

# Index

- (negation operator), 62, 91, 92
- (subtraction operator), 62
- , 227–231
- ==, 105–107
- > (object pointer), 731
- > (structure pointer), 623–624, 625–627
- !, 182, 187–188
- !=, 150–152
- " ", 30
- #, 30
- %, 62
- %, 105–107
- & (address operator), 495–497
- & (reference variables), 349
- &&, 182–184, 188
- ( ), 30
- \* (indirection operator), 501
- \* (multiplication operator), 62
- \* (pointer variable declaration), 626–627
- \*, 105–107
- . (dot operator), 604, 630, 724
- /, 62
- /\* \*/, 70
- //, 30, 70
- !=, 105–107
- ::, 722, 757
- ;, 30
- ?:, 199
- \\, 35
- \', 35
- \", 35
- \a, 35
- \b, 35
- \n, 35
- \r, 35
- \t, 35
- {, 30, 164, 236–237
- ||, 182, 184–187, 188
- }, 30, 164, 236–237
- ~ (destructor), 758
- +, 62, 586
- ++, 227–231
- +=, 105–107, 586
- <, 150–152
- <>, 30
- << (stream insertion), 31–32, 270
- <=, 150–152
- = (assignment operator), 38, 59, 161–162

- ==, 150–152, 161–162, 237
- >, 150–152
- >> (stream extraction operator), 84, 586
- [] (array subscript operator), 429, 431–433

## A

- abstract array data type case study, 782–785
- abstract base classes, 945–949
- abstract data type (ADT) definition, 1063–1064
  - in STL, 1006–1007
  - and structures, 599–601
- abstraction, 599
- accessors, 722
- access specifiers
  - base class, 918–923
  - in class declarations, 718
- accumulator, 257
- actual argument, 311
- actual parameters, 311
- addition operator (+), 600
- address, memory, 6
- address operator (&), 495–497
- ADT, *see* abstract data type
- aggregation, 860–864
  - “has a” relationship, 862
  - in UML diagrams, 864
- algebraic expressions, 92–94
- algorithm, 8
  - exhaustive, 1148–1151
  - factorial, 1126–1129
  - QuickSort, 1144–1147
  - search, 457–463
  - sorting, 470–477
  - STL, 1005–1006, 1007–1008
- Alice software, 1185–1210
- ALU (arithmetic and logic unit), 5
- American Standard Code for Information Interchange (ASCII), 49
- anonymous enumerated type, 635–636
- anonymous unions, 630–632
- append member function, *string* class, 588
- application software, 7–8
- arguments, 93–94, 311–315
  - actual, 311
  - arrays as, 407–417
  - formal, 311
  - passing, with pointers, 518–519, 578–580
  - structures as, 617–619

- arithmetic and logic unit (ALU), 5
- arithmetic assignment operators, 105
- arithmetic expressions, 89–98
- arithmetic operators, 61–69, 92
- arrays, 375–443
  - accessing elements, 377–378
  - assigning one, to another, 398–399
  - averaging values in, 400–401
  - binary search, 460–463
  - BookInfo structure, 611
  - bounds checking, 384–386
  - bubble sort, 470–474
  - comparing, 403–404
  - const key word, 412–413
  - contents of, 383–384
  - described, 375–376
  - enum with, 636–638
  - from file to, 382–383
  - as function arguments, 407–417
  - highest and lowest values, 401
  - initialization, 387–392, 613, 767
  - inputting and outputting, 378–382
  - linear search, 457–460
  - linked lists *vs.*, 1025
  - memory requirements of, 376–377
  - National Commerce Bank case study, 427–429
  - of objects, 767–768
  - off-by-one error, 386–387
  - parallel, 404–407
  - partially filled, 401–403
  - and pointers, 504–508
  - printing contents of, 399–400
  - processing array contents, 396–404
  - range-based for loop, 392–396
  - search algorithms, 457–463
  - selection sort, 474–477
  - sizeof operator, 681
  - sorting algorithms, 470–477
  - and STL vectors, 429–443, 485–490
  - structure, 611–613
  - summing values in, 400
  - three or more dimensions, 425–427
  - two-dimensional, 418–425
- array subscript operator ([]), 429, 431–433, 586
- arrow operator (->), 623, 626, 730
- ascending order, sorting in, 470
- ASCII, 49, 50, 195, 1211–1212
- assign member function, string class, 588
- assignment
  - combined, 105–107
  - memberwise, 824–825
  - multiple, 104

- assignment operator (=), 38, 59–60, 104, 161–162, 586
- assignment statement, 59
- associative containers, 429, 1006
- associativity, 91–92, 188–189, 1213
- at member function
  - string class, 588
  - vector, 442, 1009
- atof library function, 569, 570
- atoi library function, 569, 570
- atol library function, 569, 570
- attributes, 712
- auto key word, 60–61
- averages
  - in arrays, 400–401
  - calculating, 95–96

## B

- back member function, list, 1056
- bad\_alloc exception, 988–989
- bad member function, file stream, 668
- base case, recursion, 1126
- base class, 892
  - abstract, 945–949
  - multiple, 952–957
- base class access specification, 896–897, 904–905
- base class functions, redefining, 918–923
- base class pointers, 937–939
- BASIC, 11
- begin member function
  - iterator, 1010
  - string class, 588
  - vector, 1009
- bidirectional iterators, 1007
- binary digit (bit), 6
- binary files, 266, 680–681
- binary numbers, 9
- binary operators, 61, 62
- binary search, 460–463
  - efficiency, 463
  - recursive version, 1121–1124
- binary\_search algorithm (STL), 1008, 1015
- binary trees, 1155–1156
  - applications of, 1155–1156
  - child nodes, 1155
  - creating, 1159–1160
  - deleting a node, 1166–1175
  - described, 1155–1156
  - inorder traversal, 1163
  - inserting a node, 1160–1162
  - leaf nodes, 1155
  - NULL address, 1155
  - operations, 1158–1175

- postorder traversal, 1163
- preorder traversal, 1163
- root node, 1155
- searching, 1165–1166
- searching for a value in, 1145
- search trees, 1157
- subtrees, 1156
- templates for, 1175–1180
- traversing, 1162–1164
- tree pointer, 1155
- binding, 931
  - dynamic, 931
  - static, 931
- bit, 6
- blocks, of code, 211–214, 236–237
- block scope, 212
- blueprints, classes as, 715
- body
  - of function, 301, 303
  - of loop, 233
- bool
  - data type, 57
  - flag, 181
  - returning from function, 332–334
- Boolean expressions, 57, 150, 383
- bounds checking, for arrays, 384–386
- braces, 236–237
- break statement, 204, 205, 284–286
- bubble sort, 470–474
- byte, 5–6

**C**

- C, 11
- C#, 11
- C++, 11
- C++ 11, 431
  - range-based for loop with vector in, 433
- calling a function, 302–309
- capacity member function
  - string class, 588
  - vector, 442, 1012
- capitalization, 42
- case conversion, character, 551–553
- case statement, 203–204
- case study
  - abstract array data type, 782–785
  - Demetris Leadership Center, 463–470, 477–485
  - dollarFormat function, 590–591
  - General Crates, Inc., 132–135
  - Home Software Company, 590–591, 778–781
  - National Commerce Bank, 427–429
  - United Cause, 536–540
- catch block, 971
- catch key word, 971

