

Index

Numerics

- 0 (zero)
 - initializing arrays, 317
 - zero-based arrays, 315–316
- 1-based arrays, 315, 316
- 1-time pad algorithm, 446
- 4th Dimension database programming language, 79

A

- Ada language, 10, 58, 130
- address space layout randomization (ASLR), 642
- Adobe AIR RIA tool, 664
- adversarial search
 - alpha-beta pruning, 420–421
 - depth versus time in, 419–420
 - horizon effect, 420
 - library lookup, 421–422
 - overview, 418–419
- agile documentation, 287
- agile (extreme) programming, 112–114
- AI. *See* artificial intelligence
- algorithms. *See also specific kinds*
 - “best,” 9
 - comments describing, 282
 - data compression, 435–444
 - defined, 9
 - encryption, 445–459
 - overview, 9
 - searching, 409–422
 - sorting, 393–408
 - string searching, 423–433
- Alice teaching language, 60–61
- aligning text using HTML, 466
- Alpha Five database programming language, 79
- alpha-beta pruning, 420–421
- American Standard Code for Interchange (ASCII) codes, 423
- Analytical Engine, 10
- anchor points (HTML), 470–471
- And operator, 175–176. *See also* logical/Boolean operators
- AndAlso operator (Visual Basic), 597
- Apple Xcode compiler, 25, 84, 85
- AppleScript (Mac), 76, 91
- applets (Java), 66
- arrays
 - associative, 352–353, 517–518
 - in C#, 554–555
 - in C/C++, 537
 - data type limitations, 326
 - declaring, 318
 - default bounds, 315–316
 - definable bounds for, 316–317
 - defining the size, 314–317, 325–326
 - deleting elements, 327
 - disadvantages, 325–328, 329, 375–376
 - elements, defined, 314
 - for heap sort data, 402–403
 - initializing, 317–318
 - inserting elements, 326–327
 - in Java, 554–555
 - in JavaScript, 493–494
 - linked lists versus, 342
 - multi-dimensional, 321–323, 375–376
 - one-based, 315, 316
 - overview, 314
 - in Pascal/Delphi, 586–587
 - in Perl, 569–570
 - in PHP, 506
 - requirements for defining, 314
 - resizable, 319–321, 326
 - retrieving data from, 318–319
 - searching and sorting, 326
 - speed and efficiency issues, 328
 - storing data in, 318
 - for string data types in C/C++, 526
 - structures with, 314, 323–325
 - uses for, 327–328
 - in VB/RB, 603–604
 - zero-based, 315–316
- artificial intelligence (AI)
 - applications, 656
 - Bayesian probability, 653
 - camp, strong versus weak, 644
 - declarative languages, 70–73
 - expert systems, 646–648
 - game-playing, 645–646
 - image recognition, 651–652
 - machine learning, 652–655
 - natural language processing, 648–650
 - neural networks, 653–655
 - problem solving, 644–652
 - robotics, 655
 - speech recognition, 650–651
 - Turing Test, 643

ASCII (American Standard Code for Information Interchange) codes, 423
ASCII files. *See* text files
ASLR (address space layout randomization), 642
assembly language
assemblers for, 16
comments, 278
other languages compared to, 15
overview, 12–14
as processor-specific, 14
speed and efficiency of, 16, 666
typical command, 12–13
uses for, 16, 17
assignment operators
C/C++, 530–531
Java/C#, 547–548
JavaScript, 490
overview, 162
Perl/Python, 564–565
PHP, 502
Ruby, 513
associative arrays. *See* hash tables
asymmetric (public-key) encryption, 453–455
attributes, database, 616
automated documentation, 287

B

B+ trees, 384
background of Web pages, 469–470
backward chaining, 646
backward or forward searching algorithm, 411
BASIC (Beginner's All-purpose Symbolic Instruction Code). *See also* REALbasic; Visual Basic
C compared to, 50–51, 128

calling functions, 226
combining with C, 134–135
comments, 278
as database programming language, 623, 624
defining functions, 225
defining random-access files, 265
descriptive commands, 53–54
development of, 52
dialects, 52, 589–590
evolution of, 589
financial rewards less for, 51
as first language, 50–51
as good language to know, 49
GOTO command problems, 33
hardware and operating system shielded by, 54, 589
as high-level language, 14
instant feedback with, 54, 55
interpreters and compilers, 55
limitations, 589
multi-dimensional arrays in, 322, 323
practicing online, 590
principles, 53
problem-solving focus of, 128
program structure, 590
reading random-access files, 267
resizable arrays in, 319–320
True BASIC, 52
versatility of, 54
viewed as toy language, 51, 54
writing random-access files, 266–267
Basic Local Alignment and Search Tool (BLAST), 629–630

“battling robot” programming games, 61–62
Bayesian probability, 653
Big-O notation, 407
binary arithmetic, 11
binary searching algorithm, 413
binary trees
heap sort using, 399–402
overview, 382–383
bioinformatics
complimentary sequences, 628
concatenating molecules, 627, 628
described, 423, 625
languages for, 631–632
mutating molecules, 627–628
programming, 630–632
purpose of, 625–626
representing molecules, 626–627
searching databases, 628–630
bit depth, 444
BLAST (Basic Local Alignment and Search Tool), 629–630
block ciphers, 449, 450–452
block comments
C#, 543
C/C++, 525
describing code and algorithms, 282–283
documenting subprograms, 283–284
Java, 543
JavaScript, 487
overview, 279–281
PHP, 498
Ruby, 510–511
surrounding with asterisks, 281
symbols for, 280–281
block searching algorithm, 412
blocks in untyped files, 268

- blocks of commands
 - in curly bracket
 - languages, 63, 280
 - defined, 182
 - in IF-THEN statements, 182–183
 - in IF-THEN-ELSE statements, 184–185
 - in IF-THEN-ELSEIF statements, 186, 188
 - BlueJ editor (Java), 26
 - body text (HTML), 464–465
 - Boolean data type
 - in C++, 528
 - in Java/C#, 545
 - mimicking in C, 528
 - in Pascal/Delphi, 579
 - storage requirements for, 150
 - in VB/RB, 595
 - Boolean operators. *See* logical/Boolean operators
 - Borland Turbo compilers, 25
 - Boyer-Moore algorithm, 425
 - branching statements. *See also specific statements*
 - Boolean operators in, 189–190
 - branches, defined, 34
 - C/C++, 531–533
 - comparison operators in, 181
 - defined, 181
 - Java/C#, 548–551
 - JavaScript, 490–492
 - overview, 34–35, 181
 - Pascal/Delphi, 581–582
 - Perl/Python, 565–566
 - PHP, 502–504
 - Ruby, 514–515
 - VB/RB, 597–599
 - break command
 - with case statements (Java/C#), 550–551
 - with loops, 211
 - with SWITCH statement, 192–193, 503
 - breaking up programs. *See* objects; subprograms
 - breakpoints, 98–99
 - brute force attacks on encryption, 456–457
 - brute-force searching algorithms
 - adversarial search, 419
 - overview, 409–410
 - sequential search, 410–416, 424–429
 - B-trees, 383–384
 - bubble sort algorithm, 394–396, 406–408
 - buffer overflow problems, 641
 - bugs, 97. *See also* debuggers; debugging
 - Burrows-Wheeler transform (BWT) algorithm, 436–438
 - bus, defined, 12
 - buttons, best use of, 294
 - byte, defined, 149
 - bytecode. *See* p-code
-
- C
- C language. *See also* curly bracket languages
 - advantages, 15, 16, 523
 - arrays, 537
 - BASIC compared to, 50–51, 128
 - branching statements, 531–533
 - calling functions, 226
 - challenges for learning, 51
 - combining with BASIC, 134–135
 - comments, 525
 - creating functions, 534–535
 - dangers from power of, 54, 64, 523
 - data structures, 535–537
 - declaring variables, 147, 525–528
 - defining functions, 226
 - development of, 14–15
 - efficiency of, 64–65
 - as first language, 50–51, 65, 69
 - as good language to know, 49
 - high-level languages versus, 14–15
 - IF statements, 183
 - including libraries, 524
 - keywords minimal in, 64, 130
 - languages based on, 50
 - looping statements, 533–534
 - as low-level language, 666
 - operators, 528–531
 - other languages compared to, 15
 - overview, 523
 - portability, 65, 660–661
 - power of, 63–64
 - problem-solving focus of, 128
 - program structure, 524–525
 - reasons for popularity, 63–65
 - SWITCH statement, 191–193
 - as systems programming language, 73–74
 - use for major programs, 15
 - C++ language. *See also* curly bracket languages
 - advantages, 667
 - arrays, 537
 - based on C, 50, 65, 523
 - bioinformatics subprogram, 632
 - branching statements, 531–533
 - comments, 278, 280, 525
 - creating functions, 534–535
 - dangers from power of, 523

