

Index

• Numbers & Symbols •

- 0nn, printf() escape sequence, 307
 - 100.C, 192–193
 - & (ampersand), do-while loops, 227
 - & (AND) bitwise operator, 344
 - && logical operator, 180
 - <> (angle brackets), 13
 - * (asterisk)
 - comments and, 56
 - as multiplication sign, 88, 134, 313
 - \ (backslash)
 - \' (apostrophe), printf() escape sequence, 307
 - \" (double-quote), printf() escape sequence, 307
 - \0 (null), printf() escape sequence, 307
 - \? (question mark), printf() escape sequence, 307
 - escape sequences, 45
 - printf() escape sequence, 307
 - RULES.C, 37–38
 - text strings, 31
 - [] (brackets), single-character variables, 123
 - ^ (caret) symbol, 316
 - ^ (EOR/XOR) bitwise operator, 344
 - % conversion character, printf() function, 305
 - { } (curly braces)
 - comparisons and, 153
 - else keyword, 159
 - functions and, 31
 - introduction, 13
 - = (equal sign)
 - location, 314
 - numeric variables and, 80
 - == (equal to) comparison operator, if statement, 151, 160
 - (hyphen)
 - one's complement, bitwise operator, 344
 - subtraction symbol, 87, 134, 313
 - (decrementation) operator, 207–208, 322
 - > (greater than) comparison operator, if statement, 152, 160
 - >= (greater than or equal to) comparison operator, if statement, 152, 160
 - | (inclusive OR) bitwise operator, 344
 - || logical operator, 178, 180
 - < (less than) comparison operator, if statement, 151, 160
 - <= (less than or equal to) comparison operator, if statement, 152, 160
 - % (modulus)
 - introduction, 333
 - math operator, 314
 - != (not equal) comparison operator, if statement, 152, 160
 - + (plus symbol)
 - mathematical operator, 134
 - symbol, 87, 313
 - ++ (incrementation) operator
 - introduction, 202–203
 - LARD0.C, 203–204
 - location, 322–323
 - variables, 320–321
 - #define directive, 104–105, 302–303
 - #else, 303
 - #endif, 303
 - #if, 303
 - #ifdef, 303
 - #ifndef, 303
 - #include
 - construction, 294–297
 - description, 30
 - << (shift bits left) bitwise operator, 344
 - >> (shift bits right) bitwise operator, 344
 - / (slash)
 - with asterisk (/*), 56
 - division symbol, 87, 134, 313
 - double (/), 60, 63
 - // (double slash)
 - C++ comments, 60
 - nested comments and, 63
- A •
- \a, printf() escape sequence, 306
 - abs() function, 319
 - absolute value of numbers, 320
 - acos() function, 319
 - addition symbol (+), 87

alphabet trivia, 172
 ampersand (&)
 do-while loops, 227
 pointers, 343
 AND (&) bitwise operator, 344
 AND logical operator
 code example, 183
 introduction, 180
 angle brackets (< >), 13
 arguments, 277, 282
 arrays, 339–340, 341
 ASCII characters
 character variables, 129
 extended codes, 129
 table, 371–375
 typing in, 122
 ASCII.C, 193–194
 asin() function, 319
 ASSESSED.C, 140–141
 assigning pointers, 343
 assignment operators, 212
 asterisk (*), comments and, 56
 atan() function, 319
 atoi() function
 HEIGHT.C, 136
 introduction, 81–82
 returning values, 283

• B •

\b, printf() escape sequence, 306
 B programming language, 10
 backslash (\)
 escape sequences, 45
 printf(), 305
 RULES.C, 37–38
 text strings, 31
 backward counting loops, 205
 BASIC programming language, 10
 BCPL (Basic Combined Programming Language), 10
 BIGJERK1.C, 255–256
 BIGJERK2.C, 256–260
 BIGJERK3.C, 261–262
 BIGJERK4.C, 277–278
 binary numbers, integers and, 112
 binary operators, 344
 bitwise operators, 182, 344
 blank lines in code, 14
 BLOWUP1.C
 if command, 173–174
 logic, 176

BOMBER.C
 dual variables, 267
 global variable, 271–272
 variables, 265–267
 bonus() function, 288–289
 BONUS.C, 288–289
 books for further reading, 358
 BORC.C, 236
 bounds checking, do-while loops, 229–230
 brackets ([]), single-character
 variables, 123
 break command, 244, 352
 break keyword, 198–199
 for (; ;) loops, 237
 case statements, 243
 do-while loops, 228
 nested loops, 235–237
 break statements, while loops, 221
 breaking loops, 197–198
 breakpoints, 356
 bugs, 27
 bulletproofing, 229
 bytes, 128