- CD, 6
- central processing unit (CPU), 3–5
- character case conversion, 551–554
  - tolower function, 552
  - toupper function, 552
- character literals, 48, 50–52
- characters, 17, 547–553
  - character literals, 48, 50–52
  - comparing, 195–198
  - converting cases, 551–553
  - and string objects, 118–124
- character testing, 547–551
  - functions, 549–550
  - isalnum function, 548
  - isalpha function, 548
  - isdigit function, 548
  - islower function, 548
  - isprint function, 548
  - ispunct function, 548
  - isspace function, 548
  - isupper function, 548
- char data type, 48–52
- cin, 18, 83–88
  - entering multiple values, 85–87
  - getline member function, 119, 557, 572–573
  - get member function, 120–122
  - ignore member function, 123, 573
  - inputting characters, 120
  - keyboard buffer, 87
- circle pointer, 625
- circularly linked list, 1055
- class constructors, 907–911
- classes, 711–810, 811–890
  - and abstract array data types, 782–785
  - abstract base, 945–949
  - accessors, 722
  - access specifiers, 718–719
  - aggregation, 860–864
  - arguments to constructors, 750–755
  - arrays of class objects, 767–768
  - base, 892
  - as blueprint, 715
  - collaborations, 865–867
  - const member functions, 720, 723
  - constructor overloading, 762–765
  - constructors, 746–757, 906–907
  - conversion of class objects, 858–860
  - copy constructor, 825–828
  - and data hiding, 723
  - declaration statements, 718
  - default constructor, 750, 757
  - defining class objects, 723–735
  - derived, 892–893
  - described, 718

- classes (*continued*)
  - destructors, 758–761, 906–910
  - dot operator (`.`), 724
  - dynamically allocated objects, 731–732, 760–761
  - finding, 788–796
  - forward declaration, 821
  - friend functions, 819–820
  - getter function, 722
  - “has a” relationship, 862
  - hierarchies of, 923–929
  - Home Software Company case study, 772–773
  - implementation file, 738
  - include guard, 738–739
  - inheritance, 891–900
  - inline member functions, 743–745
  - instance members, 811–818
  - “is a” relationship, 892–899, 929
  - member functions, defining, 721–722
  - memberwise assignment, 824–825
  - multiple inheritance, 952–959
  - mutators, 722
  - objects *vs.*, 711–714
  - operator overloading, 831–857
  - overloading member functions, 765
  - `PassFailExam`, 926
  - placement of public and private members, 720–722
  - pointers, 730–733
  - polymorphism, 929–945
  - private member functions, 765–767
  - private members, 718, 720–721, 736–737
  - problem domain, 789
  - and procedural/object-oriented programming, 711–717
  - protected members and class access, 900–905
  - public member functions, 720–721
  - public members, 718, 720–721
  - redefining base class functions, 918–923
  - responsibilities of, 788–810
  - scope resolution operator (`::`), 722, 757
  - setter function, 722
  - specification and implementation, 737–743
  - stale data, avoiding, 730
  - static members, 811–818
  - templates, 990–995
  - `this` pointer, 835–837
  - and UML, 785–788
  - virtual functions, 931–935
  - whole-part relationship, 862
- class implementation file, 738
- class specification file, 737
- class template objects, 1000
- class templates, 996–999
  - defining objects of, 1000–1001
  - and inheritance, 1002–1004
  - linked list, 1033–1035
  - type parameter, 990
- `clear` member function
  - file stream objects, 668
  - string class, 588
  - vector, 439, 442, 996–997
- C++ library, 547, 551, 554, 714
- `close` member function, file stream objects, 269
- closing of file, 269
- `cmath` header file, 94, 124
- COBOL, 11
- code reuse, 300
- collaborations, class, 865–869
- combined assignment operators, 105–107
- comments, 27, 69–71
  - `/* */`, 70
  - `//`, 27
  - multi-line, 70–71
  - single-line, 70
- compact disc (CD), 6
- `compare` member function, string class, 587–589
- compilers, 12, 657
- compound operators, 105
- concatenation, 559–560
- conditional expression, 199–202
  - value of, using, 200–201
- conditional loop, 247
- conditional operator, 199–202
- console, 31
- console output, 31
- `const`
  - as array parameter, 412–413
  - copy constructors, 825–828
  - member functions, 720, 723
- constants, 40
  - arguments passed to, 750–757
  - in base and derived classes, 906–917
  - constructors, 746–750
  - copy, 825–830
  - default, 750, 756–757
  - default arguments with, 756–757
  - global, 338–340
  - named, 71–73
  - overloading, 762–765
  - pointers to, 517–519
- constructors
  - class, 907–911
  - derived classes, 906
- containers, 1006
  - associative, 429, 1006
  - sequence, 429, 1006
  - STL, 429

- continue statement, 286–288
  - control unit, 5
  - control variable, 235
  - conversion
    - object, 858–860
    - string/numeric, 569–575
  - copy constructors, 825–830
    - const parameters in, 828–830
    - default, 830
    - and function parameters, 830
  - copy member function, `string` class, 589
  - count algorithm (STL), 1015
  - count-controlled loops, 247, 255–256
  - counters, 241–242
  - `cout`, 18, 31–35, 153
    - fixed manipulator with, 114–115
    - left manipulator with, 116–117
    - right manipulator with, 116–117
    - `setprecision` manipulator with, 111–113
    - `setw` manipulator with, 109–110
    - `showpoint` manipulator with, 115–116
  - `cout` statement, 229
  - C programming language, 10–11, 556
  - C++ programming language, 10, 11, 22
    - arithmetic operators, 61–69
    - assignment operation, 59–60
    - auto key word, 60–61
    - `bool` data type, 57
    - `char` data type, 48–52
    - comments, 69–71
    - `cout` object, 31–35
    - floating-point data types, 54–57
    - identifiers, 41–42
    - `#include` directive, 36–37
    - initialization, 59–60
    - integer data types, 42–47
    - named constants, 71–73
    - parts of program, 27–30
    - programming style, 73–75
    - scope, 61
    - `sizeof` operator, 58
    - `string` class, 52–53
    - variables and literals, 37–41
  - CPU (central processing unit), 3–5
  - CRC cards, 868–869
  - C++ runtime library, 124
  - `cstdlib` header file, 569
  - `cstring` header file, 558, 563, 567
  - C-strings, 554–566
    - in arrays, 556–558
    - comparing, 564–566
    - concatenation, 559–560
    - copying, 560–561
    - described, 554–555
    - filenames as, 284
    - handling functions, 575–580
    - length of, 558–559
    - library functions, 558–568
    - and null terminators, 554–555
    - numeric conversion functions, 569–575
    - searching within, 560–564
    - `strcmp` function, 583
    - and string literals, 555–556
    - `strPtr` points, 580
  - `c_str` member function, 284
  - `ctime` header file, 127
- D**
- database management systems (DBMS), 657
  - data hiding, 723
  - data types
    - abstract, 599–600
    - `bool`, 57
    - `char`, 48–52
    - coercion, 98
    - conversion, 98–99
    - demotion, 98
    - `double`, 54
    - enumerated, 632–642
    - `float`, 54
    - floating-point, 54–57
    - generic, 990
    - `int`, 43, 44
    - integer, 42–47
    - `long`, 44
    - `long double`, 44
    - numeric, 43
    - primitive, 600
    - promotion, 98
    - ranking, 98–99
    - `short`, 43, 44
    - size of, determining, 58
    - `string` class, 52–53
    - type casting, 101–104
    - `unsigned int`, 43, 44
    - `unsigned long`, 43, 44
    - `unsigned short`, 43, 44
  - debugging
    - desk-checking, 21
    - hand-tracing, 130–131
    - stubs and drivers, 361–362
  - decision making, 149–226
    - blocks and scope, 211–214
    - checking numeric ranges, 189
    - comparing characters and strings, 195–198
    - conditional execution, 162–164
    - conditional operator, 199–202
    - flags, 181–182

