions, 178–179
 - protecting internal state of, 200–201
 - protecting members of, 197–199
 - Savings, 167–169, 273–277, 279–280
 - scope resolution, 174–175
 - template, 340–344
 - user-defined, operators for, 284
- class keyword, 166
- class member. *See* static data member
- classification, 163–164
- clear() member function, 298
- code
 - on CD-ROM, 2
 - cheating by copying from CD-ROM, 16–17
 - clarifying with comments, 20–21, 362
 - entering for project, 15–16
 - readability of, 65
 - snippets of, 4

Code::Blocks environment
 - accessing program arguments in, 142
 - on CD-ROM, 2
 - description of, 10
 - error reporting in, 17–18
 - installing, 11–13
 - numeric types in, 34
 - starting, 14
 - Web site, 10

Code::Blocks/gcc compiler, wild-card expansion and, 142

coding style, 65, 360

collection, 345

colon (:) in constructor, 228

commands
 - break, 73–75, 78
 - continue, 73, 75
 - #define, 146, 149–154
 - #error, 156–157
 - #if, 154–155
 - #include, 146–149, 175–176
 - nesting control, 75–77
 - preprocessor, 146

comment, clarifying code with, 20–21, 362

comment form (/* */), 21

comparison operators, 50

compiler, 10. *See also* gcc compiler

compile-time test, preprocessor and, 158

compile-time type, 267

compiling, 10

complex data member, constructing, 224–228

- computer language, 9
- Concatenate program, 105–106
- ConcatenateWide program, 109–110
- concatString() function, 107
- concept, implementing, 372–373
- concrete class, 278
- concrete function, 278
- condition is true, while loop and, 66–67
- conditional clause, for loop and, 70
- conditional compilation, 155
- Console application screen, 14
- const expression, 152
- const variable, pointer variable and, 120–121
- constant
 - to control how files are opened, 296
 - #define command and, 149–150
 - floating-point, 29
 - intrinsic, 156–158
 - npos constant, 350
- constant address of string, 135
- constant data member, constructing, 228–229
- constant value, 32–33
- constexpression phrase, 155
- ConstructArray program, 208–209
- ConstructDataMembers program, 227–228
- constructing
 - complex data members, 224–228
 - constant data members, 228–229
 - global objects, 231–232
 - local objects, 229
 - members, 233
 - objects of multiple inheritance, 335
 - static objects, 230–231
- ConstructingMembers program, 224–225
- ConstructMembers program, 209–210
- constructor. *See also* copy constructor
 - with arguments, 217–219
 - declaring virtual, 271
 - default, 217, 222–224
 - description of, 206
 - duplex and, 209–211
 - as form of conversion, 233–234
 - multiple objects and, 208–209
 - order of construction, rules for, 229
 - overloading, 220–222
 - setting breakpoint in, 231
 - single object and, 207–208
 - for subclass, 262–263
- ConstructorWArg program, 218–219
- ConstructorWDefaults program, 221–222
- ConstructSeparateID program, 226
- ConstructStatic program, 230
- container
 - description of, 345
 - list, 351–356
 - string, 346–351
- continue command, 73, 75
- ContinueDemo program, 75
- controlling format of output, 311–313
- conversion, constructor and, 233–234
- Conversion.exe example program
 - building, 17–18
 - creating project, 13–14
 - entering code, 15–16
 - executing, 18–19
 - expressions in, 23–24
- copy constructor
 - assignment operator compared to, 285–286
 - default, 238–240
 - description of, 235
 - shallow versus deep copies, 239–244
 - temporary objects and, 244–246
 - uses for, 236–238
- CopyConstructor program, 236–237
- copying
 - code from CD-ROM, 16–17
 - objects, 189, 235–238
- cout (output device)
 - boilerplate code and, 89
 - description of, 22–23

