Terms beginning with numbers, such as 2-3-4 trees, are indexed as if spelled out.

? : operator, 20t
*, See dereferencing operator
*, to represent indefinite number of multiplicities, 775
! operator, 12
!= operator, 20
# operator, 9
% operator, 20t
& operator, 24, 25
&& operator, 12
* operator, 19, 20
+ operator, 19, 20t, 31
++ operator, See increment operator
- operator, 19, 20t, 31
-- operator, See decrement operator
-> operator, 20
/ operator, 19, 20t
< operator, 20t
<< operator, 20t
<= operator, 20t
= operator, 20t
>= operator, 20t
operator, 20t
array subscript operator, 20
associative container concept, 292
ASCII (American Standard Code for Information Interchange), 17–18, 18t, 755
assertions, 166
assert macro, 168–169
assignment, 204
assignment operator, 20t, 250–251
failure to define, 252
association, UML notation, 774–775
associative array, 523
associative container, 292
associative containers, 292, 511, 774

A
abstract classes, 202–204, 771
abstract data types, 99–100
interface notation to represent, 99, 771
preconditions and postconditions, 100–101
abstract functions, 202–204
abstraction, 73–75
acceptance testing, 68t, 149
accessor member functions, 87
inline definition, 88
access specifiers, 77
accumulate function, 301t
activation bars, in UML sequence diagrams, 778
activation frames, 409–411
actual classes, 204
adapter classes
queue class as, 366
stack class as, 326
addresses, displaying for different countries case study, 206–209
address function, allocator class, 758t
address-of operator, 24
array arguments, 36
array data fields, 94
array indexing, 34
array index out of bounds error, 132t, 132–135
array-pointer equivalence, 34–35
Array Queue.tc, 370, 372, 374–375
arrays, 33–37
crude nature of, 231
disadvantages relative vectors, 233
dynamically allocated, 34–35
falling off either end, 613
multidimensional, 36–37
out-of-bound subscripts, 33
performance of sorted compared to hash tables and binary search trees, 540
recursive algorithm to search, 405
returning failure before examining all elements, 119
security holes associated with lack of range checking, 237
selection sort using, 576
for storage in queue, 370–374
telephone directory case study, 108–113
array subscript operator, 20t
artificial intelligence (AI), 403
The Art of Computer Programming, Vol 3: Sorting and Searching (Knuth), 496, 539
COPYRIGHTED MATERIAL
at function
decide class, 377t
sequences, 295t
string class, 39t
subscripting operator compared, 237
vector class, 236t, 236–237
attributes, 4, 75
UML syntax, 771–772
attribute types, UML syntax, 772–773
automobile recalls, logic errors responsible for many, 137
auto_ptr class, 765
AVLNode class, 636
AVLNode.h, 636
AVL_Tree class, 634–636
AVL_Tree.h, 635–636
AVL trees, 628–643
back edges, 722
back function
decide class, 377t
list class, 265t
sequences, 295t
backtracking, 435
finding path through maze application case study, 436–439
bad_alloc exception, 142t
bad_cast exception, 142t
bad function, stream class, 48, 48t
bad_typeid exception, 142t
balanced binary search trees, 530, 624
balanced parentheses, testing expressions for, 320–324
base case, 406
base classes, 186, 188–190
failure to qualify template base class name, 472
protected visibility for data fields, 192
virtual, 759–764
basic_ios class, 187
begin function
containers, 293t
decide class, 377t
Graph class, 704
hash_map class, 543t
list class, 265t, 266
List_Graph class, 707t, 710–711
set class, 513, 514t
vector class, 296
big-O notation, 172–178
binary openmode flag, 55t
binary search, 446, 450–451
recursive, 422–426
binary_search function, 424–425
BTree class, 675
Binary_Search_Tree class, 468–479
Binary_Search_Tree.h, 469–470
binary search trees, 446, 450–451, 466–472
AVL trees, 628–643
balanced, 530, 624
insertion into, 472–474
performance compared to hash tables and sorted arrays, 540
Red-Black trees, 643–655, 671
removal from, 474–479
rotation, 624–628
self-balancing, 623
testing, 479–484
traversal, 455–456
writing index for term paper case study, 479–484
Binary_Tree class, 457, 458–465
binary trees, 446, 448–452
fullness and completeness, 451–452, 623
bidirectional_iterator, 267
bit and keyword, digraphs, trigraphs, or equivalents for, 736t
bit or keyword, digraphs, trigraphs, or equivalents for, 756t
bitwise and operator, 20t
bitwise operator, 20t
bitwise complement operator, 20t
bitwise exclusive or operator, 20t
black-box testing, 449