• C •

%c conversion character, printf()
 function, 311
 calling functions, 254, 279–280
 caret (^) symbol, 316
 case keyword, 244, 247
 case sensitivity
 else keyword, 159
 function naming, 264
 include directives, 296
 keywords, 33
 printf(), 42
 source code, 13
 case statement, 243, 244, 247
 cat command, 351
 cd command, 351
 char keyword
 introduction, 50
 numeric data types, 108
 single-character variables, 122–123
 string variable declaration, 57
 unsigned char keyword, 109
 variable declaration, 40, 123–124
 character data types, 108
 character variables
 char keyword, 121
 characters in, 124

- as integer, 111
- quotes, 123
- value assignment, 124
- as values, 128–129
- characters
 - comparing, 166
 - conversion, 46
- clear command, 350
- cls command, 350
- code. *See* source code
- code blocks, 151
- code size, 346
- COLOR.C, 51–52
- command line, 345, 347
- command prompt
 - commands, 350–351
 - IDE and, 350
- commands
 - cat, 351
 - cd, 351
 - clear, 351
 - cls, 351
 - command prompt, 350–351
 - del, 351
 - dir, 351
 - exit, 351
 - if, 147–148
 - line numbers, text editor, 349
 - ls, 351
 - nv, 351
 - pwd, 351
 - ren, 351
 - return, 31
 - rm, 351
 - switch, 243
 - type, 351
- comments
 - /* (slash with asterisk), 56
 - C++, 60
 - compiler and, 55
 - disabling statements, 61
 - introduction, 55
 - MADLIB1.C, 56–57
 - nested, 62–63
 - reasons for, 58
 - single-line, 59
 - styles, 58–60
 - tips, 356–357
 - variables, 95
- comparisons
 - { } (curly braces) and, 153
 - characters, 166
 - else keyword and, 159
 - GREATER.C, 167–168
 - if keyword, 150–151
 - operators, 151–152
 - strings, if keyword and, 174
- compiler
 - comments and, 55
 - errors, 27
 - FreeBSD, 360
 - GCC compiler, 15, 360, 365–367
 - GOODBYE.C, 15–16
 - header files and, 296
 - introduction, 14–15
 - Linux, 360
 - Mac, 361
 - Mac OS X, 360
 - MiniGW compiler, 360
 - MSVC (Microsoft Visual C++), 360
 - setup, 359
 - Unix, 361
 - Windows, 360
- compiling
 - FreeBSD, 368–370
 - linking and, 17
 - Linux, 368–370
 - Mac OS X, 368–370
 - recompiling, 21–22
 - variable declaration and, 95
 - WHORU.C, 40–41
- conditions
 - do-while loops, 227
 - infinite loops, 196
- const keyword, 106
- constants
 - defining, 101–102
 - definition, 91
 - numeric, 101
 - numeric, shortcut, 102–104
 - semicolons, 269
 - string constants, 53, 101
 - symbolic, 103
 - variables and, 101
- contents of variables, 76
- context-colored text editors, 348–349
- continue keyword
 - loops, 237–238
 - nested loops, 235–236
- conversion characters
 - formatting strings, 46
 - printf() function, 311–312
- converting, string numbers to integer value, 81–82

`cos()` function, 319
`COUNTDOWN.C`
 do-while loops, 226–227
 error, 228–229
 critical errors, 25
 curly braces (`{ }`)
 case keyword statements, 244
 comparisons and, 153
 else keyword, 159
 functions and, 31
 introduction, 13

• D •

`%d` conversion character, `printf()`
 function, 311
 data types, numeric, 108
`DBLQUOTE.C`, 43
`dead_horse` variable, 223
 debugging
 order, 354
 tools, 357
 declaring arrays, 340
 declaring functions, 263
 declaring pointers, 343
 declaring variables
 float, 113
 global, 270–271
 integer variables, 110–111
 introduction, 40
 location in program, 95
 multiple, 100–101
 reasons for, 94–95
 values as, 276
`DECODD.C`, 322
 decrementation. *See also* incrementation
 `--` operator, 207–208, 322
 assignment operators, 212
 for loops, 206–207
 introduction, 204–205
 operator shortcuts, 212
 skipping values, 210
 default statements, `switch` structure, 244
 defining functions, 263
`del` command, 351
`delay()` function, prototyping, 269
 delay loops, 233
 development cycle, 11
`dir` command, 350
 disabling statements, comments and, 61
 disk access, 345
 displaying text, `printf()` and, 306
 division symbol (`/`), 87