- C++ language (*continued*)
 - data structures, 535–537
 - as database programming language, 78, 623, 624
 - declaring variables, 525–528
 - as first language, 65, 69
 - IF statements, 183
 - including libraries, 524
 - looping statements, 533–534
 - multi-dimensional arrays
 - in, 323
 - as object-oriented, 65
 - objects, 537–539
 - operators, 528–531
 - overview, 523
 - program structure, 524–525
 - resizable arrays in, 320–321
 - as systems programming language, 73–74
- C++ Robots programming game, 62
- C# language. *See also* curly
 - bracket languages
 - advantages, 67–68, 667
 - based on C, 50
 - branching statements, 548–551
 - comments, 542–543
 - creating functions, 552–553
 - data structures, 553–556
 - declaring variables, 543–545
 - as first language, 70
 - looping statements, 551–552
 - .NET compatibility, 68
 - as object-oriented, 67
 - objects, 556–557
 - operators, 545–548
 - program structure, 542
 - pronunciation, 66
 - as type-safe, 68
 - calling. *See also* parameter passing
 - functions, 226–227
 - polymorphism with, 243
 - recursion, 227–229
 - subprograms in files, 217–218
 - subprograms in objects, 252–253
 - capitalization
 - in curly bracket languages, 144
 - of local variables (Ruby), 511
 - variable naming conventions, 144
 - Carriage Return (CR) code, 261
 - cascading stylesheets. *See* CSS
 - CASE (Computer-Aided Software Engineering)
 - Class diagrams, 117–119
 - flowcharts, 115–116
 - formatting source code
 - automatically, 120–121
 - generating code
 - automatically, 119–120
 - modeling a large project, 115–119
 - Sequence diagrams, 119
 - tools common in, 115
 - tracking revisions, 121–122
 - UML with, 116–119
 - Use Case diagrams, 117, 118
 - case statements
 - C/C++, 532–533
 - Java/C#, 550–551
 - JavaScript, 491–492
 - Pascal/Delphi, 582
 - PHP, 504
 - Ruby, 515
 - SELECT CASE, 190–197, 598–599
 - VB/RB, 598–599
 - CBR (constant bit rate), 444
 - central processing units (CPUs). *See* processors
 - chaining hash tables, 357
 - check boxes, 297–298
 - chess-playing computers, 646
 - Chinese lottery attack, 456–457
 - Chinese Postman problem, 379–380
 - Chipmunk BASIC, 590
 - cipher-block chaining (CBC), 451–452
 - circular linked lists, 341, 371
 - Clarion database programming language, 79
 - Class diagrams (CASE), 117–119
 - class files (VB/RB), 591
 - classes
 - creating objects from, 251–252
 - defining objects with, 249–251
 - #include command for, 252
 - clear command (Python), 332
 - clustered indexes, 417–418
 - COBOL (COmmon Business Oriented Language), 14, 50
 - code generators, 119–120
 - CodeWarrior compiler, 84
 - coding phase of extreme programming, 114
 - coding securely, 640–641
 - collections
 - adding data, 346–348
 - deleting data, 348–349

- dictionaries versus, 352
 - index numbers, 347, 350–351
 - keys for identifying data, 349–350, 351
 - language features for, 345
 - overview, 345–346
 - Ruby, 517
 - searching and retrieving data, 350–351
 - in Visual Basic, 603, 604
 - collisions, hash function, 356–359
 - colors
 - for background with HTML, 469, 470
 - for text with CSS, 478–479
 - for text with HTML, 467–468
 - combo boxes, 298–299
 - comma-delimited text files, 260–261
 - command-line interfaces, 290
 - comments. *See also* block comments
 - C/C++, 525
 - consistent style for, 281
 - defined, 278
 - describing code and algorithms, 282–283
 - documenting
 - subprograms, 283–284
 - HTML, 469
 - for ignoring lines of code when testing, 284–285
 - inappropriate, 279
 - Java/C#, 542–543
 - JavaScript, 487
 - line, 279
 - Pascal/Delphi, 577
 - PHP, 498
 - Ruby, 510–511
 - symbols for, 280–281
 - VB/RB, 592
 - white space for, 279
 - COMmon Business Oriented Language (COBOL), 14, 50
 - comparison operators. *See* relational/comparison operators
 - comparison with signed result operator (Perl), 562, 563
 - compilers
 - for BASIC, 55
 - choosing carefully, 22, 84
 - choosing, steps for, 84–85
 - code generation and optimization features, 86–88
 - CodeWarrior, 84
 - compiling to a virtual machine, 93–94
 - defined, 16, 83
 - features evaluation, 86–90
 - free, 89–90
 - GCC (GNU Compiler Collection), 25, 85–86
 - inefficiency of, 16
 - interpreters versus, 92
 - keyword quantity and efficiency of, 64, 130
 - languages supported, 86
 - for Linux, 25–26, 85
 - for Mac OS X, 25, 84, 85
 - open source, 85
 - operating systems supported, 88–89
 - overview, 21–22
 - Perl/Python, 561
 - specific to operating system, 21–22
 - specific to processor, 84
 - speed issues, 87–88
 - type-safe languages with, 67
 - for Windows, 24–25, 85
- complimentary sequences, 628
- compression. *See* data compression algorithms
- computer security. *See* security
- Computer-Aided Software Engineering. *See* CASE
- computers
 - choices for programming, 18–19
 - history of, 10–11
- concatenating molecules, 627, 628
- concatenation operator, 166–167
- conditional expressions, 172. *See also* relational/comparison operators
- constant bit rate (CBR), 444
- constants, 153–154, 579, 595–596
- Context MBA, 659
- converting
 - data types, 179–180
 - keys with hash functions, 354–356
- cookies, 264
- coupling, tight versus loose, 135–136, 137–138
- CPUs (central processing units). *See* processors
- CR (Carriage Return) code, 261
- cracking encryption
 - Battle of Midway example, 459
 - brute force attacks, 456–457
 - Chinese lottery attack, 456–457
 - dictionary attacks, 457–458
 - frequency analysis, 458
 - plaintext attacks, 458
 - slot machine example, 450

critical bugs, 97
 CRobots-3D programming
 game, 62
 cross-platform
 programming. *See also*
 portability
 languages for, 661–662
 need for, 660
 rich Internet applications
 (RIAs), 664–665
 robotics, 665–666
 software as service model,
 663–664
 virtual machines for, 23,
 66, 93–94, 662–663
 CSng function, 179–180
 CSS (cascading stylesheets)
 cascading and
 precedence, 482–483
 colors for text, 478
 embedding styles, 481
 linking to external
 stylesheet files,
 481–482
 origin of name, 477
 overview, 477
 separating stylesheets in
 files, 481–482
 style classes, 479–480
 stylesheet structure,
 477–479
 curly bracket languages.
 See also specific
 languages
 break command,
 192–193, 211
 choosing, 69–70
 defined, 63
 FOR-NEXT loops in,
 202–203, 205–206
 #include command for
 subprograms, 218–219
 increment operator,
 202–203
 overview, 63–69

subprograms in, 218–219
 SWITCH statement in,
 191–193
 zero-based arrays in,
 315–316

D

data. *See also* reading files;
 writing files
 isolated with OOP, 45–46
 private versus public,
 234–235
 shielded by
 encapsulation, 236–237
 transferring, scripting
 languages for, 77
 data compression
 algorithms
 Burrows-Wheeler
 transform, 436–438
 lossless, 436–442
 lossy, 442–444
 lossy/lossless trade-offs,
 442–443
 run-length encoding, 436
 trade-offs, 435, 444
 data execution protection
 (DEP), 642
 data integrity, 621
 data mining, 622
 data structures. *See also*
 arrays; structures;
 specific structures
 C#, 553–555, 556
 C/C++, 535–537
 choosing, 328, 342,
 359, 373
 collections, 345–351, 352,
 517, 603, 604
 Delphi, 585–587
 deques, 370–373
 dictionaries, 352–359, 572,
 603, 604
 graphs, 376–380

hash tables, 353–359,
 517–518, 556
 Java, 553, 554–556
 linked lists, 337–344, 370,
 371, 555–556
 Pascal, 585–587
 Perl, 569–570
 Python, 570–572
 queues, 365–370, 556
 REALbasic, 603–604
 Ruby, 517–518
 sets, 329–337, 341–342,
 587
 stacks, 361–365, 556
 trees, 380–390
 Visual Basic, 603–604
 data types. *See also*
 declaring variables;
 strings
 for arrays, 326
 Boolean, 150, 528, 545,
 579, 595
 choosing for variables,
 148–149
 comments explaining
 choice of, 283
 converting, 179–180
 decimal, 579, 594
 declaring variables with,
 146–147
 floating point, 148–150,
 527–528, 544
 for global variables, 156
 integer, 148–150, 526–527,
 544, 578, 593–594
 range of values for, 148
 records, 265
 storage requirements for,
 149–150
 typeless versus strongly-
 typed languages, 74–75
 user-defined, 265, 312
 uses for, 148
 validation still needed
 with, 149
 variant, 326, 595