- cout (output device) (*continued*)
 - insertion and extraction operators and, 284
 - stream I/O and, 293, 294–295
 - whitespace and, 107
 - C++, description of, 2
 - C++ 2009 standard
 - compilers and, 380
 - concepts, 372–373
 - enabling features related to, 13
 - icon for, 6
 - initializing data members inline, 369–370
 - lambda expressions, 373–374
 - overview of, 2
 - rvalue references, 371–372
 - smart pointers, 367–368
 - thread local storage, 371
 - variable-length lists, 368–369
 - variadic templates, 374
 - .CPP file extension, 9
 - C-string, 104
 - Ctrl+F9 (Build), 17
 - current object
 - address of, 253–254
 - description of, 173
 - naming, 172–173
 - customer care, 381
 - CustomExceptionClass program, 321–323
- D •
- data member
 - allocating, 247
 - complex, 224–228
 - constant, 228–229
 - initializing inline, 369–370
 - static, 247–251
 - data storage, 345
 - data type, intrinsic, 283
 - debugger. *See also* debugging
 - polymorphism and, 270
 - single-stepping program with, 362–363
 - debugging. *See also* debugger
 - initializing global variables and, 231
 - warnings and, 360
 - decimal number, 28–29
 - decimal number system, and bitwise operator, 55–56
 - declaration
 - automatic, 38–39, 354
 - forward, 203
 - prototype, 91–93, 146
 - writing, 22
 - declaring
 - arrays of objects, 181–182
 - assignment operators protected, 291
 - copy constructors protected, 246
 - destructors virtual, 364–365
 - iterators, 354
 - pointer variables, 116
 - pointers to objects, 182–184
 - pure virtual functions, 278–279
 - static member functions, 251–253
 - variable types, 31–32
 - variables, 25–26
 - decrement operator (–)
 - description of, 45
 - loops and, 67–69
 - deep copy, 243–244
 - DeepCopy program, 243
 - default keyword, 223–224
 - DefaultCopyConstructor program, 239
 - #define command
 - constants of user-defined type, 153–154
 - overview of, 146, 149–151
 - when not to use, 152
 - defining
 - function prototypes, 91–93
 - lambda expressions, 373–374

- member functions inside class, 175–176
 - member functions outside class, 177–178
 - `nullptr` keyword, 289
 - static data members, 247–248
 - variadic templates, 374
 - `delete` keyword, 124–125
 - `delete[]` keyword, 190, 215
 - deleting
 - copy constructor, 246
 - default assignment operator, 291
 - DemoAssignmentOperator program, 286–289
 - demotion, 37
 - dereferencing object pointer, 183–184
 - destructive, `for` loop as, 71
 - DestructMembers program, 212–214
 - destructor
 - description of, 211
 - order of invocation of, 233
 - for subclass, 263
 - virtual, 271–272, 364–365
 - working with, 212–215
 - development environment, 10. *See also* Code::Blocks environment
 - digit, 56
 - `displayArray()` function
 - description of, 99
 - pointer version of, 130–131
 - `displayCharArray()` function, 103
 - `displayExplanation()` function, 82–84
 - `displayString()` function, 104
 - DisplayString program
 - pointer version of, 132
 - string of characters and, 104
 - `do...while` loop, 67
 - DOS style, accessing program argument, 141–142
 - double precision, 29
 - double quotes (“”) string
 - double slash (//), 20
 - duplex, constructing, 209–211
 - dynamic type, 267
- E •**
- early binding, 267, 269
 - editor, 10
 - Enable All Compiler Warnings option, selecting, 12
 - enabling warnings and error messages, 359–360
 - `endl` object, 307
 - enumeration, defining, 153–154
 - `eof()` member function, 300, 301
 - equality operator (==), 50
 - error, minimizing
 - avoiding operator overloading, 363
 - coding style and, 360
 - commenting code and, 362
 - declaring destructors virtual, 364–365
 - enabling warnings and error messages, 359–360
 - exceptions and, 363–364
 - limiting visibility of class internals, 360–362
 - managing heap systematically, 363
 - multiple inheritance and, 365
 - single-stepping program, 362–363
 - `#error` command, 156–157
 - error handling. *See* exception
 - error message
 - `cerr` object and, 295
 - enabling, 359–360
 - intrinsic constants and, 156–158
 - segment violation, 135
 - error reporting in Code::Blocks environment, 17–18
 - exception
 - catching and rethrowing up stack chain, 324
 - definition of, 315
 - floating-point overflow, 34