do keyword, 227
 dot notation, structures, 342
`do_this` statement, for loop, 189
 double keyword, numeric data types, 109
 double quotes. *See also* quotes
 `DBLQUOTE.C`, 42–43
 formatting strings, 46
 strings, 42–43
 double slash (`//`), 60
 double variables
 double precision numbers, 118
 `pow()` function, 316
 double-precision data types, 109, 118
 do-while loops
 & (ampersand), 227
 bounds checking, 229–230
 break keyword, 223
 conditions, 227
 execution, 227
 input bounds, 229–230
 introduction, 186
 number checking, 229–231
 semicolons, 227
 statements, 227
`dropBomb()` function, 265–266
 dual variables, `BOMBER.C`, 267

• E •

`%e` conversion character, `printf()`
 function, 311
 E notation, 116–117
 editing, source code, 19–21, 24–25
 elements in arrays, 340
 else keyword
 `{ }` (curly braces), 159
 case sensitivity, 159
 if statements and, 158
 semicolon, 159
 else statement, 157–158
 else-if statement, 160–163
 empty parentheses, 31
 endless loops, 186
 EOF (end of file), 170
 EOR (`&`) bitwise operator, 344
 equal sign (`=`)
 location, 314
 numeric variables and, 80
 equal to (`==`) comparison operator, if
 statement, 151
 error messages
 components of, 23
 GCC compiler, 24

- line numbers, 24
 - linker and, 26
 - semicolons, 24
 - ERROR.C, 22–27
 - errors
 - bugs, 27
 - compiler errors, 27
 - critical errors, 25
 - fatal errors, 25
 - linker and, 26
 - linker errors, 27
 - null pointer assignment, 27
 - parse errors, 23
 - source code, 22–27
 - syntax errors, 23
 - escape clause, loops, 197
 - escape sequences, printf(), 44–45, 306–308
 - execute
 - definition, 160
 - do-while loops, 227
 - exit command, 351
 - exp() function, 319
 - exponents, math operations, 314–315
 - extended ASCII codes, 129
- F •**
- %f conversion character, printf()
 - function, 311
 - \f, printf() escape sequence, 306
 - fatal errors, 25
 - fflush() function, 170–171
 - file size, 346
 - filenames, extensions, 13
 - files
 - folders, 361–362
 - header files, # include and, 294–297
 - source code, 12
 - text, size, 262
 - flexibility of C, 222–223
 - float keyword
 - format, 113
 - numeric data types, 109
 - float variable, declaring, 113
 - floating-point values
 - double keyword, 118
 - formatting, 119–120
 - JUPITER.C, 114–115
 - negative, 112
 - numeric data types and, 108
 - positive, 112
 - ranges, 114
 - variables, 99
 - folders, 361–364
 - for keyword
 - description, 188
 - parentheses, 189, 190
 - for loops
 - decrementing, 206–207
 - do_this statement, 189
 - introduction, 186
 - nested, 233
 - printf() statement, 188
 - variables, 191
 - while loop comparison, 219–220
 - while_true keyword, 189
 - for (; ;) loops
 - break keyword, 237
 - while loops and, 220–222
 - format strings, printf() function, 310
 - formats
 - char variable, 122
 - E notation, 117
 - floating-point values, 119–120
 - functions, 262–263, 289–291
 - if statement, 154–155
 - printf() and, 46, 310–311
 - scanf(), 49
 - strings, 46
 - text, 47–49
 - fpurge() function, 171
 - FreeBSD
 - compiler, 360
 - folders, 362
 - functions
 - { } (curly braces) and, 31
 - abs(), 319
 - acos(), 319
 - arguments, 46, 282
 - asin(), 319
 - atan(), 319
 - atoi(), 81–82, 136
 - bonus(), 288–289
 - calling, 254
 - calling, variable names and, 279–280
 - case sensitivity, 264
 - conversion characters, 46
 - cos(), 319
 - creating, 254
 - declaring, 263
 - defining, 263
 - defining, returning values and, 282
 - delay(), 269
 - dropBomb(), 265–266
 - exp(), 319
 - fflush(), 170–171
 - formats, 262–263, 289–291

functions (*continued*)