- database management
 - connecting to databases, 273–276
 - data integrity, 621
 - data mining, 622
 - database programming, 622–624
 - with flat-file databases, 612–613
 - with free-form databases, 611
- Join command, 619
- manipulating data, 617–622
- overview, 609
- Project command, 618, 619
- RAD tools for, 275–276
- with relational databases, 613–617
- Select command, 617–618, 619
- SQL for, 271, 620–621
- third-party toolkits for, 273–275
- writing commands, 620
- database programming languages, 78–80, 623–624
- databases
 - basics, 609–617
 - bioinformatics, 628–630
 - distributed, 617
 - flat-file, 611–613
 - free-form, 610–611
 - popular file formats, 271–272
 - relational, 273, 274, 613–617
 - structure of, 272–273
 - uses for, 276
- dBASE, 78–79
- debuggers
 - defined, 83
 - features, 95
 - with IDEs, 96
 - source-level versus machine-language, 98
- debugging
 - commenting out code for, 284–285
 - stepping or tracing, 97–100
 - variable watching, 100–101
- decimal data types. *See also* floating point data types
 - in Pascal/Delphi, 579
 - in VB/RB, 594
- declarative languages, 70
- declaring arrays, 318
- declaring variables
 - as arrays, 314, 315, 316, 317
 - in BASIC, 142–143
 - in C/C++, 525–528
 - as collections, 346
 - in commands, 144–145
 - with data type, 146–147
 - global variables, 156
 - in Java/C#, 543–545
 - in JavaScript, 487–488
 - language differences for, 147
 - in modules, 156
 - as multi-dimensional arrays, 322–323
 - naming conventions, 143–144
 - overview, 142–143
 - in Pascal/Delphi, 577–579
 - Perl/Python, 561
 - in PHP, 499
 - as queues, 366
 - in Ruby, 511
 - as sets, 329–330
 - as stacks, 362
 - as structures, 312
 - in subprograms, 157
 - at top of program, 147
 - in VB/RB, 592–595
- decrement operator
 - C/C++, 529–530
 - Java/C#, 546–547
 - JavaScript, 489, 490
 - Perl, 563–564
 - PHP, 501–502
- Deep Fritz chess-playing computer, 646
- Delphi
 - branching statements, 581–582
 - comments, 577
 - constants, 579
 - creating functions and subprograms, 584–585
 - data structures, 585–587
 - database programming features, 80
 - declaring variables, 577–579
 - looping statements, 583–584
 - objects, 587–588
 - origins of, 57
 - overview, 575–576
 - program structure, 576
 - reading untyped files, 269–270
 - Web site, 40, 57
 - writing untyped files, 268–269
- DEP (data execution protection), 642
- deque
 - adding and removing data, 371–372
 - common commands, 371
 - overview, 370–371
 - undoing commands, 372–373
 - uses for, 373
- design patterns
 - defined, 243
 - flyweight, 244–245
 - interface, 244

- design patterns (*continued*)
 - memento, 245–246
 - multiple in one
 - program, 246
 - overview, 243
- design specifications, 285
- designing user interfaces
 - as art and science,
 - 302–303
 - being consistent, 306
 - error messages, 305
 - event-driven programming
 - for, 39–40, 41, 42
 - focusing on tasks, 306–307
 - hiding unusable options,
 - 304–305
 - knowing the user, 303
 - for navigation ease, 307
 - tolerating mistakes,
 - 305–306
 - visually, 39–40, 41, 42
- desire, training versus,
 - 17–18
- Dev-C++ compiler, 25
- Dev-Pascal compiler, 25
- dialog boxes (user interface), 300–301, 494–495
- dictionaries
 - adding data, 352–353
 - collections versus, 352
 - defined, 352
 - hash tables with, 353–359
 - key-value pairs, 352, 354
 - in Python, 572
 - in REALbasic, 603, 604
 - searching and retrieving data, 353
- dictionary attacks on
 - encryption, 457–458
- dictionary encoding
 - algorithms, 439–442
- directed graphs, 377–378, 428–429
- disassemblers
 - defined, 83
 - for malware dissection,
 - 634, 635
 - obfuscators for
 - preventing, 106
 - overview, 105–106
- distributed databases, 617
- Div operator (Pascal/Delphi), 580
- dividing programs. *See* objects; subprograms
- divmod operator (Python), 562
- DO loops, 208–211
- documentation. *See also* comments
 - agile, 287
 - automated, 287
 - design specifications, 285
 - help files, 287–288
 - need for, 277
 - self-documenting code,
 - 277–278
 - technical designs, 286
 - tools, 286–287
 - types of, 285
 - user manuals, 286
 - video tutorials, 286
 - writing, 285–288
- double hashing, 358–359
- double linked lists, 340–341, 371
- DO-UNTIL loops (VB/RB), 600–601
- DO-WHILE loops
 - C/C++, 534
 - Java/C#, 552
 - JavaScript, 493
 - Visual Basic, 600, 601
- dynamic model (UML), 119
- free with operating systems, 19
- IDE, 95–97
- for Java, 26
- overview, 19–20
- standalone, 95
- electronic codebook (ECB) cipher, 451
- Electronic Numerical Integrator and Computer (ENIAC), 10
- ELIZA NLP program, 649–650
- emphasizing text using HTML, 467
- encapsulation, 235–238
- encryption algorithms
 - basics of encryption,
 - 447–448
 - block ciphers, 449,
 - 450–452
 - cracking encryption, 450,
 - 455–459
 - implementation
 - issues, 447
 - key length issues, 448
 - one-time pad, 446
 - password, 446–447
 - permutation box or P-box,
 - 447–448
 - security through
 - obscurity, 447
 - steganography, 456
 - stream ciphers, 449–450
 - substitution box or S-box,
 - 447–448
 - substitution ciphers,
 - 445–446
 - symmetric/asymmetric,
 - 452–455
 - for wireless standards, 450
- endless loops, 201–202, 207, 212, 228
- ENIAC (Electronic Numerical Integrator and Computer), 10

E

- Eclipse IDE (Java), 26, 97
- editors
 - defined, 19, 83
 - features, 96

- enumerated variables (C/C++), 536–537
 - error messages, avoiding
 - cryptic, 305
 - event handlers, 39, 40, 42–43
 - event-driven programming
 - combining with other methodologies, 48
 - designing a user interface, 39–42
 - development of, 38–39
 - languages, 40
 - parts of programs, 39
 - writing event handlers, 42–43
 - writing the program, 43
 - Excel (Microsoft), 659
 - executable (EXE) files, 21, 83
 - EXIT command with
 - loops, 211
 - experimenting, 23–24
 - expert systems, 646–648
 - exponentiation operator
 - overview, 163
 - Perl/Python, 562
 - Ruby, 512
 - VB/RB, 596
 - extreme programming (XP), 112–114
- F**
-
- factorials, calculating, 227–228
 - FBI's project failure, 8
 - Fibonacci searching, 414–416
 - FIFO (First In, First Out) structures, 366
 - files. *See also* reading files; writing files
 - for CSS, external, 481–482
 - executable (EXE), 21, 83
 - JavaScript, loading in Web page, 487
 - PDF, 264
 - proprietary formats, 271
 - random-access, 264–268
 - storing subprograms
 - separately in, 36–37, 131, 133–135
 - text, 259–264
 - Universal Binary (Mac), 21
 - untyped, 268–271
 - finite automaton string
 - search algorithm, 428–429
 - firewalls, 635, 636, 637
 - flat-file databases, 611–613
 - floating point data types. *See also* decimal data types
 - in C/C++, 527–528
 - in Java/C#, 544
 - overview, 148–150
 - storage requirements
 - for, 150
 - flyweight design pattern, 244–245
 - fonts, HTML for, 467–468
 - FOR loops
 - C/C++, 533
 - Java/C#, 551
 - JavaScript, 492
 - Pascal/Delphi, 583
 - Perl/Python, 566–567
 - PHP, 504
 - Ruby, 515
 - VB/RB, 600
 - forensics, 639
 - FOR-NEXT loops
 - counting backward, 205–206
 - counting by a range, 203–204
 - counting by different increments, 204–205
 - in curly bracket
 - languages, 202–203, 205–206
 - DOWNT0 with, 205
 - endless, 201–202
 - initializing arrays
 - using, 317
 - loop variable with, 201–203
 - nested, 209–211
 - overview, 200–201
 - STEP with, 204
 - VB/RB, 600
 - FORTRAN (FORMula TRANslator), 14, 161
 - forward chaining, 646
 - 4th Dimension database
 - programming language, 79
 - free-form databases, 610–611
 - frequency analysis, 458
 - functional languages, 70
 - functional model (UML), 117, 118
 - functions. *See also* subprograms
 - calling, 226–227
 - creating in BASIC, 225
 - creating in C, 226
 - creating in C/C++, 534–535
 - creating in Java/C#, 552–553
 - creating in JavaScript, 493
 - creating in Pascal/Delphi, 584–585
 - creating in Perl/Python, 568–569
 - creating in PHP, 505
 - creating in Ruby, 516
 - creating in VB/RB, 602
 - defined, 162
 - math (table), 165–166
 - parameter list for, 226
 - RETURN keyword, 226
 - string (table), 167

functions (*continued*)
 as subprograms with
 name as value, 225
 typical example, 225
 future of programming
 cross-platform
 programming, 660–666
 language choices,
 657–658, 666–669
 operating system choices,
 658–660

