

Index

Symbols

`*/` character, 35–36
`/*` character, 35–36
`->` operator, 316
`+` (addition) operator, 61–63, 154
`+=` (addition) operator, 159
`&` (ampersand) operator, 93
`&` (AND) operator, 256, 259
`&&` (and) operator, 107–108
`*` (asterisk), pointers and, 281, 282–283, 406
`--` (decrement) operator, 154–157
`/` (division) operator, 61–63, 154
`/=` (division) operator, 159
`//` (double-slash), 36
`=` (equal) sign, 33, 102–103, 154
`>` (greater than) operator, 100
`>=` (greater than or equal to) operator, 100
`++` (increment) operator, 154–157
`==` (is equal to) operator, 100–102, 102–103
`<` (less than) operator, 100
`<=` (less than or equal to) operator, 100
`-` (minus) sign, 33
`%` (modulo) operator, 154, 158
`%=` (modulo) operator, 159
`*` (multiplication) operator, 61–63, 154

`*=` (multiplication) operator, 159
`!` (not) operator, 107–108, 259
`!=` (not equal to) operator, 100
`~` (ones' complement) operator, 259
`|` (OR) operator, 253–256, 259
`||` (or) operator, 107–108
`+` (plus) sign, 33
`-` (subtraction) operator, 61–63, 154
`-=` (subtraction) operator, 159
`?:` (ternary) operator, 112–113
`-` (unary minus) operator, 154
`+` (unary plus) operator, 154

A

abs() function, 161
adding
 functions, 40–41
 logical operators, 107–108
 multiple *for* loop conditions, 129
 return statement, 39
 text, 47–48
addition (`+`) operator, 154
addition (`+=`) operator, 159
address() function, 303
addresses, getting for arrays, 285–287
Algol, 22

aligning output, 206–207
allocating space for structures, 314–316
ampersand (`&`) operator, 93
analytical engine, 21–22
and (`&&`) operator, 107–108
AND (`&`) operator, 256, 259
appending text to files, 345–346
arguments
 defined, 32
 for *main()* function, 225–229
 passing to functions, 140–143
arithmetic, 61–63
arrayinc() function, 191
arrays
 about, 89, 173–176
 avoiding, 173–174
 building of pointers, 296–300
 character, 177–179
 creating from structures, 216–217
 declaring multidimensional, 188–189
 empty *char*, 179–180
 functions and, 189–191
 getting addresses for, 285–287
 initializing, 177
 memory locations in, 277
 multidimensional, 183–189
 passing to functions, 189–191

- arrays (*continued*)
 - pointer math in, 287–292
 - pointers and, 285–294
 - reading size of, 272
 - resizing, 183
 - returning from
 - functions, 191
 - sorting, 181–183
 - substituting pointers for
 - notation in, 293–294
 - three-dimensional, 186–187
 - two-dimensional, 183–186
- ASCII codes, 85, 121, 413–418
- assigning
 - values on creation, 77
 - values to variables, 71
 - values using pointers, 283–284
- assignment operators, 158–160, 408–409
- asterisk (*), pointers and, 281, 282–283, 406
- attributes, of variables, 270
- auto* keyword, 240
- avoiding arrays, 173–174

B

- B programming language, 22, 23
- Babbage, Charles, 21–22
- backslash character, 54–55, 76, 363
- backwards()* function, 150
- bash shell program, 10
- BASIC, 22
- Basic Combined
 - Programming Language (BCPL), 22, 23
- beep()* function, 31, 34
- binary digits (bits)

- ! (NOT) operator, 259
- ~ (ones' complement)
 - operator, 259
- about, 249–251
- binbin()* function, 263
- bitwise & operator, 256
- bitwise | operator, 253–256
- defined, 250
- hexadecimal values, 264–266
- manipulating, 253–263
- negative binary
 - numbers, 262
- outputting values, 251–253
- reading, 348–350
- shifting values, 259–263
- writing, 346–348
- XOR operator, 257–258
- binbin()* function, 251–253, 261, 263
- bit field, 363
- bits (binary digits)
 - ! (NOT) operator, 259
 - ~ (ones' complement)
 - operator, 259
 - about, 249–251
 - binbin()* function, 263
 - bitwise & operator, 256
 - bitwise | operator, 253–256
 - defined, 250
 - hexadecimal values, 264–266
 - manipulating, 253–263
 - negative binary
 - numbers, 262
 - outputting values, 251–253
 - reading, 348–350
 - shifting values, 259–263
 - writing, 346–348
 - XOR operator, 257–258

- bitwise & operator, 256
- bitwise | operator, 253–256
- bitwise operators, 154
- _Bool data type, 70, 74
- braces, 34
- break* keyword, 110, 111–112, 128–129, 398–399
- breaking out, of endless
 - loops, 127–129
- breakpoints, setting, 387
- bubble sort, 181–182
- bugs
 - about, 381
 - debugger, 383–389
 - error messages, 390–392
 - resolving problems, 381–383
- building
 - arrays from structures, 216–217
 - arrays of pointers, 296–300
 - Code::Blocks projects, 16–17
 - custom header files, 374–377
 - empty files, 17
 - external structure
 - variables, 243–245
 - functions, 133–136
 - functions to return values, 143–145
 - linked lists, 316–323
 - logic comparisons, 106–107
 - projects, 16–17, 18–19
 - static variables, 238–241
 - string storage, 308–309
 - strings, 200–202
 - with *typedef* keyword, 235–238
- byte, 250