- formatting strings, 46
- fpurge(), 171
- getchar(), 126
- gets(), 65–67
- getval(), 284
- header files and, 296
- input requirements, 275
- introduction, 30
- jerk(), 256–258
- library, 255
- log(), 319
- log10(), 319
- main(), 30
- math, 319–320
- mathematical operators, 134
- naming, 263–264
- necessity, 253
- output, 275–276
- parameters, 279
- parentheses, 254, 262
- pow(), 315
- printf(), 305–312
- procedures and, 253
- prototyping, 258–262
- putchar(), 127–128
- puts(), 67–71
- rand(), 326–328
- redundancy and, 256
- return keyword, 285–287
- scanf(), 40
- seedrnd(), 329–331
- sending values to, 276–277
- sin(), 319
- sleep(), 234
- sqrt(), 317–319
- srand(), 329
- strings and, 42
- tan(), 319
- time(), 332
- types, 275–276
- values, declaring as variables, 276
- values, passing, 279
- values, passing multiple, 280–282
- values, returning, 255, 282–289
- variable naming and, 96
- further reading, 358

• G •

- %g conversion character, printf() function, 311
- gcc command, math library links, 317

GCC compiler

- error messages, 24
- introduction, 15
- Windows, 365–367
- GENIE1.C, 148–150
- GENIE2.C, 162–163
- GENIE3.C, 163–164
- getchar() function
 - reading text from keyboard, 126
 - returning values, 283
 - returns, 168
 - single character reading, 171
 - standard input and, 168
- gets() function
 - INSULT1.C, 66
 - introduction, 65–66
- getval() function, 284
- global variables
 - declaring, 270–271
 - description, 269
- GOODBYE.C
 - compiling, 15–16
 - creating, 13–14
 - recompiling, 21–22
 - running program, 16
 - typing tips, 14
- goto keyword, loops, 186
- greater than (>) comparison operator,
 - if statement, 152
- greater than or equal to (>=) comparison operator, if statement, 152
- GREATER.C
 - comparisons, 167–168
 - standard input reading, 170
- GRID.C, 234–235

• H •

- .H file extension, 296
- header files
 - compiler and, 296
 - functions and, 296
 - #include construction and, 294–297
 - library files and, 300
 - programming code, 300
 - STDIO.H, 297–298
 - writing, 298–299
- HEAD.H, 298–299
- HEIGHT.C, 135–136
- HEY.C, 216–217
- high-level languages, 10
- history, command line, 347
- history of C, 9–11

• 1 •

`%i` conversion character, `printf()` function, 311

ICKYGU.C, 98–99

IDE (Integrated Development Environment), command prompt window, 350

`if` command. *See* `if` keyword

`if` keyword

- BLOWUP1.C, 173–174
- comparisons and, 148, 150–151
- introduction, 147–148
- logical operators, 180–182
- math and, 148
- operators, 151–152
- parentheses, 150
- selection statements, 148
- string comparison, 174
- TAXES.C, 155–157
- test, 150
- values and, 165

`if` statement

- block, 150–151
- format, 154–155

`if-else` structure

- definition, 158
- either-or decisions, 158–159

`if-then` statement, 154

imaginary number, 319

INCODD.C, 321

incrementation. *See also* decrementation

- `++` operator, 202–203
- `a++`, 320–321
- ASSESSED.C, 140–141
- assignment operators, 212
- five count, 211
- for loops and, 188
- introduction, 137
- LARDO.C, 138–140
- loops and, 201–202, 209
- operator location, 322–323
- operator shortcuts, 212
- post-incrementing, 322–323, 351
- pre-incrementing, 322–323, 351
- skipping values, 210
- values, 138–139

indents, source code, 14

infinite loops

- conditions, 196
- FOREVER.C, 195
- while loops, 217

input

- functions, 275
- `getchar()` and, 168
- GREATER.C, 170
- STDIO.H, 297–298

input bounds, `do-while` loop number checking, 229–230

INSULT1.C

- `gets()` function and, 66
- `puts()` function, 68–69

INSULT2.C, 69–70

`int` data types, 108

`int` integer, 110–111

`int` keyword

- `main()` function and, 79
- numeric data types, 108
- placement in program, 78
- range, 78
- unsigned `int` keyword, 109

`int` main, 31

integers

- binary numbers and, 112
- floating-point number format, 119–120
- versus floating-point numbers, 110
- `int`, 110–111
- introduction, 78–79
- long `int`, 110–111
- METHUS1.C, 79–80
- negative numbers, 111
- reasons to use, 110
- types, 110–111
- value conversion from string numbers, 81–82
- value range, 78
- variables, declaring, 110–111