blobs, cell counting case study, 431–434
bool data type, 16, 17t, 18
in control statements, 12
to indicate errors, 138
returning unnecessarily, 91
using to indicate success or failure of programs, 135
Boost Graph Library, 697
topology, 71
boundary conditions, testing, 153–155
braces in control statements, 14
omitting around compound statement, 15
branch coverage, 150
breadth-first search, 715–720
Breadth_First_Search.cpp, 719–720
breadth_first_search function, 718–719
break, in switch statements, 13t, 15
breakpoints, debugger, 162
BST_With_Rotate class, 627–628
BST_With_Rotate.h, 627–628
BTreeNode class, 457–458
BTreeNode.h, 458
BTree class, 673–674
B-trees, 681
B-trees, 643, 663, 672–680
bubble sort, 576–580, 587t
BubbleSort.h, 579–580
bucket, 538
bucket hashing, 538
buffer overruns, 135
case
in switch statements, 13t, 15
case studies
building custom Huffman trees, 498–504
converting expressions with parentheses, 347–350
converting infix expressions to postfix expressions, 339–347
counting cells in blob, 431–434
displaying addresses for different countries, 206–209
Dutch National Flag problem, 616–619
evaluating postfix expressions, 333–338
finding palindromes, 315–320
finding path through maze using backtracking, 436–439
geometric shape processing, 220–225
Huffman coding, 561–564
Huffman trees, building custom, 498–504
maintaining a customer queue, 362–365
maintaining an ordered list, 285–291
shortest path through maze, 727–730
simulate strategy for serving airline passengers, 381–397
telephone directory, 102–124
telephone directory with map application, 558–560
telephone directory with vector application, 239
t testing expressions for balanced parentheses, 320–324
topological sort of graph, 731–733
Towers of Hanoi problem, 427–431
writing index for term paper, 479–484
catch blocks, 140–141
and unreachable exception handlers, 146
catching exceptions, See exception catching
C++ compiler, 3
cell counting case study, 431–434
C++ environment, 2–7
cerr object, 44–45
chaining, hash tables, 538, 540–541
char data type, 17t, 17–18
 extraction operator processing, 45t
insertion operator processing, 49t
char_traits class, 759
children, 447
cin object, 44–45
circular arrays
to implement deque, 376, 378
to implement queue, 370–374
circular linked lists, 262
class declaration, 3–4
class definition, 4, 75–79
class diagrams, See UML class diagrams
classes, 3–4
 actual and abstract compared, 204t
as components of other classes, 94
documentation style, 96–97
function, 303–306
for implementing abstract data types, 99, 100
testing, 157–158
virtual base classes, 759–764
class hierarchies, 186–192. See also inheritance
abstract classes, assignment, and casting in, 202–209
displaying addresses for different countries case study, 206–209
processing geometric shapes case study, 220–225
class implementation, 75, 79–85
class keyword, 256
class member access operator, 20t, 31, 198
class names, 21, 78
UML syntax, 770
class types, 16
client programmer, 77
clients, 76
of lists and stacks, 311
closed-box testing, 149
collisions, hash tables, 531
reducing by expanding table size, 536–537
reducing using quadratic probing, 537–538
comments
 assertions, 166
removing by preprocessor, 7
compare function, 494, 522, 528
string class, 40t
compiled program, 3
compiler, 3, 4
 switch, 7, 206
compiling, 3, 6
conditional, 9–10
complement operator, 12, 20t
complete binary tree, 452
completeness, of trees, 452, 623
complexity, using abstraction to manage, 73–75
 comp1 keyword, digraphs, trigraphs, or equivalents for, 756t
compound assignment operators, 20t
compound statements, 12
 omitting braces around, 15
copy constructor, 247
failure to define, 252
copy function, 300t
copying objects, 247–250
count function, shared_ptr class, 767t
count_occurrences function, 31
cout object, 44–45
coverage testing, 149–150
CppUnit test framework, 779–781
create_graph function, Graph class, 704t, 726
critically unbalanced trees, 632–634
cryptographic algorithms, 178
c_str function, string class, 40t
cubic growth rate algorithms, 176, 177
customer queue, 358
 maintaining case study, 362–365
cycles, graphs, 694–695, 694–696
D
data abstraction, 73–74
debuggers, 162–164
delimiters, 42
data function, string class, 40t
data members, 75. See also data fields
deallocate function, allocator class, 758t
debuggers, 162–164
debugging, 130, 160–164
dec I/O manipulator, 50t
