

# Index

## Symbols

- \t horizontal-tab escape sequence 110
- , (comma operator) **333**
- !, logical negation (NOT) operator **205**, 207
- !=, inequality operator 121
- ? **144**
- ?:, conditional operator **144**, **165**, 271, 272
- . dot operator **540**
- .wbt file (Webots) **429**
- \* assignment suppression character **525**
- \*, multiplication operator **117**, 156
- \*, pointer operators 367
- \*, multiplication assignment operator **165**
- /, division operator 156
- /\*...\*/ multi-line comment **109**
- //, single-line comment **108**
- /=, division assignment operator **165**
- \? escape sequence 519
- \' single-quote-character escape sequence 519
- \" double-quote-character escape sequence 519
- \\ backslash-character escape sequence 519
- \0 null-character escape sequence 309
- \a alert escape sequence 110, 519
- \b escape sequence 519
- \f escape sequence 445
- \f form-feed escape sequence 519
- \n newline escape sequence 110, 445, 519
- \r carriage-return escape sequence 445, 519
- \t horizontal-tab escape sequence 519
- \v vertical-tab escape sequence 445, 519
- &, address operator **114**, 367
- &, bitwise AND operator 549
- &&, logical AND operator **205**, 271, 272
- &=, bitwise AND assignment operator 557
- # formatting flag **517**
- #, preprocessor operator 109, **744**
- ##, preprocessor operator **744**
- % character in a conversion specifier 156, **505**
- %, remainder operator **117**, 250
- %% conversion specifier 511
- %=, remainder assignment operator **165**
- %c conversion specification 310
- %c conversion specifier 242, 510, **522**
- %d conversion specifier 114, 115, 242
- %E conversion specifier **509**, 522
- %e conversion specifier **509**, 522
- %f conversion specification 157
- %f conversion specifier 242
- %g conversion specifier 522
- %hd conversion specifier 242
- %hu conversion specifier 242
- %i conversion specifier **521**
- %ld conversion specifier 242
- %Lf conversion specifier 242
- %lf conversion specifier 242
- %lld conversion specifier 242
- %llu conversion specifier 242
- %lu conversion specifier 242
- %p conversion specification **367**
- %p conversion specifier **511**
- %s conversion specification **126**
- %s conversion specifier 395, 511, **522**

%u conversion specifier 242, **506**  
 %X conversion specifier 520  
 %zu conversion specification **301**  
 ^ inverted scan set 523  
 ^, bitwise exclusive OR operator 549  
 ^=, bitwise exclusive OR assignment operator 557  
 + flag 515, **516**  
 +, unary plus operator **165**  
 ++, increment operator **163**, **165**, 386  
 +=, addition assignment operator 162, **165**  
 <, less than operator 121  
 <<, left-shift operator 549, 555  
 <<=, left-shift assignment operator 557  
 =, assignment operator **115**, **165**  
 -, unary minus operator **165**  
 --, decrement operator **163**, **165**, 386  
 -=, subtraction assignment operator **165**  
 ->, structure pointer operator 540  
 >, greater than operator 121  
 >>, right-shift operator 549, 555  
 >>=, right-shift assignment operator 557  
 |, bitwise inclusive OR operator 549  
 |=, bitwise inclusive OR assignment operator 557  
 ||, logical OR operator **205**, 271  
 ~, bitwise complement operator 549, 555

## Numerics

0 Conversion specifier 114, 521, 522  
 0x 517

## A

a file open mode 600  
 a.out 74  
 a+ file open mode 600  
 ab file open mode 600  
 ab+ file open mode 600  
 abnormal program termination 763  
 abort function **745**  
 absolute value 234  
 abstraction **235**  
 accelerometer **57**  
 access privileges 373  
 access violation 114, 443, 510, 511  
 accessibility heuristic 359  
 accounts receivable 226  
 accumulator **418**, 420, 423  
 accumulator overflow **423**  
 action **110**, **121**, **138**, 139  
 action statement 139  
 action symbol **141**  
 action/decision model 143  
 add an integer to a pointer 385  
 addition 58  
 addition assignment operator (+=) **162**  
 addition program 112  
 address 657  
 address of a bit field 560  
 address operator (&) **114**, 310, **366**, 369, 380, 381  
 “administrative” section of the computer 58  
 Advanced string manipulation exercises 483  
 aggregate data types 376, **536**  
 AI 28, 38

alert (\a) 110  
 algebra 117  
 algorithm **138**  
   development 23, 39  
   insertion sort **719**  
   merge sort **722**  
   selection sort **714**  
 \_Alignas keyword 789  
 aligned\_alloc 789  
 aligning 505  
 AlphaGo 102  
 AlphaZero **102**  
 ALU (arithmetic and logic unit) **58**  
 Amazon Alexa 30, 105  
 AMD processors 791  
 American National Standards Committee on Computers and Information Processing 69  
 American National Standards Institute (ANSI) **69**  
 ampersand (&) 114  
 Analog Clock exercise 591  
 analysis **804**  
 analysis of examination results 161  
 analyzability 778  
 AND 549  
 Android  
   operating system **67**  
   smartphone 67  
 animated visualization 34  
 animation in raylib 575  
 Annex K 31, 343  
   remove from C standard 31  
 anomaly detection 98  
 Anscombe’s Quartet 24, 35, 638, 639  
 ANSI 69  
 Apache Software Foundation **66**  
 Apple 66  
   Macintosh 66

- Apple (cont.)
  - Siri 30, 105
  - TV **67**
  - Watch **67**
  - Xcode 19, 22, 51
- Apple M1 processor 791
- applied approach 28
- arc4random function
  - POSIX secure random numbers 275
- area of a circle 183
- argc **756**
- argument **110**, 114
  - of a function **233**
- argument coercion **241**
- arguments **738**
- argv **756**
- arithmetic 75
  - assignment operators 162
  - conversion rules **241**
  - expressions 384
  - mean 119
  - operations 418
  - operators 22, **117**
  - overflow **166**
- arithmetic and logic unit (ALU) **58**
- arithmetic assignment operators
  - =, +=, -=, \*=, /=, and %= 162
- arithmetic average (mean) 325
- ARPANET 88
- array **298**
  - bounds checking 306, 343
  - data structure 25
  - initializer **302**
  - JSON 645
  - notation 390
  - of pointers **392**, 401
- array (cont.)
  - of pointers to functions 416
  - subscript notation 309, 377, 391
- array of strings **392**
- arrow operator (->) **540**
- artificial general intelligence **101**, 105
- artificial intelligence (AI) 22, 24, 38, 101
- as-a-service
  - big data (BDaaS) 89
  - Hadoop (Haas) 89
  - Infrastructure (IaaS) 89
  - platform (PaaS) 89
  - software (SaaS) 89
  - storage (Saas) 89
- ASCII (American Standard Code for Information Interchange) **198**, 462
  - character set **60**, 462
- assembler **64**
- assembly language **63**
- assert macro **745**
- <assert.h> 248, **745**
- assertions 25
- assignment
  - expressions 384
  - operator = **115**, 122
  - statement **115**
- assisting people with disabilities 98
- associativity **119**
- asterisk (\*) **117**
- at\_quick\_exit **787**
- atexit function **760**
- atomic operation 799
- \_Atomic variable 799
- attribute 803
  - of a class **801**
  - of an object 803
- attributes of an object 803
- audible (bell) 519
- auto storage class specifier **260**
- automated
  - closed captioning 98
- automatic storage **260**, 314
- autonomous vehicles **58**
- average 119
  - mean 325
- Awesome C libraries list 71
- B**
- B language 68
- backslash (\) 110, 519, **740**
- bandwidth **57**, 88
- bank account program 615
- bar chart 226, 306
- base 8 number system 451
- base 10 number system 451
- base 16 number system 451
- base case(s) **265**
- basic descriptive statistics
  - 24, 32
  - mean 325
  - median 325
  - mode 326
- BASIC programming language 71
- basic time step (Webots) **436**
- BCPL 68
- BCryptGenRandom function (Microsoft secure random numbers) 275
- BDaaS (Big data as a Service) 89
- behavior
  - of a class **801**
- Bell Laboratories 68
- big data 18, 22, **62**, 97
  - analytics 98
- Big O notation 23, 25, **712**, **713**, 718
- binary 445
- binary (base-2) number system 60, 506, 550
- binary digit (bit) 60
- binary files 24, 599
- binary number 227

binary operator **115**  
 binary search 76, 273, **326**,  
     328, 329, 362  
 binary search tree **675**, 679,  
     680, 688  
 binary-to-decimal conver-  
     sion problem 182  
 binary tree 26, **675**  
     creating and traversing  
         676  
     insert 273  
     sort **679**  
 bit (“binary digit”) **58**, **60**  
 bit field **558**, 559  
 Bitcoin 95, 106, 493  
 bitwise AND (&) operator  
     **549**, 554, 572  
 bitwise AND, bitwise inclu-  
     sive OR, bitwise exclusive  
     OR and bitwise comple-  
     ment operators 552  
 bitwise assignment opera-  
     tors **557**  
 bitwise complement opera-  
     tor (~) 552, 555  
 bitwise exclusive OR (^) op-  
     erator **549**, 554  
 bitwise inclusive OR (|) op-  
     erator **549**, 554  
 bitwise operators 26, 549  
 bitwise shift operators 555  
 bitwise XOR **549**  
 Bjarne Stroustrup 71  
 blank 143  
 block 109, **146**, 238  
 block of data 468  
 block scope **262**  
 blockchain 95, 106  
 body of a function **109**, **123**  
 body of a while 148  
 Böhm and Jacopini 140  
 \_Bool **781**  
 bool 28, 778  
 bool 779  
 \_Bool Data Type 208

boolean type **208**, **781**  
 Boolean values in JSON  
     645  
 bounds checking 306, 343  
 braces ({} ) 146  
 brain mapping 98  
 branch 699  
     negative 700  
     zero 700, 703, 704, 706  
 branching instructions 422  
 break 199, 203, 204, 205,  
     229  
 Brick Game exercise 591  
 brute force computing 101  
 bubble sort **319**, 325, 378,  
     380, 381, 398  
     with pass-by-reference  
         378  
 bucket sort 732  
 buffer overflow 310, 343  
 build your own computer  
     420  
 building-block approach **70**  
 building your own compiler  
     36, 650, 690, 696, 697,  
     698, 701, 703, 704, 705,  
     707, 709  
 Building Your Own Com-  
     puter case study 28, 33,  
     63, 650  
 building-block approach  
     **803**  
 byte 58, 60, 549