- decision making (*continued*)
  - if/else if statements, 176–178, 180–181
  - if/else statements, 166–168
  - if statement, 154–162
  - logical operators, 182–189
  - menus, 190–193
  - nested if statements, 169–172
  - relational operators, 149–153
  - relationship, value of, 150–152
  - semicolons, 158
  - switch statement, 202–211
  - truth, 152–154
  - validating user input, 193–194
- decode, 5
- decrement operator (--), 227–232
  - in mathematical expressions, 231
  - postfix and prefix modes, 229–231
  - in relational expressions, 231–232
- default arguments, 345–348, 756–757
- default constructors, 750, 757
- default copy constructor, 830
- default statement, 203–204
- #define directive, 739
- delete operator, 523
- Demetris Leadership Center case study, 463–470, 477–485
- depth of recursion, 1122
- deque (STL type), 1006, 1112–1114
  - front member function, 1113
  - pop\_front member function, 1113
  - push\_back member function, 1113
- dequeue operation, 1094–1095
- dereferencing, of pointers, 501
- derived class, 892
- descending order, sorting in, 470
- designing programs, 18–22
- desk-checking, 21
- destructors, 758–760, 906
  - base and derived classes, in, 904–905
  - virtual, 929–945
- digital versatile disc (DVD), 6
- direct access file, 267
- direct recursion, 1132
- disk drive, 6
- division by zero, 167, 971–972
- dollarFormat function case study, 590–591
- dot operator (.), 604, 623, 724
- double data type, 54
- double literals, 56
- double precision, 54
- doubly linked list, 1055
- do-while loops, 242–246, 262
  - with menus, 244–247
  - as posttest loop, 243

- drivers, 361–363
- dummy parameter, 843
- DVD (digital versatile disc), 6
- dynamic binding, 931
- dynamic memory allocation, 522–526
  - bad\_alloc exception, 988–989
  - objects, 730–733, 750, 761
  - for structures, 625–626
- dynamic queues, 1093, 1105–1112
- dynamic stacks, 1064, 1080–1090

## E

- efficiency
  - binary search, 463
  - linear search, 459–460
- elements (array)
  - accessing, 377–378
  - described, 376
  - processing, 396–404
  - removing, from vectors, 438–439
- empty member function
  - list, 1056
  - string class, 587
  - vector, 440–441, 442, 1012
- encapsulation, 712
- #endif directive, 738–739
- endl, 33
- end member function
  - iterator, 1010
  - list, 1056
  - string class, 587
  - vector, 1012
- end of file, detecting, 279–281
- end-of-file marker, 660
- enqueue operation, 1093–1096
- enum, 632–642
  - anonymous, 635–636
  - with arrays, 636–638
  - assigning, to int variables, 634
  - assigning integers to enum variables, 633–634
  - comparing enumerators, 634–635
  - declaration and definition, 642
  - defining, 633–634
  - math operators with, 636
  - outputting values with, 638–640
  - and scope, 641
  - specifying values, 640–641
  - strongly-typed, 642–643
- enumerated data types, 632–642, *see also* enum
- enumerators, 633–635, 641
- eof member function, file stream, 668
- equal-to operator (==), 150–151, 161–162
- erase member function
  - list, 1056

- string class, 587
- vector, 1012
- errors
  - logical, 21
  - off-by-one, 386–387
  - recovering from, 981–983
  - syntax, 12
- error testing, files, 667–670
- escape sequences, 34–35
  - `\\`, 35
  - `\`, 35
  - `\`, 35
  - `\a`, 35
  - `\b`, 35
  - `\n`, 35
  - `\r`, 35
  - `\t`, 35
  - newline, 34
- exception handler, 972
- exceptions, 971–989
  - `bad_alloc`, 988–989
  - catch block, 972
  - dynamic memory allocation, 523
  - exceptions handler, 972
  - extracting data from, 983–987
  - handling, 972–975
  - memory allocation error, 988–989
  - multiple, handling, 978–983
  - `new` operator, 523
  - not catching, 975
  - object-oriented handling, 975–978
  - recovering from errors, 981–983
  - rethrowing, 988
  - throwing, 972
  - throw key word, 972
  - throw point, 972
  - try block, 972
  - try/catch construct, 972–975
  - unwinding the stack, 987
- executable file, 12
- execute, 5
- exhaustive algorithms, 1148–1151
- `exit()` function, 358–360
- exponents, 93–95
- expressions, 83–135
  - algebraic, 92–94
  - arithmetic, 89–98
  - case study, 132–135
  - characters and string objects, 118–124
  - `cin` object, 83–88
  - conditional, 199–202
  - C-style and prestandard C++ forms, 105–106
  - formatting output, 108–118
  - and hand tracing programs, 130–131

- initialization, 248, 252
- mathematical, 89–98, 231
- and mathematical library functions, 124–130
- multiple and combined assignment, 104–108
- overflow and underflow, 100–101
- relational, 231–232
- type casting, 101–104
- type conversion, 98–99

## F

- factorial algorithm, 1126–1129
- `fail` member function, file stream, 282, 667, 668
- false values, 150, 152–153, 160–161
- `fetch`, 5
- fetch/decode/execute cycle, 5
- Fibonacci numbers, 1134–1135
- field width, 109, 111
- FIFO (first-in first out), 1093
- file access flags, 658
  - `ios::app`, 658
  - `ios::ate`, 658
  - `ios::badbit`, 668
  - `ios::binary`, 658
  - `ios::eofbit`, 668
  - `ios::failbit`, 668
  - `ios::goodbit`, 668
  - `ios::hardfail`, 668
  - `ios::in`, 658
  - `ios::out`, 658
  - `ios::trunc`, 658
- file access methods, 267
- file buffer, 269
- filename extensions, 267
- filenames
  - and file stream objects, 267–268
  - user-specified, 282–283
- file open errors, 281–282
- file operations, 657–709
  - append mode, 658
  - binary files, 680–685
  - described, 658
  - end-of-file marker, 660
  - `fstream` data type, 658, 659, 697
  - member functions for reading and writing, 670–677
  - opening files for input and output, 697–701
  - opening files with definition statements, 662–663
  - opening multiple files, 678–680
  - open modes of `ifstream` and `ofstream`, 661–662
  - random-access files, 689–697
  - reading a character, 670
  - reading a line, 672
  - records with structures, 685–689
  - rewinding, 696–697
  - writing a character, 676–677

