

DATA STRUCTURES AND PROBLEM SOLVING USING C++

Second Edition

MARK ALLEN WEISS

Florida International University

 **ADDISON-WESLEY**

An imprint of Addison Wesley Longman, Inc.

Reading, Massachusetts • Harlow, England • Menlo Park, California
Berkeley, California • Don Mills, Ontario • Sydney • Bonn • Amsterdam
Tokyo • Mexico City

Contents

Part I: Objects and C++

Chapter 1 Arrays, Pointers, and Structures 3

- 1.1 What Are Pointers, Arrays, and Structures? 3
- 1.2 Arrays and Strings 4
 - 1.2.1 First-Class Versus Second-Class Objects 4
 - 1.2.2 Using the **vector** 6
 - 1.2.3 Resizing a **vector** 7
 - 1.2.4 **push_back: size and capacity** 11
 - 1.2.5 Parameter-Passing Mechanisms 11
 - 1.2.6 Primitive Arrays of Constants 13
 - 1.2.7 Multidimensional Arrays 14
 - 1.2.8 The Standard Library **string** Type 14
- 1.3 Pointer Syntax in C++ 15
- 1.4 Dynamic Memory Management 20
 - 1.4.1 The **new** Operator 21
 - 1.4.2 Garbage Collection and **delete** 21
 - 1.4.3 Stale Pointers, Double Deletion, and More 22
- 1.5 Reference Variables 24
- 1.6 Structures 26
 - 1.6.1 Pointers to Structures 28
 - 1.6.2 Exogenous Versus Indigenous Data and Shallow Versus Deep Copying 29
 - 1.6.3 Noncontiguous Lists: Linked Lists 30
- Summary 32
- Objects of the Game 32
- Common Errors 34
- On the Internet 35
- Exercises 35
- References 38

Chapter 2 Objects and Classes 41

- 2.1 What Is Object-Oriented Programming? 41
- 2.2 Basic **class** Syntax 43
 - 2.2.1 Class Members 43
 - 2.2.2 Extra Constructor Syntax and Accessors 45
 - 2.2.3 Separation of Interface and Implementation 48
 - 2.2.4 The Big Three: Destructor, Copy Constructor, and **operator=** 51
 - 2.2.5 Default Constructor 57
- 2.3 Additional C++ Class Features 57
 - 2.3.1 Initialization Versus Assignment in the Constructor Revisited 61
 - 2.3.2 Type Conversions 63
 - 2.3.3 Operator Overloading 64
 - 2.3.4 Input and Output and Friends 67
- 2.4 Some Common Idioms 68
 - 2.4.1 Avoiding Friends 70
 - 2.4.2 Static Class Members 71
 - 2.4.3 The **enum** Trick for Integer Class Constants 71
- 2.5 Exceptions 72
- 2.6 A **string** Class 73
- 2.7 Recap: What Gets Called and What Are the Defaults? 82
- 2.8 Composition 84
- Summary 85
- Objects of the Game 85
- Common Errors 87
- On the Internet 89
- Exercises 90
- References 96

Chapter 3 Templates 97

- 3.1 What Is a Template? 97
- 3.2 Function Templates 98
- 3.3 A Sorting Function Template 100
- 3.4 Class Templates 103
 - 3.4.1 A **MemoryCell** Template 103
 - 3.4.2 Implementing the **vector** Class Template 108
- 3.5 Templates of Templates: A **matrix** Class 108
 - 3.5.1 The Data Members, Constructor, and Basic Accessors 111
 - 3.5.2 **operator[]** 112
 - 3.5.3 Destructor, Copy Assignment, and Copy Constructor 112

- 3.6 Fancy Templates 112
 - 3.6.1 Multiple Template Parameters 112
 - 3.6.2 Default Template Parameters 113
 - 3.6.3 The Reserved Word **typename** 113
- 3.7 Bugs Associated with Templates 114
 - 3.7.1 Bad Error Messages and Inconsistent Rules 114
 - 3.7.2 Template-Matching Algorithms 114
 - 3.7.3 Nested Classes in a Template 114
 - 3.7.4 Static Members in Class Templates 115
- Summary 115
- Objects of the Game 115
- Common Errors 115
- On the Internet 116
- Exercises 117