## C

### C

code repositories 29  
 forums 43  
 language 68  
 Language Reference 43  
 Language Reference (Mi-  
     crosoft) 43  
 open-source community  
     29  
 preprocessor **74**, **109**, 736

C (cont.)  
     standard document 26  
     standard ISO/IEC  
         9899:2018 69  
 C standard library 70, 73,  
     **232**, 249, 374  
     functions 22  
     headers 45  
 C# programming language  
     **72**  
 C++ 240  
 C++ programming language  
     71  
 C11 26, 778  
 C11 headers 787  
 C18 26, 84, 778  
 C99 69, **778**  
 C99 headers 779  
 Caesar cipher 488  
 calculations 59, 115, 126  
 California Consumer Priva-  
     cy Act (CCPA) 32, 105  
 call a function **233**, **237**  
 call-by-reference 542  
 call-by-value 542  
 caller **233**  
 calling 237  
 calling function (caller) **233**  
 calloc **765**  
 camel casing **114**  
 cancer diagnosis 98  
 Cannon Game (game-pro-  
     gramming case study)  
     581  
 Cannon Game App exercise  
     enhancements 590  
 card games 411  
 card images 576, 587  
 Card Shuffling and Dealing  
     393, 395, 396, 543  
 caret (^) **524**  
 Carnegie Mellon Universi-  
     ty’s Software Engineering  
     Institute (SEI) 106

- Carnegie Mellon's Software Engineering Institute 37, 788
- carriage return ('\r') 445
- case label **199**, 200, 262
- case sensitive **113**, 151
- case studies 20
- cast 741
- cast operator 154, **156**, 242 (float) 156
- cbrt function 234
- CC2020
  - Paradigms for Future Computing Curricula 37
- CCPA (California Consumer Privacy Act) 30, 105
- ceil function 234
- Celsius 533
- central processing unit (CPU) **58**
- CERT C Coding Standard 106, 403
- CERT Division of Carnegie Mellon University's Software Engineering Institute 39, 106
- Challenge Project: The RSA Problem 502
- char \* 510
- char \*\* 450
- char fundamental type 242
- char primitive type **198**, 443
- CHAR\_BIT symbolic constant **551**
- character **60**
  - set **60**
- character and string conversion specifiers 510
- character array 309, 310
- character constant 375, **442**
- character handling library **444**
- character set 135, **198**, **442**
- character string **110**, 300
- chatbots **632**
- check if a string is a palindrome 273
- check protection 485
- checkerboard 182
- chess 101, 357
- child **675**
- Christopher Marlowe's *Edward the Second* 35, 39
- cimag function **783**
- cipher
  - algorithms 488
  - Caesar 488
  - cryptii.com 488
  - substitution 488
  - Vigenère 492, 493
- ciphertext **488**, 496
- circumference of a circle 183
- cJSON library 643, 647
- Clang compiler 51, 55, 77
- clang-tidy 31
- class **802**
  - instance variable **803**
- class-average problem 149, 154
  - counter-controlled iteration 150
  - sentinel-controlled iteration 155
- classes **801**
- classes in object-oriented languages **269**
- cleartextin cryptography 488
- client application **643**
- Climate at a Glance time series 642
- clock 253
- CloseWindow function (raylib) **579**
- cloud 18, 35, 89, 643
- cloud (cont.)
  - computing **89**
- cloud-based
  - services 35, **89**, 644
  - tools 39
- clusters of computers 100
- code **54**
- code repositories 29
- coding standards 31
- coercion of arguments **241**
- coin tossing 288
- collision detection in raylib 575
- Color type in raylib **577**
- colors in raylib 575
- column 332
- comma operator (,) 271, 272, **333**
- comma-separated list 333
- comma-separated values (CSV) file 24, 35
- Command Line Tool** project in Xcode 81
- Command Prompt window 77, 79
- command-line arguments 25, 584, 647, 756, 757
- comment 108
- commission 177, 352
- Common programming errors 42
- Communications of the ACM* 140
- communications systems 30
- comparing strings **457**
- comparison expressions 384
- compilation error **74**, 209
- compilation process 703
- compile **73**
- compile and run a program in Xcode 82
- compile phase **73**
- compile-time error **74**



- compiler [64](#), 74, 109, 110
  - Apple Xcode (macOS) 51
  - Clang 55, 77
  - GNU gcc 51, 55, 77
  - Microsoft Visual Studio 51
  - Visual Studio Community edition 55, 77
  - Xcode on macOS 55, 77
- compiler optimization [705](#)
- compiling multiple-source-file programs 25
- complement operator (~) [549](#)
- complete algorithm 141
- complex 783
- \_Complex keyword [783](#)
- complex number 28, 782
- complex number [783](#)
- complex numbers 778, 779
- complex.h 779, 783
- complex.h header 234
- component 801
- compound interest 192, 193, 225
- compound literal 26, [780](#)
- compound literals 778
- compound statement [146](#)
- computational thinking 39
- computer dump [421](#)
- computer hardware 17
- computer memory concepts 22
- computer networks 669
- computer program [56](#)
- computer science 23
- Computer Science and Artificial Intelligence Laboratory (CSAIL) 28
- Computer Science Curricula 37, 38
- computer-science topics 25
- computer simulator 420
- computer software 17
- computer vision 98, 105
- computer-vision applications 101
- Computer-Assisted Instruction (CAI) 293, 294
- Computer-Assisted Instruction (CAI): Difficulty Levels 294
- Computer-Assisted Instruction (CAI): Monitoring Student Performance 294
- Computer-Assisted Instruction (CAI): Reducing Student Fatigue 294
- Computer-Assisted Instruction (CAI): Varying the Types of Problems 294
- computers in education 293
- computing the sum of the elements of an array 304
- concatenating strings [457](#)
- concurrent operations 790
- condition [121](#), 205
- conditional compilation 27, [736](#), [741](#)
- conditional execution of preprocessor directives [736](#)
- conditional expression [144](#)
- conditional operator (?:) [144](#), 165
- conditional transfer of control 420
- connector symbol [141](#)
- const keyword [317](#), 373, 376, 392
- constant 691
- constant integral expression [201](#)
- constant pointer 376, 377, 387
- constant pointer to constant data [373](#), 377
- constant pointer to non-constant data [373](#), 376, 377
- constant run time [713](#)
- constraint violation 403
- container (Docker) 39, 54
- continue 203, 205, 229
- control characters [448](#)
- control statement 23
  - nesting 141
  - stacking [141](#), [142](#)
- control structure [140](#), [142](#)
- control variable [186](#), 192
  - increment [187](#)
  - initial value [187](#)
  - name [187](#)
- controller (Webots) [434](#), [436](#), [438](#), [438](#)
- controlling expression in a switch [199](#)
- conversion rules [241](#)
- conversion specification [114](#), 115, [505](#)
  - %c 310
  - %d 114, 115
  - %p [367](#)
  - %s [126](#)
  - %X 520
  - %zu [301](#)
- conversion specifier [505](#), 517
  - 0 (zero) flag 518
  - c [510](#)
  - e and E [508](#)
  - f [508](#)
  - g (or G) [508](#)
  - s [510](#)
- convert lowercase letters to uppercase letters 248
- Cooking with Healthier Ingredients 486
- coprime [497](#)
- copy 249
- copying strings [457](#)
- corpus [631](#)
  - corpora (plural of corpus) [631](#)
- correction 75
- cos function 235
- cosine 235

- count statistic [637](#)
- counter [149](#), 179
- counter-controlled iteration 21, [149](#), 187, 188
- counter-controlled looping 158, 159
- counting letter grades 199
- counting loop 188
- counting word frequencies [632](#)
- CPU (central processing unit) [58](#), 75
- cracking RSA ciphertext 502
- Craigslist 90
- crash a program 443, 510
- “crashing” [153](#)
- creal function [783](#)
- create sentences 481
- creating algorithms 41
- credit limits 225
- credit scoring 98
- crime prevention 98
- CRISPR gene editing 98
- crop yield improvement 98
- crossword puzzle generator 487
- crowdsourced data [99](#)
- cryptocurrency 95, 96, 106, 493
- cryptography 33, 488
  - cleartext 488
- Cryptography API
  - Next Generation (Microsoft) 275
- CSV (comma-separated value) file 24, 35
- .csv filename extension 640
- <Ctrl> c 763
- <ctype.h> header file 444, 248, 740
- cube a variable
  - using pass by reference 370
  - using pass by value 369
- cube root function 234
- current technology trends 22
- custom functions 23
- custom header 248
- customer
  - churn 98
  - retention 98
  - service agents 98
- Cyberbotics Ltd. 425
- cybersecurity 40, 98
- Cybersecurity Curricula 37, 40
- D**
- dangling pointer 681
- dangling-else problem 146, 180, 181
- data 56
- data counter (Simple compiler) [703](#)
- data hierarchy 22, [60](#)
- data mining [62](#)
  - Twitter 98
- data munging [634](#)
- data samples 638
- data science 22, 23, 24, 40, 98, 636, 637
  - get to know your data 636
  - use cases 98
- data science curriculum proposal 39
- data scientist 634
- data structure 25, 40, 650
- data visualization 98
- data wrangling [634](#)
- database 39, [61](#)
- data-interchange format
  - JSON 644
- dataset 638
- date 248
- \_\_DATE\_\_, predefined symbolic constant 745
- deallocate memory 652
- debug 140
- Debug area (Xcode) 82
- debugging 26, 803
- debugging (online appendices) 26, 44
- decimal 229, 445, 451
  - digit [60](#)
- decision [121](#), 126
- decision making 22
- decision symbol 143
- deck of cards 392
- decomposition 236
- decrement [187](#), 191, 386
- decrement a pointer 385
- decrement operator (--) [163](#)
- decrypt 184
- deep learning 38, 100, 105
- DeepBlue [101](#)
- default case 199, 200
- default precision [157](#), 508
- #define preprocessor directive [303](#), 737
- definite iteration [149](#)
- definition [113](#)
- delimiting characters [467](#)
- DeMorgan's Laws 228
- dependent variable [639](#), 639
- depth of a binary tree 688
- dequeue 668, [669](#)
- dereferencing a pointer [367](#)
- dereferencing a void \*
  - pointer 387
- dereferencing operator (\*) [367](#), 540, 542
- derived data type [537](#), 546
- descriptive statistics 26, 34, [637](#), 637
- design pattern [91](#)
- design process [804](#)
- designated initializer 28, 778, [779](#), 779, 780, 798
- destructive [116](#), [117](#)