- files
  - closing of, 269
  - for data storage, 265–284
  - detecting end of, 279–281
  - file open errors, 281–282
  - input/output program, 268
  - opening, and creating file objects, 268–269
  - processing, with loops, 278–279
  - reading from, 274, 276–277
  - read position, 275–276
  - types of, 266–267
  - user-specified filenames, 282–283
  - writing to, 270
- file stream objects, 267
  - closing, 269
  - creating, 268–269
  - member functions, 670–677
  - passing to functions, 665–667
- filters, 678
- final, 943–945
- find algorithm (STL), 1007, 1016–1017
- finding classes, 788–796
- find member function, string class, 589
- first-in first-out (FIFO), 1093
- fixed manipulator, 114–115, 663
- flags, 181–182
  - integer, 182
- flash memory, 6
- float data type, 54
- floating-point data types, 54–57
  - comparing, 159–160
  - and integer variables, 56–57
- floating-point literals, 55–56
- floating point numbers, 43
- float literals, 56
- floppy disk drive, 6
- flowcharts, 20
- for\_each algorithm (STL), 1008, 1017–1018
- for loops, 247–257, 262
  - counter variable, 251
  - initialization expression, 248, 252
  - omitting expressions of, 254–256
  - as pretest loop, 251
  - test expression, 248
  - update expression, 248, 252, 253–254
  - user-controlled, 252–253
  - while and do-while *vs.*, 250–251
- formal argument, 311
- formal parameters, 311
- formatting output, 108–118, 663–665
  - fixed manipulator, 114–115
  - left manipulators, 116–117
  - right manipulators, 116–117
  - setprecision manipulator, 111–113
  - showpoint manipulator, 115–116
- FORTRAN, 11
- forward declaration, 821
- forward iterators, 1007
- friend class, 823
- friend functions, 819–823
- friend key word, 819
- front member function
  - deque, 1112
  - list, 1056
  - vector, 1012
- fstream header file, 268
- fstream objects, 658
- function arguments
  - file stream objects as, 665–667
  - structures as, 617–619
- function call statements, 302
- function declarations, 309
- function header, 301
- function parameters
  - pointers as, 513–521
  - reference variables as, 348–353
- function prototypes, 309–310
- functions, 28
  - bool value, returning, 332–334
  - calling, 302–309
  - default arguments, 345–348, 756
  - defining, 300–301
  - exit(), 358–360
  - friend, 819–823
  - generic, 990
  - inline, 743–745
  - local and global variables, 334–341
  - main, 28–29, 302, 537–538
  - member, 712
  - menu-driven programs, 318–322
  - modular programming, 299–300
  - overloading, 354–358
  - overriding, 940
  - passing data by value, 316–317
  - pointers, returning, 526–528
  - prototypes, 309–310
  - pure virtual, 945–949
  - recursive, 1121–1125
  - redefining base class, 918–923
  - reference variables as parameters, 348–353
  - return statement, 322–323
  - sending data into, 311–315
  - static local variables, 342–345
  - static member, 816–819
  - string handling, 575–580
  - structures, returning, 620–622
  - stubs and drivers, 361–363



- value-returning, 324–332
- virtual, 931–934, 945–949
- void, 301
- function signature, 355
- function templates, 990–995
  - with multiple types, 994–995
  - overloading with, 995
  - using operators in, 994

## G

- games, 265
- GCD (greatest common divisor), 1133–1144
- General Crates, Inc. case study, 132–135
- generalization, inheritance and, 891–892
- generic data types, 990
- generic functions, 990
- getArea member function, 716
- getCircleData function, 621
- getItem function, 619
- getline member function, 119
  - cin, 557, 572–573
  - file streams, 672–675
- get member function, 120–122
  - file streams, 675–676
- getter functions, 722
- global constants, 338–340
- global variables, 336–338
- good member function, file stream, 668
- greatest common divisor (GCD), 1133–1134

## H

- handler, exception, 972
- hand tracing programs, 130–131
- Hanoi, Towers of, 1141–1144
- hardware, 3–7
  - CPU, 3–5
  - input devices, 7
  - main memory, 5–6
  - output devices, 7
  - secondary storage, 6
- “has a” relationship, 862
- header
  - function, 301
  - loop, 233, 248
- header file, 28
  - cmath, 94, 124
  - cstdlib, 359, 569
  - cstring, 558
  - ctime, 127
  - fstream, 268
  - iomanip, 110, 113
  - iostream, 28, 36–37
  - string, 52, 581

- hexadecimal literals, 47
- hiding data, 712, 723
- hierarchies, class, 923–929
- hierarchy chart, 20
- high-level languages, 10
- Hoare, C.A.R., 1144
- Home Software Company case study, 590–591

## I

- identifiers, 41–42
  - capitalization, 42
  - legal, 42
- if/else if statements, 176–178
  - nested decision structures *vs.*, 180–181
  - trailing else, 179
- if/else statements, 166–168
- #ifndef directive, 738–739
- if statement
  - conditionally executed code, 156, 162–164
  - expanding, 162–165
  - floating-point comparisons, 159–160
  - nested, 169–172
  - programming style, 159
  - semicolon in, 158
- ifstream objects, 268, 661
  - close member function, 269
  - open member function, 269
  - >> with, 274
- ignore member function, cin, 123, 572
- image editors, 265
- implementation file, class, 737
- implicit sizing, of arrays, 391–392
- #include directive, 28, 36–37, 717, 738, 742
- include file directory, 741
- include guard, 738–739
- increment operator (++), 227–232
  - in mathematical expressions, 231
  - postfix and prefix modes, 229–231
  - in relational expressions, 231–232
- indirection operator (\*), 501
- indirect recursion, 1132
- infinite loops, 236
- inheritance, 891–899
  - base class, 892
  - class hierarchies, 923–929
  - and class templates, 1000–1004
  - constructors and destructors, 906–911
  - derived class, 892–893
  - “is a” relationship, 892–899
  - multiple, 952–959
  - redefining functions, 918–923
- initialization, 59–60
  - array, 387–392, 421–422, 613
  - for loops, 248, 252

- initialization (*continued*)
  - pointers, 510–511
  - structure, 608–610
  - structure array, 613
- initialization list, 388
- inline expansion, 745
- inline member functions, 743–745
- inorder traversal, binary trees, 1163
- input, 17–18
  - array contents, 378–382
  - with `cin`, 83–87
  - reading, into string objects, 581
- input devices, 7
- input file, 265, 268
- input iterators, 1007
- input–output stream library, 36
- input validation
  - and decision making, 193–194
  - and while loops, 239–241
- insert member function
  - list, 1056
  - string class, 590
  - vector, 1012
- instances
  - of class, 715
  - of classes, 723–733
  - of structures, 604
  - variables, 811–812
- instantiation, 723
- int, 43, 44, 634
- integer data types, 42–47
- integer division, 62, 64, 99
- integer flags, 182
- IntegerList class, 782–785
- integer literals, 46
- integer variables, 56–57
- integrated development environments (IDE), 12
- iomanip header file, 110, 113
- ios::app access flag, 658
- ios::ate access flag, 658
- ios::badbit status flag, 668
- ios::binary access flag, 658, 681
- ios::eofbit status flag, 668
- ios::failbit status flag, 668
- ios::goodbit status flag, 668
- ios::hardfail status flag, 668
- ios::in access flag, 658
- ios::out access flag, 658
- iostream header file, 28, 36–37
- ios::trunc access flag, 658
- isalnum library function, 548
- isalpha library function, 548
- “is a” relationship, 892–899, 929
- isdigit library function, 548

