INDEX

Page numbers in italics refer to figures and tables.

Symbols

+ (addition operator), 10
&& (AND operator), 13
-> (arrow symbol), 20, 35
* (asterisk), 28
@ (at sign), 25
\\ (backslash escape sequence), 16
{ } (braces), 16
: (colon), 29
+ (concatenation operator), 14
-- (decrement operator), 12
/ (division operator), 10
$ (dollar sign), 15
\$ (dollar sign escape sequence), 16
!! (double-bang or null assertion operator), 18
?: (Elvis operator), 17–18
== (equality operator), 12
> (greater-than operator), 12
++ (increment operator), 12
!= (inequality operator), 12
< (less-than operator), 12
= (main assignment operator), 12
:: (member reference operator), 33
% (modulo operator), 11, 155
* (multiplication operator), 10
! (NOT operator), 13
+= operator, 53
-= operator, 53
.. operator, 22
|| (OR operator), 13
?. (safe call operator), 17–18
- (subtraction operator), 10

A

A* search algorithm, xxix, *288*, 288–300
  code for, 292–297
    display helper functions, 294–295
  g-score, h-score, and f-score, 288
  heuristic functions, 289–291
    admissible and consistent, 289
    approaches for generating, 289–290
  result, 298–299
  steps in algorithm, 291–292
abstract classes, *68*, 71–72
  declaring, 71
  interfaces vs., 74
abstraction, heuristic function generation through, 290
abstract keyword, 71–72
acceleration coefficients, 348
access modifiers, 63–64
ACS algorithm. *See* ant colony system algorithm
action event listeners, 120–123
addAll() method, 99, 101, 105, 119
add() function, 30–34, 53, 55, 80, 98–99, 145, 282
addition operator (+), 10
addTask() method, 80
addToProcessedText() function, 163
ad hoc selection of h-scores, 289–290
agent-based algorithms, xxv–xxvi, 345–377. *See also* ant colony system algorithm; particle swarm optimization

alphabet in L-system, *242*, 242
amplitude, 189
`AnchorPane` container, 97
AND operator (`&&`), 13
angular displacement, 190–191, *198*
angular velocity, 191, *198*
animation
  frame rate, 120
  JavaFX, 115–123
    bouncing ball animation, 118–123
    square animation, 116–117
ant colony system (ACS) algorithm, 356–361, *357*
  NP-hard problems, 358
  pseudocode, 360–361
  symbols, *358*, 358–359
  tour construction, 359–360
  traveling salesman problem, 361–376
  updating pheromone trails, 360
`appendText()` method, 40
`Application` class, 92
`applyFunction()` function, 34
apps. *See* computer programs
`area()` method, 72–73
argmax function, 358
arguments, 29
  named, 31–32
arithmetic operators, 10–11
`Array` constructor, 49–50
`ArrayDeque` class, 282
`arrayOf()` function, 48, 50–52
arrays, xxiv, 48–52
  `Array` constructor, 49–50
  indexes, 48
  methods, *50*, 50–51
  multidimensional, 51–52
  operations, 50
  primitive, 49
arrow symbol (`->`), 20, 35
assignment operators, 12
`aStar()` function, 295–297
asterisk (*), 28
astronomical unit (AU), 211
at sign (`@`), 25
axiom in L-system, *242*, 242
Azul JDK FX, 90
Azul Zulu JDK, 90

B

Babylonian square root algorithm, xxvii, 128–130
  code for, 129–130
  origin of, 128
  result, 130
  steps in, 128
`babylonianSquareRoot()` function, 129
backslash escape sequence (`\\`), 16
bar charts, xxvi, 99–102
  code for, 99–101
  result, 101–102, *102*
BFS. *See* breadth-first search
`bfsQueue()` function, 286
big O notation, 266–267
binary star system simulation, xxviii, 209–220
  code for, 212–219
  result, *220*, 220
  science behind, *210*, 210
  strategy for, 211–212
`bisection()` function, 180
bisection method, *178*, 178–180
`Boolean` data type, 7, *8*
`BorderPane` container, 97
bouncing ball animation, xxvii, 118–123, *123*
  action event listeners, 120–123
  setting keyframes explicitly, 118–119
`bouncyBall()` method, 122
braces ({ }), 16
breadth-first search (BFS), xxix, 284–287
  code for, 285–286
  FIFO principle, 284–285
  result, 286–287, *287*
  steps in algorithm, 285
  time and space complexity, 281
`break` keyword, 23–24
Brown, Robert, 165
Brownian motion, 165
`buildAntTour()` function, 370–371
bytecode, 383
`Byte` data type, *8*