IQ.C

- `getval()` function, 284–285
- type-casting and, 286

• J •

`jerk()` function, 256–258, 278–279

JUPITER.C, 114–115

justified text, 47–49

JUSTIFY.C, 47–49

• K •

keyboard

- numeric values, 81
- reading text from, 125–126
- reading text to, 127–128

keyboard overflow, 67

keywords

break, 198–199

break, nested loops and, 235–237

case, 244

case sensitivity, 33

char, 50, 108

const, 106

continue, 235–236, 237–238

do, 227

double, 109

float, 109, 113

goto, 186

if, 148

int, 78, 108

introduction, 32

long, 109

return, 285–287

short, 108

short int, 108

struct, 341–342

switch, 243–244

unsigned char, 109

unsigned int, 109

unsigned long, 109

unsigned short, 109

variable naming and, 96

while, 218

kitty variable, 76–77

● L ●

languages, high-level, 10

LARDO.C

++ operator, 203–204

incrementation, 138–140

leaping loops, 210

Learn directory/folder, 363–364

less than (<) comparison operator, if statement, 151

less than or equal to (<=) comparison operator, if statement, 152

libm library, 319

library files

header files and, 300

libm, 319

math library links, 317

library of functions, 255

LIGHTS1.C, 316–317

LIGHTS2.C, 318–319

line numbers

commands, text editor, 349

error messages, 24

linked lists, 343

linker

errors and, 26, 27

introduction, 15

linking

compiling and, 17

FreeBSD, 368–370

Linux, 368–370

Mac OS X, 368–370

math library, 317

Linux

compiler, 360

folders, 362

lists, linked, 343

literal strings, 53

LOBBY1.C, 240–243

LOBBY2.C, 248–250

local variables, 270

log() function, 319

log10() function, 319

logic

if command and, 180–182

introduction, 175–176

logical operators

!, 180

||, 178, 180

AND, 180

if command, 180–182

OR, 178

long int integer, 110–111

long keyword

numeric data types, 109

unsigned long keyword, 109

loops. *See also* for loops

100.C, 192–193

;; in for loops, 220–222

backward counting, 205

break keyword, 198–199, 235–236, 352

conditions, 196

continue keyword, 235–236, 237–238

delay loops, 233

description, 185

do-while, 186

endless loops, 186

escape clause, 197

for loops, 186, 188–189

goto keyword, 186

incrementation, 201–202, 209

infinite loops, 195–196

leaping loops, 210

looping statements, 185

nested, 231–235

OUCH.C, 187–188

stopping, 193–195, 197–198

switch-case, 239–247

variable initialization, 217
 while, 186, 215–216
 lowercase text, 13, 33
 low-level programming languages, reasons
 to use, 10
 ls command, 350
 Lvalue errors, math, 314

• M •

Mac
 compiler, 361
 folders, 362
 Mac OS X
 compiler, 360
 folders, 362
 machine language, 10
 macros, 303–304
 MADLIB1.C
 comment styles, 58–59
 comments and, 56–57
 magic pellets problem, order of
 precedence, 144–145
 main() function
 int keyword, 79
 introduction, 30
 returning values and, 287–288
 math
 exponents, 314–315
 functions, 319–320
 if command and, 148
 imaginary number, 319
 incrementation, 137–139
 Lvalue errors, 314
 order of precedence, 314
 pow() function, 315
 square root operations, 314, 317–319
 math library, links, 317
 mathematical operators, 86–88
 + (addition), 87, 134
 / (division), 87, 134
 * (multiplication), 87, 134
 order of precedence, 141–146
 shortcuts, 212
 - (subtraction), 87, 134
 values, 134
 variables, 134
 MATH.H header, pow() function, 315
 MDAS mnemonic, 142–143
 METHUS1.C, 79–80
 METHUS2.C, 83–85
 METHUS3.C, 85–86
 METHUS4.C, 88–90

METHUS5.C, 90–92
 MiniGW compiler, 360
 mnemonic for order of precedence,
 142–143, 335
 modulus (%)
 introduction, 333
 math operator, 314
 MSVC (Microsoft Visual C++) compiler, 360
 multiplication symbol (*), 87
 My Dear Mother's Aunt Sally mnemonic,
 142–143