C

- C compiler, 11, 76
- C library references, 19, 32
- `%c` placeholder, 88
- C programming language
 - about, 22–23, 29
 - comments, 35–36
 - components of, 29–36
 - functions, 31–32
 - keywords, 30–31
 - operators, 33
 - statements, 33–35
 - structure, 33–35
 - typical program in, 37–41
 - values, 33
 - variables, 33
- C++ programming language, 22, 23
- Caesar, Julius, 330
- calendar, 330
- calling directories, 359–361
- `calloc()` function, 309–311
- `case` statement, 109–110
- cc compiler, 11
- `ceil()` function, 161, 162
- changing characters, 197–198
- `char` arrays, 89, 92, 94, 95, 178–180
- `char` data type, 69, 70, 74, 81
- character arrays. *See* strings
- character I/O, 83–88
- character variables, 87–88
- characters, fetching with `getchar()` function, 84–86
- `chdir()` function, 364–365
- checking
 - clocks, 331–332
 - location of variables, 275–278
- `clang` compiler, 26, 160, 372, 379
- `clock()` function, 331
- clocks, checking, 331–332
- closing
 - command prompt, 223
 - terminal window, 223
- COBOL, 22
- code
 - constants in, 80–81
 - reading, 409
 - running in Text mode, 223–224
 - splitting, 377
 - talking through, 382
 - working on, 405
- code listings
 - `%` (modulo) operator, 158
 - `--` (decrement) operator, 156, 157
 - `=` (equal) sign, 102, 157
 - `++` (increment) operator, 156, 157
 - `==` (is equal to) operator, 101–102
 - `|` (OR) operator, 254, 255
 - `?:` (ternary) operator, 113
- adding multiple `for` loop conditions, 129
- adding `return` statement, 39
- aligning output, 206
- allocating input buffers, 309
- allocating space for structures, 315–316
- appending text to files, 345
- argument counter, 228
- array of strings, 185
- array program, 293
- arrays, 176
- arrays and pointer math, 288, 290, 291
- assigning values, 77
- assigning values using pointers, 283–284
- assignment operators, 159
- avoiding arrays, 174
- avoiding function prototype, 137–138
- basic function, 135
- `binbin()` function, 252
- breaking out of endless loops, 128
- bubble sort, 181–182
- `calloc()` function, 310
- changing characters, 197
- character variables, 88
- checking location of variables, 276
- Code::Blocks C skeleton, 4, 15, 37
- comparing values, 100–101
- computer math, 62
- constants, 149
- converting degrees to radians, 164
- copying files, 367
- counting by letters, 121
- counting by two, 120
- counting with `for` loops, 119
- creating arrays from structures, 216–217
- creating custom header files, 374–375, 376
- creating file-dumper programs, 349
- creating files, 366
- creating functions to return values, 143, 145
- creating linked lists, 317–318, 320–322
- creating static variables, 238–239
- creating structures, 236

- code listings (*continued*)
 - debugger, 385
 - decision-making, 111–112
 - declaring initialized structures, 215
 - declaring strings using pointers, 295
 - declaring variables, 77
 - deleting files, 368
 - directories, 364
 - disabling statements, 49
 - displaying char *arrays*, 178
 - displaying humorous messages, 46
 - displaying text with *printf()* function, 52
 - displaying two lines of text, 47
 - displaying values, 59
 - double-slash comments, 36
 - do-while* loop, 125
 - editing linked lists, 324–327
 - endless loops, 127
 - enum* keyword, 245–246
 - error messages, 390–391
 - errors in source code, 50, 55
 - exiting functions with *return*, 146
 - external variables, 242
 - fetching characters with *getchar()* function, 84
 - fgets()* function and, 344–345
 - Fibonacci sequence, 125
 - filling *char* arrays, 179
 - formatting floating points, 203
 - free()* function, 313–314
 - fseek()* function, 356
 - ftell()* function, 354–355
 - getting addresses for arrays, 286
 - Hello World program, 25
 - hexadecimals, 265
 - if-else* comparison, 105
 - integers, 65
 - interacting variables, 78
 - linking two source files, 370, 372
 - local variables in functions, 139
 - logic comparisons, 107
 - for* loops, 116, 130–131
 - main()* function, 38
 - malloc()* function, 307
 - math functions, 161–162
 - memory locations in arrays, 277
 - multiline comments, 36
 - multiple-choice selections, 109
 - multivariable, 212
 - nested *for* loops, 122
 - nested structures, 218–219
 - overly commented source code, 35
 - passing arrays to functions, 190
 - passing pointers to functions, 302
 - passing structures to functions, 244
 - passing values to functions, 141
 - pointer arrays, 296–297, 297–298
 - pointers, 280, 282
 - preparing for constant updates, 82
 - pulling files from directories, 360–361
 - quitting programs, 229–230
 - random numbers, 166, 168, 169
 - reading binary data, 348
 - reading command line, 225
 - reading size of arrays, 272
 - reading size of structures, 273
 - reading size of variables, 271–272
 - reading strings with *fgetc()* function 5, 95
 - reading strings with *scanf()* function, 92
 - reading text from files, 342–343
 - reading values with *scanf()* function, 93
 - realloc()* function, 311–312
 - renaming files, 366
 - reusing variables, 78
 - rewind()* function, 354–355
 - semicolons, 103
 - setting output width, 204–205
 - sharing variables between modules, 373, 374
 - shifting binary values, 260
 - simple comparison, 98
 - simple program that does nothing, 38
 - single-character input function, 209
 - snoozing time, 336
 - sorting strings, 300–301
 - stat()* function, 362
 - storing strings, 89
 - strcat()* function, 201
 - strcmp()* function, 200
 - stream input, 207, 208
 - strings and pointers, 294
 - system()* function, 230–231