G

game-playing in AI, 645–646
 GCC (GNU Compiler
 Collection), 25, 85–86
 GCJ Java compiler, 94
 generic values (VB/RB), 595
 global variables, 154–156
 glue, scripting languages as,
 77, 668
 GNU Emacs editor, 95
 GOTO command
 problems, 33
 grafting sub-trees, 388, 390
 graphical user interface
 (GUI). *See* user
 interface
 graphics
 adding to Web page
 (HTML), 469
 background for Web page
 (HTML), 470
 image recognition,
 651–652
 graphs. *See also* trees
 connecting nodes, 380
 connections or edges,
 defined, 377
 nodes or vertices,
 defined, 377
 overview, 376–377
 for shortest path
 solutions, 379–380

for single path solutions,
 378–379
 topological graph theory,
 380
 types of, 377–378
 GUI (graphical user
 interface). *See* user
 interface

H

hackers, defenses against
 firewall, 637
 forensics, 639
 intrusion detection
 systems, 637–638
 rootkit detectors, 638
 secure computing,
 639–642
 hash arrays (Perl), 570
 hash tables
 in C#, 556
 chaining, 357
 collisions, 356–359
 complexity added by, 359
 converting keys with hash
 functions, 354–356
 double hashing, 358–359
 overview, 353–356
 Ruby, 517–518
 searching improved by,
 353–354
 headings in Web pages
 CSS for, 478–480, 482–483
 HTML for, 465
 style classes for (CSS),
 479–480
 tables, HTML for, 472–473
 titles, HTML for, 464
 heap, defined, 399
 heap sort algorithm,
 399–403, 406–408
 help file creators, 83,
 103–104, 288
 help files, 287–288

heuristic searching
 algorithms, 409–410
 hexadecimal numbers, 11
 high coupling, 135–136
 high-level languages, 13, 14,
 15. *See also specific
 languages*
 history of computer
 programming
 assembly language, 12–14
 C language, 14–15
 early computers, 10–11
 high-level languages, 14
 machine language, 11
 pros and cons of
 languages, 15–17
 honeypot programs, 638
 horizon effect, 420
 HTML (HyperText Markup
 Language)
 <html> and </html>
 tags, 463–464
 aligning text, 466
 as basis of all Web
 pages, 463
 body text, 464–465
 colors for background,
 469, 470
 colors for text, 467–468
 comments, 469
 document structure,
 463–469
 emphasizing text, 467
 font size, 468
 graphics for background,
 470
 graphics on pages, 469
 headings, 465
 hyperlinks, 470–471
 JavaScript markup,
 486–487
 line breaks, 464–465
 PHP markup, 497–498
 tables, 471–475
 titles, 464

hybrid OOP languages,
246–247
Hydra chess-playing
computer, 646
hyperlinks, 470–471

I

IBM's Jikes Java compiler,
94

IDE (integrated
development
environment), 95–97

identifying the problem, 8

IDSs (intrusion detection
systems), 637–638

IF statements

C/C++, 531

Java/C#, 548

JavaScript, 490

Pascal/Delphi, 581

Perl/Python, 565

PHP, 502–503

Ruby, 514

IF-ELSE statements

C/C++, 531

Java/C#, 548

JavaScript, 491

Perl/Python, 565–566

PHP, 503

Ruby, 514

IF-ELSEIF statements

C++, 531–532

Java/C#, 548–549

Perl/Python, 566

PHP, 503

Ruby, 514–515

IF-THEN statements

blocks with, 182–183

C/C++, 183

overview, 182

Pascal/Delphi, 582

SELECT CASE statements

versus, 196–197

VB/RB, 597

IF-THEN-ELSE statements
blocks with, 184–185

IF-THEN-ELSEIF

statements versus, 185

overview, 184

Pascal/Delphi, 581, 582

VB/RB, 598

IF-THEN-ELSEIF

statements

checking a condition for
each command,

186–187

IF-THEN-ELSE

statements versus, 185

Pascal/Delphi, 582

for three or more choices,
187–189

VB/RB, 598

image recognition, 651–652

increment operator

C/C++, 529–530

Java/C#, 546–547

JavaScript, 489

overview, 202–203

Perl, 563–564

PHP, 501

Ruby, 513

indexes

for collections, 347,

350–351

for searching algorithms,
416–418

inference engine, 646, 647

informed searching

algorithms, 409–410

inheritance

advantages, 240

in C++, 538–539

inheriting an object,

253–256

in Java/C#, 557

multiple, 240, 539

need for, 238–240

overview, 47, 48, 240–242

in Pascal/Delphi, 588

in Perl/Python, 573

in PHP, 507

polymorphism with,
242–243, 256–258

in Ruby, 519

in VB/RB, 605

initializing

arrays, 317–318

loop variables, 207, 209

in-order traversal of trees,
385–386

in-place sorting algorithms,
394

insertion sort algorithm,
397–398, 406–408

installer programs, 83,
104–105

instructions, defined, 128

instrumentation mode of
profilers, 103

integer data types

in C/C++, 526–527

in Java/C#, 544

overview, 148–150

in Pascal/Delphi, 578

storage requirements for,
150

in VB/RB, 593–594

integer division operator,
163, 596

integrated development
environment (IDE),
95–97

interface design pattern,
244

Internet resources

Alice site, 60

“battling robot”

programming

games, 62

bioinformatics

subprograms, 632

BLAST, 629

Chipmunk BASIC site, 590

database programming
languages, 79

Internet resources (*continued*)

Delphi site, 40, 57
 free compilers, 89–90
 GCC compiler, 85
 IDEs, 97
 interpreters, 92
 Java compilers, 26, 94
 Java editors, 26
 Java site, 94
 KPL site, 59
 Liberty BASIC site, 590
 Linux compilers, 25–26, 90
 Loebner Prize, 643
 Mac OS X compilers, 25, 90
 Mac OS X editors, 19
 Phrogram site, 60
 PowerShell interpreter, 91
 REALbasic site, 40, 589
 Run BASIC site, 590
 software for running
 Windows and Linux on
 Mac OS X, 19
 standalone editors, 95
 ThinkFree Microsoft Office
 clone, 663
 True BASIC, 52
 Visual Basic site, 589
 Windows compilers, 24–25, 89
 interpolation searching
 algorithm, 414–416
 interpreters
 advantages, 92
 for BASIC, 55
 compilers versus, 92
 for Logo, 57
 for operating systems, 91
 overview, 22–23, 90–91
 for p-code, 23
 for scripting languages,
 22–23, 74
 for Web pages, 91–92
 intrusion detection systems
 (IDSs), 637–638

J

Java. *See also* curly bracket
 languages; virtual
 machines (VMs)
 advantages, 667
 applets, 66
 based on C, 50, 66
 bioinformatics in, 631, 632
 branching statements,
 548–551
 comments, 278, 542–543
 compilers, 26, 66, 94
 creating functions,
 552–553
 data structures, 553–556
 declaring variables,
 543–545
 editors for, 26
 as first language, 70
 IDEs for, 97
 JavaScript versus, 486
 looping statements,
 551–552
 objects, 556–557
 operators, 545–548
 overview, 541
 p-code used by, 23, 66
 portability, 66, 541
 program structure, 542
 safety features, 66
 starting with, 26
 Web site, 94
 JavaScript
 arrays, 493–494
 branching statements,
 490–492
 comments, 487
 creating functions, 493
 declaring variables,
 487–488
 external files for, 487
 HTML markup for, 486–487
 interpreter, 91–92
 Java versus, 486

operators, 488–490
 overview, 485–486
 structure of programs,
 486–487
 for transferring data
 among programs, 77
 user interface design,
 494–496
 jEdit editor (Mac), 19
 Jikes Java compiler
 (IBM), 94
 Join command for
 databases, 619
 JScript interpreter
 (Windows), 91
 jump searching algorithm,
 412
 Just BASIC compiler, 25