C

calcRMS1d() function, 169–170
calculateEarthMetrics() function, 138–139
calculateEdges() function, 366–367
calculatePheromone0() function, 365, 367–368
calculator, console-based, xxvi, 40–44
  code for, 41–44
  result, 44
camelCase, 5
cannonball flight prediction, xxvii, 175–182, *176*
  bisection method, *178*, 178–180
  code for, 179
  numerical method, 178
  result, *181*, 181
  strategy for, 178–179
  trajectories, 181, *182*
canvas in JavaFX, 107–115
  drawing simple shapes, 107–108, *108*
  graphics context, 107
    common methods, *109*, 109–110
  spiral seashell drawing, 110–117
Char data type, 7, *8*
charts, 98–105
  bar charts, 99–102
  multiseries line charts, 102–105
  steps for creating, 99
child classes, 65–67
child nodes, 97–98
ciphertext, 154
circumference of Earth, calculating, xxvii, *136*, 136–139
classes, xxiv, 57
  common, 67–75
    abstract, *68*, 71–72
    data, *68*, 68–69
    enum, *68*, 74–75
    interface, *68*, 72–73
    pair, *68*, 70–71
    triple, *68*, 70
  constructors, 58–61
    primary, 58–59
    secondary, 59–61
  declaring, 57–58
  encapsulation, 63–64
  inheritance, 65–67
  init block, 61–62
  instances and instantiation, 57–58
  methods, 62
  naming, *7*
  polymorphism, 65–67
  subclasses, 57
  this keyword, 64–65
::class.java construct, 94
class keyword, 57
clear() method, 54
coding, xxii
  benefits of learning, xxii
  code examples on GitHub, xxx
  key terms, xxx
  workflow, 383–384
coffee cooling physics, xxviii, 200–209
  code for, 202–209
    calculating temperature changes, 206–207
    plotting temperature profiles, 207–209, *208*
  effect of mixing liquids, 201
  Newton's law of cooling, 200
  strategy for, 201–202, *202*
collections, xxiv, 52–57
  lists, 52–54
    mutable, 53–54
    read-only, 52–53
  maps, 56
  sets, 54–55
colon (:), 29
comments, 4–5
  documentation, 4–5
  multiline, 4–5
  single-line, 4
  syntax, 4–5
compilers, 383
computer programs (software; apps), 382
  creation workflow, 383–384
concatenation, 14–15
  method chaining, 15
  operator (+), 14–15
  techniques for, 15–16

conditional statements, 18–21
    if statements, 19–20
    when statements, 20–21
console-based calculator, xxvi, 40–44
    code for, 41–44
    result, 44
constants, 6–7, *7*
const keyword, 6–7
constructors, 49–50, 58–61
    primary, 58–59
    secondary, 59–61
contains() function, 55
containsKey() function, 56
continue keyword, 23
copying objects, 75–78
    deep copy, 76–78
    shallow copy, 75–76
copy() method, 68, 75–78
    deep copy, 77–78
    shallow copy, 75–76
cos() function, 112–113
createCoolingChart() function, 207
createRWChart1() and createRWChart2() functions, 170, 172
crossover, 311, 314–315
    real-coded genes, 314–315
    single-point crossover, *314*, 314
crossover() function, 319, 321, 329, 336