- islower library function, 548
- isprint library function, 548
- ispunct library function, 548
- isspace library function, 548
- isupper library function, 548
- iteration, 234
- iterators, 1006
  - begin member function, 1011
  - end member function, 1011
  - [] operator, 1009
- itoa library function, 574

## J

- Java, 11
- JavaScript, 11

## K

- keyboard buffer, 87
- key words, 14

## L

- language elements, 14
- last-in-first-out (LIFO), 1063
- left manipulator, 116–117
- legacy code, 556
- legal identifiers, 42
- length, of C-strings, 558–559
- length member function, string class, 587, 590
- library functions, 93
  - atof, 569, 570
  - atoi, 569, 570
  - atol, 569, 570
  - for C-strings, 558–568
  - isalnum, 548
  - isalpha, 548
  - isdigit, 548
  - islower, 548
  - isprint, 548
  - ispunct, 548
  - isspace, 548
  - isupper, 548
  - itoa, 574
  - strcat, 559–560, 567
  - strcmp, 564–566
  - strcpy, 560–561, 568
  - strlen, 558–559, 567
  - strncat, 561
  - strncpy, 561
  - strstr, 562–564, 568
  - tolower, 551
  - toupper, 551
- lifetime, of variables, 335–336
- LIFO (last-in, first-out), 1063

- linear search
    - algorithm for, 457–459
    - efficiency, 459–460
  - lines, 16
  - LinkedList class template, 1049
  - linked lists
    - appending a node, 1028–1033
    - arrays and vectors *vs.*, 1025–1057
    - circularly, 1055
    - circularly linked, 1055
    - class as node type, 1049–1055
    - composition of, 1026
    - counting nodes, 1136–1137
    - declarations, 1026–1027
    - deleting a node, 1039–1042
    - described, 1025
    - destroying, 1041–1042
    - displaying nodes in reverse, 1137–1139
    - doubly linked, 1055
    - inserting a node, 1035–1039
    - list head, 1026
    - NULL address, 1026
    - operations, 1027–1043
    - recursion with, 1135–1139
    - self-referential data structure, 1027
    - singly linked, 1055
    - template for, 1043–1055
    - traversing, 1033–1035
    - variations of, 1055
  - linker, 12
  - list (STL type), 1006, 1056–1057
    - back member function, 1056
    - empty member function, 1056
    - end member function, 1056
    - erase member function, 1056
    - front member function, 1056
    - insert member function, 1056
    - merge member function, 1056
    - pop\_back member function, 1056
    - pop\_front member function, 1056
    - push\_back member function, 1056
    - push\_front member function, 1056
    - reverse member function, 1057
    - size member function, 1057
    - swap member function, 1057
    - unique member function, 1057
  - list head, linked list, 1026
  - literals, 39–40
    - character, 48, 50–52
    - double, 56
    - float, 56
    - floating-point, 55–56
    - hexadecimal, 47
    - integer, 46
    - long integer, 46
    - octal, 47
    - string, 40, 50–52, 555–556
  - local scope, 212
  - local variables, 334–336
    - initializing, with parameter values, 336
    - lifetime of, 335–336
    - with same name as global, 341
    - static, 342–345
  - logical errors, 21
  - logical operators, 182–189
    - && (AND), 182–184
    - ! (NOT), 187–188, 566
    - || (OR), 184–187
    - associativity, 188–189
    - and numeric ranges, 189
    - precedence, 188–189
    - short-circuit evaluation, 183, 185
  - long data types, 43, 44
  - long double data types, 43, 44
  - long double precision, 54
  - long integer literals, 46
  - long long integer literal, 46
  - loop header, 233, 248
  - loops, 227–288
    - breaking and continuing, 284–288
    - conditional, 247
    - control variable, 235
    - count-controlled, 247, 255–256
    - counters, 241–242
    - described, 232
    - do-while, 242–246, 262
    - and files, 265–284
    - for, 247–257, 262
    - and increment/decrement operators, 227–232
    - infinite, 236
    - input validation with, 239–241
    - nested, 262–264
    - posttest, 242–244
    - pretest, 235, 251
    - processing files with, 278–279
    - programming style, 237–238
    - running total, 257–259
    - selecting, 261–262
    - sentinels, 260–261
    - user-controlled, 244, 252–253
    - while, 232–238, 261
  - lowercase conversion, character, 551–553
  - low-level languages, 10
- M**
- machine language, 9
  - main function, 28–29, 537–538

- main memory, 5–6
- manipulators
  - fixed, 114–115
  - left, 116–117
  - right, 116–117
  - setprecision, 111–113
  - showpoint, 115–116
  - stream, 33, 117
- mantissa, 54
- map (STL type), 1007
- mathematical expressions, 89–98, 231
  - algebraic, to programming statements, 92–93
  - associativity, 91–92
  - exponents, 93–95
  - grouping with parentheses, 92
  - operator precedence, 90–91
- mathematical library functions, 124–130
  - random numbers, 126–129
- mathematical operators, with enum, 636
- max\_element algorithm (STL), 1008, 1016
- member access specification
  - defined, 904
  - inherited, 904
- member functions, 712, 927
  - binding, 931
  - dynamic binding, 931
  - getLength, 736
  - getObjectCount, 814
  - getTaxRate, 754
  - other overloaded, 765
  - overriding, 940
  - private, 765–767, 771
  - public, 771–772
  - redefining, 918–923, 940
  - setLength, 736, 737
  - static, 816–818
  - static binding, 931
  - static stack class, 1065
  - this pointer, 835–837
  - virtual, 931–934
  - withdraw, 773
- members, of structures, 601, 604–607, 626–627
- member variable
  - static stack class, 1065
- memberwise assignment, 824–825
- memory
  - flash, 6
  - main, 5–6
  - random-access, 5
- memory address, 6, 398
- memory allocation, dynamic, *see* dynamic memory allocation
- memory leak, 524
  - avoiding, 533–535

- memory requirements of arrays, 376–377
- menu-driven programs, 190, 318–322
- menus, 190–193, 207–209
- merge member function, list, 1056
- message, 23
- methods, 711
- microprocessors, 4
- min\_element algorithm (STL), 1008, 1013–1016
- modular program, 318
- modular programming, 299–300
- multi-line comments, 70–71
- multimap (STL type), 1007
- multiple assignment, 104
- multiple inheritance, 952–959
- multiset (STL type), 1007
- mutators, 722