decrement operator, 19, 20t
  using in expression with other operators, 22
deep copy, 247–250
default allocator, 757
default constructor, 87
default copy constructor, 247
default destructor, 251
#define directive, 77
delaboration, in Unified Model, of software life cycle, 66, 67
#define directive, 10
  to select stack implementation, 343
  empty function
  containers, 294t
destructor, 251
  declaring virtual, 200
  failure to define, 252
detailed design, 68t
digraph, 693–694
digraphs, 756, 756t
Dijkstra.cpp, 737–738
Dijkstra’s algorithm, 734, 736–738
directed acyclic graphs, 731
directed graphs, 693–694
discovery order, in depth-first search, 722
divide-and-conquer, 569
division by zero error, 131–132, 132t
DNode class, 259, 260
DNode.h, 260
do … while statement, 13t
DOC++ program, 97
documentation comment tags, 97t
documentation style, 96–97
domain_error exception, 142t
dot notation, 32
double data type, 17, 17t
  compatibility with other data types, 22
  conversion of int data type to, 22
  extraction operator processing, 45t
  insertion operator processing, 49t
double-ended queue, 376
double-linked lists, 258–262
  implementation, 271–284
  implementing queues as, 365–367
downcast, 205
Doxygen program, 96–97, 114
driver program, 92, 157
drivers, for testing, 157
Dutch National Flag problem case study, 616–619
dynamically allocated arrays, 34–35
  failure to delete, 120
  dynamically created objects, 27–28
  dynamic casting, 206–207
  and run-time type identification, 206
driver program, 92, 157

dequeue, 365, 376–380
dequeue class, 376–380
  and container class hierarchy, 292
  relation to other container concepts, 774
dereferencing operator (*), 24, 198
  syntax, 26
derived classes, 186, 188–190
  data field initialization, 190–191
  omitting abstract function definition, 203
descendant, 447
design
  in software life cycle, 65, 66, 68t, 70–72
  telephone directory case study, 102–124
destination, directed graphs, 693
destroy function, allocator class, 757, 758t
destroy function, all, 147
  with try and catch blocks, 140–141
exception classes, 141
   creating your own, 338
   hierarchy, 145–146
exception handler, 139
   unreachable, 146
exception handling, 138
   error recovery, 143–145
   with try and catch blocks, 140–141
   exceptions, 131, 138–147
   to enable straightforward code, 146
   standard, 141–143, 142
   ways to indicate errors, 138
   uncaught, 139–140
friend declaration, 89, 217–218
front function, 245, 246
   function, 300
   find_first_of function, 471
   find_first_not_of function, 471
   file streams, 54–57
   Fibonacci numbers, 417–419
   factorial growth rate algorithms, 177, 178
   factorial, recursive algorithm for computing, 412–413
   factorial growth rate algorithms, 177, 178
   expression trees, 446, 449
   traversal, 455–456
   extending classes, 186
   external nodes, 447
   extraction operator, 45, 92
   processing of primitive data types and strings, 45
   Extreme Programming, 156
F
   factorial, recursive algorithm for computing, 412–413
   factorial growth rate algorithms, 177, 178
   factory function, 223
   Fibonacci numbers, 417–419
   FIFO (First-In-First-Out), 357, 489
   file streams, 54–57
   find_first_not_of function, string class, 40t, 42
   find_first_of function, string class, 40t, 41–42
   find function, 300t
   Binary_Search_Tree class, 468t, 469, 471
   hash_map class, 543t
   set class, 513, 514t, 517
   string class, 40t
   template function, 297–299
   find_if function, 300t, 304t
   finish order, in depth-first search, 722
   First-In-First-Out (FIFO), 357, 489
   fixed floating-point format, 52t
   fixed I/O manipulator, 50t
   float data type, 17, 17t
   compatibility with other data types, 22
   extraction operator processing, 45t
   insertion operator processing, 49t
   floating-point formats, 52t, 52–54
   for_each function, 300t
   forest, 724
   for statement, 13
   purpose and syntax, 13t
   4-node, 663–664
   friend declaration, 89, 217–218
   try catch blocks, 140–141
   exceptions, 131, 138–147
   to enable straightforward code, 146
   standard, 141–143, 142t
   ways to indicate errors, 138
   uncaught, 139–140
   global objects, 24
   graph abstract data type, 697–700
   implementing, 701–714
   Graph class, 697–700, 704–705
   Graph.cpp, 705–706
   Graph.h, 698–700
   Graph::iter_impl class, 711
   graphs
   adjacency list, 701–702
   adjacency matrix, 701, 702–703