determining the length of strings [457](#)  
 developing algorithms 23  
 development environments 44  
 devices 73, 76  
 diagnose medical conditions 101  
 diagnostic medicine 98  
 diagnostics 248  
 diameter of a circle 183  
 diamond symbol [143](#)  
 dice rolling 251, 307  
   simulation 29  
   using arrays instead of switch 307  
 dictionary 629  
 die-rolling simulation 574, 583  
 differential wheels (Webots) [428](#)  
 difftime function (header time.h) 795  
 digit 135  
 Digital Clock exercise 591  
 direct-access files 24, 34  
 directly reference a value 365  
 disk 75  
 displacement [611](#)  
 display  
   a binary tree 689  
   an unsigned integer in bits 550  
   value of a union in both member data types 548  
 divide and conquer [232](#), [235](#)  
 divide by zero [423](#)  
 division 58, 117  
   by zero 76, [153](#)  
 do...while iteration statement 141  
 do...while statement example 202  
 Docker 41, 54  
   container 41, 54

Docker (cont.)  
   Desktop installer 54  
   GNU Compiler Collection (GCC) container 55, 77, 86  
   image 54  
 Docker Hub account 54  
 document a program [108](#)  
 DOS (Disk Operating System) 65  
 dot operator (.) 540  
 (double) cast operator 156  
 double complex 783  
 double-ended queue 687  
 double fundamental type [154](#), 156, 241  
 double indirection (pointer to a pointer) [657](#)  
 double primitive type 193  
 double quote character (") 110  
 double-selection statement [141](#), 159  
 download examples 44  
 DrawGame function in a raylib game 579  
 drawing graphs 226  
 DrawRectangleLines function (raylib) [589](#)  
 DrawTextureEx function (raylib) 589  
 dual-core processor [59](#)  
 dummy value [151](#)  
 dump [421](#)  
 duplicate elimination 361, [680](#), 688  
 duration 260, 262  
 dynamic  
   driving routes 98  
   pricing 98  
 dynamic animated visualization 33  
 dynamic array [765](#)  
 dynamic data structure 23, 364, [650](#)

dynamic memory  
   allocation 25, 765  
 dynamic memory management 364, [652](#)

## E

Eclipse Foundation [66](#)  
 edit phase [73](#), 75  
 editor [73](#), 442  
**Editor** area (Xcode) 82  
*Edward the Second* 35, 39, 633, 636  
 EEPs (examples, exercises and projects) 28  
 efficiency of  
   insertion sort 722  
   merge sort 727  
   selection sort 718  
 Eight Queens 273, 360, 362  
   Brute Force approach 361  
 electronic health records 98  
 element of an array [298](#), 299  
 element positions in raylib 578  
 #elif [742](#)  
 ellipsis (...) in a function prototype [754](#)  
 #else [742](#)  
 emacs 73  
 e-mail (electronic mail) 88  
 embedded system 32, [56](#), 66, [69](#)  
 Embedded Systems Programming case study 32  
 emotion detection 98  
 employee identification number 61  
**Empty Project** template 77  
 empty statement 123  
 encrypt 184  
 “end of data entry” 151  
 end-of-file 199, 444, 453, [594](#), 597, 598  
 #endif [742](#)



Enforcing Privacy with  
 Cryptography 184  
 English-like abbreviations  
 63  
 enqueue **669**  
*Enter* key 74, **114**, 200  
 enum **259**, 561  
 enumeration 24, **258**, 562  
 enumeration constant **259**,  
**561**, 741  
 enumeration example 562  
 environment 73  
 EOF 198, 199, 444  
 e-puck robot **428**  
 e-puck robot (Webots) **428**  
 e-puck\_avoid\_obstacles  
 controller (Webots) **434**  
 equality and relational oper-  
 ators 387  
 equality operator (==) **121**  
 e-reader device 67  
 <errno.h> 248  
 error 75  
   condition 248  
   fatal 118  
   message 75, 76  
   nonfatal 118  
 error checking (in file pro-  
 cessing) 612  
 #error preprocessor direc-  
 tive **743**  
 escape character **110**, 519  
 escape sequence **110**, 519,  
 533  
 Ethereum 95, 106  
 ethics 30, 39, 106  
 Euler 357  
 Euler's totient function **496**  
 event **762**  
 exabytes (EB) 93  
 exaflops 94  
 exam results analysis 161  
 examination results prob-  
 lem 160  
 examples (download) 44

examples, exercises and  
 projects (EEPs) 28, 31  
 exclusive write mode 600  
 executable image **74**  
 executable program **110**  
 execute **75**  
   a program **57**  
   in parallel 36  
   phase **73**  
 execution-time error 76  
 exit function **760**  
   atexit functions 761  
 EXIT\_FAILURE **760**, 787  
 EXIT\_SUCCESS **760**, 787  
 exp function 234  
 expand a macro **739**  
 explicit conversion **156**  
 exponential complexity **272**  
 exponential format 505, 506  
 exponential function 234  
 exponential notation **508**  
 exponentiation 120  
   modular 499  
 exponentiation operator  
 193  
 expression 196, 201, 238  
 extensible languages 269  
 extern 260, **758**  
 external linkage **759**, 784

## F

f or F for a float **762**  
 fabs function 234  
 Facebook 66  
 Facial Recognition 99  
 factorial 183, 225  
 factorial function 266, 273  
 Fahrenheit temperatures  
 533  
 false boolean value **121**,  
 779  
 fatal error 76, 118, 135, **153**,  
 423, 511  
 fatal logic error 147  
 FCB 595  
 fclose function **598**

fenv.h 779  
 feof function **598**, 613  
 fetch 421  
 fetch the next instruction  
**422**  
 fgetc function **595**, 629  
 fgets function **453**, **595**  
 Fibonacci function 271,  
 273  
 Fibonacci series 269, 289  
 field **61**  
 field width **194**, 505, 512,  
 514, 524  
   inputting data 524  
 fields (Webots) **431**  
 FIFO (first-in first-out) **668**  
 file **61**, **594**  
   name 73  
   scope **262**  
 file control block (FCB)  
**595**  
 file descriptor **595**  
 file-matching program 626  
 file offset **602**  
 file open mode **597**, 600  
 FILE pointer **595**  
 file position pointer **602**  
 \_\_FILE\_\_, predefined sym-  
 bolic constant 745  
 file processing  
   error checking 612  
 files for long-term data re-  
 tention 24  
 filter project templates in  
 Visual Studio 78  
 final value **187**  
 final value of a control vari-  
 able 192  
 find the minimum value in  
   an array 362  
 first-in first-out (FIFO) **668**  
 first refinement **152**, 159  
 Fisher-Yates Shuffling Algo-  
 rithm 546  
 five-card poker 411  
 fixed-point notation 508

flag value [151](#)  
 flagged 699  
 flags [505](#), 515  
 flexible array member [785](#)  
 flexible array members 26,  
     778  
 flipped classroom 40  
 float fundamental type  
     156, 242  
 float type [194](#)  
 <float.h> 248  
 floating point 509  
     number 151, [154](#), 156,  
         157, 158  
     size limits 248  
 floating-point literal [194](#)  
     double by default 194  
 floating-point conversion  
     specifiers 509, 513, 521  
     using 509  
 floating-point suffix  
     f or F for a float [762](#)  
     l or L for a long double  
         [762](#)  
 floating-point types 778  
 floor function 234  
 FLOPS (floating-point op-  
     erations per second) 94  
 flow of control [127](#)  
 flowchart [140](#), 143  
     sequence structure 140  
 flowcharting the do...while  
     iteration statement 203  
 flowline [140](#)  
 Floyd's Triangle problem  
     181-182  
 fmod function 235  
 Folding@home network 94  
 font conventions in this  
     book 41  
 fopen function [597](#)  
 for iteration statement 141,  
     192  
 format control string [114](#),  
     115, 505, 513, 520

formatted input/output  
     model [605](#)  
 formatting 24  
 form-feed character (\f)  
     445  
 forward reference (Simple  
     compiler) [699](#)  
 four V's of big data 97  
 fprintf function [595](#)  
 fprintf\_s function 620  
 fputc function [595](#)  
 fputs function [595](#)  
 fractional parts 156  
 frame-by-frame animation  
     [578](#)  
 frames-per-second 579  
 fraud detection 99  
 fread function [595](#), 607  
 free function 652, 667  
 front of a queue 650  
 fscanf function [595](#)  
 fscanf\_s function 620  
 fseek function [609](#)  
 \_\_func\_\_ predefined identi-  
     fier [786](#)  
 function 23, 70, 74, [109](#),  
     215, [232](#)  
     argument [233](#)  
     body [238](#)  
     call [233](#), 238  
     call and return 248  
     call stack 23  
     call/return mechanism 25  
     caller [233](#)  
     header [238](#), 400, 402  
     invoke [233](#), [237](#)  
     name 237, 261, 274, 398  
     parameter 237, 371, 377  
     pointer 398, [401](#)  
     prototype [194](#), [237](#), 238,  
         240, 261, 371, 381  
     prototype scope 261, [262](#)

function (cont.)  
     return from [233](#), 234  
     scope [262](#)  
 functional-style program-  
     ming 31, 801  
 function-call stack [243](#),  
     376  
 fundamental data types 24  
 fundamental types  
     long double 194  
 fwrite [595](#), 607, 609