- testing characters, 195–196
 - tic-tac-toe board, 184, 187
 - time()* function, 332, 335
 - trigonometry, 164–165
 - typecasting, 234
 - typedef* keyword, 235–236, 237
 - unsigned int, 74
 - using *putchar()* function, 86
 - values, 79
 - variables, 68, 71
 - viewing timestamps, 333
 - watching variables, 388
 - while* loop, 124
 - writing binary data, 346
 - writing structures to files, 351, 352–353
 - writing text to files, 341
 - XOR operator, 257–258
 - Code::Blocks IDE
 - about, 11
 - building projects, 16–17
 - installing, 12–13
 - skeleton code listing, 4, 15
 - workspace, 13–14
 - coding. *See* programming
 - command line
 - reading, 225–227
 - tools, as requirement for programming, 10
 - command prompt
 - about, 221
 - arguments for *main()* function, 225–229
 - launching terminal window, 221–224
 - programming in, 10–11
 - quitting programs, 229–230
 - reading command line, 225–227
 - running another program, 230–231
 - running code in Text mode, 223–224
 - command prompt programming, 15–16
 - commas, in *for* loops, 398
 - comments
 - about, 35–36
 - statements and, 49
 - writing, 382–383
 - comparing
 - conditions and, 395–397
 - logic, 106–108
 - simple, 97–99
 - text, 199–200
 - values, 100–102
 - comparison operators, 100–102
 - compiler messages, 138
 - compilers/compiling, 26–27, 35
 - complex decisions, 104–105
 - computer, as requirement for programming, 10
 - conditions, 395–397
 - const* type, 147, 240
 - constants
 - about, 79, 147
 - in code, 80–81
 - defined, 147–150
 - reusing values, 79–80
 - using, 81–82
 - conversion characters, 59–60, 427–428
 - convert()* function, 143–145
 - copying files, 367–368
 - cos()* function, 164
 - counting, with *for* statement, 119–120
 - creating
 - arrays from structures, 216–217
 - arrays of pointers, 296–300
 - Code::Blocks projects, 16–17
 - custom header files, 374–377
 - empty files, 17
 - external structure variables, 243–245
 - functions, 133–136
 - functions to return values, 143–145
 - linked lists, 316–323
 - logic comparisons, 106–107
 - projects, 16–17, 18–19
 - static variables, 238–241
 - string storage, 308–309
 - strings, 200–202
 - with *typedef* keyword, 235–238
 - creativity, in names, 404–405
 - ctime()* function, 331, 333, 334–336
 - CTYPE* functions, 193–195
 - curly brackets, 34, 109, 115, 123, 134, 213, 399
- ## D
- %d placeholder, 59, 65, 68
 - D programming language, 23
 - data types, 69–70, 423–424
 - debugger, 383–389
 - decision making
 - about, 97
 - adding logical operators, 107–108
 - adding third options, 105–106
 - break* statements, 111–112

- decision making (*continued*)
 - building logical comparisons, 106–107
 - comparing = and ==, 102–103
 - comparing values, 100–102
 - complex, 104–105
 - if* keyword, 99–100
 - multiple, 104–106
 - multiple comparisons with logic, 106–108
 - semicolons, 103–104
 - simple comparisons, 97–99
 - switch-case* structure, 108–112
 - ?: (ternary) operator, 112–113
- declaring
 - multidimensional arrays, 188–189
 - strings using pointers, 295–296
 - variables, 70
- decrement (--) operator, 154–157
- #define directive, 147, 148
- defined constants
 - about, 147–148
 - using, 148–150
- deleting files, 368
- devices, input/output (I/O), 83–84
- difftime()* function, 331, 336
- directories
 - calling, 359–361
 - separating files from, 363
 - stat()* function, 361–363
- directory tree, 364–365
- disabling statements, 49
- discount()* function, 302
- displaying

- humorous messages, 45–46
- strings using pointers, 294–295
- text on-screen, 45–51
- text with *printf()* function, 52
- values with *printf()* function, 59–60
- division (/) operator, 154
- division (/=) operator, 159
- documenting flow, 382
- double data type, 69, 74
- double-pointer notation, 298
- double-slash (//), 36
- double-word, 250
- do-while* loop, 125–126
- dump, 350

E

- editing linked lists, 323–328
- Editor, Code::Blocks workspace, 14
- elements, in arrays, 175
- else if* statement, 105–106
- emacs text mode editor, 11
- endless loops, 126–128, 400–401
- enum* keyword, 240, 245–247, 294
- enumerating, 245–247
- equal (=) sign, 33, 102–103, 154
- error messages, 390–392
- errors
 - defined, 51
 - in source code, 49–51, 55–56
- escape sequences
 - about, 425
 - using, 54–55

- escaping, 76
- ex0201 program, 28
- exit condition, 116, 117
- exit()* function, 229–230
- extern* keyword, 240, 243, 373
- external variables, 241–243

F

- %f placeholder, 59
- feof()* function, 353
- fgetc()* function, 342–343, 350, 367
- fgets()* function, 94–96, 179, 309, 340, 344–345, 401, 405
- Fibonacci sequence, 125–126
- files
 - appending text to, 345–346
 - copying, 367–368
 - creating empty, 17
 - deleting, 368
 - directories, 359–365
 - header, 41
 - library, 41
 - linking two source, 370–372
 - reading data from, 353–355
 - reading text from, 342–345
 - renaming, 365–367
 - rewinding data in, 353–355
 - saving linked lists to, 357
 - separating from
 - directories, 363
 - writing structures to, 351–353
 - writing text to, 341–342
- fill_data()* function, 303
- filling structures, 215–216
- fillstructure()* function, 377
- finding