- exception (*continued*)
 - justification for mechanism of, 317–318
 - program example, 315–316
 - steps in handling, 318–321
 - throw keyword and, 321–323
 - using, 363
 - exclusive OR operator, truth table for, 59
 - .exe (executable) file, 10, 146
 - executing first program, 18–19
 - explicit keyword, 234
 - expression
 - assignment operator and, 290
 - calculating, 23–24
 - const, 152
 - lambda, 373–374
 - mathematical, 43
 - mixed mode, 37–38
 - extended function name, 90–91, 265
 - extensions to C++, 2
 - extern template, instantiating, 370–371
 - extraction operator (>>), 284, 294
- F •**
- FactorialException program, 315–316
 - factoring, definition of, 273, 277
 - fail() member function, 293
 - file
 - machine-executable, 19
 - opening, 296–297
 - positioning pointer within, 307–308
 - source, 9, 16
 - Template.cpp file, 20
 - __FILE__ (intrinsic #define), 323
 - file extensions
 - .CPP, 9
 - .exe file, 10, 146
 - .o file, 146
 - FileCopy program, 304
 - flag, using static data member as, 251
 - flags() method, 305, 306
 - floating-point overflow, 34
 - floating-point variable
 - description of, 28–29
 - limitations of, 29–30
 - logical operations and, 53–54
 - flow control
 - autoincrement/autodecrement features and, 67–69
 - branch commands and, 63–65
 - break commands and, 73–75
 - continue command and, 73, 75
 - for loop and, 69–72
 - infinite loop and, 72
 - loops and, 65–75
 - nesting control commands, 75–77
 - overview of, 63
 - switch statement and, 77–78
 - while loop, 66–67
 - flushing buffer, 307
 - F9 (Build and Run), 18
 - for loop, 69–72
 - ForDemo program, 70
 - ForDemo2 program, 71
 - format flags for stream I/O, 305–307
 - format of output, 311–313
 - forward declaration, 203
 - friend keyword, 201–204
 - fstream object, 298
 - __func__ (intrinsic #define), 323
 - function declaration, 84
 - functional approach, 162, 164
 - FunctionDemo program
 - displayExplanation() and sumSequence() functions in, 82–83
 - overview of, 81
 - functions. *See also* constructor; virtual function
 - accessing other members from, 172–173
 - add(), 342
 - adding, 169–170
 - with arguments, 87–89
 - assert(), 157

- `bad()`, 298
 - base class, overriding, 267
 - calling, 85, 170–172, 179, 185–189
 - character string, 337
 - `clear()`, 298
 - `concatString()`, 107
 - concrete, 278
 - construct for, 86
 - constructor, 206–211
 - current object and, 172–173
 - defining inside class, 175–176
 - defining outside class, 177–178
 - defining prototypes, 91–93
 - describing, 85
 - `displayArray()`, 99, 130–131
 - `displayCharArray()`, 103
 - `displayExplanation()`, 82–84
 - `displayString()`, 104
 - `eof()`, 300, 301
 - extended function names, 90–91
 - `fail()`, 298
 - `flags()`, 305, 306
 - `gcount()`, 303
 - generalizing into template, 338–340
 - `get()`, 303, 342
 - `getline()`, 300, 303
 - inlining, 176
 - `intFn()`, 342
 - list container, 351–353
 - `main()`, 89–90, 140–143
 - making polymorphic, 269
 - `maximum()`, 337–338
 - with multiple arguments, 89
 - naming, 84
 - operators compared to, 283–284
 - outline, 177–178
 - overloading, 178–179, 265–268
 - overview of, 81, 85, 168–169
 - `parseString()`, 310–311
 - passing pointers to, 117–120, 184–189
 - positioning pointer within file, 307–308
 - `precision()`, 312
 - `put()`, 303
 - `read()`, 303–305
 - `readArray()`, 99
 - `seekg()`, 308
 - `setf()`, 305, 306
 - simple, 86–87
 - static, 251–253, 271
 - `strcpy()`, 135
 - of stream classes, 301–303
 - string container, 346–348
 - string-manipulation, 107–108
 - `sumArray()`, 99
 - `sumSequence()`, 82–85
 - `tellg()`, 307–308
 - `tellp()`, 308
 - `typeid()`, 374–375
 - `unsetf()`, 305, 306
 - variable storage types and, 93
 - `void()`, 86–87
 - wide string-handling, 109–110
 - `width()`, 306–307
 - `write()`, 303–305
- G •
- garbage collection, 367
 - gcc compiler, 142, 380
 - `gcount()` method, 303
 - `get()` method, 303, 342
 - `getline()` method, 300, 303
 - Global compiler settings dialog box, 12, 13
 - global object
 - constructing, 231–232
 - description of, 206
 - global variable
 - description of, 229
 - static data members and, 248, 249
 - greater-than (>) operator, 51
 - greater-than-or-equal-to (<=) operator, 51

• H •

handling errors. *See* exception
 HAS_A relationship, 263–264
 head pointer, 192, 251
 heap memory
 description of, 122
 destructor and, 212
 managing, 363
 pointers to objects and, 189–190
 scope and, 122–123
 scope problem and, 123–124
 solution to scope problem using, 124–125
 hexadecimal system, 57