## N

- named constants, 71–73
- names
  - of functions, 301
  - of variables, 42, 213–214, 341
- nameSlice functions, 576
- namespaces, 28
- National Commerce Bank case study, 427–429
- negation operator (–), 62, 91, 92
- nested if statements, 169–172
- nested loops, 262–264
  - break statement in, 286
- nested structures, 613–616
- newline escape sequence, 34
- new operator, 522–524
- nodes, 1025
  - appending, 1028–1033
  - of binary trees, 1155, 1160–1162, 1166–1175
  - class as node type, 1049–1055
  - counting, 1136–1137
  - deleting, 1039–1042, 1166–1175
  - displaying, in reverse, 1137–1139
  - inserting, 1035–1039, 1160–1162
  - of linked lists, 1025, 1028–1039, 1049–1055, 1136–1139
- NOT (!) operator, 566
- null character, 50, 554
- null pointer, 500
- nullptr, 500
- null statement, 158
- null terminator, 50, 554–555
  - userName[count], 577, 578
- numbers, 17
- numbers, random, 126–129
- numeric data
  - checking ranges of, 189

- integer data types for, 43
  - from text files, 276–277
- O**
- object aggregation, 860–864
  - object code, 12
  - object conversion, 858–860
  - object file, 12
  - object-oriented design
    - aggregation, 860–864
    - class collaborations, 865–867
    - classes, finding, 788–797
    - CRC cards, 868–869
    - generalization and specialization, 891–892
    - inheritance, 891–900
    - problem domain, 789
    - responsibilities, identifying, 794–796
    - UML, 785–788, 864
  - object-oriented programming (OOP), 22, 23, 711–717
    - abstract array data type, 782–785
    - game simulation, 869–875
    - problem solving, 771–778
    - Unified Modeling Language, 785–788
  - object reusability, 714
  - objects, 711, 714–716, class *vs.*
    - array of, 767–768
    - attributes, 712
    - data hiding, 712–713
    - dynamically allocated, 731–733, 750
    - encapsulation, 712
    - methods, 713
    - pointers, 730–733
    - state, 727
  - octal literals, 47
  - off-by-one error, 386–387
  - off position, 6
  - ofstream objects, 268, 661–662
    - close member function, 269
    - open member function, 269
    - <<, used with, 270
  - one-dimensional arrays, 419
  - on position, 6
  - OOP, 22, 23, 711–717
  - open member function, file stream objects, 269
  - operands, 59, 98
  - operating systems, 7
  - operator functions, 831
  - operator overloading, 831–857
    - [ ] operator, 586, 852–855
    - >> and << operators, 848–852
    - = operator, 831–835
    - general issues, 837–838
    - math operators, 838–843
    - postfix ++ operator, 843–844
    - prefix ++ operator, 843
    - relational operators, 846–848
  - operators, 14, 15
    - (-) negation, 62
    - (subtraction), 62
    - , 227–231
    - ==, 105–107
    - > (object pointer), 731
    - > (structure pointer), 623–624, 625–627
    - !, 182, 187–188, 566
    - !=, 150–152
    - % (modulus), 62, 91
    - %=, 105–107
    - &(address), 495–497
    - &&, 182–184, 383
    - \* (indirection), 501
    - \* (multiplication), 62, 91–92
    - \* (pointer variable declaration), 626–627
    - \*=, 105–107
    - . (dot operator), 604, 630, 724
    - / (division), 62, 91
    - /=, 105–107
    - ?:, 199
    - || (OR), 182, 184–187
    - +, 62, 91, 586
    - ++, 227–231
    - +=, 105–107, 586
    - <, 150–152
    - <<, 31–32, 270
    - <=, 150–152
    - =, 586
    - ==, 150–152, 161–162, 237
    - >, 150–152
    - >=, 150–152
    - >>, 84, 586
    - [] operator, 429, 431–433, 586
    - associativity, 91–92, 188–189, 1213
    - binary, 61, 62
    - overloading, 831–857
    - precedence of, 90–91, 188–189, 1213
    - relational, *see* relational operators
    - scope resolution (: :), 722
    - string class, 586
    - ternary, 61
    - unary, 61
  - OR
    - with enum, 638–640
    - formatting, 663–665
    - || logical operator, 184–187
  - output, 18
  - output devices, 7
  - output file, 265, 268

- output iterators, 1007
- overflow, 100–101
- overhead, 1126
- overloading functions, 354–358
  - constructors, 762–764
  - member functions, 758
  - templates, 995
- override, 943–945
- overriding, 940
- P**
- parallel arrays, 404–407
- parameters
  - array, 411
  - pointers as, 513–521
  - reference variables as, 348–353
- parentheses, 92
- partially filled arrays, 401–403
- Pascal, 11
- passing by value, 316–317
- passing to functions
  - with pointers, 578–580
- percentage discounts, 66–68
- pointers, 495–540
  - address operator (&), 495–497
  - arithmetic with, 508–509
  - and arrays, 504–508
  - base class, 937–939
  - comparing, 511–513
  - constant, 520–521
  - constant, to constants, 521
  - to constants, 517–519
  - creating and using, 499–504
  - dynamic memory allocation, 522–526
  - as function parameters, 513–521
  - initializing, 510–511
  - to objects, 730–733
  - passing C-string arguments with, 578–580
  - returning, from a function, 526–528
  - smart, 533–535
  - structure pointer operator, 623–624, 626, 627
  - to structures, 623–625
  - structures containing, 689
  - United Cause case study, 536–540
- pointer variables, *see* pointers
- polymorphism, 929–945
  - abstract base classes, 945–949
  - base class pointers, 937–939
  - dynamic binding, 931
  - overriding, 940
  - pure virtual function, 945–949
  - and references or pointers, 935–937
  - references/pointers, 935–937
  - static binding, 931
  - virtual destructors, 929–945
  - virtual functions, 931–934, 945–949
- pop\_back member function
  - list, 1056
  - vector, 442, 1009
- pop\_front member function
  - deque, 1112
  - list, 1056
- pop operation (stacks), 1064
- postfix mode, 229–231
- postorder traversal, binary trees, 1163–1164
- posttest loop, 242–244
- pow function, 93–95, 126
- precedence, operator, 90–91, 188–189, 1213
- prefix, template, 990, 995
- prefix modes, 229–231
- preorder traversal, binary trees, 1163
- preprocessor, 11
- preprocessor directive, 28, 29
- prestandard C++
  - standard *vs.*, 73–75
  - type cast expressions, 105–106
- pretest loop, 235
- priming read, 240
- primitive data types, 600
- private member functions, 765–767
- private members, class, 720–721, 736–737
- problem domain, 789
- procedural programming, 22–23, 711–714
- processing, 19
- processing array contents, 396–404
- programmability, of computers, 1–2
- programmer-defined data types, 429
- programmer-defined identifiers, 14, 15
- programmers, 2, 9
- programming, 1–23
  - computer systems
    - input, processing, and output, 17–18
    - procedural and object-oriented, 22–23
    - process of, 18–22
  - programability of computers, 1–2
  - program elements, 14–17
  - programs and programming languages, 8–13
  - programming languages, 8–11, *see also* specific languages
    - high-level, 10
    - low-level, 10
- programming process, 18–22
- programming style, 73–75
  - and if statements, 159
  - and nested decision structures, 172–173
  - and while loops, 237–238
- programs
  - defined, 1

- described, 8–9
- designing/creating, 18–22
- elements, 14–17
- primary activities of, 17–18
- protected members, 900–905
- prototypes, function, 309–310
- pseudocode, 20–21, 63, 460
- public member functions, 720–721
- public members, class, 720–721
- punctuation, 14, 15–16
- pure virtual function, 945–949
- push\_back member function
  - deque, 1112
  - list, 1056
  - vector, 435–436, 442, 1009
- push\_front member function, list, 1056
- push operation (stacks), 1064
- put member function, file streams, 676–678
- Python, 11