D

data classes, *68*, 68–69
data structures
    arrays, 48–52
        Array constructor, 49–50
        methods, *50*, 50–51
        multidimensional, 51–52
        primitive, 49
    classes, 57–75
        constructors, 58–61
        encapsulation, 63–64
        inheritance, 65–67
        init block, 61–62
        methods, 62
        polymorphism, 65–67
        this keyword, 64–65
    collections, 52–57
        lists, 52–54
        maps, 56
        sets, 54–55
    console-based task manager, 78–85
    copying objects, 75–78
        deep copy, 76–78
        shallow copy, 75–76
data types, 7–10
    casting (conversion), 9–10
    common, *8*
    inferring, 8–9
    suffixes, 9
data visualization. *See* visualization
debugging, 383–384
deconstruction, 69
decrement operator (--), 12
decrypt() function, 163
decryptIndexBlock() function, 163
deepCopyPerson() function, 77
deleteTask() method, 83
depth-first search (DFS), xxix, 280–284
depth in L-system, *242*, 242
    code for, 281–282
    LIFO principle, 281
    result, 282–284, *283*
    steps in algorithm, 280–281
    time and space complexity, 285
describe() method, 72
determinants in encryption, 155
deterministic systems, 164
dfsStack() function, 282
displayGraph() function, 294
displayInfo() method, 65
displayList() function, 294
displayProduct() function, 161
displayShortestPath() function, 294–295
division operator (/), 10
documentation comments, 4–5
dollar sign ($), 15
dollar sign escape sequence (\$), 16
domain-specific knowledge, 289
Dorigo, Marco, 356
dot notation, 10
double-bang operator (!!), 18
Double data type, 7, *8*
do...while loops, 27
downTo keyword, 23
drawCircle() method, 112–114

drawFibonacciSpiral() function, 145–146
drawFountain() function, 188
draw() function, 247–248
drawing
  Fibonacci spiral, 141–142, *142*, 145–147
  JavaFX
    canvas, 107–115
    graphics context, 107, *109*, 109–110
    simple shapes, 107–108
    spiral seashell, 110–115
drawMSet() function, 256, 259
drawMultiTurnSpiral() method, 112–113
drawNozzles() function, 188
drawRectangle() method, 108
drawSquares() function, 231–232
drawStars() function, 217–218
drawTree() function, 240–241
drawTriangle() function, 237

E

Earth's circumference, xxvii, *136*, 136–139
  code for, 138–139
  Eratosthenes's calculation, 136–138
  exterior angle theorem, 137
  result, 139
Eggholder function, 308, *309*, 332–336, 339–340, *340*, 350–351, *355*, 355–356
Einstein, Albert, 165
elitism, 315
else clause, 19–20
Elvis operator (?:), 17–18
encapsulation, 63–64
  access modifiers, 63–64
encrypt() function, 163
encryptIndexBlock() function, 163
encryption, 154
  Hill cipher, xxvii, 154–164
enums, *68*, 74–75
equality operator (==), 12
equals() method, 68
Eratosthenes, 133, 136–138
escape sequences, 16
Euclid's formula
  code for, 130–131
  creating Pythagorean triples with, xxvii, 130–132
  primitive triples, 131
  result, 132
  steps in, 130
Euler-Cromer numerical method, 191–192
evolving gibberish into Shakespeare, xxix, 316–323
  code for, 316
    driver function, 318–320
    global declarations, 316–317
    initializing population and fitness evaluation, 317–318
    operator functions, 320–321
  result, 321–322
  strategy for, 316
executable files, 383
exitProcess() function, 42
expressions, 9

F

Fibonacci (Leonardo of Pisa), 139–140
Fibonacci sequence, xxvii, 139–148
  code for, 143–147
    drawing the spiral, 145–147
    generating the sequence, 145
    printing the sequence, 147
    setting up, 143–145
  Fibonacci spiral, 141–142, *142*
  golden ratio, *141*, 141
  rabbit analogy, *140*, 140
  result, 147–148
  steps in sequence, 140
Fibonacci spiral, 141–142, *142*
FibonacciSpiral application class, 143–144
File class, 39–40
fillOval() method, 114
fillText() function, 114
fitness, 310
  initializing evaluation of, 317–318, 326–327, 335–336
Float data type, 7–8, *8*