   algorithms using weighted graphs, 734–742
   applications, 696–697
   dense and sparse contrasted, 713
   directed acyclic, 731
   directed and undirected, 693–694
   minimum spanning trees, 738–742
   paths and cycles, 694–695, 694–696
   terminology, 692–697
   and trees, 694–696, 696
   graph traversal, 697
   breadth-first search, 715–720
   depth-first search, 715, 720–726
   shortest path through maze case study, 727–730
   topological sort of graph case study, 731–733
   greater than operator, 20t
   greater than or equal operator, 20t
   greatest common divisor, recursive algorithm for computing, 414–415
   growth rate, algorithm performance compared, 176, 177
   guillemets, 771
G
   gcc compiler, 6
   general floating-point format, 52t
   generalization, UML notation, 773–774
   general trees, 452–453
   generic classes, UML notation, 776
   generic functions, 9
   geometric shape processing case study, 220–225
   get_data function, Binary_Tree class, 460t
   get_dest function, Edge class, 700t
   get_edge function
   Graph class, 704
   List_Graph class, 707t, 710
   Matrix_Graph class, 713
   get function, shared_ptr class, 767t
   get_left_subtree function, Binary_Tree class, 460t, 462, 465
   getline function, stream class, 46–47, 47t
   get_num_v function, Graph class, 704t
   get_right_subtree function, Binary_Tree class, 460t, 462, 465
   get_source function, Edge class, 700t
   get_weight function, Edge class, 700t
   glass-box testing, 149
   global namespace, 214
   global objects, 24
   graphs
   adjacency list, 701–702
   adjacency matrix, 701, 702–703
   algorithms using weighted graphs, 734–742
   applications, 696–697
   dense and sparse contrasted, 713
   directed acyclic, 731
   directed and undirected, 693–694
   minimum spanning trees, 738–742
   paths and cycles, 694–695, 694–696
   terminology, 692–697
   and trees, 694–696, 696
   graph traversal, 697
   breadth-first search, 715–720
   depth-first search, 715, 720–726
   shortest path through maze case study, 727–730
   topological sort of graph case study, 731–733
   greater than operator, 20t
   greater than or equal operator, 20t
   greatest common divisor, recursive algorithm for computing, 414–415
   growth rate, algorithm performance compared, 176, 177
   guillemets, 771
H
   handling exceptions, See exception handling
   has-a relationship, 187–188
   and composition, 775–776
   hash codes, 531–533
   hash function class (proposed), 533
   hashing, 531
   hash_map class, 542–554
   Hash_Table_Chain.h, 551–552
   Hash_Table_Open.h, 544–545
   hash tables, 512, 530–532
   chaining, 538, 540–541
   expanding size, 536–537
   implementing, 542–554
indexes, using size_t versus int, 547
index for term paper case study, 479–484
infinite recursion, 413
infinity function, numeric_limits
class, 19t
infix expressions, converting to postfix
case study, 339–347
infix notation, 332
inheritance, 186–192. See also class
hierarchies
is-a and has-a relationships con-
trasted, 187–188
member function overloading,
194–196
member function overriding,
193–194, 200
multiple, 210–212
polymorphism, 197–201
and virtual functions, 196–197
initialization, in control statements,
13t
initialization expressions, using in
constructors, 86
inline definition, 88
inline functions, 303
inner classes, UML syntax, 774
in openmode flag, 55t
inorder predecessor, 475
inorder traversal, 454
input streams, 44, 45–49
class hierarchy, 187
using fail state to indicate errors,
138, 143–144
insert function
AVL_Tree class, 634, 637–638
Binary_Search_Tree class, 468, 468t,
469, 473–474
B_Tree class, 673–676
dequ class, 377t
Graph class, 704
hash_map class, 543t, 547–548,
552–553
list class, 265t, 265–266, 274–276
List_Graph class, 707t, 710
multiset class, 518
Red_Black_Tree class, 650, 653–654
sequences, 294t
set class, 513, 514t, 515, 517
Two_Three_Four_Tree class, 668–670
vector class, 236, 236t, 244,
295–296, 517
insert_into_node function
B_Tree class, 675
Two_Three_Four_Tree class, 670
insertion
into AVL tree, 636–642
into binary search tree, 472–474
into B-trees, 672–679
into heaps, 485, 487–488
into linked lists, 254
nodes into double-linked lists, 260,
261
nodes into single-linked lists, 257
into Red-Black tree, 644–655
into sequences, 231
into 2–3 trees, 658–661
into 2–3–4 trees, 668–670
insertion operator, 44
processing of primitive data types