• I •

`#if` command, 146, 154–155
`if` statement, branch statement and, 64–65
`#ifdef` conditional compilation, 155
`#ifndef` conditional compilation, 155
`ifstream` class, 295–296, 298–301
`#include` command
 defining member function in class and, 175–176
 preprocessor and, 146–149
`#include <cstring>` statement, 108
 include file `iostream`, 284, 293–294
 include statement, function prototypes and, 92–93
 increment operator (`++`)
 description of, 45
 loops and, 67–69
 index
 accessing arrays with, 133
 applying to arrays, 129
 referring to character strings by, 139
 infinite loop, 72
 inheritance. *See also* multiple inheritance
 definition of, 257–258
 example of, 259–260

 factoring and, 277
 HAS_A relationship, 263–264
 introduction of, into C++, 258
 IS_A relationship, 258, 273
 misrepresentations and, 275–276
 polymorphism and, 268
 purposes of, 273
 subclass, constructing, 262–263
 subclass, destructing, 263
 subclass, using, 261–262
 InheritanceExample program, 259–260
 initialization clause, for loop and, 69
`initializer_list<T>` template class, 368–369
 initializing
 arrays, 100
 data members inline, 369–370
 global variables, 231
 link pointers, 195
 pointer variables, 135
 variables, 32
 inline keyword, 152
 inlining member function, 176
 input/output technique, 293. *See also* stream I/O
 insertion operator (`<<`), 284, 294
 installing
 Code::Blocks environment, 11–13
 programs from CD, 378
 instance
 of class, 166
 definition of, 163
 instantiating
 extern template, 370–371
 templates, 339
`int` return type, 84
`int2Month()` program, 139
 integer overflow, 34
 integer variable type
 description of, 27
 limitations of, 27–29
 logical operators and, 53
 signed and unsigned, 32

`intFn()` function, 342
intrinsic constant, 156–158
intrinsic data type, 283
intrinsic `#define`, 323
`iostream` include file, 284, 293–294
IS_A relationship, 259, 273
`istream` class
 description of, 294
 methods of, 301–303
 read pointer, 307–308
iterate, definition of, 100
iterator, 353–354

• K •

keywords

`catch`, 315
`class`, 166
`default`, 223–224
`delete`, 124–125
`delete[]`, 190, 215
`explicit`, 234
`friend`, 201–204
`inline`, 152
`main()`, 89–90
`new`, 124
`nullptr`, 136–137, 192, 289
`operator`, 284
`private`, 199
`protected`, 197
`public`, 166, 198, 261
`sizeof`, 111
`static`, 93, 247–248
`static_assert()`, 158
`struct`, 166
`template`, 339
`throw`, 315, 321–323, 324
`try`, 315
`typedef`, 158
`using`, 148
`virtual`, 269–270, 334

• L •

lambda expression, 373–374
late binding, 267, 269
Layout program, 114
LayoutError program, 117
less-than (<) operator, 51
less-than-or-equal-to (>=) operator, 51
level of abstraction, 162
limited range
 of floating-point variable, 30
 of integer type variable, 28
limiting visibility of class internals,
 360–362
`__LINE__` (intrinsic `#define`), 323
`LinkableClass` class, 191–192
linked list
 creating, 191–192
 performing operations on, 192–193
 sample program, 193–196
`LinkedListData` program, 193–195
`LinkedListForward` program, 196
linker program, 145
linking, definition of, 10
list, linked. *See* linked list
list container
 iterating through lists, 353–354
 methods of, 351–353
 operations on lists, 354–355
 program example, 355–356
local object
 constructing, 229
 description of, 206
localization, 110
logical operators
 bitwise, 55–57
 floating-point variables and, 53–54
 int variables, 53
 overview of, 49–50
 simple, 50–51
 storing logical values, 51–52
 types of, 49
 use of, 62