flow control, 18–27
conditional statements, 18–21
if statements, 19–20
when statements, 20–21
loops, 22
break keyword, 23–24
continue keyword, 23
for loops, 22–23
named for loops, 24–26
nested for loops, 24–26
while loops, 26–27
FlowPane container, 97
for loops, 22–26
break keyword, 23–24
continue keyword, 23
named, 24–26
nested, 24–26
step value, 22
fractals, 225–263. *See also* L-system; Mandelbrot set
concept of, 226
fractal dimension, 227
fractal trees, xxviii, 239–242
code for, 240–241, *241*
strategy for, *239*, 239–240
"Hello, World!" project, 229–234
recursive functions, 227–229, *228*
Sierpiński triangles, *226*, 226–227, *227*, 234–239
frame rate, 120
free diffusion, 165
frequency, 189
function overloading, 29, 32–33
functions, 27–34
custom, 29–34
arguments, 29
conditional expression syntax, 33
declaring, 29
named arguments, 31–32
overloading, 29, 32–33
parameters, 29
providing default parameter values, 31
referencing functions without calling, 33–34
return type, 29
signatures, 29
single-expression functions, 30–31
higher-order, 35
importing, 28
invoking, 10
mathematical, 28–29
methods vs., 10
naming, *7*
scope, 34–35

G

generateFibonacciNumbers() function, 145
generate() function, 246–247
generatePythagoreanTriple() function, 131–132
genetic algorithms, xxv, 305–343. *See also* genetic operators; nature-inspired algorithms
chromosomes, 310
evolving gibberish into Shakespeare, 316–323
fitness, 310
key components of, 310–311
knapsack problem, 323–332
multivariate function optimization, 332–341
stopping condition for, 341–342
genetic operators, 310, 311–315
crossover, 311, 314–315
elitism, 315
mutation, 310–311, 315
selection, 311–314
getAngleAndVel() function, 186–187
getArithmeticOperation() function, 43
getConvergence() function, 256–258
getData() function, 101
getFactorial() function, *228*, 228–229
getFitness() function, 317–318, 321, 327, 334, 336, 353
getGraphicsContext2D() method, 108
getHScore() function, 291, 297
getIndexBlock() function, 163

getMaleData() and getFemaleData() functions, 104
get() method, 14, 56
getText() function, 159, 161
getTrajectories() function, 186–188
GitHub, xxx
global minima and optima, *308*, 308
globalPheromoneUpdate() function, 372
golden ratio, *141*, 141
graphical user interfaces (GUIs), 37
greater-than operator (>), 12
greet() function, 31
GridPane container, 97

H

hashCode() method, 68
hash codes, 68
hasNextLine() method, 39
haversineDistance() function, 152–153
haversine formula, 148–153
heap sort
  compared to other sorting algorithms, 268
  features of, *267*
"Hello, world!" projects
  creating, 385–387, *386*, *387*, *388*
  fractals, xxviii, 229–234
    code for, 231–234, *232*
    strategy for, *230*, 230–231
  JavaFX, xxvi, 90–95
    code for, *91*, *92*, 92–94
    result, *93*, 95
Heron of Alexandria, 128
heuristic search, 288–300, 309
higher-order functions, 35
Hill, Lester S., 154
Hill cipher, xxvii, 154–164
  code for, 157–163
    helper functions, 159–163
    variables and data structures, 157–158
  decryption steps, 157
  encryption steps, 156–157
  multiplying two matrices, 160–161
  result, 163–164
  terminology, 155–156
hyperspaces, 309

I

identity matrices, 155
IDEs (integrated development environments), 4, 382–383
if...else structure, 19–20
if statements, 19–20
import keyword, 28
increment operator (++), 12
indexed characters, 14
inequality operator (!=), 12
inheritance, 65–67
init block, 61–62
initializeAnts() function, 369
initializePheromone() function, 365, 368
initialPositions() function, 215–216
initPopulation() function, 317–318, 321, 326, 330, 338
initSwarm() function, 351–352, 354
input and output, 37–40
  console based, 37–38
  file operations, 39
insertion sort, xxviii, 268–270
  code for, 269–270
  compared to other sorting algorithms, 267
  features of, *267*
  result, 270
  steps in algorithm, 268–269
insertionSort() function, 269–270
intArrayOf() function, 52
Int data type, 7–8, *8*
integrated development environments (IDEs), 4, 382–383
IntelliJ IDEA, 4, 90–91
  creating new projects, 385–387, *386*, *387*, *388*
  downloading and installing, 384
interfaces, *68*, 72–73
  abstract classes vs., 74
  defining, 73
intermediate code, 382
internal access modifier, 64
intersect() function, 55
introduce() method, 63
inverse matrices, 155