and strings, 45t
insertion sort, 581–586, 587t
InsertionSort.h, 583–584
installation, 68t
instance variables, 4, 75
int data type, 17, 17t
compatibility with other data types,
22
conversion to other data types, 22
extraction operator processing, 45t
insertion operator processing, 49t
Integrated Development Environment
(IDE), 6
integration testing, 68t, 149, 159–160
Intel x86
division by zero, 132
primitive data types, 17t
interfaces, 63–64, 99, 771
internal nodes, 447
international character set (ISO 646-
1983), 755
invalid_argument exception, 142t
I/O manipulators, 50t, 50–54, 51t
ios_base class, 187
ios_base::failure exception, 141,
142t, 144, 145
iostream class, 50, 54, 187
is-a relationship, 187–188
is_directed function, Graph class, 704t
is_edge function
Graph class, 704
List_Graph class, 7–7t, 709–710
Matrix_Graph class, 713
is_leaf function, Binary_Tree class,
460t, 463
is_null function, Binary_Tree class,
460t
ISO 646-1983, 755
is_red function, Red_Black_Tree class,
654–655
linear probing, hash tables, 533, 537
linear search, recursive, 420–421
linear_search function, 420–421
linked lists, 252–254. See also double-linked lists; single-linked lists
circular, 262
falling off end of, 259
removing elements from, 269–270
traversing, 258, 259
Linked_Queue.tc, 368–370
Linked_Stack.tc, 330–331
linker, 3, 4, 6
links, 255
Linux operating system testing for, 148–159
structured walkthroughs, 148–149
failure to declare virtual in base class, 199
and implicit parameters, 32
modifier, 87
specified as public in class definition, 75, 77–79
members, 4
memory-leak, 120
merge, 592
merge sort, 569, 597–598
merge_sort function, 597–598
MergeSort.h, 597–598
messages, 71
in UML sequence diagrams, 778
message to self, 107
Microsoft C++ compiler, 7
Microsoft Visual Studio .NET development environment, 779
Microsoft Windows operating system, See Windows operating system
M
machine language, 3
macros, 7, 8–9
main function, 6
passing file names as parameters to, 134
maintenance, 68
many-to-one mapping, 521
map class, 513, 521–528
and container class hierarchy, 292
relation to other container concepts, subconcepts, and classes, 774
map.h, 524–525
mapping, 521
maps, 511, 521–528
telephone directory application, 558–560
Mars Lander spacecraft, crash due to logic error, 136
Matrix_Graph class, 703, 713–714
Matrix_Graph::iter_impl class, 713
max_element function, 300
max function, numeric_limits class, 19
max heap, 485
max_size function, allocator class, 758
maze, path finding case study, 436–439
member access operator, 20t, 31, 198
member function overloading, 194–196
member function overriding, 193–194
must have same return type, 200
member functions, 4, 74
access, 87
constant, 79
failure to declare virtual in base class, 199
and implicit parameters, 32
modifier, 87
specified as public in class definition, 75, 77–79
members, 4
memory-leak, 120
merge, 592
merge sort, 569, 592–598
merge_sort function, 595–596
MergeSort.h, 597–598
messages, 71
in UML sequence diagrams, 778
message to self, 107
Microsoft C++ compiler, 7
Microsoft Visual Studio .NET development environment, 779
Microsoft Windows operating system, See Windows operating system
min_char function, 29
min_element function, 300
min function, numeric_limits class, 19
min_heap, 485
minimum spanning trees, 738–742
modeling language, 769
modifier member functions, 87
inline definition, 88
multidimensional arrays, 36–37
multimap class, 513, 528–529
and container class hierarchy, 292
relation to other container concepts, subconcepts, and classes, 774
multimaps, 528–529
multiple inheritance, 210–212
multiplicity, in association, 774–775
multiset class, 513, 518–519
and container class hierarchy, 292
relation to other container concepts, subconcepts, and classes, 774
multisets, 518–519
mutator, 87
my_vector class, 242–243
namespace declaration, 213–214
namespaces, 5–6
explicit namespace qualification in include files, 85
global, 214
and visibility, 213–218
when to use, 215
namespace std, 5–6
narrowing conversion, 22
nested classes, UML syntax, 774
nested if statements, 14–15
new operator, 20, 27
new object creation using, 197
syntax, 28
new[] operator, 35
Node class, 255–256
Node.h, 255, 367
nodes, 255–256
connecting, 256–257
inserting into double-linked list, 260, 261
inserting into single-linked list, 257
removing from double-linked list, 260–261
removing from single-linked list, 257–258