## G

game loop [578](#)  
 game playing 99, 249  
 game programming 29, 33  
 Game Programming Case  
     Study 580  
     Cannon Game 581  
     SpotOn Game 580  
 game systems 30  
 "garbage value" [151](#)  
 Gary Kasparov 101  
 gcc compilation command  
     [74](#)  
 GDPR (General Data Pro-  
     tection Regulation) 30,  
     105  
 Gender Neutrality 487  
 general utilities library (st-  
     dlib) [450](#)  
 generating mazes randomly  
     415  
 \_Generic keyword [788](#)  
 generic math 778  
 generic pointer 386  
 generic programming 33,  
     801  
 get to know your data 636,  
     639  
 getc 740  
 getchar [455](#), 629, 740  
 getting questions answered  
     41  
 gigabytes (GB) [58](#), 92

gigaflops 94  
 GitHub 28, 29, 43, **65**  
 global variable 261, 262, 381, 758  
 GNU C Standard Library Reference Manual 43  
 GNU Compiler Collection (GCC) Docker container 22, 55, 77, 86, 793  
 GNU gcc 22, 42, 51, 55, 77  
 GNU Scientific Library **637**, 640  
   `gs1_fit_linear` function **640**  
 gnuplot 37, **637**  
   install 641  
 Go board game 102  
 golden mean 269  
 golden ratio 269  
 good programming practices 42  
 Google Assistant 30, 105  
 Google Maps 90  
 Gosling, James 72  
 goto elimination **140**  
 goto-less programming 140  
 goto statement 140, 262, 767, **767**, 767  
 GPS (Global Positioning System)  
   device 57  
 GPS sensor 100  
 GPU (graphics processing unit) 791  
 graphical user interface (GUI) **66**  
 graphics processing unit (GPU) 791  
 gravity in Webots **429**  
 greatest common divisor 273, 500  
 grouping of operators 299, 368, 557  
`gs1_fit_linear` function (GNU Scientific Library) **640**

guess the number exercise 289  
 GUI (Grahical User Interface) **66**  
 Guido van Rossum 71

## H

Hadoop (Apache)  
   as a Service (HaaS) 89  
 halt 421  
 halt instruction 699  
 hands-on implementation  
   case studies 21  
 hard disk 57  
 hard drive 56, 73  
 hardware 17, 22, 54, 56, 63  
 hardware independent 68  
 hardware platform **69**  
 head of a queue **650**, **668**  
 header (file) **109**, **208**, 247, 737  
   `complex.h` 234, 779, **783**  
   `ctype.h` 444  
   `fenv.h` 779  
   `inttypes.h` 779  
   `stdbool.h` 779, **781**  
   `stdint.h` 779  
   `stdio.h` 453  
   `stdlib.h` 450  
   `string.h` 457  
   `tgmath.h` 779  
 Health Insurance Portability and Accountability Act (HIPAA) 30  
 health outcome improvement 99  
 heuristic 359  
 hexadecimal 227, **445**, 451, 505, 506  
 hexadecimal integer 367  
 high-level language **64**  
 highest level of precedence 118  
 high-order bit 551

High-performance card  
   shuffling and dealing simulation 543, 544  
 HIPAA (Health Insurance Portability and Accountability Act) 30, 105  
 Histogram printing 306  
 hook (Simple compiler) **700**  
 horizontal tab (`\t`) 110, 445  
 HTML (HyperText Markup Language) **89**  
 HTTP (HyperText Transfer Protocol) **89**  
 HTTPS protocol 488  
 human genome sequencing 99  
 HyperText Markup Language (HTML) **89**  
 HyperText Transfer Protocol (HTTP) **89**  
 hypotenuse of a right triangle 286

## I

IBM DeepBlue **101**  
 IBM Watson 30, **101**, 105  
 identifier(s) **113**, 738  
 identity theft prevention 99  
`#if` **742**  
   if selection statement **121**  
   if statements, relational operators, and equality operators 122  
   if...else selection statement 141, 144  
`#ifdef` preprocessor directive **742**  
`#ifndef` preprocessor directive **742**  
 image **74**  
 image (Docker) **54**  
 immunotherapy 99

- immutable (not modifiable) 444
- implicit conversion [156](#)
- in parallel [790](#)
- #include preprocessor directive [737](#)
- including headers 248
- increment a control variable [187](#), 192
- increment a pointer 385
- increment operator (++) [163](#)
- incremented 386
- indefinite iteration [151](#)
- indefinite postponement [394](#), 412, 545
- indentation 143, 146
- independent variable [639](#), 639
- index (subscript) [299](#)
- indirection [365](#), 369
- indirection operator (\*) [367](#), 369
- indirectly reference a value 365
- infinite loop [148](#), 156, 190
- infinite recursion 268
- infix notation [691](#)
- infix-to-postfix conversion [691](#)
- Infix-to-Postfix Converter exercise 691
- information hiding [262](#), 380
- Information Revolution [57](#)
- information technology (IT) 62
- Information Technology Curricula 37
- Information Technology Curricula 2017 40
- Infrastructure as a Service (IaaS) 89
- inheritance [803](#)
- InitGame function in a raylib game 578
- initial value of a control variable [187](#), 192
- initialization phase [154](#)
- initialize [113](#)
- initializer list 309, 333
- Initializing multidimensional arrays 333
- initializing structures 540
- Initializing the elements of an array to zeros 301
- Initializing the elements of an array with an initializer list 302
- InitWindow function (raylib) [578](#)
- inline function 778, 786
- inner block 262
- innermost pair of parentheses 118
- inorder traversal of a binary tree 273, [676](#), 679
- input 24
- input device [57](#)
- input events in raylib 575
- input unit [57](#)
- input/output operators 418
- inputting data with a field width 524
- inserting literal characters 505
- insertion sort algorithm [719](#), 720, 722
- instance [802](#)
- instance variable [803](#)
- instantiation 802
- instruction 75, 694
- instruction counter 703
- instruction execution cycle [421](#), [421](#)
- Instructor Solutions Manual 44
- instructor supplements 44
- int type 109, [113](#), 242
- integer 109, 113
- integer array 298
- integer constant 377
- integer conversion specifiers [506](#)
  - using 506
- integer division [118](#), [156](#)
- integer promotion 242
- integer suffix
  - l or L for a long int [762](#)
  - ll or LL for a long long int [762](#)
  - u or U for an unsigned int [762](#)
- integral size limits 248
- integral types
  - portable 779
- integrated development environments (IDEs) 73
- Intel processors 791
- intelligent assistants 30, 99, 105
  - Amazon Alexa 30, 105
  - Apple Siri 30, 105
  - Google Assistant 30, 105
  - IBM Watson 30, 105
  - Microsoft Cortana 30, 105
- intelligent virtual assistants [631](#)
- interactive attention signal 763
- interactive computing [115](#)
- intercept [639](#)
- Interface Builder 66
- interlanguage translation [632](#)
- internal linkage [759](#), 784
- International Standards Organization (ISO) [69](#)
- Internet [88](#), 643
- Internet bandwidth 17
- Internet of Things (IoT) 18, 67, 90, 97, 100, 106
  - medical device monitoring 99
- Internet Protocol (IP) [88](#)
- interpreter [64](#)
- interrupt 763

intersection of sets 354  
 Intro to Data Science  
   Dynamic Visualization of  
     Coin Tossing 590  
   Dynamic Visualization of  
     Rock, Paper, Scissors  
     Game Statistics 591  
   Dynamic Visualization of  
     Rolling Two Dice 591  
 inttypes.h 779  
 invalid operation code **423**  
 inventory 628  
 Inventory Control 99  
 inverted scan set **524**  
 invoke a function **233, 237**  
 iOS 65, **67**  
 IoT (Internet of Things) 97  
 IP address **88, 90**  
 iPad **67**  
 iPadOS **67**  
 iPhone **67**  
 isalnum function **445**  
 isalpha function **445**  
 isblank function 445  
 iscntrl function **448**  
 isdigit function **445**  
 isgraph function **448**  
 islower function **447**  
 ISO (International Standards Organization) **69**  
 ISO/IEC 9899  
   2018 (C standard document) 69  
 isprint function 445, **448**  
 ispunct function 445, **448**  
 isspace function 445, **448**  
 Issue navigator **82**  
 isupper function **447**  
 isxdigit function **445**  
 iteration 272  
 iteration statement 24, **140, 142, 148**  
 iterative function 328

## J

Jacopini, G. 140

Java programming language  
   67, 72  
 JavaScript 72  
 Jobs, Steve 66  
 JSON (JavaScript Object  
   Notation) 36, 100, **644**  
   array 645  
   Boolean values 645  
   cJSON library 643, 647  
   data-interchange format  
     644  
   false 645  
   JSON object 644  
   null 645  
   number 645  
   string 645  
   true 645

## K

kernel of an operating system **65**  
 Kernighan, B. W. 69  
 key value **326**  
 keyboard 56, 112, 114, 453  
 keywords **124**  
   added in C11 124  
   added in C99 124  
 Knight's Tour 357  
   Brute Force approaches  
     360  
   Closed tour test 361  
 Kotlin programming language 67