logical variable, 37

loops

 autoincrement/autodecrement feature
 and, 67–69

 break commands and, 73–75

 continue command and, 73, 75

 do...while, 67

 for, 69–72

 infinite, 72

 nested, 75–77

 overview of, 65

 switch statement and, 77–78

 while, 66–67, 69

• M •

machine language, 9

machine-executable file, 10

MacroConfusion program, 150

main() function

 accessing arguments to, 140–143

 description of, 89–90

Management window, 15

manipulating string of characters,
 105–107

manipulator, 312

mathematical operations

 assignment operators, using, 46–47

 binary arithmetic, 41–42

 decomposing expressions, 43

 hexadecimal numbers and, 57

 increment operator, 45

 order of, determining, 43–44

 unary, 44–45

maximum() function, 337–338

MaxTemplate program, 338–339

member. *See also* nonmember

 complex data member, constructing,
 224–228

 constant data member, constructing,
 228–229

 constructing, 233

 protecting, 197–199

 member function. *See also* constructor;
 virtual function

 accessing other members from,
 172–173

 add(), 342

 adding, 169–170

 with arguments, 87–89

 assert(), 157

 bad(), 298

 base class, overriding, 267

 calling, 85, 170–172, 179, 185–189

 character string, 337

 clear(), 298

 concatString(), 107

 concrete, 278

 construct for, 86

 constructor, 206–211

 current object and, 172–173

 defining inside class, 175–176

 defining outside class, 177–178

 defining prototypes, 91–93

 describing, 85

 displayArray(), 99, 130–131

 displayCharArray(), 103

 displayExplanation(), 82–84

 displayString(), 104

 eof(), 300, 301

 extended function names, 90–91

 fail(), 298

 flags(), 305, 306

 gcount(), 303

 generalizing into template, 338–340

 get(), 303, 342

 getline(), 300, 303

 inlining, 176

 intFn(), 342

 list container, 351–353

 main(), 89–90, 140–143

 making polymorphic, 269

 maximum(), 337–338

 with multiple arguments, 89

 naming, 84

 operators compared to, 283–284

- outline, 177–178
 - overloading, 178–179, 265–268
 - overview of, 81, 85, 168–169
 - `parseString()`, 310–311
 - passing pointers to, 117–120, 184–189
 - positioning pointer within file, 307–308
 - `precision()`, 312
 - `put()`, 303
 - `read()`, 303–305
 - `readArray()`, 99
 - `seekg()`, 308
 - `setf()`, 305, 306
 - simple, 86–87
 - static, 251–253, 271
 - `strcpy()`, 135
 - of stream classes, 301–303
 - string container, 346–348
 - string-manipulation, 107–108
 - `sumArray()`, 99
 - `sumSequence()`, 82–85
 - `tellg()`, 307–308
 - `tellp()`, 308
 - `typeid()`, 374–375
 - `unsetf()`, 305, 306
 - variable storage types and, 93
 - `void()`, 86–87
 - wide string-handling, 109–110
 - `width()`, 306–307
 - `write()`, 303–305
 - member-by-member copy, 238–239
 - memory. *See also* heap memory
 - neighborhood model of, 128
 - overuse of common, 251
 - referencing with uninitialized pointer variable, 135
 - template classes and, 344
 - troubleshooting and, 380
 - method. *See* member function
 - Microsoft Foundation Classes, 336
 - minimizing error
 - avoiding operator overloading, 363
 - coding style and, 360
 - commenting code and, 362
 - declaring destructors virtual, 364–365
 - enabling warnings and error messages, 359–360
 - exceptions and, 363–364
 - limiting visibility of class internals, 360–362
 - managing heap systematically, 363
 - multiple inheritance and, 365
 - single-stepping program, 362–363
 - minus sign (-) operator, 23, 44
 - misrepresentations of inheritance, 275–276
 - mixed mode expression, 37–38
 - modes, binary and text, 297
 - module, definition of, 91
 - module header, standard, 360
 - modulus operator (%), 42
 - multiple inheritance
 - ambiguities in, 327–328
 - avoiding, 365
 - constructing objects of, 335
 - contrary opinions of, 335–336
 - description of, 325
 - graphical expression of, 325–326
 - program example, 326–327
 - virtual inheritance and, 328–334
 - MultipleInheritance program, 326–327
 - MultipleInheritanceFactoring program, 329–330
- N ●
- name collision, 148–149, 328
 - NameDataSet program, 254
 - NamespaceExample program, 149
 - naming
 - classes, 166
 - conventions for, 38, 360
 - current objects, 172–173
 - functions, 84
 - objects, 166
 - variables, 22