testing if there are more to process, 267
nodes, in graphs, 692
as inner classes, 775
nodes, in linked lists, 253
no-parameter constructor, 191
noshowpoint I/O manipulator, 50
noskipws I/O manipulator, 50
not_eq keyword, digraphs, trigraphs, or equivalents for, 756
not equal to operator, 20
notes, in UML sequence diagrams, 778
not keyword, digraphs, trigraphs, or equivalents for, 756
not operator, 12
null character, 36
null pointer, 27
error in dereferencing, 132
135–136
numeric_limits class, 19
O
object code, 3
object file, 4
Object Modeling Group, 769
object-oriented design, 71–72
object-oriented programming (OOP), 186
objects, 4, 23
copying, 247–250
dynamically created, 27–28
function, 303–306
lifetimes, 23–24
referencing actual, 203–204
in UML sequence diagrams, 776–777
object type, 23
offset, to pointer, 602
ofstream class, 54, 187
onto mapping, 521
open addressing, hash tables, 532–535, 536, 540–541, 554
open-box testing, 149
openmode flags, 54
operations, 773
operator functions, 31–32
operator precedence, 20
operators, 19, 71, 73–74, 88–89
order, of B-tree, 672
ordered list case study, 285–291
order of magnitude, 173
or_eq keyword, digraphs, trigraphs, or equivalents for, 756
or keyword, digraphs, trigraphs, or equivalents for, 756
or operator, 12, 20
ostream class, 44, 49, 54, 187
reading binary tree, 464–465
ostringstream class, 57, 187
out_of_range exception, 138–139, 140, 142
out openmode flag, 55
t
output streams, 44, 49–50
class hierarchy, 187
using fail state to indicate errors, 138
overflow_error exception, 141, 142
overloading, of member functions, 194–196
overloading operators, 88
overriding, of member functions, 193–194, 200
pair class, 515, 518–519
palindrome case study, 315–320
@param documentation comment tag, 97, 97
parameters, 30
referring when hidden by local declaration, 92
parent, 447
parentheses
converting expressions with case study, 347–350
testing expressions for balanced case study, 320–324
partition function (Quicksort), 607–613
path coverage, 150
paths, graphs, 694–695, 694–696
Pez dispenser, as example of stack, 312
phases, in Unified Model, software life cycle, 66
physical view, 73
pivot, in quicksort, 604
Plain Old Data (POD), 256
plug-ins, 164
pointers, 24–26. See also smart pointers
accessing object through (dereferencing operator), 198
array-pointer equivalence, 34–35
declaring multiple in one declaration, 27
declaring type, 197
initializing to NULL, 136
invalid arithmetic, 602
null, 27, 135–136
when to reference an object through, 199
polymorphism, 200–201
pop_back function
deque class, 376, 377
nodes from double-linked lists using iterators, 269–270
nodes from single-linked lists, 257–258
from Red-Black tree, 655
from sequences, 231
from 2-3 trees, 661–662
remove function
deque class, 377t
list class, 265t
repetition control, 12–14
replace_parent function,
Binary_Search_Tree class, 478–479
requirements, in software life cycle, 65, 66
requirements specification, 65, 68–69
telephone directory case study, 102–108
reserve function, vector class, 244, 245–246
resetiosflags I/O manipulator, 51t
resize function
deque class, 377t
vector class, 236t
@return documentation comment tag, 97, 97t
reusability, 185
reusable code, 98
reverse function, 300t
rfind function, string class, 40t
right I/O manipulator, 50t
right rotation, of binary search trees, 625
rotate_left function, BST_With_Rotate class, 627, 628
rotate_right function, BST_With.Rotate class, 627, 628
rotation, of binary search trees, 624–628
rule of three, 252
run-time errors, 131–136
run-time type identification, 206
r-value, 204

S
scientific floating-point format, 52t
scientific I/O manipulator, 50t
scope resolution operator, 31
search. See also binary search
breadth-first, 715–720
depth-first, 715, 720–726
recursive linear, 420–421
2-3 trees, 657
secondary branches, of trees, 445
security concerns
buffer overruns, 135
and lack of range checking for arrays, 237
seed, for rand function, 395–396
selection control, 12–14
selection sort, 572–576, 587t
SelectionSortArray.h, 576
SelectionSort.h, 574–576
selector, in control statements, 13t
self-balancing search trees, 623
AVL trees, 628–643
B+ trees, 681
B-trees, 672–680
Red-Black trees, 643–655, 671
2-3 trees, 656–662
2-3-4 trees, 663–671
sequence concept, 292, 774
sequence diagrams, See UML sequence diagrams
sequences, 231–232, 292, 294–295, 771
sequence statements, 12
sequential access, 232