## L

l or L suffix for a long double literal **762**  
 l or L suffix for a long int literal **762**  
 label 262, **767**  
 language identification **632**  
 language translation 99  
 larger of two numbers 176  
 largest number problem  
   134

last-in, first-out (LIFO)  
   **243, 663**  
 Law of Large Numbers 583  
 leading asterisks 485  
 leaf node **675**  
 least access privilege 377,  
   378  
 least common multiple **501**  
 left align 194  
 left child **675**  
 left justify 198, **505**  
   strings in a field 516  
 left justify in a field 516  
 left subtree **675**  
 left-shift operator (<<) **549, 572**  
 legacy code 373  
 lemmatization **632**  
 length modifier **506**  
 letter **60**  
 level order binary tree traversal **689**  
 lexicographical comparison  
   462  
 libcurl library 643  
 libcurl library (for invoking  
   web services) 646  
 libcurl open source library  
   35  
 library function 70  
 LIFO (last-in, first-out)  
   243, **663**  
 limerick exercise 481  
 <limits.h> header 248, 551  
 line number 694, 698  
 \_\_LINE\_\_, predefined symbolic  
   constant 745  
 #line preprocessor directive  
   **744**  
 linear data structure **653**  
 linear regression **639**  
 linear relationship **639, 639**  
 linear run time **713**  
 linear search 273, **326, 327, 362**



link (pointer in a self-referential structure) **651**  
 link phase **73**  
 linkage 260  
 linkage of an identifier **260**  
 linked list 364, 536, **650**, **653**  
 linked lists 25  
 linker **74**, 110, 758  
 linker error 758  
 linking **74**  
 links 653  
 Linux 73, 74, 754  
   shell prompt 55, 77  
 Linux operating system **65**  
   kernel **66**  
 literal **110**  
   floating point **194**  
 literal characters **505**  
 live-code approach 17  
 live-code examples 42  
 ll or LL suffix for a long  
   long int literal **762**  
 -lm command line option  
   for using the math library  
   194  
 load 707  
 load a program into memory **418**  
 load phase **73**  
 load/store operations 418  
 loader **75**  
 loading **75**  
 loading phase **423**  
 local variable **237**, 238, 260,  
   261, 312  
 locale 248  
 <locale.h> header 248  
 location 116  
 location-based services 99  
 log function 234  
 log10 function 234  
 log<sub>2</sub>*n* comparisons 680  
 logic error 147, 151, 190,  
   210, 304, 547

logical AND operator (&&) **205**, 551  
 logical decision 56  
 logical negation (NOT) operator (!) **205**, 207  
 logical operators 23, **205**  
 logical OR operator (||) **205**  
 logical page 519  
 logical unit **57**  
 Logo language 356  
 long 201  
 long double 242  
 long double fundamental  
   type 194  
 long int 242  
 long long int 242  
 loop 148, 151, 157, 189  
 loop continuation condition **186**, 188, 189, 202  
 loop-continuation condition 187, 190  
 lowercase letter 61, 135, 248  
 low-order bit 551  
 lvalue ("left value") **210**,  
   299

## M

M1 processor (Apple) 791  
 machine dependent 63  
 machine independent 68  
 machine language **63**, 74  
   programming 33, 63, 417  
 machine learning 28, 35, 38,  
   100, 636, 640  
 Macintosh 66  
 macOS **66**  
 macro 248, **736**, **738**  
   arguments 739  
   complex **783**  
   definition 739  
   expansion 739  
   identifier **738**  
   variable-length argument  
   list 784

main 109  
 main window in Visual Studio 78  
 malloc function **652**, 765  
 Malware Detection 99  
 "manufacturing" section of  
   the computer 58  
 marketing  
   analytics 99  
 mashup 35, **90**  
 mashups 643  
 mask **551**  
 massively parallel processing  
   100  
 master file 626  
 math library functions 248,  
   292  
 <math.h> header file 194,  
   234, 248  
 mathematics 39  
 matrix multiplication 356  
 maximum 179  
 maximum 239  
 maximum statistic **637**  
 maze traversal 273, 415  
 mazes of any size 415  
*m*-by-*n* array **333**  
 mean 321  
 mean (average) 32, 325  
 measures of central tendency **637**  
 measures of dispersion **637**  
   standard deviation 637  
   variance 637  
 measures of variability **637**  
 median 32, 321, 325  
 megabytes (MB) 92  
 member of a struct 537  
 members **537**  
 memchr function 469, **471**  
 memcmp function 469, **470**  
 memcpy function 468, **469**,  
   785  
 memmove function **470**  
 memory 57, 58, 75  
   unit **58**

memory addresses 365  
 memory alignment control 778  
 memory allocation 248  
 memory boundaries 538  
 memory footprint 707  
 memory functions of the string handling library 468  
 memory utilization 558  
 memset function 469, **471**  
 menu-driven system 401  
 merge sort algorithm **722**, 723, 727  
 merge two arrays 722  
 message **110**  
 method **802**  
   call **803**  
 metric conversion program 485  
 Microsoft 31  
   Cortana 30, 105  
   Visual C++ 22, 31  
   Visual Studio Community edition 51, 73, 77  
 Microsoft's Cryptography API  
   Next Generation 275  
 minimum statistic **637**  
 minimum value in an array 273  
 MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) 28  
 MIT Project MAC 30  
 mixed-type expressions **242**  
 mobile application 67  
 mode 32, 321, 326  
 modular architecture of this book 18  
 modular exponentiation 499  
 Moore's Law **56**, 97  
 mortgage problem 177  
 motion information 57  
 mouse 56

Mozilla Foundation **66**  
 multicore computers 28  
 multicore processor **36**, **59**  
 multicore systems 31, 36, 791  
 multidimensional array 23, 332, 333, 335  
   initialize 333  
 multiple selection statement **141**, 199  
 multiple-source-file programs 260, 261, 758, 759  
 multiples of an integer 182  
 multiplication 117  
 multiplicative operators 156  
 multiply two integers 273  
 Multipurpose sorting program using function pointers 398  
 multithreaded applications 36  
 multithreading 25, 26, 30, 36, 669, **790**  
   -pthread option 795  
 multivariate time series **642**  
 mutex (multithreading) 799

## N

n factorial (n!) 266  
 name 187, **299**  
 name of a control variable **187**  
 name of a variable 116, 117  
 name of an array 299  
 named entity recognition **632**  
 National Oceanic and Atmospheric Administration (NOAA) 642  
 natural language 100, **630**  
 natural language processing (NLP) 24, 28, 34, 100, **631**, **632**  
   case study 24

natural language translation 99  
 natural logarithm 234  
 Navigator area (Xcode) 82  
   Issue **82**  
   Project **82**  
 nested **159**  
 nested building block 213  
 nested control statement 158  
 nested if...else statement **145**, 146  
 nested parentheses **118**, 119  
 nesting **158**  
 nesting rule **213**  
 neural network 102  
 new pharmaceuticals 99  
 newline (\n) **110**, 143, 310, 443, 444, 445, 525  
 NeXTSTEP operating system 66  
 n-grams **632**  
 node (Webots) **431**  
 NodeJS 72  
 nodes 652, **653**  
 non-constant pointer to constant data **373**, 375  
 non-constant pointer to non-constant data **373**  
 nondestructive **117**  
 nonfatal error 76, 118, 135, 147, 241  
 nonlinear data structure 675  
 nonrecursive function 289  
 \_Noreturn function specifier 778, **788**  
 not modifiable (immutable) 444  
 Notepad++ text editor 576  
 noun phrase extraction **632**  
 NULL 365, 387, 391, **597**, 658

null character ('\0') **309**,  
310, 375, 391, 443, 693  
null in JSON 645  
NULL pointer 651, 766  
null-terminated string 392  
number systems 26  
numbers in JSON 645  
numeric codes **461**

## O

$O(1)$  **713**  
 $O(n \log n)$  time **728**  
 $O(n)$  time **713**  
 $O(n^2)$  time **714**, **718**, 722  
object **801**  
object code **74**  
object-oriented analysis and  
design (OOAD) **804**  
object-oriented languages  
20, 26, 31, 804  
object-oriented program-  
ming (OOP) 31, **66**, 71,  
235, **804**  
terminology and con-  
cepts 26  
object program 110  
Objective-C 66  
object-oriented languages  
801  
object-oriented program-  
ming 801  
observations in a time series  
**642**  
octa-core processor **59**  
octal number 227, 445, 451,  
505, 506  
off-by-one error **190**  
off-screen buffer **578**  
offset **387**, **611**  
one-dimensional array 380,  
393  
one's complement **555**  
online C forums 43  
OOAD (object-oriented  
analysis and design) 804  
OOP (object-oriented pro-  
gramming) **804**  
open file table **595**  
open source **65**, 67  
increases productivity  
**65**  
libraries 99  
movement 19  
software 30, 99  
OpenAI 66  
OpenCV **66**  
OpenML **66**  
OpenWeatherMap 37  
Current Weather Data  
644  
One Call API 644  
OpenWeatherMap web ser-  
vice 643  
operand **115**, **418**, 699  
operating system 32, **65**, 66,  
68  
operation code **418**, 699  
operator precedence 124,  
299  
rules **118**  
operator precedence chart  
773  
Operator sizeof when ap-  
plied to an array name re-  
turns the number of bytes  
in the array 382  
operators 22, 162  
optimize (Simple compiler)  
707  
optimized code 708  
optimizing the simple com-  
piler 707  
order **138**, 140, 142  
order of evaluation of oper-  
ands 271  
ordinary least squares **640**  
orientation information 57  
OS X 66  
outer block 262, 265  
out-of-bounds array ele-  
ments 343  
output device **58**  
output unit **58**  
oval symbol **141**  
overflow 763  
overlapping regions of  
memory 785  
overtime pay problem 178

## P

$\pi$  227  
packets **88**  
padding **561**  
page layout software 442  
palindrome 362  
parallel **790**  
parallel operations 790  
parallel threads 36  
parameter **237**  
parameter list **238**, 277  
parameter of a function  
**237**  
parameter types 381  
parent node **675**  
parentheses () 118, 124  
partitioning step of Quick-  
sort 732  
parts-of-speech (POS) tag-  
ging **632**  
pass-by-reference 23, 314,  
315, 364, **369**, 371, 380  
pass-by-value **369**, 371  
passing an array 316  
passing an array element  
316  
Passing arrays and individu-  
al array elements to func-  
tions 316  
pattern of 1s and 0s 60  
percent sign (%) **117**  
perfect number 288  
performance 42, 70  
performance requirements  
261  
performance tuning 803