- NestedDemo program
 - functions and, 81–82
 - nested loops example, 76–77
 - nesting control commands, 75–77
 - new keyword, 124
 - newline, 21
 - newline character (\n)
 - binary and text modes and, 297
 - endl object and, 307
 - in strings, 35
 - nonmember
 - definition of, 170
 - giving access to protected members, 201–204
 - nonstatic object method, 253–254
 - NOT operator, 58, 60
 - npos constant, 350
 - null character, 103–104
 - nullptr constant
 - defining, 289
 - initializing pointers to, 137, 192
 - properties of, 136
 - null-terminated byte string, 104–105, 131
 - numeric types, 33–34
- ○ •
- .o (object) file, 146
 - object
 - abstract classes and, 279
 - accessing property of, 167
 - cerr, 294, 295
 - classes and, 167, 205
 - copying, 189, 235–238
 - creating, 205–206
 - current, 172–173, 253–254
 - declaring arrays of, 181–182
 - declaring pointers to, 182–184
 - definition of, 165
 - destructor and, 211–215
 - duplex, constructing, 209–211
 - endl object, 307
 - fstream object, 298
 - global, 206, 231–232
 - keeping count of, 250–251
 - local, 206, 229
 - multiple, constructing, 208–209
 - of multiple inheritance, constructing, 335
 - naming, 166
 - passing by reference, 266–267
 - passing to functions, 184–189
 - pointers to, 189–190
 - simulating real-world, 168
 - single, constructing, 207–208
 - SleeperSofa object, 328–334
 - standard stream I/O, 294–295
 - static, constructing, 230–231
 - temporary, and copy constructor, 244–246
 - object-oriented programming (OOP). *See also* class
 - abstraction, 161–163
 - classification, 163–164
 - inheritance and, 258
 - multiple inheritance and, 335
 - polymorphism and, 268
 - ObjPtr program, 182–183
 - octal counting system, 56, 57
 - offset, adding to pointer variable, 128, 129
 - ofstream class, 295–296
 - OOP. *See* object-oriented programming
 - opening file, 296–297
 - operator. *See also* AND operator; bitwise operator; OR operator
 - applying to address of arrays, 130–131
 - applying to pointer types, 134
 - assignment, 23, 46, 285–291
 - binary, 41–42
 - comparison, 50–51
 - definition of, 23
 - exclusive OR, 59
 - extraction, 284, 294

- functions compared to, 283–284
 - insertion, 284, 294
 - for lists, 354–355
 - logical, 49–54, 62
 - minus sign (-), 23, 44
 - modulus (%), 42
 - NOT, 58, 60
 - pointer, 113, 116
 - precedence of, 43–44
 - reference, 188–189
 - scope resolution (: :), 174
 - subscript, overloading, 291–292
 - unary, 42, 44–45
 - for user-defined class, 284
 - operator keyword, 284
 - operator overloading, 283, 284, 363
 - OR (||) operator
 - bitwise, 58
 - short-circuit evaluation and, 54–55
 - simple, 49, 51
 - truth table for, 59
 - order of operations, 43–44
 - ostream class
 - description of, 294
 - methods of, 301–303
 - outline function, 177–178
 - output
 - controlling format of, 311–313
 - generating, 22–23
 - overflow, 34
 - overflowing character buffer, 300
 - OverloadConstructor program, 220–221
 - overloading
 - assignment operators, 285–291
 - constructors, 220–222
 - function names, 90–91
 - member functions, 178–179, 265–268
 - operators, 283, 284, 363
 - pass by value functions, 120
 - subscript operators, 291–292
 - OverloadOverride program, 266–267
 - overriding base class function, 267
- p •
- parameter, 140. *See also* argument
 - pArray, charArray compared to, 134–136
 - parseString() method, 310–311
 - passing
 - abstract classes, 280
 - objects by reference, 266–267
 - objects to functions, 184–189
 - passing argument to function
 - by reference, 119–120
 - by value, 118
 - PassObjPtr program, 186–187
 - PassObjRef program, 188
 - PassObjVal program, 185
 - pNext pointer, 192
 - pointer operators, 113, 116
 - pointer variable
 - arrays and, 128–130, 134–135
 - arrays of, 137–143
 - const variable and, 120–121
 - declaring pointers to objects, 182–184
 - defining operations on, 127–128
 - description of, 111, 115–116
 - initializing, 135
 - linked list and, 191
 - nullptr constant and, 136–137
 - object and, 189–190
 - passing to functions, 117–120, 184–189
 - pNext, 192
 - positioning within files, 307–308
 - reference compared to, 191
 - referencing memory with
 - uninitialized, 135
 - smart, 367–368
 - strings and, 131–133
 - this, 254
 - types of, 116–117
 - to virtual function, 272