J

Java, xxii
Java Archive (JAR) files, 384
Java Development Kit (JDK), 89, 383
    downloading and setting up, 385
JavaFX, 88, 89–123. *See also* charts
    animation, 115–123
        bouncing ball animation, 118–123
        square animation, 116–117
    creating projects in, 91
    drawing with the canvas, 107–115
    "Hello, world!" project, 90–95
    key functionalities of, 89–90
    object hierarchy, 96–98
        child nodes, 98
        layout containers, 97–98
        `Scene` objects, 96–97
        `Stage` object, 96
    overview, 89–95
    setting up, 90
Java Runtime Environment (JRE), 383
Java Virtual Machine (JVM), xxii, 383
JDK. *See* Java Development Kit
JetBrains, xxii
Jetpack Compose, 88
`joinToString()` method, 54
JRE (Java Runtime Environment), 383
JVM (Java Virtual Machine), xxii, 383

K

`KeyFrame` objects, 118–119, 122
keywords, 5
knapsack problem, xxix, 323–332
    code for, 324–330
        driver function, 327–328
        global parameters, 324–326
        initializing population and fitness evaluation, 326–327
        operator functions, 328–329
        problem definition, 324–326
    result, 330–332
    strategy for, 323–324
Kotlin
    advantages of, xxiii
    app creation workflow, 383–384
    basic skills, xxiv, 3–45
        comments, 4–5
        flow control, 18–27
        functions, 27–34
        input and output, 37–40
        lambda expressions, 35–36
        null and nullable types, 17–18
        operators, 10–14
        scope functions, 34–35
        strings, 14–16
        variables, 5–10
    code examples on GitHub, xxx
    creating new projects, 385–387, *386*, *387*, *388*
    Java and, xxii–xxiii
    origin of, xxii

L

lambda expressions (lambdas), 35–36
    parameters, 36
layout containers, 96–98
    `Group` container, 97
    `HBox` container, 98
    `Pane` container, 97
    `VBox` container, 98
Leonardo of Pisa (Fibonacci), 139–140
less-than operator (<), 12
Lets-Plot, 88
Liber Abaci (Fibonacci), 140
Liberica Full JDK, 90
libraries, 382
Lindenmayer, Aristid, 241
`listOf()` function, 52–53
lists, 40, 52–54
    mutable, 53–54
    read-only, 52–53
`listTasks()` method, 81–82
local minima and optima, *308*, 308–309, *309*
logical operators, *13*, 13–14
`Long` data type, *8*, 8
loops, 22–27
    `do...while` loops, 27
    `for` loops, 22–26
        `break` keyword, 23–24
        `continue` keyword, 23

named, 24–26
nested, 24–26
`while` loops, 26–27
L-system, 241–245
components of, *242*, 242
drawing patterns with Turtle Graphics, 243–245, *244*
notations and procedures, *243*, 243
`LSystemApp` class, 249–251
L-system simulator, xxviii, 245–252
code for, 245–251
global declarations, 245
problem definition, 246
fractal examples, *251*, 252
result, *251*, 251, *252*

M

machine code, 382
main assignment operator (=), 12
Mandelbrot, Benoit, 226
Mandelbrot set, 252–254
bulbs, *260*, 260
cardioid, *260*, 260
complex plane, 254
elephant valley, *260*, 260–261, *261*
orbits, *253*, 253–254, *254*
other fractals vs., 252
quadratic and recursive functions, 252–253
seahorse valley, *260*, 260–261, *261*
Mandelbrot set project, xxviii, 254–262
code for, 255–259
checking for convergence, 257–258
combining code segments, 258–259
finding and drawing members, 256–257, *257*
global variables, 255–256
result, 259–261, *260*, *261*
`map()` method, 77–78
`mapOf()` function, 56
maps, 56
`markTaskAsDone()` method, 82–83
mathematical problems, xxv, 127–174
Babylonian square root algorithm, 128–130
Earth's circumference, 136–139
Euclid's formula and Pythagorean triples, 130–132
Fibonacci sequence, 139–148
Hill cipher, 154–164
one-dimensional random walk simulation, 164–173
shortest distance between two locations, 148–153
sieve of Eratosthenes and prime numbers, 133–136
mathematical spaces, 164
matrices in encryption, 155
multiplying two, 160–161
member reference operator (::), 33
merge sort, xxviii–xxix, 270–274, *271*
code for, 271–273
compared to other sorting algorithms, 268
features of, *267*
result, 273–274
`mergeSort()` function, 271–273
methods, 62
array operation, *50*, 50–51
class, 62
common, for type casting, 9
functions vs., 10
invoking, 10
object-oriented programming, 38
Metrica (Heron), 128
modeling and simulation, xxv, 175–222
binary star system simulation, 209–220
cannonball flight prediction, 175–182
coffee cooling physics, 200–209
pendulum motion and phase tracking, 189–199
water jet fountain design, 182–189
modular multiplicative inverse (MMI), 155
modules, 64
modulo operator (%), 11, 155
modulus, 155
multidimensional arrays, 51–52
multiline comments, 4–5
multiplication operator (*), 10
`multiply()` function, 33–34
`multiplyMatricesMod29()` function, 160