K

keys for encryption
 asymmetric (public-key),
 453–455
 ECB ciphers, 451
 length issues, 448
 stream ciphers, 449
 symmetric (private-key),
 452–453
 keys in data structures
 collections, 349–350, 351
 dictionaries, 352–353,
 354–356
 hash function for
 converting, 354–356
 keywords
 in C, minimal, 64, 130
 defined, 64
 efficiency, ease, and
 number of, 64, 130
 overview, 128–129
 subprograms similar
 to, 132
 Kid's Programming
 Language (KPL), 59–60

knowledge base, 646, 647
 KPL (Kid's Programming Language), 59–60

L

languages. *See also specific languages*

artificial intelligence, 70–73
 assembly, 12–14, 15–16
 for bioinformatics programming, 631–632
 choosing compiler for, 84–85
 choosing the best, 668–669
 choosing your first, 50–51, 81
 collections with, 345
 compiler support for, 86
 cross-platform, 661–662
 curly bracket, 63–70
 database programming, 78–80
 for event-driven programming, 40
 features, 30
 future of, 657–658, 666–669
 goal of, 12
 high-level, 14
 hybrid OOP, 246–247
 improvements in, 12, 29–30
 knowing, knowing programming versus, 26–27
 learning two or more, 72
 low-level, 666
 machine, 11, 666
 multiple in one program, 134–135
 object-oriented, 246–248
 popular, reasons for knowing, 49

problem-solving focus of, 128
 proprietary, 58–59
 pros and cons of, 15–17
 queues with, 366
 scripting, 73–78
 sets with, 330
 shifting popularity of, 27
 systems programming, 73–74, 668
 teaching, 52–62
 variable declaration differences, 147
 variety of choices available, 49
 Lego Mindstorms NXT-G, 61, 666
 level order traversal of trees, 386
 LF (Line Feed) code, 261
 Liberty BASIC, 590
 libraries
 breaking programs into subprograms for, 133–134
 commercial, 133
 features added through, 330
 math functions, 165
 PHP array functions, 506
 for queues, 366
 for sets, 330
 for user interfaces, 139, 293
 line breaks (HTML), 464–465
 line comments, 279
 lines of code
 commenting out, 284–285
 comments on, 279
 defined, 19, 128
 minimizing, 128
 linked lists
 adding data, 338
 arrays versus, 342
 circular, 341, 371

complexity of creating, 343–344
 creating, 338–339
 deleting data, 340
 disadvantages, 341–342
 double, 340–341, 371
 in Java, 555–556
 nodes, defined, 337
 overview, 337–338
 pointers, defined, 338
 for queues, 370, 371
 rearranging pointers, 339–340
 Linux
 based on UNIX, 18
 bioinformatics subprogram, 632
 compilers, 25–26, 90
 described, 19
 editors, 19
 running on Mac OS X, 19
 LISP (LISt Processing), 72–73, 278, 653
 list boxes, 298
 lists (Python), 571
 Loebner Prize, 643
 logical/Boolean operators
 And (overview), 175–176
 in branching statements, 189–190
 C/C++, 529
 Java/C#, 546
 JavaScript, 489
 multiple expressions using, 189–190
 Not (overview), 175
 Or (overview), 176–177
 overview, 174–175
 Pascal/Delphi, 580–581
 Perl/Python, 563
 PHP, 500–501
 Ruby, 512–513
 VB/RB, 596–597
 in WHILE loops, 207
 Xor (overview), 177–178
 Logo language, 55–57

loop variable
 adding 1 to, 202
 counting backward, 205–206
 counting by a range, 203–204
 counting by different increments, 204–205
 FOR-NEXT loop, 201–206
 overview, 201–203
 WHILE loop, 208
 looping statements. *See also specific statements*
 C/C++, 533–534
 endless loops, 201–202, 207, 212
 EXIT command, 211
 initializing arrays using, 317
 Java/C#, 551–552
 JavaScript, 492–493
 nested loops, 209–211
 overview, 35, 199–200
 parts of loops, 35
 Pascal/Delphi, 583–584
 PHP, 504–505
 Ruby, 515–516
 tips, 211–212
 VB/RB, 600–601
 loose coupling, 135–136, 137–138
 lossless compression algorithms
 Burrows-Wheeler transform, 436–438
 dictionary encoding, 439–442
 lossy algorithms versus, 435, 442–443, 444
 run-length encoding, 436
 lossy compression algorithms
 bit depth, 444
 CBR versus VBR, 444

lossless algorithms
 versus, 435, 442–443, 444
 overview, 442–444
 Lotus 1-2-3, 659
 low coupling, 135–136, 137–138
 lowercase. *See* capitalization
 LZ77 algorithm, 440
 LZ78 algorithm, 440
 LZW algorithm, 440–442

M

Mac OS X
 AppleScript with, 76
 compilers, 25, 84, 85, 90
 described, 18
 editors, 19
 software for running
 Windows and Linux on, 19
 Universal Binary files, 21
 machine language
 assemblers for, 16
 created by compilers, 21
 defined, 11
 other languages compared to, 15
 overview, 11
 speed and efficiency of, 16, 666
 uses for, 16, 17
 machine learning
 Bayesian probability, 653
 neural networks, 653–655
 overview, 644, 652–653
 robotics, 655
 machine-language debuggers, 98
 macro languages. *See* scripting languages
 malware, 633–636

math functions, 165–166
 mathematical operators
 C/C++, 528
 common (table), 163
 Java/C#, 545
 JavaScript, 488
 Pascal/Delphi, 580
 Perl/Python, 561–562
 PHP, 500
 precedence, 164–165
 Ruby, 511–512
 VB/RB, 596
 memento design pattern, 245–246
 memory
 ASLR, 642
 OOP requirements, 248
 sorting algorithm requirements, 394
 storage requirements for data types, 149–150
 menus (user interface), 290–291, 294, 295
 merge sort algorithm, 403–404, 406–408
 methodologies. *See* programming methodologies
 methods. *See* subprograms
 microprocessors. *See* processors
 Microsoft. *See also* Windows (Microsoft)
 C# development by, 541
 compilers no longer supported by, 84
 Excel, 659
 JScript interpreter, 91
 Office suite compilers, 84
 PowerShell interpreter, 91
 Robotics Studio, 665
 Silverlight RIA tool, 665
 Mod operator
 (Pascal/Delphi), 580

modeling, 44–45
 Modula-2 language, 58
 module files (VB/RB), 591
 modules, restricting
 variable scope to,
 156–157
 MP3 compression, 443, 444
 multi-dimensional arrays
 creating, 322–323
 limitations, 375–376
 overview, 321–322
 retrieving data from, 323
 storing data in, 323
 multiple inheritance,
 240, 539
 mutating molecules,
 627–628

N

names
 storing values in
 subprogram name,
 225–227
 for subprograms, 216
 variable naming
 conventions, 143–144
 natural language
 processing (NLP),
 648–650
 .NET framework, 68–69,
 133, 541
 NetBeans IDE (Java), 26, 97
 neural networks, 653–655
 niche markets, 8
 NLP (natural language
 processing), 648–650
 NOT operator, 175. *See also*
 logical/Boolean
 operators
 numbers. *See also* decimal
 data types; floating
 point data types;
 integer data types

data types for, 148–150
 initializing arrays for, 317
 in variant data type,
 326, 595
 NXT-G language (Lego),
 61, 666

O

obfuscators, 106
 object model (UML),
 117–119
 Objective-C language, 667
 object-oriented
 programming (OOP).
 See also objects
 advantages, 47, 232, 235
 combining with other
 methodologies, 48
 complexity eased by,
 44–45
 data isolated with, 45–46
 defining an object with a
 class, 249–251
 design patterns, 243–246
 development of, 232
 disadvantages, 248
 encapsulation, 235–238
 examples, 249–258
 inheritance, overview, 47,
 48, 238–242
 inheriting an object,
 253–256
 languages, 246–248
 loose coupling enforced
 by, 136
 modeling with, 44
 modifications simplified
 by, 46–48
 need for, 43
 not foolproof, 235
 overview, 232–235
 polymorphism, overview,
 242–243

polymorphism, using to
 rewrite a subprogram,
 256–258
 reusability with, 44, 46
 structured programming
 versus, 44–47
 objects. *See also* object-
 oriented programming
 (OOP)
 breaking programs into,
 135–138
 C++, 537–539
 C#, 556–557
 creating from classes,
 251–252
 data isolated by, 45–46
 defined, 45
 defining with classes,
 249–251
 Delphi, 587–588
 inheriting, 253–256
 Java, 556–557
 modifications simplified
 by, 46–48
 parts of, 234
 Pascal, 587–588
 Perl/Python, 572–573
 PHP, 507
 physical items
 represented by,
 136–137
 private versus public data
 in, 234–235
 private versus public
 subprograms in,
 234–235
 REALbasic, 605
 Ruby, 518–519
 running subprograms
 stored in, 252–253
 subprograms compared
 to, 136–137, 232–233
 Visual Basic, 605
 OCR (optical character
 recognition), 651