- polymorphism
 - definition of, 267
 - inheritance and, 268
 - support for, 269
 - virtual keyword and, 269–270
 - precedence of operators, 43–44
 - precision() member function, 312
 - preprocessor
 - #define command, 149–154
 - description of, 145–146
 - #if command, 154–155
 - #include command, 146–149
 - intrinsic constants and, 156–158
 - typedef keyword and, 158
 - PrintArgs program, 140
 - private keyword, 199
 - processing error return, 317
 - program. *See also specific programs*
 - basing on C++ statements, 21
 - building, 17
 - cheating by copying code, 16–17
 - declarations, writing, 22
 - description of, 9
 - entering code, 15–16
 - executing, 18–19
 - framework for, 19
 - output, generating, 22–23
 - project, creating, 13–14
 - promotion, 37
 - properties. *See also inheritance*
 - of nullptr constant, 136
 - of Savings class, 168
 - protected inheritance, 261
 - protected keyword, 197
 - protected member, giving nonmember
 - functions access to, 201–204
 - protecting
 - internal state of class, 200–201
 - member of class, 197–199
 - prototype declaration
 - description of, 91–93
 - include files and, 146
 - public keyword, 166, 198, 261
 - pure virtual function, 278
 - put() method, 303
- Q •**
- quotes, include file in, 147
- R •**
- read() method, 303–305
 - readability of code, 65
 - readArray() function, 99
 - reading stream directly, 303–305
 - real number, 28
 - reference
 - to nonstatic objects, 253–254
 - passing argument to functions by, 119–120
 - passing objects by, 266–267
 - pointers compared to, 191
 - rvalue, 371–372
 - reference operator, calling function
 - using, 188–189
 - referencing
 - memory with uninitialized pointer variable, 135
 - static data members, 249–250
 - rerouting standard input and standard output, 295
 - resizing buffer dynamically, 300
 - resolve, definition of, 179
 - rethrowing exception up stack chain, 324
 - return type
 - int, 84
 - void, 84, 86
 - return value to function, 86
 - returning pointer, 122
 - reverse iterator, 353
 - round-off error, 27–29, 54
 - runtime type, 267
 - rvalue references, 371–372



- saving source file, 16
- Savings class, 167–169, 273–277, 279–280
- scope
 - heap and, 122–123
 - problem with, 123–124
 - solution to problem with, 124–125
- scope resolution, 174–175
- seekg() method, 308
- segment violation error, 135
- sequence of numbers, and array, 95–97
- setf() method, 305, 306
- shallow (byte-by-byte) copy
 - assignment operator and, 285–286
 - creating, 240–244
- ShallowCopy program, 240–241
- shared_ptr<T> template class, 368
- short-circuit evaluation, 54–55
- single-bit operator, 58–59
- single-stepping program with debugger, 362–363
- size of variable, 111–113
- sizeof keyword, 111
- SleeperSofa object, 328–334
- smart pointer, 367–368
- software modification, and inheritance, 258
- source code
 - on CD-ROM, 2
 - cheating by copying from CD-ROM, 16–17
 - clarifying with comments, 20–21, 362
 - entering for project, 15–16
 - readability of, 65
 - snippets of, 4
- source file
 - description of, 9
 - saving, 16
- special characters, 35–36
- splat character, 116
- spreadsheet, and array, 101–102
- SquareDemo function example, 87–89
- stack chain, rethrowing exception up, 324
- stack unwinding, 318
- Standard Template Library (STL)
 - contents of, 337
 - description of, 146, 345–346
 - initializer_list<T> template class, 368–369
 - list container, 351–356
 - shared_ptr<T> template class, 368
 - string container, 346–351
- standards, 2, 6. *See also* C++ 2009 standard
- starting Code::Blocks, 14
- statement, definition of, 21
- static data member
 - defining, 247–248
 - description of, 247
 - referencing, 249–250
 - uses of, 248–249, 250–251
- static keyword, 93, 247–248
- static member function, declaring, 251–253, 271
- static object, constructing, 230–231
- static type, 267
- static variable, 229
- static_assert() keyword, 158
- std namespace, 149
- STL (Standard Template Library)
 - description of, 345–346
 - initializer_list<T> template class, 368–369
 - list container, 351–356
 - shared_ptr<T> template class, 368
 - string container, 346–351
- STLListStudents program, 355–356
- STLString program, 349–350
- storage type of variable, 93
- storing
 - binary numbers, 55
 - data, 345
 - logical values, 51–52