- specific records, 355–357
- Windows files in
 - Ubuntu, 224
- float data type, 69, 74
- float variable, 233–234
- floating points. *See* floats
- floating-point numbers. *See* floats
- floats
 - about, 58
 - formatting, 202–204
 - integers compared with, 62–63
 - pretending integers are, 64–65
- floor()* function, 161
- flow, documenting, 382
- folders. *See* directories
- open()* function, 340, 341, 345–346
- for* loops
 - about, 116, 117–119
 - adding multiple conditions, 129
 - commas in, 398
 - counting with, 119–120
 - creating structures, 322
 - for letters, 120–121
 - nesting, 121–123
 - for a specific numbers of times, 116–117
 - for* statement, 119–120
- formatting
 - floating points, 202–204
 - with *printf()* function, 202–207
- FORTRAN language, 22
- forward()* function, 150
- fprintf()* function, 340, 342, 347
- fputc()* function, 367
- fputchar()* function, 340
- fputs()* function, 340, 344
- fread()* function, 348–349, 350, 353, 357, 360
- free()* function, 313–314
- freeing memory, 313–314
- fscanf()* function, 340
- fseek()* function, 355–357
- ftell()* function, 354–355
- functions
 - about, 31–32
 - abs()*, 161
 - adding, 40–41
 - address()*, 303
 - anatomy of, 133–138
 - arrayinc()*, 191
 - arrays and, 189–191
 - backwards()*, 150
 - beep()*, 31, 34
 - binbin()*, 251–253, 261, 263
 - calloc()*, 309–311
 - ceil()*, 161, 162
 - character manipulation, 193–198
 - chdir()*, 364–365
 - clock()*, 331
 - constants, 147–150
 - constructing, 133–136
 - convert()*, 143–145
 - cos()*, 164
 - creating that return values, 143–145
 - ctime()*, 331, 333, 334–336
 - CTYPE*, 193–195
 - difftime()*, 331, 336
 - discount()*, 302
 - exit()*, 229–230
 - feof()*, 353
 - fgetc()*, 342–343, 350, 367
 - fgets()*, 94–96, 179, 309, 340, 344–345, 401, 405
 - fill_data()*, 303
 - fillstructure()*, 377
 - floor()*, 161
 - open()*, 340, 341, 345–346
 - forward()*, 150
 - fprintf()*, 340, 342, 347
 - fputc()*, 367
 - fputchar()*, 340
 - fputs()*, 340, 344
 - fread()*, 348–349, 350, 353, 357, 360
 - free()*, 313–314
 - fscanf()*, 340
 - fseek()*, 355–357
 - ftell()*, 354–355
 - fwrite()*, 347, 350, 357
 - getc()*, 85
 - getch()*, 209–210
 - getchar()*, 84–86, 87, 89, 143, 209
 - getcwd()*, 364–365
 - gets()*, 96
 - graph()*, 142–143
 - initialize()*, 245
 - isalnum()*, 194
 - isalpha()*, 194
 - isblank()*, 197
 - iscntrl()*, 194
 - isdigit()*, 194
 - isgraph()*, 194
 - islower()*, 194
 - isprint()*, 194
 - ispunct()*, 194
 - isspace()*, 194
 - isupper()*, 194
 - isxdigit()*, 194
 - limit()*, 146
 - load()*, 357

functions (*continued*)

localtime(), 331, 334–336

main()

about, 70

arguments for, 136,
225–229

code listing, 38

constants and, 80

missing parentheses
in, 399

passing pointers to
functions, 303

prompt() function and, 137

return statement and, 145

setting, 38–39

using, 138

variables in, 140

make_structure(), 322–323

malloc(), 303, 306–309, 311,
312, 314, 315

math, 160–165

mkdir(), 364–365

monster(), 303

nice(), 190

opendir(), 359–361

passing arguments to,
140–143

passing arrays to, 189–191

passing pointers to, 302

passing structures
to, 219

pointers in, 302–303

pow(), 161, 378

printf()

about, 19, 40–41, 51,
52–53, 72, 85

arguments for, 176, 213

binary functions and,
251, 252

constants in, 82

conversion character in,
334, 400

counting by letters, 121

counting by two, 120

for debugging, 381

displaying strings using
pointers, 294–295

displaying text with, 52

displaying values with,
59–60

external variables, 243

formatting with, 202–207

nested *for* loop, 122

newline character and, 53

prompt() function and, 136

random numbers
and, 167

returning values with,
143, 144

stat() function and, 363

storing strings, 89

wrapping statements, 162

printstructure(), 377

proc(), 239

prompt(), 136, 137

prototyping, 134, 136–138

putchar(), 86–87, 120, 122,
143, 296

puts(), 31, 32, 46–49, 51, 53,
294–295, 381, 382

railway(), 142

rand(), 32, 166–170

readdir(), 360–361

read_next_character(), 405

realloc(), 311–313

rename(), 365–367, 391

returning arrays from, 191

returning early, 145–146

returning pointers
from, 303

rewind(), 354–355

rmdir(), 364–365

save(), 357

scanf(), 90–94, 167, 180,
202, 212, 239, 278, 308,
323, 401

second(), 370, 372

sending values to, 140–143

setringervolume(), 405

showarray(), 190–191

sin(), 164

sleep(), 331, 336

sqrt(), 32, 161

srand(), 167, 169

stat(), 361–363

for storage, 339–357

strcat(), 199, 200–202

strchr(), 199

strcmp(), 199–200

strcpy(), 199

string, 198–202

strlen(), 199, 273

strncat(), 199

strncmp(), 199–200

strncpy(), 199

strrchr(), 199

strstr(), 199

swap(), 303

system(), 230–231

tan(), 164

time(), 168–169,
331–332, 333

toascii(), 198

tolower(), 194, 197–198

toupper(), 194, 197–198

trigonometric, 163–165

unlink(), 368

variables and, 139–146

vegas(), 140

verify(), 150
whatever(), 189
writeln(), 56
writing, 405
fwrite() function, 347,
350, 357