multiseries line charts, xxvi, 102–105
code for, 103–105
result, 105
multivariate function optimization
with genetic algorithm, xxix, 332–341
code for, 333–338
result, 338–341
strategy for, 333
with particle swarm, xxix, 350–356
code for, 350–354
result, 354–356
`mutableListOf()` function, 53–54
mutable lists, 53–54
`mutableMapOf()` function, 56
`mutableSetOf()` function, 54
mutation, 310–311, *315*, 315
`mutation()` function, 321, 329, 336–337

N

named `for` loops, 24–26
namespace pollution, 28
nature-inspired algorithms (NIAs), 306–310. *See also* genetic algorithms
decision variables, 306–307
global minima and optima, *308*, 308
hyperspaces, 309
local minima and optima, *308*, 308–309, *309*
objective functions, 306–308, *307*
optimization problems, 306–310
well-behaved functions, 308
when to use, 310
nested `for` loops, 24–26
newline escape sequence (`\n`), 16, 40
`newtonCooling()` function, 205–207
Newton's law of cooling, 200
NOT operator (!), 13
NP-hard problems, 358
null, 17–18
nullable types, 17–18
null assertion operator (!!), 18
null pointer exceptions, 17
numerical method for predicting trajectory of projectiles, 178

O

objective functions, 306–308, *307*
object-oriented programming, 38
objects, 38, 49
one-dimensional random walk simulation, xxvii, 164–173
code for, 166–170
one-dimensional model, *165*, 165–166
random walks, 164
result, 170–173, *171*, *172*
RMS distance, 166, 168–173, *172*
OpenFX project, 89
`open` keyword, 66
operating systems (OSs), 382
operators, 10–14
arithmetic, 10–11
assignment, 12
logical, 13–14, *13*
relational, 12–13
unary, 12
Oracle, 89
OR operator (||), 13
OSs (operating systems), 382
out-of-place sorting, 267
`override` keyword, 67

P

packages, naming, *7*
pair classes, *68*, 70–71
parameters, 29, 31
parent classes, 65–67
particle swarm optimization (PSO), 346–349
acceleration coefficients, 348
implementing for function minimization, 348–349
initializing particle position and velocity, 346
multivariate function optimization, 350–356
pseudocode, 349
updating particle position, 346, 348
updating particle velocity, 346–348, *347*
velocity explosion, 348
`partition()` function, 275–277

pattern databases, heuristic function generation through, 290
pendulum motion and phase tracking, xxviii, 189–199
angular displacement, 190–191, *198*
angular velocity, 191, *198*
code for, 192–197
Euler-Cromer numerical method, 191–192
result, 197–199, *198*, *199*
simple harmonic motion, 189–190
simple pendulum, *190*, 190–191
strategy for, 191–192
`performCalculation()` function, 43
`perimeter()` method, 73
period, 189
periodic motion, 189
plaintext, 154
`play()` method, 119, 122
Plotly, 88
`plus()` method, 14
polymorphism, 65–67
`pow()` function, 28
primary constructors, 58–59
prime numbers, 133
identifying with sieve of Eratosthenes, xxvii, 133–136
code for, 134–135
result, 135–136
steps in algorithm, 133
strategy for, 133–134
primitive arrays, 49
`printBestOverallFitnessAndTour()` function, 373
`printFibonacciSequenceAndRatios()` function, 147
`print()` function, 42
`printLatLong()` function, 151–152
`println()` function, 35, 42, 69, 114
`printMessage()` function, 34
`printOptions()` function, 79
`printParams()` method, 112–114
`printPrimes()` function, 135
`printProcessedText()` function, 163–164
`printTimeAndTemp()` function, 205
`private` access modifier, 63
`processInterimResults()` function, 365, 372–373
programming languages, 382
`projectile()` function, 180
projectiles, 176
cannonball flight prediction, 175–182
water jet fountain design, 182–189
properties, 38
`protected` access modifier, 64
PSO. *See* particle swarm optimization
`public` access modifier, 63
`put()` method, 56
Pythagoras, 130
Pythagorean triples, 130
creating with Euclid's formula, xxvii, 130–132
code for, 130–131
primitive triples, 131
Pythagorean theorem, 130, *131*
result, 132
steps in formula, 130