offline sorting algorithms, 394

one-based arrays, 315, 316

one-time pad algorithm, 446

online sorting algorithms, 394

open source compilers, 85

operating systems. *See also specific kinds*

choices for programming, 18–19

choosing compiler for, 84–85

compiler support for, 88–89

compilers specific to, 21–22

defined, 18

editors free with, 19

future of, 658–660

interpreters, 91

p-code compatibility issues, 23

shielded from programmers by BASIC, 54

operators. *See also specific kinds*

assignment (overview), 162

C/C++, 528–531

defined, 162

increment (overview), 202–203

Java/C#, 545–548

JavaScript, 488–490

language differences for, 161

logical/Boolean (overview), 174–178

mathematical (overview), 162–165

Pascal/Delphi, 580–581

Perl, 561–563

PHP, 500–502

precedence, 164–165

Python, 561–563, 564–565

regular expressions, 168–171

relational/comparison (overview), 172–174

Ruby, 511–513

VB/RB, 596–597

optical character recognition (OCR), 651

optimizing programs

compiler features for, 86–87

profilers for, 102–103

Or operator, 176–177. *See also logical/Boolean operators*

ordered trees

binary, 382–383

B-trees, 383–384

ordinary, 381–382

OrElse operator (Visual Basic), 597

organizing programs

breaking into subprograms, 131–135

creating a user interface, 138–140

dividing into objects, 135–138

need for, 130–131

storing subprograms in separate files, 36–37, 131, 133–135

organizing user interfaces, 301–302

p

parameter passing

calling functions, 226–227

calling subprograms, 219–222

parameter list for, 219, 226

parameters, defined, 219

by reference, 222–225, 602

scope of variables with, 158–159

by value, 223, 602

PARRY NLP program, 650

Pascal language

advantages, 57–58

branching statements, 581–582

comments, 281, 577

constants, 579

creating functions and subprograms, 584–585

data structures, 585–587

declaring variables, 147, 577–579

development of, 57

fading popularity of, 58

as high-level language, 14

linked lists with, 343–344

looping statements, 583–584

objects, 587–588

overview, 575–576

program structure, 576

structured programming in, 37, 57–58

passing parameters. *See parameter passing*

password algorithms, 446–447. *See also keys for encryption*

patching, 640

pattern matching

combining regular expressions, 170–171

with multiple-character wildcards, 169–170

with ranges, 170

with single-character wildcard, 168

for specific characters, 169

payload of viruses, 634

P-box algorithms, 447–448

p-code

- advantages and
 - disadvantages, 23
 - compatibility issues, 23
 - compiling to a virtual machine for, 93–94
 - with Java, 23, 66
 - with .NET framework, 68
 - overview, 23
 - with Revolution, 77–78
 - virtual machines for, 662
 - with Visual Basic, 77
- PDF (Portable Document Format), 264
- performance
 - arrays, 328
 - assembly language, 16
 - BASIC limitations, 589
 - compiler issues, 16, 64, 87–88, 130
 - of compilers, 87–88
 - keywords' affect on, 64, 130
 - machine language, 16
 - sorting algorithms, 394, 406–408
 - string searching
 - algorithms, 433
 - structures, 328
 - virtual machines, 94
- Perl
 - based on C, 50
 - bioinformatics in, 627, 628, 631, 632
 - branching statements, 565–566
 - comments, 278, 561
 - creating functions, 568–569
 - data structures, 569–570
 - default sorting algorithm, 404
 - defining variables, 561
 - interpreter for, 92
 - looping statements, 566–568
 - objects, 572–573
 - operators, 561–565
 - overview, 559–560
 - program structure, 560–561
 - Python compared to, 559–560
 - for transferring data
 - among programs, 77
- permutation box
 - algorithms, 447–448
- PGP (Pretty Good Privacy), 454, 455
- phonetic string searching, 431–433
- PHP (PHP Hypertext Processor)
 - arrays, 506
 - branching statements, 502–504
 - comments, 498
 - creating functions, 505
 - creating objects, 507
 - declaring variables, 499
 - HTML markup for, 497–498
 - interpreter for, 92
 - looping statements, 504–505
 - operators, 500–502
 - program structure, 497–498
- Phrogram teaching language, 60
- plain text files. *See* text files
- plaintext attacks on
 - encryption, 458
- pointers, 338. *See also* linked lists
- polymorphism, 242–243, 256–258
- popping data from a stack, 363–364
- portability. *See also* cross-platform programming
 - of C, 65, 660–661
 - defined, 65
 - of Java, 66, 541
 - of REALbasic, 589–590
 - Visual Basic issues, 589
- Portable Document Format (PDF), 264
- postorder traversal of trees, 386
- PowerShell interpreter (Windows), 91
- precedence of operators, 164–165
- preorder traversal of trees, 385
- Pretty Good Privacy (PGP), 454, 455
- private versus public data and subprograms, 234–235
- private-key (symmetric) encryption, 452–453
- problem solving in AI
 - expert systems, 646–648
 - game-playing, 645–646
 - image recognition, 651–652
 - natural language processing, 648–650
 - overview, 644
 - speech recognition, 650–651
- problem solving in programming, 7–9, 127–128
- procedural languages, 70
- procedures. *See* subprograms
- processors
 - assembly language
 - particular to, 14
 - C manipulation of, 15
 - compilers specific to, 84
 - defined, 11
 - functions of, 12
 - native language of, 11
 - registers, 12, 13
- profilers, 83, 96, 102–103

- programming basics
 - assemblers or compilers, 21–22
 - computer and operating system choices, 18–19
 - defining the steps, 9
 - desire versus training, 17–18
 - editors, 19–20
 - history, 10–17
 - identifying the problem, 8
 - interpreters, 22–23
 - knowing programming
 - versus knowing a language, 26–27
 - p-code (pseudocode), 23
 - starting with Java, 26
 - starting with Linux, 25–26
 - starting with Mac OS X, 25
 - starting with Windows, 24–25
 - subjects involved, 1
 - taking time to experiment, 23–24
- programming languages.
See languages
- programming
 - methodologies
 - combining, 48
 - event-driven
 - programming, 38–43
 - object-oriented
 - programming, 43–48
 - spaghetti programming, 31–32, 33
 - structured programming, 32–37
 - tools, 29–30
 - top-down programming, 35–37
- programming tools. *See* tools
- Project command for
 - databases, 618, 619
- Prolog language, 70–72, 73, 653
- proprietary file formats, 271

- proprietary languages, 58–59
- pruning sub-trees, 387–388, 389
- pseudocode. *See* p-code
- pseudorandom numbers, 449–450
- public versus private data and subprograms, 234–235
- public-key (asymmetric) encryption, 453–455
- Python
 - bioinformatics in, 631, 632
 - branching statements, 565–566
 - calling functions, 227
 - comments, 278, 561
 - creating functions, 568–569
 - data structures, 570–572
 - defining variables, 561
 - interpreter for, 92
 - looping statements, 566–568
 - objects, 572–573
 - operators, 561–563, 564–565
 - overview, 559–560
 - Perl compared to, 559–560
 - program structure, 560–561
 - reading text files, 263
 - sets in, 330–337
 - subprogram creation in, 220
 - for transferring data among programs, 77
 - writing text files, 262

Q

- queues
 - adding data, 367
 - in C#, 556
 - counting and searching, 368–370

- declaring variables
 - for, 366
- deleting data, 368
- overview, 365–366
- quick sort algorithm, 405–408

R

- Rabin-Karp algorithm, 426
- RAD (rapid application development) tools
 - database, 275–276
 - user interface, 139, 140, 293
- radio buttons, 297
- random-access files
 - overview, 264–266
 - proprietary formats, 271
 - reading, 267–268
 - text files versus, 264
 - uses for, 276
 - writing, 266–267
- RAT (remote access Trojan), 636
- RB. *See* REALbasic
- RC4 stream cipher, 450
- reading files
 - defined, 263
 - random-access, 267–268
 - text, 263
 - untyped, 269–271
- REALbasic. *See also* BASIC (Beginner's All-purpose Symbolic Instruction Code)
 - branching statements, 597–599
 - class files, 591
 - comments, 592
 - compiler, 26
 - constants, 595–596
 - creating functions and subprograms, 601–602
 - data structures, 603–604
 - declaring variables, 592–595

- looping statements, 600–601
 - module files, 591
 - objects, 605
 - operators, 596–597
 - portability, 589–590, 661
 - program structure, 590–591
 - Visual Basic versus, 667–668
 - Web site, 40, 589
 - window files, 590–591
 - records (data structures). *See* structures
 - records in random-access files, 264–265
 - recursion, 227–229, 394
 - reference, passing
 - parameters by, 222–225
 - registers, 12, 13
 - regular expressions (RegEx)
 - combining, 170–171
 - defined, 168
 - multiple-character wildcards in, 169–170
 - ranges of characters in, 170
 - single-character wildcard in, 168
 - square brackets in, 169
 - for string searching, 429–431
 - relational databases, 273, 274, 613–617
 - relational/comparison operators
 - in branching statements, 181
 - C/C++, 529
 - common (table), 172
 - Java/C#, 546
 - JavaScript, 488–489
 - overview, 172–174
 - Pascal/Delphi, 580
 - Perl/Python, 562–563
 - PHP, 500
 - Ruby, 512
 - VB/RB, 596
 - Remember icon, 4
 - remote access Trojan (RAT), 636
 - REPEAT-UNTIL loops (Pascal/Delphi), 583–584
 - repository, 121–122
 - reserved words. *See* keywords
 - resizable arrays, 319–321, 326
 - reusability. *See also* libraries
 - inheritance for, 241–242
 - object-oriented programming for, 44, 46
 - polymorphism for, 243
 - structured programming issues, 46–47
 - subprogram issues, 231
 - revision control programs, 121–122
 - Revolution scripting language, 77–78
 - RIAs (rich Internet applications), 664–665
 - Robocode programming game, 62
 - robotics, 61, 655, 665–666
 - Robotics Studio (Microsoft), 665
 - rootkit detectors, 638
 - RTF (Rich Text Format), 261
 - Ruby
 - branching statements, 514–515
 - comments, 510–511
 - creating functions, 516
 - creating objects, 518–519
 - data structures, 517–518
 - declaring variables, 511
 - interpreter for, 92
 - looping statements, 515–516
 - operators, 511–513
 - overview, 509
 - program structure, 510
 - Ruby on Rails framework for, 509
 - for transferring data among programs, 77
 - Run BASIC site, 590
 - run-length encoding (RLE) algorithm, 436
-
- ## S
-
- S-box algorithms, 447–448
 - scope of variables
 - described, 154
 - general rule for, 159
 - global, 154–156
 - module, 156–157
 - parameter passing, 158–159
 - subprogram, 157–159
 - Script Editor interpreter (Mac), 91
 - scripting languages. *See also specific languages*
 - for automating repetitive tasks, 75
 - for customizing programs, 76
 - with database programs, 80
 - defined, 74
 - future of, 668
 - as glue, 77, 668
 - interpreters for, 22–23, 74
 - overview, 74–75
 - for standalone applications, 77–78
 - traditional programming languages versus, 74–75
 - for transferring data among programs, 77
 - as typeless, 74–75
 - uses for, 75
 - search space, 409