sequential containers, 292. See also list class; vectors
set class, 513–517, 514t
and container class hierarchy, 292
relation to other container concepts, subconcepts, and classes, 774
set difference, 513
set_difference function, 514t, 516–517
setfill I/O manipulator, 51t
set hierarchy, 513
set intersection, 513
set_intersection function, 514t, 516–517
setiosflags I/O manipulator, 51t
set operators, 516–517
setprecision I/O manipulator, 51t
set_red function, Red_Black_Tree class, 654–655
sets, 511, 512–513
vectors compared, 517–518
setter, 87
set union, 513
set_union function, 514t, 515, 516–517
setw I/O manipulator, 51t
shared_ptr class, 765–767, 767–768
~shared_ptr function, 767t
Shell sort, 569, 586, 588–592
ShellSort.t.h, 591–592
short-circuit evaluation, 12
short data type, 17, 17t
extraction operator processing, 45t
insertion operator processing, 49t
showpoint I/O manipulator, 50t
siblings, 447
signature, of a function, 190
signed bit shift left operator, 20t
signed bit shift right operator, 20t
signed char data type, 17t, 17–18
simple paths, graphs, 694, 695
simulation, 380–381
single-linked lists, 252–258
implementing queues as, 367–370
implementing stacks as, 329–331
problems with, 258–259
traversing, 258
single-step execution, debugger, 162
size, of vector, 234
size_t function, hash_map class, 556
skips I/O manipulator, 50t
smart pointers, 764–767
for BTNodle manipulator, 50t
software life cycle, 64–72
software life cycle activities, 67–72
Unified Model, 66–67
software life cycle models, 65–67
software performance, 176–178
sort function, 300t
sorting, 569
bubble sort, 576–580, 587t
comparison of quadratic, 586–588
comparisons and exchanges contrasted, 588
Dutch National Flag problem case study, 616–619
heapsort, 569, 599–603
insertion sort, 581–586, 587t
merge sort, 569, 592–598
quicksort, 569, 604–613
selection sort, 572–576, 587t
Shell sort, 569, 586, 588–592
testing, 614–616
using C++ functions, 570–572
source, directed graphs, 693
source character set, 755–756
source file, 4
spanning trees, 738–742
sparse graph, 713–714
specialization, of swap function, 302, 303
split_node function
B_Tree class, 675, 676–679
Two_Three_Four_Tree class, 670
stable_sort function, 290
stack abstract data type, 312–315
stack class, 312–315
stack container, 312–313
stack.h, 313–315, 327–329
stacks, 311, 312–315
activation frames and recursive functions, 409–411
applications, 315–324, 332–350
attempting to pop empty, 324
converting expressions with parentheses case study, 347–350
converting infix to postfix case study, 339–347
evaluating postfix expressions case study, 333–338
finding palindromes case study, 315–320
implementing, 325–331
overflow from infinite recursion, 413
testing expressions for balanced parentheses case study, 320–324
stack.tc, 326–327, 328
standard exceptions, 141–143, 142
standard library, 2–7, 186
abstract data type implementations, 99
algorithms, 299–301
Standard Template Library (STL), 231
static attributes, 772
std::less function class, 495
step into, 162
step over, 162
stepwise refinement, 70
streams, 44
file, 54–57
input, 44, 45–49
output, 44, 49–50
string, 57–58
str function, 57t
string class, 36, 38–43
functions, 39–40t
string function, string class, 39t
~string function, string class, 39t
string literals, 36
strings, 38–43
extraction operator processing, 45t
insertion operator processing, 49t
splitting into tokens, 42
as templates, 43
stringstream class, 187
string streams, 57–58
strongly typed language, 204, 205
struct keyword, 255, 256
structure chart, 70
structured walkthroughs, 148–149
subs, 156–157
subclasses, 186
subscription
out-of-bound, 33
using postfix and prefix increment with, 243
subscripting operator
deque class, 377t, 379
hash_map class, 543t, 549
member function at compared, 237
vector class, 236t, 236–237, 242–243
substr function, string class, 40t, 41
subtrees, 447
sum_each_row function, 37
superclasses, 186
swap function, 300t, 302–303
containers, 293t
deque class, 377t
list class, 265t
shared_ptr class, 767t
specialization, 302, 303
vector class, 236t, 241–242
switch statement, 15–16
and program testing, 150
purpose and syntax, 13t
syntax
abstract function definition, 203
address-of operator, 25
alias for type, 289
assert statement, 169
calling base-class functions, 194
class definition, 78
constant pointer declaration, 546
const_cast, 523–524
constructor definition, 86