- performance-intensive systems 32
- performing operations
  - concurrently **790**
- persistent storage **59**
- personal assistants 99
- personalized medicine 99
- personally identifiable information (PII) 105
- petabytes (PB) 93
- petaflops 56, 94
- phases of basic programs
  - initialization phase **154**
  - processing phase **154**
  - termination phase **154**
- phishing 630
- phishing elimination 99
- Phishing Scanner 630
- physics engine (Webots) **434**
- physics in Webots 440
- Pig Latin exercise 481
- plaintext 496
- Platform as a Service (PaaS) 89
- pointer 25, **364**, 366, 368, 369
  - arithmetic 385, 386, 388, 483
  - arrow (->) operator 540
  - comparisons 387
  - expression 387, 388
  - notation 371, 388, 390
  - parameter 370
  - subscripting **388**
  - to a function **398**
  - to a pointer (double in-direction) 452, **657**
- pointer (cont.)
  - to a structure **540**
  - to void (void \*) **386**
  - variable 377
- pointer/offset notation **388**
- pointer/subscript notation 388
- poker 411
- poll 304
- pollution reduction 99
- polynomial 120
- pop 663
- pop off a stack **243**
- portability 70
- portable 70
- portable code **68**, 70
- portable integral types 779
- position number **299**
- postdecrement **163**
- postfix evaluation 700
- postfix-expression evaluation algorithm **691**
- Postfix Expression Evaluator exercise 693
- postfix increment and decrement operators 163
- postfix notation **691**
- postincrement **163**
- postorder traversal **676**, 679, 680
- postorder traversal of a binary tree 273
- pow (power) function 120, **194**, 234
- power 234
- PowerPoint slides 46
- #pragma processor directive 743, **743**
- precedence **118**, 299, 368
  - of arithmetic operators 24
- precedence of arithmetic operators 124
- precision **156**, 194, 505, 506, 508
  - default 508
- precision for integers, floating-point numbers and strings 513
- precision medicine 99
- predecrement operator **163**
- predefined symbolic constants **745**
- predicate function **661**
- predicted value in simple linear regression 639
- predicting
  - disease outbreaks 99
  - weather-sensitive product sales 99
- predictive analytics 99
- prefix increment and decrement operators 163
- preincrement 163
  - operator **163**
- preorder traversal of a binary tree 273, **676**, 679
- preprocess phase **73**
- preprocessor 27, **74**, 248, 736
- preprocessor directive **74**, 736, **737**, **740**
- preventative medicine 99
- preventing
  - disease outbreaks 99
- primary memory **58**
- prime number 288, 500
- principle of least privilege **262**, 317, 370, 373, 376, 381
- print
  - trees 689
- print a hollow square 182
- print a linked list backwards 273
- print a square 181
- print a string backwards 273, 362
- print an array 273, 362
- print an array backwards 273
- print characters 447

print patterns 225  
 printf **504**  
 printf function 110  
 printing a string input at the keyboard backwards 273  
 Printing a string one character at a time using a non-constant pointer to constant data 375  
 printing character **448**  
 printing dates in various formats 484  
 printing multiple lines with a single printf 111  
 printing one line with two printf statements 111  
 printing positive and negative numbers with and without the + flag 516  
 privacy 30, 105, 488  
 private decryption key **494**  
 private key **494**, 496, 498  
 probability 250  
 problem solving 23, 26  
 procedural programming 31, 801  
 procedure **138**  
 Processing a queue 669  
 processing phase 152, **154**, 157  
 processing unit 56  
 product 132  
 production (Simple compiler) **707**  
 program **56**  
 program control **139**  
 program execution stack **243**  
 Program to simulate the game of rock, paper, and scissors 255–258  
 ProgrammableWeb 37, 90, 648  
 programmer **56**

Programmer-defined maximum function 239  
 programming fundamentals 18, 39  
 programming paradigms 31, 801  
 Project Gutenberg 29, 633  
 project in Visual Studio **77**  
 project in Xcode **81**  
 Project MAC (MIT) 30  
 Project navigator **82**  
 promotion **242**  
 prompt **114**  
 proprietary software **65**  
 protecting the environment 99  
 PROTO nodes (Webots) **432**  
 pseudo-random numbers **252**  
 pseudocode **139**, 154, 157  
 -pthread option (multithreading) 795  
 public domain  
   card images 576, 587  
   images 588  
 public-domain card images 587  
 public encryption key **494**, 497  
 public key **494**, 496, 497  
 public-key cryptography 35, **493**, **495**  
 public-key/private-key pair 496  
 push 663, 666  
 push onto a stack **243**  
 putchar **453**  
 puts 455, 629  
 puts function **126**  
 Pythagorean Triples 228  
 Python 71  
 Python Software Foundation **66**

**Q**  
 quad-core processor **59**  
 quadratic run time **714**  
 quantum computers 95  
 questions  
   getting answered 43  
 queue 25, 364, 536, **650**, **668**, 669  
 quick\_exit function 778  
 quicksort 273, 732  
  
**R**  
 r file open mode 600  
 R programming language 71, 73  
 r+ file open mode 600  
 radians 235  
 radius 183  
 raise **763**  
 raising an integer to an integer power 273  
 ralib game-programming library  
   element positions 578  
 RAM (Random Access Memory) **58**  
 rand 249  
 RAND\_MAX 250, 253  
 random function POSIX secure random numbers 275  
 random number 248  
   generation 23, 32  
 random number generation 393, 481  
 random-access file **606**, 609  
 randomizing **252**  
 range checking **216**  
 range statistic **637**  
 raylib cheat sheet **577**  
 raylib game-programming library 33, **574**  
   animation 34, 575



- raylib game-programming (cont.)
  - C programming demos 575
  - Cannon game 581
  - CloseWindow function 579
  - collision detection 36, 575
  - color constants 577
  - colors 575
  - Color type 577
  - custom types 577
  - DrawGame function 579
  - DrawRectangleLines function 589
  - DrawTextureEx function 589
  - frame-by-frame animation 578
  - game loop 578
  - InitGame function 578
  - InitWindow function 578
  - input events 34, 575
  - Law of Large Numbers 583
  - Rectangle type 577
  - rFXGen online sound-effect generator 580
  - RGBA color 577
  - sample games 575
  - shapes 34, 575
  - SetTargetFPS function 578
  - Sound type 577
  - sounds 34, 575
  - types 577
  - UnloadGame function 579
  - UpdateGame function 579
  - Vector2 type 577
  - WindowShouldClose function 579
- raylib.h header 577
- rb file open mode 600
- rb+ file open mode 600
- read 699
- readability 123, 631
- reading and discarding characters from the input stream 525
- reading characters and strings 523
- reading input with floating-point conversion specifiers 522
- reading input with integer conversion specifiers 521
- readLine function (non-standard) 444
- real-time systems 32
- real-world data 39
- realloc 765
- “receiving” section of the computer 57
- recommender systems 99
- record 61, 376, 596
- record key 596
- rectangle 143
- rectangle symbol 141
- Rectangle type in raylib 577
- RectangleArena (Webots) 427, 431
- recursion 23, 265, 271
  - recursion step 266
  - recursive call 266
  - recursive definition 266
  - recursive function 265
  - recursive function gcd 291
  - recursive function power 289
  - vs. iteration 272
- recursion examples
  - binary search 273
  - binary tree insert 273
  - check if a string is a palindrome 273
  - Eight Queens 273
- recursion examples (cont.)
  - Factorial function 273
  - Fibonacci function 273
  - Greatest common divisor 273
  - inorder traversal of a binary tree 273
  - linear search 273
  - maze traversal 273
  - minimum value in an array 273
  - multiply two integers 273
  - postorder traversal of a binary tree 273
  - preorder traversal of a binary tree 273
  - print a linked list backwards 273
  - print a string backwards 273
  - print an array 273
  - print an array backwards 273
  - printing a string input at the keyboard backwards 273
  - quicksort 273
  - raising an integer to an integer power 273
  - recursive main 273
  - search a linked list 273
  - selection sort 273
  - sum of the elements of an array 273
  - Towers of Hanoi 273
  - visualizing recursion 273
- recursive prime problem 291
- recursive search of a list 688
- recursive main 273
- recursive selection sort 731
- recursive step of Quicksort 732
- reddit 43

redirect input or output 504, 505  
 reducing carbon emissions 99  
 redundant parentheses **120**  
 refactoring **91**  
 register 260  
 regression line **639**  
 reinforcement learning 102  
 reinventing the wheel 232  
 relational database **61**  
 relational operators **121**  
 reliable integer division 778, 785  
 remainder 235  
 remainder operator (%) **117**, 134, 250  
 repeatability 252  
 replacement text **303**, 738  
 representational error in floating point **194**  
 Representational State Transfer (REST) **644**  
 reproducibility 41, 42, 105  
 request to a web service **643**  
 requirements 261  
 requirements statement **804**  
 research and project exercises 28  
 reserved word **124**  
 response from a web service **643**  
 RESTful web services **644**  
 restrict 784  
 restricted pointer **784**  
 restricted pointers 778  
 return 369  
 return a result 109, 237  
 return from a function **233**, 234  
 return key 74, 422  
 return statement 237, 239  
 return type 381  
 return value type **238**, 277

reusable software components **801**  
 reuse 802  
 rewind function 602  
 rFXGen online sound-effect generator (raylib) **580**  
 RGBA (red, green, blue, alpha) color **577**  
 Richards, Martin 68  
 ride sharing 99  
 right align in a field 194, 512  
 right brace { } **109**  
 right child **675**  
 right justify in a field **505**, 512, 516  
 right subtree **675**  
 right-justifying integers in a field 512  
 right-shift (>>) operator 572  
 rise-and-shine algorithm 138  
 risk minimization 99  
 risk monitoring and minimization 99  
 Ritchie, D. 68  
 robo advisers 99  
 robot  
   e-puck **428**  
 robotics simulations 28, 32  
 robotics simulator 424, 440  
 Robotics with Webot Simulator 32  
 roll a six-sided die 251  
 rock, paper and scissors game 254, 355  
 root node of a binary tree **675**, 689  
 rounded **157**  
 rounding 132, 265, **505**  
 rounding a number 195  
 rounding toward negative infinity 785  
 rounding toward zero 785

rows 332  
 RSA algorithm 33  
 RSA ciphertext cracking 502  
 RSA Problem 502  
 RSA Public-Key Cryptography algorithm **494**  
 rules of operator precedence **118**  
 runtime constraint 403  
 runtime error 76  
*rvalue* ("right value") **210**