- searching. *See also*
 - searching algorithms
 - arrays, challenges for, 326
 - collections, 350–351
 - dictionaries, 353
 - queues, 369–370
 - regular expressions for, 429–431
 - stacks, 365
 - trees (traversal), 382, 383, 384, 385–386
- searching algorithms
 - adversarial search, 418–422
 - backward or forward searching, 411
 - binary searching, 413
 - BLAST, 629
 - block searching, 412
 - indexes for, 416–418
 - informed or heuristic, 409–410
 - interpolation searching, 414–416
 - search space for, 409
 - sequential search, 410–416
 - string searching, 423–433
 - uninformed or brute-force, 409–410
- secure computing
 - coding securely, 640–641
 - overview, 639–640
 - patching, 640
 - security by design, 641–642
- Secure Sockets Layer (SSL), 454
- security
 - firewalls, 635, 636, 637
 - need for, 633–634
 - secure computing, 639–642
 - stopping hackers, 637–639
 - stopping malware, 634–636
- through obscurity, 447
- SELECT CASE statements
 - checking range of values, 194–195
 - comparing values, 195
 - ELSE statement with, 195–196
 - general rule for, 196–197
 - IF-THEN statements versus, 196–197
 - matching multiple values, 193–194
 - overview, 190–191
 - running at least one command, 195–196
 - SWITCH statement, 191–193
 - VB/RB, 598–599
- Select command for
 - databases, 617–618, 619
- selection sort algorithm, 396–397, 406–408
- self-documenting code, 277–278
- self-synchronizing stream ciphers, 450
- Sequence diagrams (CASE), 119
- sequences, 33–34
- sequential search algorithms
 - backward or forward searching, 411
 - binary searching, 413
 - block searching, 412
 - Fibonacci searching, 414–416
 - interpolation searching, 414–416
 - overview, 410
 - text search, 424–429
- sequential text search
 - Boyer-Moore algorithm, 425
 - finite automaton algorithm, 428–429
- overview, 424–425
- Rabin-Karp algorithm, 426
- Shift Or algorithm, 427–428
- sets
 - adding data, 331
 - advantages over arrays, 330, 332
 - checking for membership, 332–333
 - combining elements with difference
 - command, 335–337
 - combining elements with intersection
 - command, 334–335
 - combining with union
 - command, 333–334
 - deleting data, 331–332
 - drawbacks, 341–342
 - language features for, 330
 - in Pascal/Delphi, 587
- Seven Bridges of Königsberg problem, 378–379
- SharpDevelop IDE (Windows), 97
- Shell, Donald (algorithm inventor), 398
- shell sort algorithm, 398–399, 406–408
- Shift Or algorithm, 427–428
- show stopper bugs, 97
- Silverlight RIA tool (Microsoft), 665
- SIMULA language, 232
- single inheritance, 240
- sliders, 299
- slot machine hacking, 450
- SNOBOL operators, 161
- software as service model, 663–664
- software engineering
 - CASE (Computer-Aided Software Engineering), 115–122

- extreme programming
 - method, 112–114
- goals, 108
- need for, 107–108
- pros and cons, 117, 122–123
- waterfall model, 108–112
- sorting algorithms
 - Big-O notation for, 407
 - bubble sort, 394–396
 - comparison of, 406–408
 - factors to consider, 393–394
 - heap sort, 399–403
 - in-place, 394
 - insertion sort, 397–398
 - merge sort, 403–404
 - online versus offline, 394
 - overview, 393–394
 - quick sort, 405–406
 - recursion in, 394
 - selection sort, 396–397
 - shell sort, 398–399
- Soundex algorithm, 431–433
- source code
 - commenting out, 284–285
 - comments, 278–285
 - compilers or assemblers for, 21–22
 - defined, 20
 - disassemblers for, 83, 105–106
 - formatting automatically, 120–121
 - generating automatically, 119–120
 - interpreters for, 22–23
 - obfuscators for, 106
 - self-documenting, 277–278
 - source code formatters, 120–121
 - source-level debuggers, 98
 - spaces, initializing arrays with, 317
 - spaghetti programming, 31–32, 33
 - specifications, defined, 109
 - speech recognition, 650–651
 - spreadsheet programs, 659
 - spyware, 636
 - SQL (Structured Query Language), 271, 620–621
 - SSL (Secure Sockets Layer), 454
 - stacks
 - adding data, 362–363
 - in C#, 556
 - counting and searching, 365
 - declaring variables for, 362
 - deleting data, 363–364
 - overview, 361–362
 - standalone editors, 95
 - statements, 128. *See also*
 - branching statements;
 - looping statements
 - steganography, 456
 - stepping through code
 - breakpoints for, 98–99
 - overview, 97–98
 - stepping out, 100
 - stepping over, 99
 - stream ciphers, 449–450
 - string searching algorithms
 - bioinformatics, 423
 - Boyer-Moore, 425
 - finite automaton, 428–429
 - performance, 433
 - phonetically, 431–433
 - Rabin-Karp, 426
 - regular expressions for, 429–431
 - sequential text search, 424–429
 - Shift Or, 427–428
 - Soundex, 431–433
 - strings
 - in C/C++, 526
 - common functions (table), 167
 - comparison operators for, 172–174
 - concatenation operator, 166–167
 - initializing arrays for, 317
 - in Java/C#, 543
 - in Pascal/Delphi, 577–578
 - regular expressions for, 168–171
 - storage requirements for, 150
 - in variant data type, 326, 595
 - in VB/RB, 592
 - strong coupling, 135–136
 - strongly-typed languages. *See* type-safe languages
 - structured programming
 - branches in, 34–35
 - combining with other methodologies, 48
 - loops in, 35
 - need for, 32
 - object-oriented programming versus, 44–47
 - Pascal designed for, 37
 - reusability issues, 46–47
 - sequences in, 33–34
 - top-down programming, 35–37
 - Structured Query Language (SQL), 271, 620–621
 - structures
 - arrays with, 314, 323–325
 - in C#, 554
 - in C/C++, 536
 - defined, 311–312, 536
 - as records in some languages, 311
 - retrieving data from, 313
 - speed and efficiency issues, 328
 - storing data in, 312–313
 - in Visual Basic, 603
 - style classes (CSS), 479–480

stylesheets. *See* CSS
(cascading stylesheets)
subprograms. *See also*
 functions
 for bioinformatics, 632
 calling, 217–218
 comments documenting,
 283–284
 creating in Pascal/Delphi,
 584
 creating in VB/RB,
 601–602
 creating (overview),
 216–217
 in curly bracket
 languages, 218–219
 grouped by encapsulation,
 237
 #include command for,
 218–219
 libraries of, 133
 limitations and problems,
 135–136, 231, 235–236,
 238–239
 methods as, 234
 module files for (VB/RB),
 591
 for multiple languages in
 one program, 134–135
 naming, 216
 need for, 131
 objects compared to,
 136–137, 232–233
 parameter passing,
 158–159, 219–227
 private versus public,
 234–235
 protected by
 encapsulation, 238
 repeating with recursion,
 227–229
 rewriting using
 polymorphism, 256–258
 stored in objects, running,
 252–253