G

gcc compiler, 11
generating
 arrays from structures,
 216–217
 arrays of pointers,
 296–300
 Code::Blocks projects,
 16–17
 custom header files,
 374–377
 empty files, 17
 external structure
 variables, 243–245
 functions, 133–136
 functions to return values,
 143–145
 linked lists, 316–323
 logic comparisons, 106–107
 projects, 16–17, 18–19
 static variables, 238–241
 string storage, 308–309
 strings, 200–202
 with *typedef* keyword,
 235–238
getc() function, 85
getch() function, 209–210
getchar() function, 84–86, 87,
89, 143, 209
getcwd() function, 364–365
gets() function, 96
goto keyword, 131

graph() function, 142–143
greater than (>)
 operator, 100
greater than or equal to (>=)
 operator, 100
groupings, binary, 250

H

header files, 32, 41, 374–377
Hello World program,
14–19, 25
hexadecimals, 264–266
high-level languages, 30
Homebrew package
manager, 11
Hopper, Grace, 22
humorous messages,
displaying, 45–46

I

icons, explained, 4–5
IDE. *See* integrated
development
environment (IDE)
if statement
 CTYPE functions and, 196
 in simple comparisons,
 97–99
 using, 99–100
if-else comparison, 104–105,
196, 407–408
immediate value, 67
#include directives, 27,
34, 373
increment (++) operator,
154–157
initialization, of loops,
115, 117
initialize() function, 245
initializing

arrays, 177
 pointers, 279–280
input devices, 83–84
input/output (I/O)
 character, 83–88
 fetching characters with
 getchar() function, 84–86
 input devices, 83–84
 output devices, 83–84
 reading strings with *scanf()*
 function, 91–93
 reading values with *scanf()*
 function, 93–94
scanf() function, 90–91
storing strings, 89–90
text, 88–96
 using *fgets()* function for
 text input, 94–96
 using *putchar()* function,
 86–87
working with character
variables, 87–88
installing
 clang, 11
 Code::Blocks IDE, 12–13
 Ubuntu, 10
 Xcode, 11
int data type, 69, 70, 74
integers
 about, 58
 compared with floats,
 62–63
 pretending they're floats,
 64–65
integrated development
environment (IDE)
 Code::Blocks, 11–14
 programming in, 11–14
as requirement for
programming, 10

- Internet access, as requirement for programming, 10
- Internet resources
 - Code::Blocks IDE, 11
 - Homebrew package manager, 11
 - VIM editor, 11
- is equal to (==) operator, 100–102, 102–103
- isalnum()* function, 194
- isalpha()* function, 194
- isblank()* function, 197
- iscntrl()* function, 194
- isdigit()* function, 194
- isgraph()* function, 194
- islower()* function, 194
- isprint()* function, 194
- ispunct()* function, 194
- isspace()* function, 194
- isupper()* function, 194
- isxdigit()* function, 194

J

- Julian calendar, 330

K

- Kernighan, Brian, 22
- keywords, 30–31, 419–420

L

- launching terminal windows, 221–224
- less than (<) operator, 100
- less than or equal to (<=) operator, 100
- letters, looping, 120–121
- levels, of programming languages, 30

- libraries
 - defined, 32
 - files for, 41
 - linking, 378–379
- limit()* function, 146
- line numbers, 51
- linked lists
 - about, 305–306, 314
 - allocating space for structures, 314–316
 - creating, 316–323
 - editing, 323–328
 - saving, 328
 - saving to files, 357
- linker, 26
- linking
 - about, 26–27
 - libraries, 378–379
 - two source code files, 370–372
- Linux
 - Code::Blocks IDE, 11, 12
 - command prompt programming in, 10
 - opening Terminal app, 222
- listings, code
 - % (modulo) operator, 158
 - (decrement) operator, 156, 157
 - = (equal) sign, 102, 157
 - ++ (increment) operator, 156, 157
 - == (is equal to) operator, 101–102
 - | (OR) operator, 254, 255
 - ?: (ternary) operator, 113
 - adding multiple *for* loop conditions, 129
 - adding *return* statement, 39
 - aligning output, 206

- allocating input buffers, 309
- allocating space for structures, 315–316
- appending text to files, 345
- argument counter, 228
- array of strings, 185
- array program, 293
- arrays, 176
- arrays and pointer math, 288, 290, 291
- assigning values, 77
- assigning values using pointers, 283–284
- assignment operators, 159
- avoiding arrays, 174
- avoiding function prototype, 137–138
- basic function, 135
- binbin()* function, 252
- breaking out of endless loops, 128
- bubble sort, 181–182
- calloc()* function, 310
- changing characters, 197
- character variables, 88
- checking location of variables, 276
- Code::Blocks C skeleton, 4, 15, 37
- comparing values, 100–101
- computer math, 62
- constants, 149
- converting degrees to radians, 164
- copying files, 367
- counting by letters, 121
- counting by two, 120
- counting with *for* loops, 119
- creating arrays from structures, 216–217