## S

sales tax problem 176  
 samples (in datasets) 638  
 Satya Nadella 30  
 savings account example 192  
 scalar **315**, 380  
 scaling **250**  
 scaling factor **250**, 254  
 scan characters 521  
 scan set **523**  
   inverted 524  
 scanf **504**  
 scanf function **114**  
 scanf\_s function 33, 343  
 scanning images 57  
 scene tree (Webots) **431**, 431  
 science, technology, engineering and math (STEM) 18  
 scientific computing 71  
 scientific notation **508**  
 scope of an identifier **260**, 262, 740  
 Scoping example 263  
 screen 56, 58, 76  
 SDK (Software Development Kit) **91**  
 search a linked list 273

- search functions of the
  - string handling library 462
- search key **326**
- search strings 462
- searching **326**, 328
  - arrays 23
- searching a binary tree 680
- searching strings 457
- second refinement **152**, 153, 160
- secondary storage 57
  - device 73
  - unit **59**
- secure C 125
- Secure C Programming
  - sections 37
- Secure Coding in C and C++, 2/e* 216
- secure random numbers
  - arc4random function 275
  - BCryptGenRandom function 275
  - random function 275
- security 33, 37, 41, 488
- security vulnerabilities 126, 526
- seed 253
- seed the rand function **252**
- SEEK\_CUR **612**
- SEEK\_END **612**
- SEEK\_SET **611**, 612
- segmentation fault 114, 443, 510
- SEI (Carnegie Mellon University's Software Engineering Institute) 106
- SEI CERT C Coding Standard 106, 125
- selection sort 273, 353, 731
  - recursive 731
- selection sort algorithm **714**, 715, 718
- selection statement 23, 142
- selection structure **140**, **142**
- Self Check exercises 29
- self documenting 113
- self-driving cars 99, 101
- self-referential structure **537**, **651**
- semicolon (;) 110, 123
- send a message to an object 803
- sentiment analysis 99, **631**
- sentinel-controlled iteration 23, 153
- sentinel value **151**, 153, 154, 176
- sequence structure **140**, **142**
- sequence structure flow-chart 140
- sequential access file 596
- sequential execution **140**
- sequential file 596
- service-oriented architecture (SOA) **89**
- <setjmp.h> 248
- SetTargetFPS function (raylib) **578**
- Shakespeare 35, 633
- shapes in raylib 575
- share memory (union) 546
- sharing economy 99
- shell prompt on Linux 55, 77
- shift **250**
- Shifted, scaled integers
  - produced by `1 + rand() % 6` 250
- shifting value **254**
- "shipping" section of the computer 58
- short 201, 241
- short-circuit evaluation **207**
- sibling **675**
- side effect **249**, 261, 271
- Sieve of Eratosthenes 361
- SIGABRT 763
- SIGFPE 763
- SIGILL 763
- SIGINT 763
- sign bit 506
- signal **763**
- signal handling 25, 763
  - library **763**
- signal value **151**
- <signal.h> 248, **763**
- signed decimal integer 506
- SIGSEGV 763
- SIGTERM 763
- silicon 56
- similarity detection 99, **632**, 633
- Simple
  - made up programming language 690
- Simple compiler
  - symbol table **698**
- Simple compiler **699**
  - case study 36, 697
  - data counter **703**
  - enhancements 708
  - first pass 698
  - hool **700**
  - optimize 707
  - production **707**
  - second pass 698
  - token **698**
  - unresolved forward reference 699
- Simple programming language 694
- simple condition 206
- simple interest problem 178
- simple linear regression 24, 35, 640
- simplest flowchart 212
- Simpletron 629

- Simpletron Machine Language (SML) 33, 36, [417](#), 650
- Simpletron simulator 417, 420, 423
  - modifications 423
- Simpletron virtual machine 691, 695
- simulated robots 32
- simulation 28, 249, 393
  - techniques 23, 32
- sin function 235
- sine 235
- single entry/single exit control statement [141](#), 143, 213
- single-selection statement [141](#)
- sinking sort [319](#)
- size\_t [301](#), 458
- sizeof operator [382](#), 538, 629, 652, 741
- slope [639](#)
- smallest number problem 134
- smart cities 99
- smart homes 99
- smart thermostats 99
- smart traffic control 99
- smartmeters 99
- smartphone 67
- SML [417](#), 420, 423
  - instruction [418](#)
- SMS Language 487
- social graph analysis 99
- software 17, 28, 40, 44, [54](#)
- Software as a Service (SaaS) 89
- software-based simulation 33, 417, [420](#)
- Software Development Kit (SDK) [91](#)
- software engineering 205, 262, 381
- Software Engineering Institute (Carnegie Mellon) 37, 788
- Software Engineering Institute (SEI) 106
- software engineering observations 42
- software model [420](#)
- software reuse [70](#), [235](#), 381, 759
- solid-state drive 56, 57
- Solution Explorer** 78
  - add an existing file 78
  - display 78
- solution in Visual Studio [77](#)
- sort algorithms
  - bucket sort 732
  - insertion sort [719](#)
  - merge sort [722](#)
  - Quicksort 732
  - recursive selection sort 731
  - selection sort [714](#)
- sort key 712
- sorting 319, 712
  - arrays 23
- Sound type in raylib [577](#)
- sounds in raylib 575
- source code [64](#)
- space 525
- space flag [516](#), 517
- space–time trade-off [728](#)
- spam
  - detection 99
- Spam Scanner 486
- speaking to a computer 57
- special characters [443](#)
- Special Section: Advanced String Manipulation Exercises 483
- special symbol [60](#)
- speech recognition 40, 100, [631](#)
- speech synthesis 38, 100, [631](#)
- spell checking [632](#)
- spelling correction [632](#)
- split the array in merge sort 722
- SpotOn Game (game-programming case study)’ raylib 580
- SpotOn Game exercise enhancements 589
- sprintf 453, [455](#)
- sqrt function 234
- square brackets ([]) 299
- square root 234
- srand 252
- sscanf 453, [456](#)
- stack [243](#), 364, 536, [650](#), [662](#)
- stack frame [244](#)
- Stack program 663
- stacked building blocks 213
- stacking rule [212](#)
- StackOverflow 28, 29, [244](#)
- stacks 25
- Standard C 69
- standard data types 383
- standard deviation 637
- standard error stream (stderr) [76](#), [504](#), [594](#)
- standard input 114
- standard input stream (stdin) 24, [76](#), 504, [594](#)
- standard input/output header (stdio.h) [109](#)
- standard input/output library (stdio) 453
- standard library 74
  - header [247](#), [737](#)
- standard output stream 504
- standard output stream (stdout) 24, [76](#), 504, [594](#)
- standard version of C 69

- statement **110**, **140**, 694
  - return 237
- statement terminator (;) **110**
- static **260**, 261, 262, 312
- Static arrays are automatically initialized to zero if not explicitly initialized by the programmer 312
- \_Static\_assert 746
- static assertions 778
- static code analysis tools 32
- static data structures **765**
- static global variable 579
- static storage duration **260**
- \_Static\_assert **789**
- statistical thinking 40
- statistics
  - count **637**
  - maximum **637**
  - measures of central tendency **637**
  - measures of dispersion **637**
  - measures of variability **637**
  - minimum **637**
  - range **637**
  - standard deviation 637
  - sum **637**
  - variance 637
- stdalign.h header 789
- stdarg.h 248, 754
- stdbool.h **208**, 779, **781**
- stddef.h 248, 366
- stderr (standard error stream) **76**, **595**
- stdin (standard input stream) **76**, 453, **595**
- stdint.h 779
- stdio.h **109**, 198, 248, 261, 453, **504**, 595, 740
- stdlib.h 248, 249, 250, **450**, 760
- stdout (standard output stream) **76**, **595**, 598
- stemming **632**
- stepwise refinement 393
- stepwise refinement, **152**
- stock market forecasting 99
- stop word elimination **632**
- Storage as a Service (SaaS) 89
- storage class 260
- storage class of an identifier **260**
- storage duration **260**, 314
- storage duration of an identifier 260
- storage unit boundary 561
- storage-class specifiers **260**
- Store 418
- store 707
- stored array 654
- straight-line form **118**
- strcat function **459**
- strchr function **463**
- strcmp function **460**
- strcpy function 458
- strcspn function **464**
- stream **504**, **594**
- strerror **473**
- string **110**, **443**
  - processing 25
- string array **392**
- string built-in type in JSON 645
- string comparison lexicographical 462
- string comparison functions **460**
- string concatenation 483
- string constant **443**
- string conversion functions **450**
- string copy 483
- string is a pointer 443
- string literal 310, **443**, 444
- string literals separated only by whitespace 321
- string manipulation functions of the string handling library 457, 461
- string processing 248, 309
- <string.h> 457
- <string.h> header file 248
- strlen function **473**
- strncat function **458**, 459
- strncmp function **460**
- strncpy function **458**
- strpbrk **464**
- strpbrk function 463, 465
- strrchr function 463, **465**
- strspn function **465**, 466
- strstr function 463, **466**
- strtod function **450**
- strtok function 463, **467**, 467
- strtol function **451**
- strtoul function 450, **452**
- struct 298, **537**
- structure 24, **376**, 536
  - definition 537, 538
  - member (.) operator **540**, 541, 547
  - pointer (->) operator **540**, 541, 547
  - tag name **537**, 538
  - type **537**
- structured programming 108, 127, 138, 140, 767
- structured programming summary 210
- Structures **536**
- student poll analysis program 305
- subclass **803**
- subscript **299**, 306
- subscript notation 377
- substitution cipher 488, 489



subtract an integer from a pointer 385  
 subtracting one pointer from another 385  
 subtraction 58  
 suffix  
   floating point **762**  
   integer **762**  
 sum 132  
 sum of numbers 175  
 sum of the elements of an array 273, 304  
 sum statistic **637**  
 summarizing text 99  
 superclass **803**  
 supercomputer **56**  
 supercomputing 95  
 supplements for instructors 44  
 survey data analysis 25, **321**  
 Survey data analysis program 321  
 swapping values 714, 719  
 Swift programming language 67, 72  
 Swiss Federal Institute of Technology (EPFL) 425  
 switch multiple-selection statement 141, 196, 199  
 symbol 135, **141**  
 symbol table 698  
 symbol table (Simple compiler) **698**  
 symbolic constant 198, **303, 736**  
 symmetric encryption **494**  
 syntax coloring conventions in this book 41  
 syntax error **74**, 147, 164, 166, 210  
 Systems Software case studies 21