• N •

\n (newline character)
 printf() escape sequence, 306
 RULES.C, 36–37
 naming
 functions, 263–264
 variables, 95
 variables, calling functions and, 279–280
 variables, guidelines, 95–96
 variables, tips for, 351
 negative numbers
 E notation, 117
 floating-point, 112
 integers, 111
 numeric data types, 111–113
 nested comments, problems with, 62–63
 nested loops
 break keyword, 235–237
 continue keyword, 235–236, 237–238
 definition, 231
 for loops, 233
 GRID.C, 234–235
 while loops, 233
 newline character, 31, 71
 Onn, printf() escape sequence, 307
 not equal to (!=) comparison operator,
 if statement, 152
 NULL character, strings, 341
 null pointer assignment error, 27
 numbers
 absolute value, 320
 ASCII characters, 122
 checking in do-while loops, 229–231
 converting string to integer values, 81–82
 floating-point *versus* integers, 110
 precision, 118
 random, 325–326
 scientific notation, 115
 strings and, 82
 variable naming and, 96

numeric constants
 description, 101
 shortcuts, 102–104

numeric data types
 character types, 108
 double-precision types, 109
 integer types, 108, 109
 introduction, 107–108
 negative numbers, 111–113
 positive numbers, 111–113
 ranges, 108–109
 short integer types, 108
 signed, 111–113
 single-precision types, 109
 unsigned, 111–113
 unsigned character types, 109
 unsigned integer types, 109
 variables, 108–109

numeric variables
 description, 93
 integers, 78–79
 introduction, 77–78
 value assignment, 80–81, 97

nv command, 350

• 0 •

%o conversion character, `printf()`
 function, 311

object code files, OBJ extension, 15

OLLYOLLY.C, 204

operating system interaction, 345

operators
 -- (decrement), 207–208, 322
 ++ (incrementation), 202–203
 binary, 344
 bitwise, 344
 comparison operators, 151–152
 comparison operators with `if`
 statement, 160
 description, 34
`if` comparisons, 151–152
 logical, 178
 mathematical, + (addition), 134
 mathematical, / (division), 134
 mathematical, * (multiplication), 134
 mathematical, - (subtraction), 134
 shortcut for math, 212

optimizers, 357–358

OR (&) bitwise operator, 344

OR logical operator
 examples, 181
`if` command and, 178

order of precedence
 DENTIST.C, 141
 introduction, 141
 magic pellets problem, 144–145
 mnemonic, 142–143, 337
 parentheses, 144, 145–146
 PELLETS.C, 144–145

OS (operating system), 27

OUCH.C, 187–188

output
 functions, 275–276
 STDIO.H, 297–298

• p •

PacMan example of variables, 76

parameters, functions, 279

parentheses
 empty, 31
 for keyword, 189, 190
 functions, 254, 262
`if` command, 150
 order of precedence and, 144, 145–146
 source code, 13
 statements, 34
 strings, 31, 42
 switch-case loops, 247

parse errors, 23

passing values, to functions, 279–282

PELLETS.C, 144–145

pipe character, 178–179

pointers, 343

positive numbers
 floating-point, 112
 numeric data types, 111–113

post-incrementing, 322–323, 351

`pow()` function
 double variables, 316
 introduction, 315
 MATH.H header, 315

pre-incrementing, 322–323, 351

precedence. *See* order of precedence

precision, numbers, 118

preprocessor directives, 30

`printf()`
 backslash, 305
 conversion characters, 311
 description, 31
 escape sequences, 44–45, 306–308
 flags, 312
 for loops, 188
 format, 42, 46, 310–311

- format strings, 310
- input-size specifiers, 312
- newline character and, 71
- precision specifiers, 312
- puts() and, 71
- review, 305
- special characters, 306
- text display, 306
- variable contents, 71
- variable display, 305
- variables, 46
- width specifiers, 312
- PRINTFUN.C, 307–309
- printing
 - text, 42–44
 - variables, 70
- procedures, functions and, 253
- program size, 346
- programmer, 11
- programming
 - breaking up code, 354
 - breakpoints, 356
 - talking through program, 355
- programming code. *See* source code
- programming languages
 - B, 10
 - BASIC, 10
 - low-level, 10
 - machine language, 10
- programs
 - running, 17
 - saving to files, 361–362
 - troubleshooting, 353–358
- prototyping functions
 - BIGJERK3.C, 261–262
 - delay() function, 269
 - introduction, 258–259
 - semicolons, 260
 - value returning and, 283
- pseudorandom numbers, 328
- putchar() function, reading text to
 - keyboard, 127–128
- puts() function
 - INSULT1.C, 68–69
 - INSULT2.C, 69–70
 - introduction, 67
 - printf() function and, 71
 - STOP.C, 68
 - string variables, 71
 - text, 67, 71
 - variable printing, 70
- pwd command, 351