Q

quadratic equations, 177
queue-based searching, 284–287
quick sort, xxix, 274–278
alternative techniques to avoid time complexity, 275
code for, 275–277
compared to other sorting algorithms, 268
features of, *267*
result, *277*, 277
steps in algorithm, 274
`quickSort()` function, 275–277

R

rabbit analogy by Fibonacci, *140*, 140
randomization, 309
`RandomWalk1D` application class, 168
`randomWalk1d()` function, 168–169
random walks, 164
one-dimensional random walk simulation, 164–173
code for, 166–170
one-dimensional model, *165*, 165–166

random walks *(continued)*
one-dimensional random walk simulation *(continued)*
result, 170–173, *171*, *172*
RMS distance, 166, 168–173, *172*
rank-based selection, 314
`readCities()` function, 366
`readDoubleInput()` function, 42
`readIndex()` function, 79, 82–83
`readLines()` method, 40
`readln()` function, 37, 44
read-only lists, 52–53
recursive functions, xxv, 227–229, *228*. *See also* fractals
relational operators, 12–13
relaxation heuristics, 289–290
`removeAll()` method, 53–54
`removeAt()` method, 53
`remove()` function, 55, 56
root-mean-square (RMS) distance, *165*, 168–173
roulette wheel selection, *312*, 312–313, *313*
`roundToInt()` method, 325
rules in L-system, *242*, 242
`runACS()` function, 365, 368–369
`runGA()` function, 318–319, 321, 327–328, 338
`runPSO()` function, 352–354
`runValidation()` function, 159–160

S

safe call operator (`?.`), 17–18
`sayHello()` method, 62
`Scanner` class, 39–40
`Scene` objects, 96–97
scope functions, 34–35
secondary constructors, 59–61
`selectElites()` function, 319–321
selection, 311
rank-based, 314
roulette wheel, *312*, 312–313, *313*
tournament, *312*, 312
`selectNodeToVisit()` function, 370–372
`setOf()` function, 54
sets, 54–55
`ShapeOfWater` application class, 185–186
SHM (simple harmonic motion), 189
`Short` data type, *8*
shortest distance between two locations, finding, xxvii, 148–153
code for, 150–153
great circle concept, 148–149, *149*
haversine formula, 148–153
latitude and longitude, 149
result, 153
`showChoices()` function, 42
`show()` method, 94, 99, 101, 105, 108, 122
Sierpiński triangles, xxviii, *226*, 226–227, *227*, 234–239
code for, 236–238, *238*
L-system, *251*, 252
strategy for, *235*, 235
sieve of Eratosthenes, identifying prime numbers with, xxvii, 133–136
code for, 134–135
result, 135–136
steps in algorithm, 133
strategy for, 133–134
`sieveOfEratosthenes()` function, 134–135
simple harmonic motion (SHM), 189
`simplePendulumWithDrag()` function, 194–196
`SimulateBinarySystem` application class, 212–216
simulation. *See* modeling and simulation
`sin()` function, 112–113
single-line comments, 4
single-point crossover, *314*, 314
`singleXYChart()` function, 196–197
software. *See* computer programs
solar year (yr), 211
sorting and searching, xxv, 265–301
importance of, 265–266
search algorithms, 278–280
A* search algorithm, 288–300
breadth-first search, 284–287
depth-first search, 280–284
graphs, 278–279, *279*
searching graphs, 279–280
sorting algorithms, 266–268
heap sort, *267*, 268
insertion sort, *267*, 267, 268–270