## T

tab 110, 135, 143, 519, 525  
 tables of values 332

tablet computer 67  
 tabular format 301  
 tail of a queue **650, 668**  
 tail recursion 293  
 tan 235  
 tangent 235  
 TCP (Transmission Control Protocol) **88**  
 TCP/IP **88**  
 telemedicine 99  
 telephone number program 482  
 telephone-number word problem 628  
 temporary copy 156  
 temporary double representation 194  
 terabytes (TB) **59**, 92  
 teraflop 94  
 terminate 76  
 terminating null character 309, 443, 455, 510  
 termination phase **154**, 157  
 termination request 763  
 ternary operator **144**, 271  
 terrorist attack prevention 99  
 Test Item File 46  
 testing 803  
 text analysis 484  
 text processing 442  
 text summarization **631**  
 tgmath.h 779  
 The CERT Division of Carnegie Mellon's Software Engineering Institute 788  
 the cloud 18, 35, 89, 643  
*The Twelve Days of Christmas* 199  
 theft prevention 99  
 Thinking Like a Developer 28, 39  
 Thompson, Ken 68  
 thrd\_create function **798**

thrd\_error **799**  
 thrd\_join function **799**  
 thrd\_nomem **799**  
 thrd\_success **799**  
 thrd\_t type **798**, 798  
 thread of execution **790**  
 thread ID 798  
 thread local storage 799  
 \_Thread\_local storage class specifier 260  
 threads 36  
 <threads.h> header **792**  
 time 248  
 time function of header time.h 253  
 \_\_STDC\_\_, predefined symbolic constant 745  
 \_\_TIME\_\_, predefined symbolic constant 745  
 time series **642**  
   Climate at a Glance 642  
   observations **642**  
 <time.h> 248  
 timing operations 30  
 Tiobe Index 54  
 toggling bits 549  
 token 463, 698, 744  
 tokenization **632**  
 tokenize a string 467  
 tokenizing a string 467  
 tokenizing strings **457**  
 tokens **467, 632**  
 tokens (Simple compiler) **698**  
 tokens in reverse 482  
 toLower function **447**  
 top **152**  
 top-down, stepwise refinement 23, **152**, 154, 157, 158, 159, 393, 394  
 top of a stack **650**  
 Tortoise and the Hare Race 294  
   multimedia with raylib 585

total **151**  
 totient **496**  
 toupper function 374, **447**  
 Towers of Hanoi 273, 290  
 trailing zeros 508  
 transaction file 626  
 transaction-processing program **606**, 614  
 transaction-processing system 24, 34  
 transfer of control **140**, 418, 422  
 translate speech 101  
 translation **63**  
 translator program **64**  
 Transmission Control Protocol (TCP) **88**  
 trap **763**  
 trap a SIGINT 765  
 traversing a binary tree 676  
 Treating character arrays as strings 310  
 tree 119, 364, 536, **675**  
 Trend spotting 99  
 trigonometric cosine 235  
 trigonometric sine 235  
 trigonometric tangent 235  
 true 779  
 true boolean value **121**  
 truncated **156**  
 truth **206**  
 truth table **206**  
 turtle graphics 356  
 tvOS **67**  
 two-dimensional array **332**, 336, 392  
 type 116, 117  
 type checking 241  
 typedef **542**  
 typedef keyword 26  
 type-generic expressions 778  
 type-generic macro 786  
 types of programming languages 22

## U

u or U for an unsigned int **762**  
 Ubuntu Linux 81  
   in the Windows Subsystem for Linux 81  
 unary operator **156**, 165, 366  
   sizeof 382  
 unbiased shuffling algorithm 546  
 unconditional branch 706, 767  
 #undef preprocessor directive **740**, 745  
 undefined behavior 788  
 undefined behaviors 526  
 underscore (\_) 113  
 Unicode 462  
 Unicode character set **60**, 198, 462  
 Unicode support 778  
 union **546**, 547, 548, 571  
 unions 24  
 union of sets 354  
 univariate time series **642**  
 UNIX 199  
 UNIX operating system 68  
 UnloadGame function in a raylib game 579  
 unnamed bit field **561**  
 unnamed bit field with a zero width **561**  
 unresolved forward reference (Simple compiler) 699  
 unresolved references 758  
 unsafe macro 746  
 unsigned decimal integer 506  
 unsigned hexadecimal integer 506  
 unsigned int 242  
 unsigned integer 549  
 unsigned long int 452

unsigned long long int 267, 268, 269  
 unsigned octal integer 506  
 unsigned short 242  
 UpdateGame function in a raylib game 579  
 uppercase letter 135, 248  
 URL (Uniform Resource Locator) **644**  
 use cases 98  
 using the # flag with 517  
 usual arithmetic conversion rules **241**  
 Utilities area (Xcode) 82  
 utility function 248

## V

V's of big data 97  
 va\_arg **755**  
 \_\_VA\_ARGS\_\_ **784**  
 va\_copy macro **787**  
 va\_end **755**  
 va\_list **755**  
 va\_start **755**  
 validate data **216**  
 value **299**  
 value of a variable 116, 117  
 variable **113**  
 variable arguments header stdarg.h **754**  
 variable initialization 391  
 variable-length argument list 25  
 variable-length array (VLA) 23  
 variable name 695, 698  
 variable-length argument list **754**, 755  
   macro 784  
 variable-length array (VLA) **340**  
 variance 637  
 variety (in big data) 97  
 Vector2 type in raylib **577**  
 velocity (in big data) 97  
 veracity (in big data) 97

version control tools 30  
 vertical tab ('\v') 445  
 vi editor 73  
 Vigenère secret-key cipher 33, [488](#), 489, 492, 493  
   Vigenère square [489](#), [492](#)  
 virtual machine 23, 28, 33, [417](#), 690  
 virtual reality [424](#)  
 Virtual Reality Modeling Language (VRML) [429](#)  
 virtual time [436](#)  
 Visual C++ compiler 31, 40, 43, 52, 77  
 Visual C++ programming language 72  
 visual product search 99  
 Visual Studio 73  
   add an existing file to a project 78  
   Command Prompt window 79  
   Community Edition 44  
   Community edition 51, 55, 77  
   compile and run a program 79  
   display the **Solution Explorer** 78  
   **Empty Project** template 77  
   filter project templates 78  
   main window 78  
   project [77](#)  
   **Search for templates** 78  
   solution [77](#)  
   **Solution Explorer** 78  
 visualization 29, 32, 33, 34, 100, 639  
 Visualization with raylib  
   Law of Large Numbers Animation 583  
 visualizing recursion 273, 291  
 voice recognition 99

void \* (pointer to void) [386](#), 469, 652  
 volatile information [58](#)  
 volume (in big data) 97  
**W**  
 w file open mode 600  
 w+ file open mode 600  
 W3C (World Wide Web Consortium) [89](#)  
 “warehouse” section of the computer 59  
 watchOS [67](#)  
 Waze GPS navigation app 99  
 wb file open mode 600  
 wb+ file open mode 600  
 Weather Forecasting 99  
 web 643  
 web service 35, 89, 644  
   invoke with libcurl 646  
   request [643](#)  
   response [643](#)  
 web service host [643](#)  
 web services 643  
   web service host [643](#)  
 Webots 32, [424](#)  
   .wbt file [429](#)  
   avoid obstacles 440  
   basic time step [436](#)  
   controller [434](#), [436](#), [438](#), [438](#)  
   **Create a Webots project directory** wizard 429  
   differential wheels [428](#)  
   e-puck robot [428](#)  
   e-puck\_avoid\_obstacles [434](#)  
   fields [431](#)  
   gravity [429](#)  
   Guided Tour 426  
   lighting effects 439  
   node [431](#)  
   physics 440  
   physics engine [434](#)

Webots (cont.)  
   physics options 439  
   PROTO nodes [432](#)  
   RectangleArena [427](#), 431  
   scene tree [431](#), 431  
   textures 439  
   WoodenBox [427](#), 432  
   world 429  
 Webots  
**Welcome to Xcode** window 81  
 while iteration statement [148](#)  
 whitespace character [109](#), [143](#)  
   string literals separated 321  
 width of a bit field [558](#), 561  
 William Gates 29  
 Windows operating system [65](#), 754, 763  
 Windows Subsystem for Linux (WSL) 24, 44, [53](#), 81  
 WindowShouldClose function (raylib) [579](#)  
 WoodenBox (Webots) [427](#), 432  
 word boundary 539  
 word frequency counting [632](#)  
 words [418](#)  
 workspace window in Xcode [82](#)  
 world in Webots 429  
 World Population Growth exercise 183  
 World Wide Web [89](#)  
 worst-case run time for an algorithm 713  
 worst-case runtime for an algorithm 712  
 Wozniak, Steve 66  
 write 699  
 writing to a file 598

**X**

Xcode 51, 55, 73, 77

**Command Line Tool** project 81

    compile and run a program 82

**Debug** area 82

**Editor** area 82

**Navigator** area 82

    project **81**

**Utilities** area 82

**Welcome to Xcode** window 81

    workspace window **82**

Xcode navigators

**Issue** **82**

**Project** **82**

Xerox PARC (Palo Alto Research Center) 66

**Y**

$y$ -intercept **639**, 640

**Z**

0 (zero) flag **517**

zettabytes (ZB) 93