- creating custom header files, 374–375, 376
- creating file-dumper programs, 349
- creating files, 366
- creating functions to return values, 143, 145
- creating linked lists, 317–318, 320–322
- creating static variables, 238–239
- creating structures, 236
- debugger, 385
- decision-making, 111–112
- declaring initialized structures, 215
- declaring strings using pointers, 295
- declaring variables, 77
- deleting files, 368
- directories, 364
- disabling statements, 49
- displaying *char arrays*, 178
- displaying humorous messages, 46
- displaying text with *printf()* function, 52
- displaying two lines of text, 47
- displaying values, 59
- double-slash comments, 36
- do-while* loop, 125
- editing linked lists, 324–327
- endless loops, 127
- enum* keyword, 245–246
- error messages, 390–391
- errors in source code, 50, 55
- exiting functions with *return*, 146
- external variables, 242
- fetching characters with *getchar()* function, 84
- fgets()* function and, 344–345
- Fibonacci sequence, 125
- filling *char* arrays, 179
- formatting floating points, 203
- free()* function, 313–314
- fseek()* function, 356
- ftell()* function, 354–355
- getting addresses for arrays, 286
- Hello World program, 25
- hexadecimals, 265
- if-else* comparison, 105
- integers, 65
- interacting variables, 78
- linking two source files, 370, 372
- local variables in functions, 139
- logic comparisons, 107
- for* loops, 116, 130–131
- main()* function, 38
- malloc()* function, 307
- math functions, 161–162
- memory locations in arrays, 277
- multiline comments, 36
- multiple-choice selections, 109
- multivariable, 212
- nested *for* loops, 122
- nested structures, 218–219
- overly commented source code, 35
- passing arrays to functions, 190
- passing pointers to functions, 302
- passing structures to functions, 244
- passing values to functions, 141
- pointer arrays, 296–297, 297–298
- pointers, 280, 282
- preparing for constant updates, 82
- pulling files from directories, 360–361
- quitting programs, 229–230
- random numbers, 166, 168, 169
- reading binary data, 348
- reading command line, 225
- reading size of arrays, 272
- reading size of structures, 273
- reading size of variables, 271–272
- reading strings with *fgetc()* function 5, 95
- reading strings with *scanf()* function, 92
- reading text from files, 342–343
- reading values with *scanf()* function, 93
- realloc()* function, 311–312
- renaming files, 366
- reusing variables, 78
- rewind()* function, 354–355
- semicolons, 103
- setting output width, 204–205
- sharing variables between modules, 373, 374
- shifting binary values, 260
- simple comparison, 98
- simple program that does nothing, 38

- listings, code (*continued*)
 - single-character input function, 209
 - snoozing time, 336
 - sorting strings, 300–301
 - stat()* function, 362
 - storing strings, 89
 - strcat()* function, 201
 - strcmp()* function, 200
 - stream input, 207, 208
 - strings and pointers, 294
 - system()* function, 230–231
 - testing characters, 195–196
 - tic-tac-toe board, 184, 187
 - time()* function, 332, 335
 - trigonometry, 164–165
 - typecasting, 234
 - typedef* keyword, 235–236, 237
 - unsigned int, 74
 - using *putchar()* function, 86
 - values, 79
 - variables, 68, 71
 - viewing timestamps, 333
 - watching variables, 388
 - while* loop, 124
 - writing binary data, 346
 - writing structures to files, 351, 352–353
 - writing text to files, 341
 - XOR operator, 257–258
- LLVM clang compiler, 11
- load()* function, 357
- localtime()* function, 331, 334–336
- logic comparisons, 106–108
- logical operators, 107–108
- Logs, Code::Blocks workspace, 14
- long, 250
- long int data type, 74

- loops
 - for*, 116–123, 129
 - about, 115–116
 - breaking out of, 128–129
 - endless, 126–128, 400–401
 - goto*, 131
 - semicolons and, 397
 - troubleshooting, 130–131
 - while*, 123–126
- low-level languages, 30

M

- Mac
 - opening Terminal app, 222
 - Xcode, 11
- Mac OS X, command prompt programming in, 11
- macros, 85
- main()* function
 - about, 70
 - arguments for, 136, 225–229
 - code listing, 38
 - constants and, 80
 - missing parentheses in, 399
 - passing pointers to functions, 303
 - prompt()* function and, 137
 - return statement and, 145
 - setting, 38–39
 - using, 138
 - variables in, 140
- make_structure()* function, 322–323
- malloc()* function, 303, 306–309, 311, 312, 314, 315
- man program, 19
- Management pane, Code::Blocks workspace, 14

- manipulating binary digits (bits), 253–263
- math
 - operator, 156–157
 - ++ operator, 156–157
 - about, 57, 153
 - arithmetic, 61–63
 - assignment operators, 158–160
 - decrementing, 154–156
 - float-integer, 63–64
 - functions, 160–165
 - incrementing, 154–156
 - integers, 64–66
 - modulus operator, 158
 - operators, 153–160
 - order of precedence, 170–171
 - for pointers in arrays, 287–292
 - random numbers, 166–170
 - trigonometry, 163–165
- math operators, 61–63
- member operator, 214
- memory
 - about, 305–306
 - calloc()* function, 309–311
 - creating string storage, 308–309
 - freeing, 313–314
 - malloc()* function, 306–309
 - reallocating, 311–313
- midlevel languages, 30
- minus (-) sign, 33
- MJD (Modified Julian Date), 330
- mkdir()* function, 364–365
- mode, 340
- Modified Julian Date (MJD), 330
- modules

- about, 369
- breaking into, 406
- creating custom header files, 374–377
- defined, 370
- libraries, 378–379
- linking two source code files, 370–372
- multi-, 369–377
- sharing variables between, 372–374
- modulo (%) operator, 154, 158
- modulo (%)= operator, 159
- monster()* function, 303
- multidimensional arrays
 - about, 183
 - declaring, 188–189
 - three-dimensional, 186–187
 - two-dimensional, 183–186
- multiple decisions, 104–106
- multiple-choice selections, 108–110
- multiplication (*) operator, 154
- multiplication (*=) operator, 159
- multivariables, 211–213
- “My Dear Aunt Sally” mnemonic, 170

N

- names
 - creativity in, 404–405
 - of variables, 70
- negative binary numbers, 262
- nested *for* loops, 121–123, 167
- nested structures, 218–219
- newline character, 53
- nice()* function, 190

- not equal to (!=) operator, 100
- not (!) operator, 107–108, 259
- notation, pointer vs. array, 299
- NULL value, 92, 306, 333
- numbers, random, 166–170, 333