merge sort, *267*, 268, 270–274
quick sort, *267*, 268, 274–278
space complexity, 267
stability, 267
time complexity, 266
spiral seashell drawing, xxvi, 110–115
code for, 111–114
result, *115*, 115
strategy for, *110*, 110–111
`sqrt()` function, 28–29
square animation, xxvi, 116–117
code for, 116–117
result, 117
square roots, finding with Babylonian algorithm, xxvii, 128–130
code for, 129–130
origin of algorithm, 128
result, 130
stable solution, 191
steps in algorithm, 128
stack-based searching, 280–284
`StackPane` container, 97
stack space, 229
`Stage` object, 96
standard libraries, 382
`start()` method, 100–101, 108, 112, 120–121
stochasticity, 164, 310
`String` data type, 7, *8*
strings, 14–16
concatenation, 14–15
escape sequences, 16
indexed characters, 14
string templates, 15–16
string templates, 15–16
complex expressions, 16
syntax, 15
`strokeLine()` function, 241
`strokeOval()` method, 114
`strokePolygon()` function, 231, 237
`strokeRect()` method, 108
`subtract()` function, 55
subtraction operator (-), 10
Sun Microsystems, 89

T

`\t` (tab escape sequence), 16
tail call optimization (TCO), 229
tail recursion, 229
task manager, console-based, xxvi, 78–85
code for, 78–84
adding tasks, 80
deleting tasks, 83
exiting program, 84
listing tasks, 81
marking tasks as done, 81–83
result, 84–85
TCO (tail call optimization), 229
`tempAfterMixing()` function, 205–207
text editors, 382
thermal mass, 201
`this` keyword, 64–65
`Timeline` class, 118–119, 122
`toMutableList()` function, 78
`toRadians()` function, 112–113
`toString()` method, 59, 68
tournament selection, 312, *312*
`TranslateTransition` class, 117
traveling salesman problem, xxix, 361–376
code for, 361
`buildAntTour()` function, 370–371
`calculateEdges()` function, 366–367
`calculatePheromoneO()` function, 367–368
global declarations, 362–364
`globalPheromoneUpdate()` function, 372
`initializeAnts()` function, 369
`initializePheromone()` function, 368
`main()` block, 364–366
problem definition, 362–364, *363*
`readCities()` function, 366
`runACS()` function, 368–369
`selectNodeToVisit()` function, 371–372
result, 373–376, *375*
triple classes, *68*, 70
truth tables, 13, *13*
`try...catch` blocks, 38, 40

`Turtle` class, 248–249
Turtle Graphics, 243–245, *244*

U

unary operators, 12
`union()` function, 55
`updateAndDrawTrails()` function, 219
`updateInfo()` method, 65
`updateStarPositions()` function, 217

V

`val` keyword, 5
variables, 5–10. *See also* data types
    constants, 6–7
    declaring, 5–6
    initializing, 5–6
    mutable, 5
    naming, 5, *7*
    read-only, 5
    syntax, 5
`var` keyword, 5–6
vectors
    in encryption, 155
    forces with magnitude and direction, 210
velocity explosion, 348
visualization, xxiv, 87–124
    animation, 115–123
        bouncing ball animation, xxvii, 118–123
        square animation, xxvi, 116–117
    charts, 98–105
        bar charts, xxvi, 99–102
        multiseries line charts, xxvi, 102–105
    drawing with the canvas, xxvi, 107–115
    "Hello, world!" project, xxvi, 90–95
    object hierarchy, 96–98
    overview, 89–95
    tools for, 88

W

Wacław, Sierpiński, 226
water jet fountain design, xxviii, 182–189
    code for, 184
        `getAngleAndVel()` function, 186–187
        `getTrajectories()` function, 187–188
        `ShapeOfWater` application class, 185–186
    displacement equations, 177
    nozzles, *183*, *183*
    parameters, *183*
    result, *188*, 188–189
    strategy for, 184
well-behaved functions, 308
`when` expression, 80
`when` statements, 20–21
`while` loops, 26–27, 38, 80

Y

Yale University Library, 128
yr (solar year), 211