## Q

- queue (STL type), 1114–1115
- queue container adapter, 1114–1115
- queues, 1093–1115
  - applications of, 1093
  - array-based, 1096
  - crawling problem with array, 1095
  - dequeuing, 1093
  - described, 1093
  - dynamic, 1093, 1105–1108, 1105–1112
  - dynamic template, 1109–1112
  - empty, detecting, 1096
  - enqueueing, 1093–1096
  - first-in, first-out, 1093
  - full, detecting, 1096
  - linked list-based, 1105–1108
  - operations, 1093–1096
  - static, 1093, 1096–1100
  - static template, 1101–1104
  - STL queue and dequeue containers, 1112–1115
- QuickSort algorithm, 1144–1148

## R

- RAM, 5
- random-access files, 267, 689–697
- random-access iterators, 1007
- random-access memory (RAM), 5
- random file access, 689
- random numbers, 126–129
  - limiting range of, 128
  - seeding, 127
  - time function with, 127
- random\_shuffle algorithm (STL), 1008, 1013–1014

- range-based for loop, 392–396
  - modifying array with, 394–396
  - versus* regular for loop, 396
- range variable, 392
- reading data, from file, 274, 276–277
- read member function, file stream objects, 681–683
- read position, 275–276
- records, 685–689
- recursion, 1121–1151
  - base case, 1126
  - binary search, 1139–1141
  - counting characters, 1129–1132
  - depth of, 1122
  - direct, 1132
  - exhaustive algorithms, 1148–1151
  - factorial algorithm, 1126–1129
  - Fibonacci numbers, 1134–1135
  - greatest common divisor (GCD), 1133–1134
  - indirect, 1132
  - infinite, 1121
  - iteration *vs.*, 1151
  - linked list operations, 1135–1139
  - problem solving with, 1125–1132
  - QuickSort algorithm, 1144–1148
  - recursive functions, 1121–1125
  - recursively defined problems, 1134–1135
  - Towers of Hanoi, 1141–1144
- recursive case, 1126
- redefining base class functions, 918–923, 930
- reference parameters, constant, 639
- reference variables
  - as parameters, 348–353
  - pointers *vs.*, 513–514
- reinterpret\_cast, 683, 685
- relational expressions, 150, 231–232
- relational operators, 149–153
  - and characters, 195–196
  - and pointers, 511
  - and string class, 196–198
  - truth, 152–154
  - value of relationship, 150–152
- relationships
  - goAgain variable, 553
  - “has a,” 862
  - “is a,” 892–899, 929
  - whole-part, 862
- replace member function, string class, 590
- reserved words, 15
- resize member function, vector 1013
  - string class, 588
  - vector, 443
- responsibilities, identifying, 794–797
- rethrowing an exception, 988



- returning
  - bool value from functions, 332–334
  - pointers from functions, 526–531
  - structures from functions, 620–622
  - values from functions, 324–332
- return statement, 322–323
- reusability, object, 714
- reverse member function
  - list, 1057
  - vector, 442, 1013
- rewinding a file, 696–697
- right manipulators, 116–117
- Ruby, 11
- running, of programs, 5
- running total (for loops), 257–259
- run-time library, 12
- rvalue, 59
- S**
- scope, 61
- scope resolution operator (::), 722, 757
- search algorithm
  - binary search, 460–463, 1139–1141
  - Demetris Leadership Center case study, 463–470
  - linear search, 457–460
  - for STL vector, 485–490
- search trees, binary, 1157
- secondary storage, 6
- seekg member function, file stream objects, 689–694
- seekp member function, file stream objects, 689–694
- selection sort, 474–477
- self-referential data structure, 1027
- semicolons, 158
- sentinels, 260–261
- sequence containers, 429, 1006
- sequence structure, 154
- sequential file access, 267, 689
  - seekg member function, 696–697
- sequential search, 457
- set (STL type), 1007
- setprecision manipulator, 111–113, 663
- setter functions, 722
- setw manipulator, 664–665
- shared\_ptr, 533
- short, 43, 44
- short-circuit evaluation, 183, 185
- showItem function, 619
- showpoint manipulator, 115–116
- single-line comments, 70
- single precision, 54
- singly linked list, 1055
- size declarators, of arrays, 376
- size member function
  - list, 1057
  - string class, 590
  - vector, 437–438, 1009
- sizeof operator, 58
- smart pointers, 533–535
- software
  - application, 7–8
  - system, 7
- software developers, 1
- software development tools, 7
- software engineering, 22, 522–528, 996, 1151
- sort algorithm (STL), 1008, 1013–1014
- sorting, of strings, 566–568, 582–584
- sorting algorithm
  - bubble sort, 470–474
  - Demetris Leadership Center case study, 477–485
  - QuickSort, 1144–1147
  - selection sort, 474–477
  - for STL vector, 485–490
- source code, 11
- source files, 11
- specialization, inheritance and, 891–892
- specialized templates, 1004–1005
- specification file, class, 737
- spreadsheets, 265, 657
- stack (STL type), 1091–1092
- stacks, 1063–1115
  - applications of, 1064
  - array-based, 1064
  - cafeteria plates, 1063–1064
  - described, 1063–1064
  - dynamic, 1064, 1080–1090
  - implementing, 1071–1073
  - isEmpty function, 1083
  - isEmpty operation, 1065
  - isFull operation, 1065
  - LIFO, 1063
  - linked list-based, 1080–1085
  - for math, 1071–1073
  - operations, 1064–1065, 1071–1073
  - pop, 1064–1065
  - push, 1064
  - static, 1064–1065, 1074–1080
  - STL stack container, 1091–1092
  - unwinding, 987
- stale data, 730
- Standard Template Library (STL), 429, 1005–1018
  - abstract data types, 1006–1007
  - algorithms, 1007–1008, 1013–1018
  - associative containers, 429, 1006, 1007
  - binary\_search algorithm, 1007, 1012–1013
  - count algorithm, 1007, 1012–1013
  - deque, 1006, 1112–1114
  - find algorithm, 1007, 1016–1017
  - for\_each algorithm, 1008, 1017–1018