• Q •

- quotes. *See also* double quotes
 - char variable, 123
 - formatting strings, 46
 - strings, 42–43

• R •

- \r, printf() escape sequence, 306
- rand() function
 - introduction, 326–328
 - seeds, 328–329
- RAND_MAX, value, displaying, 330
- random numbers
 - introduction, 325–326
 - pseudorandom numbers, 326
 - seeding, 328–329
- random-sampler variable program, 98–99
- RANDOM1.C, 327–328
- RANDOM2.C, 329–330
- RANDOM3.C, 331–333
- RANDOM4.C, 335–336
- ranges
 - floating-point numbers, 114
 - int integer, 111
 - long int integer, 111
 - numeric data types, 108–109
- reading text
 - from keyboard, 125–126
 - to keyboard, 127–128
- recompiling source code, 21–22
- ren command, 350
- reserved words, 33. *See also* keywords
- resources, 358
- return command, 31
- return keyword, 285–287
- return statements, BONUS.C, 288–289
- returning values
 - atoi() function, 283
 - functions, 255, 282–289
 - getchar() function, 283
 - main() function and, 287–288
- Ritchie, Dennis, 11
- rm command, 351
- RULES.C, 36–37

• S •

- %s conversion character, printf()
 - function, 311
- saving, source code, 16

- scanf() function
 - COLOR.C and, 51–52
 - format, 49
 - introduction, 40
 - null pointer assignment errors and, 51
 - reading text from keyboard, 125–128
 - string variables, 49
- scientific notation
 - E notation, 116–117
 - introduction, 115
- seeding random numbers, 326, 328–329
- seedrnd() function, 329–331
- selection statements
 - if keyword and, 148
 - switch-case loops, 241
- semicolons
 - case statement, 247
 - compilers and, 31
 - constants, 269
 - do-while loops, 227
 - else keyword, 159
 - errors and, 24
 - gets(), 66
 - if test, 150
 - prototypes, 260
 - variable declaration, 276
 - while keyword, 219
- shell command, 345
- short int keyword, numeric data types, 108
- short keyword
 - numeric data types, 108
 - unsigned short keyword, 109
- shortcuts for math operations, 212
- signed numeric data types, 111–113
- sin() function, 319
- single quotes, char variable, 123
- single-character variables, 122–123
- single-line comments, 59
- single-precision data types, 109
- slash (/)
 - with asterisk (/*), 56
 - double (//), 60
- sleep() function, 234
- source code
 - blank lines, 14
 - case sensitivity, 13
 - coding programs, 12
 - compiler and, 14–15
 - editing, 19–21, 24–25
 - errors, 22–27
 - filename extensions, 13
 - files, 12
 - header files, 300
 - indents, 14
 - linker, 15
 - parentheses, 13
 - recompiling, 21–22
 - saving, 16
 - text editors, 12
 - tweaking, 20
 - twiddling, 20
 - typing tips, 14
- special characters, printf() function, 306
- SPEED.C, 101–104
- split lines, 37–38
- sqrt() function, 317–319
- square root math operations
 - introduction, 314
 - sqrt() function, 317–319
- srand() function, 329
- standard input, getchar() and, 168
- statements
 - case, 243
 - case keyword, 244
 - description, 34
 - disabling with comments, 61
 - do-while loops, 227
 - else, 157–158
 - else-if, 160–163
 - if block, 150–151
 - if-then, 154
 - selection statements, 148
 - while keyword, 218
- STDIO.H, 31, 297–298
- STDLIB.H, 83
- STOP.C, 68
- string constants
 - #define directive and, 105
 - description, 53, 101
- string variables
 - description, 94
 - scanf() and, 49, 57
- strings
 - comparing with if keyword, 174
 - description, 32, 340–341
 - formatting strings, 46
 - functions and, 42
 - gets() function, 66–67
 - literal strings, 53
 - NULL character, 341
 - number conversion to integer value, 81–82
 - numbers and, 82
 - parentheses, 31, 42
 - printing, 42–44
 - struct keyword, 341–342

structures, 341–342
 styles of comments, 58–60
 subtraction symbol (-), 87
 switch command, 243, 247
 switch keyword, 243–244
 switch-case loops
 break command, 244–245
 case keyword, 244
 case statements, 244
 default statements, 244
 introduction, 239
 LOBBY1.C, 241–243
 parentheses, 247
 selection statements, 241
 while loops and, 248–250
 symbolic constants, 103
 symbols, mathematical, 86–88
 syntax, 24
 syntax errors, 23