collection statements, 13t
declaring a virtual base class, 764
define preprocessor directive, 8
delete operator, 28
delete[] operator, 35
dereferencing operator, 26
derived class definition, 190
destructor definition, 251
dynamic cast, 206
friend declaration, 217
inline function declaration, 303
instantiating static data members of
template class, 545
member function definition, 79
namespace declaration, 213–214
new operator, 28
new[] operator, 35
object declaration, 23–24
operator definition, 88
pointer declaration, 25
template class instantiation, 233
template function call, 299
template function definition, 298
throw statement, 139
try block, 141
UML syntax, 109
using declaration, 215
using directive, 215
virtual function definition, 199
syntax errors, 131
system analysts, 65
system testing, 149

T
tail recursion, 416
telephone directory case study, 102–124
map application, 558–560
vector application, 239
template classes, 232–233, 241
accessing type defined inside template class from within template class, 281
failure to qualify template base class name, 472
instantiating static data members of, 545
UML notation, 776
template functions, 297–299
term paper index case study, 479–484
test cases, 136
test frameworks, 158–159
CppUnit, 779–781
test harness, 159
testing
binary search trees, 479–484
boundary conditions, 153–155
classes, 157–158
and drivers, 157
hash table implementations, 553–554
levels and types of, 149–151
logic errors, 148–159
for more nodes to process, 267
preparations for, 151
in software life cycle, 65, 66
sorting, 614–616
structured walkthroughs, 148–149
and stubs, 156–157
telephone directory case study, 121
test data development, 152–153
tips for program systems, 152
who performs?, 156
test suite, 159

this parameter, 91
throws an exception, 138
throw statement, 138–139

time axis, in UML sequence diagrams, 776
tokens, 482
splitting a string into, 42
top-down design, 70–71
top function
priority_queue class, 491t
stack class, 313t

Topological_Sort.cpp, 732–733
to_string function
Binary_Tree class, 460t, 463–464
BNode class, 457
Edge class, 700t
Towers of Hanoi case study, 427–431
traits, 757–759
traits class, 757–759
transform function, 300t, 306t
transition, in Unified Model, of software life cycle, 66–67
trees, 445–446. See also binary search trees; expression trees; heaps;
Huffman tree
B+ trees, 681
B-trees, 672–680
forest, 724
fullness and completeness, 451–452, 623
general, 452–453
and graphs, 696
terminal, 447–453
2-3 trees, 656–662
2-3-4 trees, 663–671
tree traversal, 454–456
trigraphs, 755–756, 756t
trunc openmode flag, 55t
trunk, 445
try blocks, 140–141
2-nodes, 656, 663
3-nodes, 656, 663
Two_Three_Four_Tree class, 666–667
2-3 trees, 656–662
2-3-4 trees, 663–671
and Red-Black trees, 671
type alias, 289
type casting, 133
typedef keyword, 289
types, 16–19

U
UML class diagrams, 72, 770–776
telephone directory case study, 104–106, 108–109
UML sequence diagrams, 72, 776–778
telephone directory case study, 106–108
UML (Unified Modeling Language) association, 774–775
composition, 775–776
as design tool, 72
generalization, 773–774
overview, 769–778
unary plus/minus operator, 20t
unbalanced binary search trees, 624
uncaught exceptions, 139–140
underflow_error exception, 141, 142t
undirected graphs, 693–694
Unicode character set, 18
Unified Model, of software life cycle, 66–67
Unified Modeling Language (UML), See UML
unit testing, 68t, 92, 149
unnamed pointer, 205
unreachable exception handler, 146
unsigned char data type, 17t, 17–18
unsigned int data type, 17, 17t
conversion with int data type, 22
unsigned long data type, 17, 17t
unsigned short data type, 17, 17t

while statement, 13
purpose and syntax, 13t
white-box testing, 149–150
widening conversion, 22
Windows operating system
attempting to pop empty stack, 324
division by zero, 132
null pointer reference, 135
uncaught exceptions, 140
workflows, in Unified Model, software life cycle, 66, 67
wrapper function, 418
ws I/O manipulator, 50t

X
xor_eq keyword, digraphs, trigraphs, or equivalents for, 756t
xor keyword, digraphs, trigraphs, or equivalents for, 756t
zero, division by, 131–132

W
waterfall model, of software life cycle, 65–66
wchar_t data type, 17t, 18
weighted graphs, 694
algorithms using, 734–742
minimum spanning trees, 738–742
weights, 694
while statement, 13
purpose and syntax, 13t
waterfall model, of software life cycle, 65–66