- `strcpy()` function, 135
 - stream I/O
 - classes and constants, 295–297
 - controlling format of output and, 311–313
 - description of, 293
 - `endl` object and, 307
 - format flags, 305–307
 - methods of stream classes, 301–303
 - open modes and, 297
 - positioning pointer within file, 307–308
 - prototypes appearing in, 294
 - reading and writing streams directly, 303–305
 - standard objects, 294–295
 - state information and, 298
 - `stringstream` classes and, 308–311
 - StreamInput program, 298–299
 - StreamOutput program, 296
 - string
 - character, 131–133, 138–139
 - constant address and, 135
 - expanding pointer operations to, 131–133
 - wide character, 109–110
 - zero-terminated, 105
 - string container
 - methods of, 346–348
 - program example, 349–350
 - string of characters
 - creating, 103–105
 - manipulating, 105–107
 - string type, 110, 228
 - string variable, 35
 - string-manipulation functions, 107–108
 - `stringstream` classes, 308–311
 - StringStream program, 308–310
 - strongly typed language, 26
 - `struct` keyword, 166
 - Student class
 - with constructor, 207
 - defining, 169–170
 - overloading member functions, 178–179
 - protected members and, 199
 - scope resolution and, 174–175
 - subclass
 - abstract classes and, 278
 - constructing, 262–263
 - description of, 259
 - destructing, 263
 - graphical expression of, 275
 - propagation of changes in base class to, 276
 - using, 261–262
 - subscript operator, overloading, 291–292
 - `sumArray()` function, 99
 - `sumSequence()` function, 82–85
 - switch statement, 77–78, 138
 - system requirements for CD-ROM, 377–378
- 7 •
- tab character (`\t`), 35
 - `tellg()` method, 307–308
 - `tellp()` method, 308
 - template class
 - `ifstream` class as, 295
 - `initializer_list<T>` template class, 368–369
 - `shared_ptr<T>` template class, 368
 - using, 340–344
 - template definition
 - description of, 338
 - generalizing function into, 338–340
 - tips for using, 343–344
 - template keyword, 339
 - Template.cpp file, 20
 - TemplateVector program, 340–342
 - temporary object, and copy constructor, 244–246
 - testing, and `displayArray()` function, 131
 - text mode, 297

`this` pointer, 254
thread local storage, 371
`throw` keyword, 315, 321–323, 324
tilde (~) destructor symbol, 212
transitive, inheritance property as, 259
troubleshooting CD-ROM, 380
truncation, 27–29
`try` keyword, 315
type
 assignment operator and, 290
 character, 36
 charArray, 130, 134–136
 compile-time, 267
 dynamic, 267
 numeric, 33–34
 runtime, 267
 static, 267
 string, 110, 228
 user-defined enumerated, 154
TypeConversion program, 234
typedef keyword, 158
typeid() member function, 374–375

• U •

unary operator, 42, 44–45
Unicode Transformation Format (UTF), 36
unsetf() method, 305, 306
unwinding stack, 318
user-defined class, operators for, 284
user-defined enumerated type, 154
using keyword, 148
UTF (Unicode Transformation Format), 36

• V •

value
 assignment operator and, 290
 constant, 32–33

 definition of, 22
 expressions and, 23
 passing argument to function by,
 118, 185–186
 storing logical, 51–52
variable. *See also* array; pointer variable
 address assigned to, 113, 115
 automatic declaration, 38–39
 bool, 37, 51–52
 char, 35, 36
 const, pointer variable and, 120–121
 constant, 33
 declaring, 25–26, 32
 definition of, 22, 25
 floating-point type, 28–30, 53–54
 global, 229, 248, 249
 initializing, 32
 integer type, 27–29, 32, 53
 logical, 37
 mixed mode expression and, 37–38
 naming conventions for, 38
 size of, 111–113
 special characters in, 35–36
 static and global, 229
 storage types, 93
 string, 35
 types of, 31–32
VariableSize program, 112
variadic template, 374
vector, 340
virtual destructor, 271–272, 364–365
virtual function
 abstract class and, 278
 recognizing, 270
 using, 271–272
virtual inheritance, 328–334
virtual keyword, 269–270, 334
VirtualInheritance program, 332–333
void function, 86–87
void return type, 84, 86

• *W* •

warnings, enabling, 359–360
wcerr object, 294, 295
wcin object, 294, 295
wcout object, 294, 295
weakly typed language, 26
Web sites
 of author, 10
 Code::Blocks environment, 10
 GNU gcc, 380
 Wiley Product Technical Support, 381
while loop
 description of, 66–67
 for loop compared to, 69
WhileDemo program, 66
whitespace, 21, 65
wide string-handling functions, 109–110

width() method, 306–307
wild-card expansion, 142
Wiley Product Technical Support, 381
Windows programming, 10, 19
Windows style, accessing program
 argument, 142–143
write() method, 303–305
writing stream directly, 303–305
wstring class, 351

• *X* •

XOR operator, truth table for, 59

• *Z* •

zero-terminated string, 105