• T •

\t, printf() escape sequence, 307
 talking through program, 355
 tan() function, 319
 TAXES.C, 155–157
 text
 display, printf(), 306
 formatting, 47–49
 justified, 47–49
 lowercase, 13
 printing, 42–44
 puts() function, 67, 71
 reading from keyboard, 125–126
 reading to keyboard, 127–128
 strings, 32
 text editors
 context-colored, 348–349
 line-number commands, 349
 running, 364–365
 source code, 12
 windows, 348
 text files
 size, 262
 source code, 12
 text strings, 31
 time() function, 332
 tweaking source code, 20
 twiddling source code, 20
 type command, 351
 TYPER1.C, 197–198
 TYPER2.C, 220–222
 typing, source code, 14

• U •

%u conversion character, printf()
 function, 311
 underline, variable naming and, 96
 Unix, compiler, 361
 unsigned char keyword, numeric data
 types, 109
 unsigned character data types, 109
 unsigned int keyword, numeric data
 types, 109
 unsigned integer data types, 109
 unsigned long keyword, numeric data
 types, 109
 unsigned numeric data types, 111–113
 unsigned short keyword, numeric data
 types, 109

• V •

\v, printf() escape sequence, 307
 value
 absolute, 320
 arrays, 340
 constants, 91
 declaring as variables, 276
 floating-point, 99
 functions, returning, 282–289
 functions, sending to, 276–277
 if keyword, 165
 incrementation, 138
 keyboard entry, 81
 mathematical operators, 134
 numbers and, 82
 numeric variables and, 80–81, 97
 parameters, 279
 passing multiple to functions, 280–282
 passing to functions, 279
 predefining in variables, 124
 return keyword, 285–287
 returning from functions, 255
 returning, main() function and, 287–288
 variables, 96–98
 variables, char, 124
 variables
 ++ operator, 321
 arrays, 339–340
 BOMBER.C, 265–269
 char keyword and, 40, 123–124
 character, comparing, 166
 comments, 95
 constants and, 101
 contents, 76

variables (*continued*)

- dead_horse, 223
- declarations, 40
- declarations, multiple, 100–101
- declarations, reasons for, 94–95
- declaring values as, 276
- description, 75–76
- double, pow() function, 316
- dual, BOMBER.C, 267
- float variable, 113
- floating-point values, 99
- for loop, 191
- global, 269
- global, declaring, 270–271
- initializing, loops and, 217
- integer, declaring, 110–111
- kitty (string variable), 76–77
- local, 270
- mathematical operators, 134
- METHUS1.C, 79–80
- names, calling functions and, 279–280
- naming, 95
- naming guidelines, 95–96
- naming tips, 351
- numeric, 77–86, 93
- numeric data types, 109
- PacMan example, 76
- printf() function, 46, 71, 305
- printing, 70
- programming tips, 356
- puts() function and, 70
- random-sample program, 98–99
- scanf() and, 49
- single-character, 122–123
- storage location, 76
- string, 94
- string, scanf() and, 49
- value assignment, 80–81
- value assignment, char, 124
- value incrementation, 138
- value, predefining, 124
- values, 96–98
- vertical bar (pipe character), 178
- vim text editor
 - context-colored, 348–349
 - line-number commands, 349
- viruses, keyboard overflow and, 67

• W •

- WHICH.C, 129
- while keyword, statements, 218
- while loops
 - break statement, 221
 - controls, 216
 - do-while loops, 225–231
 - for loop comparison, 219–220
 - for (; ;) loops and, 220–222
 - infinite loops, 217
 - introduction, 186
 - nested, 233
 - overview, 215–216
 - repeat rate, 216
 - switch-case structure and, 248–250
- while true condition, 218–219
- while_true statement
 - for loops, 189
 - while loop, 218
- WHORU.C
 - compiling, 40–41
 - I/C and, 39
- Windows
 - compiling, 367–368
 - folders, 361–362
 - GCC compiler, 365–367
 - linking, 367–368
- windows, editor view, 348
- Windows Notepad, filename extensions, 13
- worms, keyboard overflow and, 67
- writing, header files, 298–299

• X •

- %x conversion character, printf()
 - function, 311
- xnnn, printf() escape sequence, 307
- XOR (&) bitwise operator, 344

• Z •

- zeroes, floating-point number format,
 - 119–120