 storing in separate files,
 36–37, 131, 133–135
 storing values in name,
 225–227
 tight versus loose
 coupling, 135–136
 in top-down programming,
 35–36
 uses for, 133–135, 213–215
 variable scope isolation
 in, 157–158
 window files (VB/RB),
 590–591
substitution box
 algorithms, 447–448
substitution ciphers,
 445–446
sub-trees
 defined, 387
 grafting, 388, 390
 pruning, 387–388, 389
Sun Microsystems. *See also*
 Java
 Java compiler and VM
 from, 26
 Java development by, 541
SWITCH statements. *See*
 also case statements
 break command with,
 192–193, 503
 C/C++, 532–533
 general rule for, 196–197
 IF-THEN statements
 versus, 196–197
 Java/C#, 549–551
 JavaScript, 491–492
 matching multiple values,
 194
 overview, 191–192
 PHP, 503–504
 SELECT CASE statement
 versus, 195
symmetric (private-key)
 encryption, 452–453

synchronous stream
 ciphers, 450
systems programming
 languages, 73–74, 668

T

tab-delimited text files,
 260–261
tables (HTML)
 captions, 474, 475
 creating, 471–472
 header and footer,
 474–475
 headings, 472–473
 rows and data, 473–474
tabs (user interface), 302
teaching languages
 Alice, 60–61
 BASIC as, 52–55
 “battling robot”
 programming games,
 61–62
 goal of, 52
 KPL (Kid’s Programming
 Language), 59–60
 Lego Mindstorms NXT-G,
 61, 666
 Logo, 55–57
 Pascal as, 57–58
 proprietary, 58–62
technical design
 documents, 286
Technical Stuff icon, 4
text boxes, 296–297
text files
 comma-delimited, 260–261
 cookies as, 264
 creating, 261–262
 defined, 259
 overview, 259
 random-access files
 versus, 264
 reading, 263
 RTF, 261

- source code as, 264
 - tab-delimited, 260–261
 - uses for, 276
 - text (HTML)
 - aligning, 466
 - body, 464–465
 - colors, 467–468
 - comments, 469
 - emphasizing, 467
 - font size, 468
 - headings, 465
 - line breaks, 464–465
 - titles, 464
 - text searching. *See* string searching algorithms
 - TextWrangler editor (Mac), 19
 - ThinkFree Microsoft Office clone, 663
 - third-party components, 102, 273–275
 - Three Cottage problem, 380
 - tight coupling, 135–136
 - Tip icon, 4
 - titles (HTML), 464
 - toolbars and toolboxes, 294–296
 - tools
 - CASE, 115
 - choosing a compiler, 84–90
 - choosing an interpreter, 90–92
 - compiling to a virtual machine, 93–94
 - database toolkits, 273–275
 - debuggers, 97–101
 - disassemblers, 105–106
 - documentation, 286–287
 - editors, 19–20, 94–97
 - efficiency improved by, 29–30
 - help file creators, 83, 103–104, 288
 - installer programs, 104–105
 - minimum needed, 106
 - profilers, 102–103
 - third-party components, 102
 - for user interfaces, 139, 140
 - variety available, 83
 - top-down programming, 35–37
 - topological graph theory, 380
 - tracing code. *See* stepping through code
 - training, desire versus, 17–18
 - Traveling Salesman problem, 380
 - traversal of trees
 - B+ trees, 384
 - binary trees, 383
 - B-trees, 384
 - defined, 385
 - in-order, 385–386
 - level order, 386
 - ordered trees, 382
 - postorder, 386
 - preorder, 385
 - trees
 - adding data, 386, 387
 - B+ trees, 384
 - binary, 382–383
 - B-trees, 383–384
 - deleting data, 386–387
 - grafting sub-trees, 388, 390
 - heap sort algorithm using, 399–402
 - internal nodes, 380
 - leaf nodes, 380
 - ordered, 381–382
 - overview, 380–381
 - pruning sub-trees, 387–388, 389
 - root node, 380
 - sub-trees, defined, 387
 - traversing (searching), 382, 383, 384, 385–386
 - unordered, 381
 - uses for, 388–389
 - Trojan horses, 636
 - True BASIC, 52
 - tuples
 - database, 616
 - Python, 571
 - Turbo compilers (Borland), 25
 - Turing, Alan (mathematician), 643
 - Turing Test, 643
 - typeless versus type-safe languages, 74–75, 178–179
 - type-safe languages
 - advantages and disadvantages, 67, 68
 - C# as, 68
 - converting data types, 179–180
 - as strongly-typed, 74
 - typeless languages versus, 74–75, 178–179
-
- U
- UML (Unified Modeling Language)
 - dynamic model, 119
 - functional model, 117, 118
 - generating code automatically, 119–120
 - object model, 117–119
 - overview, 116–117
 - problems encountered, 116, 117
 - unclustered indexes, 417–418
 - undirected graphs, 377–378
 - Unicode character codes, 423

uninformed searching
 algorithms. *See* brute-force searching algorithms
 Universal Binary files (Mac), 21
 UNIX, 18. *See also* Linux
 unordered trees, 381
 UNTIL loops (Ruby), 516
 untyped files
 overview, 268
 proprietary formats, 271
 reading, 269–271
 uses for, 276
 writing, 268–269
 uppercase. *See* capitalization
 Use Case diagrams (CASE), 117, 118
 user interface
 advantages for users, 38, 292–293
 basic functions of, 293
 buttons, 294
 challenges for programmers, 39
 check boxes, 297–298
 combo boxes, 298–299
 command-line, 290
 creating, 139–140
 customizing, 40–42
 defined, 39
 design tips for, 302–307
 designers with IDEs, 96
 designing visually, 39–40, 41, 42
 dialog boxes, 300–301
 displaying commands to users, 293–296
 displaying information to users, 300–301
 before event-driven programming, 39
 event-driven programming for, 38–43
 evolution of, 290–293
 of expert systems, 646

getting data from users, 296–299
 JavaScript for designing, 494–496
 libraries of subprograms for, 139, 293
 list boxes, 298
 menus, 290–291, 294, 295
 organizing, 301–302
 overview, 138–139
 purpose of, 289
 RAD tools for, 139, 140, 293
 radio buttons, 297
 sliders, 299
 tabs, 302
 text boxes, 296–297
 toolbars, 294–295
 toolboxes, 295–296
 transparent to user, 293
 writing event handlers, 42–43
 writing from scratch, 293
 user manuals, 286
 user-defined data types, 265, 312
 utilities. *See* tools

V

variable bit rate (VBR), 444
 variables. *See also* data types; declaring variables
 assigning value of one to another, 152
 assigning values to, 150–151, 162
 data overwritten by new values, 151
 defined, 142, 152
 enumerated (C/C++), 536–537
 global, 154–155
 initializing before looping, 207, 209
 limitations, 311

loop variable, 201–206
 modifying a variable by itself, 152
 module, 156–157
 passing data among subprograms, 158–159
 retrieving data from, 151–152
 scope, 154–159
 in subprograms, 157–159
 watching with debugger, 100–101
 variant data type, 326, 595
 VB. *See* Visual Basic
 VBA (Visual Basic for Applications), 75
 VBR (variable bit rate), 444
 version control programs, 121–122
 video games, 645
 video tutorials, 286
 VIM editor, 95
 virtual machines (VMs)
 compiling to, 93–94
 for cross-platform programming, 662–663
 Java, 66
 for p-code, 23
 speed issues, 94
 viruses, 634
 Visual Basic. *See also* BASIC (Beginner's All-purpose Symbolic Instruction Code)
 branching statements, 597–599
 class files, 591
 collections in, 346–351
 comments, 592
 constants, 595–596
 creating functions and subprograms, 601–602
 data structures, 603–604
 database programming features, 80
 declaring variables, 147, 592–595

as event-driven programming language, 40
 looping statements, 600–601
 module files, 591
 .NET compatibility, 68
 objects, 605
 operators, 596–597
 p-code used by, 77
 portability issues, 589
 program structure, 590–591
 queues in, 366–370
 REALbasic versus, 667–668
 stacks in, 362, 363–365
 for standalone applications, 77
 Web site, 589
 window files, 590–591
 Visual Basic for Applications (VBZ), 75
 VMs. *See* virtual machines

W

Warning! icon, 4
 watching variables, 100–101
 waterfall model, 108–112
 weak coupling, 135–136, 137–138

Web resources. *See* Internet resources
 weighted graphs, 377–378
 WHILE loops
 C/C++, 534
 counting, 208
 endless, 207
 initializing the variable, 207
 Java/C#, 551–552
 JavaScript, 492
 nested, 209–211
 overview, 206–208
 Pascal/Delphi, 583
 Perl/Python, 568
 PHP, 505
 reading text files using, 263
 reading untyped files using, 269–270
 Ruby, 516
 wildcards for pattern matching, 168–171
 window files (VB/RB), 590–591
 windows in user interface (JavaScript), 496
 Windows (Microsoft) compilers, 24–25, 89
 as largest market, 22
 as largest market for programs, 18

.NET framework, 68–69
 Notepad editor with, 19
 running on Mac OS X, 19
 worms, 635
 writing files
 random-access, 266–267
 text, 261–262
 untyped, 268–269

X

xBASE languages, 79
 Xcode compiler, 25, 84, 85
 XOR operator, 177–178. *See also* logical/Boolean operators
 XP (extreme programming), 112–114

Z

zero (0)
 initializing arrays, 317
 zero-based arrays, 315–316