O

- object code, 26
- octal, 266
- OCTO constant, 147–148
- ones’ complement (~) operator, 259
- on-screen, displaying text, 45–51
- Open command, 17
- opendir()* function, 359–361
- operating system, returning something to the, 39–40
- operators. *See also specific operators*
 - about, 33, 421–422
 - assignment, 158–160, 408–409
 - comparison, 100–102
 - logical, 107–108
 - math, 61–63, 153–160
 - member, 214
 - shift, 259–263
- or (| |) operator, 107–108
- OR (|) operator, 253–256, 259
- order of precedence, 170–171, 429–430
- output
 - aligning, 206–207
 - of binary values, 251–253
 - devices for, 83–84
 - setting width for, 204–206

P

- parentheses
 - in functions, 31, 134
 - missing, 399
 - order of precedence and, 171
 - pointers and, 292
- Pascal, 22
- passing
 - arguments to functions, 140–143
 - arrays to functions, 189–191
 - pointers to functions, 302
 - structures to functions, 219, 243–245
- plus (+) sign, 33
- pointers
 - about, 269, 279–282, 406
 - arrays and, 285–294
 - building arrays of, 296–300
 - declaring strings using, 295–296
 - displaying strings using, 294–295
 - in functions, 302–303
 - getting addresses of arrays, 285–287
 - math in arrays, 287–292
 - passing to functions, 302–303
 - problems with, 269–270
 - returning from functions, 303
 - sorting strings, 300–301
 - strings and, 294–301
 - substituting for array notation, 293–294
 - variable storage, 270–278
 - working with, 282–284
- posture, 404
- pow()* function, 161, 378

- powers of 2, 250
- PowerShell, 222
- precedence, order of, 170–171
- precision, 59
- preprocessor directives, 27
- print()* statement, 65
- printf()* function
 - about, 19, 40–41, 51, 52–53, 72, 85
 - arguments for, 176, 213
 - binary functions and, 251, 252
 - constants in, 82
 - conversion character in, 334, 400
 - counting by letters, 121
 - counting by two, 120
 - for debugging, 381
 - displaying strings using pointers, 294–295
 - displaying text with, 52
 - displaying values with, 59–60
 - external variables, 243
 - formatting with, 202–207
 - nested *for* loop, 122
 - newline character and, 53
 - prompt()* function and, 136
 - random numbers and, 167
 - returning values with, 143, 144
 - stat()* function and, 363
 - storing strings, 89
 - wrapping statements, 162
- printstructure()* function, 377
- proc()* function, 239
- programming
 - command prompt, 10–11, 15–16
 - defined, 21

- history of, 21–23
- in integrated development environment (IDE), 11–14
- process of, 23–28
- requirements for, 9–10
- programs
 - quitting, 229–230
 - running other, 230–231
- projects
 - building, 18–19
 - creating, 16–17
 - running, 18–19
- prompt()* function, 136, 137
- prototype, 32
- prototyping functions, 134, 136–138
- pseudo-random numbers, 166
- putchar()* function, 86–87, 120, 122, 143, 296
- puts()* function, 31, 32, 46–49, 51, 53, 294–295, 381, 382

Q

- quitting programs, 229–230

R

- radians, 163–164
- railway()* function, 142
- rand()* function, 32, 166–170
- random file access
 - about, 350–351
 - finding specific records, 355–357
 - reading, 353–355
 - rewinding, 353–355
 - saving linked lists to files, 357
 - writing structures to files, 351–353

- random numbers, 166–170, 333
- readdir()* function, 360–361
- reading
 - binary data, 348–350
 - code, 409
 - command line, 225–227
 - data from files, 353–355
 - size of arrays, 272
 - size of structures, 273
 - size of variables, 271–275
 - strings with *scanf()* function, 91–93
 - text from tiles, 342–345
 - values with *scanf()* function, 93–94
- read_next_character()* function, 405
- realloc()* function, 311–313
- reallocating memory, 311–313
- records, finding specific, 355–357
- redimensioning arrays, 183
- register* keyword, 240
- release number, 12
- Remember icon, 4
- rename()* function, 365–367, 391
- renaming files, 365–367
- repeat each condition, 117
- resizing arrays, 183
- return* statement, 39, 134, 144, 145–146, 245
- returning
 - arrays from functions, 191
 - pointers from functions, 303
- reusing
 - values, 79–80
 - variables, 77–79

rewind() function, 354–355
rewinding data in files,
353–355
Ritchie, Dennis, 22
rmdir() function, 364–365
Run button, 18
running
about, 27–28
code in Text mode,
223–224
other programs, 230–231
projects, 18–19

S

save() function, 357
saving linked lists, 328, 357
scanf() function, 90–94, 167,
180, 202, 212, 239, 278,
308, 323, 401
second() function, 370, 372
semicolons, 34, 103–104, 397
sending values to functions,
140–143
separating files from
directories, 363
sequential file access,
339–350
setringervolume()
function, 405
setting
breakpoints, 387
main() function, 38–39
output width, 204–206
setup
debugger, 383–384
variables, 68–69
sharing variables between
modules, 372–374
shift operators, 259–263
shifting binary values,
259–263

short int data type, 74
showarray() function,
190–191
showing
humorous messages,
45–46
strings using pointers,
294–295
text on-screen, 45–51
text with *printf()*
function, 52
values with *printf()* function,
59–60
sin() function, 164
single precision float data
type, 72
sizeof operator, 273–275, 306
sleep() function, 331, 336
snoozing time, 336
sorting
arrays, 181–183
strings, 300–301
source code
defined, 23
errors in, 49–51, 55–56
writing, 24–26
splitting code, 377
sqrt() function, 32, 161
square brackets, 175, 177
srand() function, 167, 169
stat() function, 361–363
statements
comments and, 49
disabling, 49
statements about, 33–35
static keyword, 240
static variables, 238–241
Status bar, Code::Blocks
workspace, 14
stdin, 84