- iterators, 1006, 1010–1011
- list, 1006, 1056–1057
- map, 1007
- max\_element algorithm, 1008, 1016
- min\_element algorithm, 1007, 1016
- multimap, 1007
- multiset, 1007
- queue, 1114–1115
- random\_shuffle algorithm, 1008, 1013–1014
- sequence containers, 1006
- set, 1007
- sort algorithm, 1008, 1013–1014
- stack, 1091–1092
- storing objects, 1091–1092
- vector, 429–443, 485–490, 1006, 1009–1010, 1012–1013
- state, object, 727
- statements, 16
- static binding, 931
- static key word, 812
- static local variables, 342–345
- static member functions, 816–819
- static member variables, 812–816
- static queues, 1093, 1096–1100
  - template, 1101–1104
- static stacks, 1064–1051, 1074–1080
- STL, *see* Standard Template Library
- storage, secondary, 6
- strcat library function, 559–560, 567
- strcmp library function, 564–566
- strcpy library function, 560, 568
- stream extraction operator, 84
- stream insertion operator, 31–32, 270, 586
- stream manipulator, 33, 117
- stream object, 31
- string class, 581–589, 717
  - append member function, 588
  - assign member function, 588
  - begin member function, 589
  - capacity member function, 589
  - clear member function, 589
  - compare member function, 589
  - comparing and sorting, 582–584
  - constructors, 762
  - copy member function, 589
  - defining string objects, 581–582, 585–586
  - described, 52–53
  - empty member function, 589
  - end member function, 589
  - erase member function, 589
  - find member function, 589
  - Home Software Company case study, 590–591
  - input, reading into a string object, 582
  - insert member function, 590
  - length member function, 587, 589
  - at member function, 588
  - member functions, 587–590
  - operators, 586
    - and relational operators, 196–198
  - replace member function, 589
  - resize member function, 589
  - size member function, 589
  - substr member function, 589
  - swap member function, 589
  - using, 52–53
- string constant, 29
- stringCopy function, 576
- string header file, 52, 581
- string literals, 29, 40, 50–52
- string literals, 555–556
- string objects
  - and characters, 118–124
  - comparing, 196–198
  - defining, 581–582, 585–586
  - functions for handling, 575–580
  - member functions and operators, 124
  - numeric conversion functions, 569–575
  - reading input into, 582
  - sorting, 566–569
  - strings, *see also* C-strings
- strlen library function, 558–559, 567
- strncat library function, 568
- strncpy library function, 568
- strongly typed enum, 642–643
- strstr library function, 562–564, 568
- struct, 601, *see also* structures
- structure pointer operator (-?), 623–626
- structures, 599–643
  - and abstract data types, 599–601
  - accessing structure members, 604–607
  - arrays of, 611–613
  - combining data into, 601–604
  - containing pointers, 689
  - and enumerated data types, 632–643
  - as function arguments, 617–620
  - initializing, 608–610
  - nested, 613–616
  - pointers as members of, 626–627
  - pointers to, 623–626
  - records, creating with, 685–689
  - returning, from a function, 620–622
  - self-referential, 1027
  - and unions, 628–632
- structure variables, 603–604, 607
- stubs, 361–363
- subscript, of array element, 377
- substr member function, string class, 589

- swap member function, 443
  - list, 1057
  - string class, 590
- switch statement, 202–211
  - break, 204, 205
  - case, 203–204
  - default, 203–204
  - fallthrough capability, 205–207
  - with menus, 207–209
- syntax, 14
- syntax errors, 12
- system software
  - operating system, 7
  - software development tools, 7
  - utility program, 7

## T

- tags, of structures, 602, 604
- tellg member function, file stream objects, 694–696
- tellp member function, file stream objects, 694–696
- template function, 990
- template prefix, 990, 995, 997
- templates
  - binary trees, 1175–1181
  - class, 996–1005
  - defining, 995, 996
  - dynamic queue, 1109–1112
  - dynamic stack, 1085–1090
  - function, 990–995
  - linked list, 1043–1055
  - prefixes, 990, 995
  - specialized, 1004–1005
  - static queue, 1096–1100
  - static queues, 1101–1104
  - static stack, 1074–1079
  - type parameter, 990
- ternary operators, 62, 199
- text editor, 11
- text files
  - described, 266
  - numeric data from, 276–277
- this pointer, 835–837
- throwing an exception, 523, 972
- throw key word, 972, 987
- throw point, 972
- time library function, 127
- tolower library function, 551
- top-down design, 20
- top member function, stack, 1191
- to\_string function, 571
- toupper library function, 551
- Towers of Hanoi, 1141–1144
- trailing else, 179
- traversing
  - binary tree, 1162–1165
  - linked list, 1033–1035
- true values, 150, 152–153, 160–161
- truth, 152–154, 160–161
- try block, 972
- try/catch construct, 972–974
- try key word, 972
- two-dimensional arrays, 418–425
  - initializing, 421–422
  - passing, to functions, 422–423
  - summing columns, 425
  - summing elements of, 424
  - summing rows, 424–425
- type cast expression, 101
- type casting, 101–104
- type coercion, 98
- type conversion, 98–99
- type parameters, 990–992, 994

## U

- UML, *see* Unified Modeling Language
- unary operator, 61, 62
- underflow, 100–101
- Unified Modeling Language (UML), 785–788, 864
  - access specification, showing, 786–787
  - aggregation, showing, 864
  - class diagram, 785–788
  - constructors, showing, 788
  - data type notation, 782
  - destructors, showing, 788
  - parameter notation, 787
- unions, 628–632, structures *vs.*
  - anonymous, 630–632
- unique member function, list, 1057
- unique\_ptr, 533–535
- United Cause case study, 536–540
- unsigned int, 43, 44
- unsigned long, 43, 44
- unsigned short, 43, 44
- unwinding the stack, 987
- update expression (for loop), 248, 252, 253–254
  - multiple statements in, 253–254
- uppercase conversion, character, 551–553
- USB drives, 6
- user-controlled loops, 244, 252–253
- user-specified filenames, 282–283
- utility programs, 7

## V

- value, passing by, 638
- value-returning functions, 324–332
  - calling, 326–332
  - defining, 324–325

- variable declaration, 17
- variable definitions, 17, 37
- variables, 16–17
  - control, 235
  - flag, 181–182
  - global, 336–338
  - instance, 811–818
  - integer, 43, 182
  - local, 334–336, 341, 342–345
  - loop control, 235
  - names, 42
  - overflow and underflow, 100–101
  - with same name, 213–214, 341
  - static member, 812–818
  - structure, 603–604, 607
- vector, 1006, 1009–1011
  - back member function, 1009
  - begin member function, 1010
  - capacity member function, 442, 1012
  - clear member function, 439, 442, 1012
  - compared to linked list, 1025
  - defining, 430–431
  - empty member function, 440–441, 442, 1012
  - end member function, 1012
  - erase member function, 1012
  - front member function, 1012
  - initialization list, with C++ 11, 431
  - insert member function, 1013
  - at member function, 442, 1009
  - [ ] operator, 1009
  - pop\_back member function, 442, 1009
  - push\_back member function, 435–436, 442, 1009
  - range-based for loop, 433–435
  - removing elements from, 438–439

- resize member function, 443, 1013
- reverse member function, 442, 1013
- searching and sorting, 485–490
- size member function, 437–438, 1009
- STL, 429–430
- storing and retrieving values in, 431–433
- swap member function, 443
- virtual destructors, 929–945, 940–945
- virtual functions
  - and polymorphism, 935–937
  - pure, 945–949
- virtual key word
- virtual member functions, 929–945
- Visual Basic, 11
- void function, 301
- volatile memory, RAM as, 5

## W

- weak\_ptr, 533
- Web browsers, 265
- while loops, 232–238, 261
  - input validation with, 239–241
  - logic of, 233–235
  - as pretest loop, 235
  - programming style, 237–238
- whitespace characters, 118
- whole-part relationship, 862
- word processors, 265, 657
- write member function, file stream objects, 681

## Z

- zero(es)
  - division by, 167, 971–972
  - trailing, 115, 116