stdout, 84
storage
permanent functions for,
339–357
random file access,
350–357
sequential file access,
339–350
strings, 89–90
variable, 270–278
Stoustroup, Bjarne, 22
strcat() function, 199,
200–202
strchr() function, 199
strcmp() function, 199–200
strcpy() function, 199
stream input, 207–210, 402
stream-oriented input, 86
strings
about, 177–179
building, 200–202
creating storage for,
308–309
declaring using pointers,
295–296
defined, 31, 193
displaying using pointers,
294–295
pointers and, 294–301
reading with *scanf()*
function, 91–93
sorting, 300–301
storing, 89–90
time, 334–336
using, 177–179
strlen() function, 199, 273
strncat() function, 199
strncmp() function, 199–200
strncpy() function, 199
strrchr() function, 199

strstr() function, 199
struct keyword, 213–214
structures
 about, 33–35
 allocating space for, 314–316
 creating arrays from, 216–217
 filling, 215–216
 multivariables, 211–213
 nested, 218–219
 passing to functions, 219
 putting within other structures, 218–219
 reading size of, 273
 struct keyword, 213–214
 writing to files, 351–353
substituting pointers for array notation, 293–294
subtraction (-) operator, 154
subtraction (-=) operator, 159
swap() function, 303
switch structure, 111, 398–399
switch-case structure, 108–112, 247, 407–408
syntax, 33
system() function, 230–231

T

tan() function, 164
Technical Stuff icon, 5
Terminal app, 11, 222
terminal windows, launching, 221–224
ternary (?:) operator, 112–113
testing
 about, 27–28
 characters, 195–197

text
 adding, 47–48
 aligning output, 206–207
 appending to files, 345–346
 building strings, 200–202
 changing characters, 197–198
 character manipulation functions, 193–198
 comparing, 199–200
 CTYPE functions, 193–195
 displaying on-screen, 45–51
 displaying with *printf()* function, 52
 formatting floating point, 202–204
 input of using *fgets()* function for, 94–96
 printf() function formatting, 202–207
 reading from files, 342–345
 setting output width, 204–206
 stream based input, 207–210
 string functions, 198–202
 testing characters, 195–197
 writing to files, 341–342
text I/O, 88–96
Text mode, running code in, 223–224
text part, 47
three-dimensional arrays, 186–187
time
 about, 329
 calendar, 330
 checking clock, 331–332
 random numbers and, 333
 snoozing, 336
 strings and, 334–336

 viewing timestamps, 333–334
 working with, 330–331
time() function, 168–169, 331–332, 333
timestamps, viewing, 333–334
Tip icon, 5
toascii() function, 198
tolower() function, 194, 197–198
toolbars, Code::Blocks workspace, 14
toupper() function, 194, 197–198
trigonometry, 163–165
triple nested loop, 123
troubleshooting loops, 130–131
2, powers of, 250
two-dimensional arrays, 183–186
type data type, 70
typecasting, 169, 233–235
typedef keyword, 235–238, 274, 319

U

Ubuntu, 10, 224
unary minus (-) operator, 154
unary plus (+) operator, 154
undefined reference, 162
union keyword, 240
Unix
 command prompt programming in, 10
 displaying math functions, 162
 locating files in, 41
 opening Terminal app, 222
 string functions, 199
 time and, 330

- Unix epoch, 330
- unlink()* function, 368
- unsigned char data type, 74
- unsigned int data type, 74
- unsigned long int data type, 74
- unsigned short int data type, 74

V

- value range, 73
- values. *See also* arguments
 - about, 33, 57–58, 70–71
 - assigning on creation, 77
 - assigning to variables, 71
 - assigning using pointers, 283–284
 - comparing, 100–102
 - creating functions to return, 143–145
 - displaying with *printf()* function, 59–60
 - floating-point, 59
 - hexadecimal, 264
 - outputting binary, 251–253
 - reading with *scanf()* function, 93–94
 - reusing, 79–80
 - sending to functions, 140–143
 - variables, 67–73
 - zeros, 61
- variables
 - about, 33, 67
 - assigning values on creation, 77
 - assigning values to, 71
 - attributes of, 270
 - checking location of, 275–278
 - controlling, 233–241

- creating external structure, 243–245
- data types, 69–70
- declaring, 70
- enumerating, 245–247
- external, 241–243
- functions and, 139–146
- more-specific data types, 73–75
- reading size of, 271–275
- reusing, 77–79
- setting up, 68–69
- sharing between modules, 372–374
- static, 238–241
- storage of, 270–278
- typecasting, 233–235
- typedef* keyword, 235–238
- using, 70–73
- watching, 388–389
- working with several, 75–76
- vegas()* function, 140
- verify()* function, 150
- verifying
 - clocks, 331–332
 - location of variables, 275–278
- vi text mode editor, 11
- View menu, Code::Blocks workspace, 14
- viewing timestamps, 333–334
- VIM editor, 11
- void* type, 69, 75, 134
- volatile* keyword, 240
- von Neumann, John, 22

W

- Warning icon, 5
- warnings, 50–51, 399–400

- websites
 - Code::Blocks IDE, 11
 - Homebrew package manager, 11
 - VIM editor, 11
- whatever()* function, 189
- while* loop
 - about, 123, 178–179, 196
 - creating structures, 322
 - do-while* loop, 125–126
 - feof()* function and, 343
 - structure of, 123–124
- white space, 34, 407
- Windows 10
 - Code::Blocks IDE, 11
 - command prompt programming in, 10
 - locating files in, 41
 - opening Terminal app, 222
- word, 250
- workspace, Code::Blocks IDE, 13–14
- writeln()* function, 56
- writing
 - binary data, 346–348
 - comments, 382–383
 - functions, 405
 - source code, 24–26
 - structures to files, 351–353
 - text to files, 341–342

X

- Xcode, 11
- XOR operator, 257–258, 259

Z

- zeros, extra, 61