Chapter 4 Inheritance 119

- 4.1 What Is Inheritance? 119
- 4.2 Inheritance Basics 123
 - 4.2.1 Visibility Rules 124
 - 4.2.2 The Constructor and Base Class Initialization 125
 - 4.2.3 Adding Members 126
 - 4.2.4 Overriding a Method 128
 - 4.2.5 Static and Dynamic Binding 129
 - 4.2.6 The Default Constructor, Copy Constructor, Copy Assignment Operator, and Destructor 131
 - 4.2.7 Constructors and Destructors: Virtual or Not Virtual? 132
 - 4.2.8 Abstract Methods and Classes 133
- 4.3 Example: Expanding the **Shape** Class 136
- 4.4 Tricky C++ Details 142
 - 4.4.1 Static Binding of Parameters 142
 - 4.4.2 Default Parameters 143
 - 4.4.3 Derived Class Methods Hide Base Class Methods 144
 - 4.4.4 Compatible Return Types for Overridden Methods 145
 - 4.4.5 Private Inheritance 146
 - 4.4.6 Friends 146
 - 4.4.7 Call by Value and Polymorphism Do Not Mix 146
- 4.5 Multiple Inheritance 147
- Summary 149
- Objects of the Game 149
- Common Errors 150
- On the Internet 152
- Exercises 152
- References 154

Chapter 5 Design Patterns 155

- 5.1 What Is a Pattern? 155
- 5.2 The Functor (Function Objects) 156
- 5.3 Adapters and Wrappers 162
 - 5.3.1 Wrapper for Pointers 162
 - 5.3.2 A Constant Reference Wrapper 168
 - 5.3.3 Adapters: Changing an Interface 169
- 5.4 Iterators 170
 - 5.4.1 Iterator Design 1 171
 - 5.4.2 Iterator Design 2 174
 - 5.4.3 Inheritance-Based Iterators and Factories 174
- 5.5 Composite (Pair) 179
- 5.6 Observer 179
- Summary 184
- Objects of the Game 184
- Common Errors 185
- On the Internet 186
- Exercises 186
- References 190

Part II: Algorithms and Building Blocks**Chapter 6 Algorithm Analysis 193**

- 6.1 What Is Algorithm Analysis? 193
- 6.2 Examples of Algorithm Running Times 198
- 6.3 The Maximum Contiguous Subsequence Sum Problem 199
 - 6.3.1 The Obvious $O(N^3)$ Algorithm 200
 - 6.3.2 An Improved $O(N^2)$ Algorithm 203
 - 6.3.3 A Linear Algorithm 204
- 6.4 General Big-Oh Rules 206
- 6.5 The Logarithm 211
- 6.6 Static Searching Problem 214
 - 6.6.1 Sequential Search 214
 - 6.6.2 Binary Search 215
 - 6.6.3 Interpolation Search 217
- 6.7 Checking an Algorithm Analysis 219
- 6.8 Limitations of Big-Oh Analysis 220

Summary	221
Objects of the Game	221
Common Errors	222
On the Internet	223
Exercises	223
References	228

Chapter 7 The Standard Template Library 231

7.1	Introduction	232
7.2	Stacks and Queues	233
7.2.1	Stacks	233
7.2.2	Stacks and Computer Languages	235
7.2.3	Queues	236
7.3	Containers and Iterators	237
7.3.1	Containers	238
7.3.2	The iterator	238
7.4	STL Algorithms	240
7.4.1	STL Function Objects	240
7.4.2	Binary Search	243
7.4.3	Sorting	244
7.5	Implementation of vector with an Iterator	245
7.6	Sequences and Linked Lists	247
7.6.1	The list Class	247
7.6.2	Stacks and Queues	249
7.7	Sets	249
7.8	Maps	251
7.9	Priority Queues	253
	Summary	257
	Objects of the Game	257
	Common Errors	259
	On the Internet	259
	Exercises	260
	References	264

Chapter 8 Recursion 265

8.1	What Is Recursion?	265
8.2	Background: Proofs by Mathematical Induction	267
8.3	Basic Recursion	269
8.3.1	Printing Numbers in Any Base	271
8.3.2	Why It Works	274

8.3.3	How It Works	275
8.3.4	Too Much Recursion Can Be Dangerous	276
8.3.5	Preview of Trees	278
8.3.6	Additional Examples	279
8.4	Numerical Applications	284
8.4.1	Modular Arithmetic	285
8.4.2	Modular Exponentiation	285
8.4.3	Greatest Common Divisor and Multiplicative Inverses	287
8.4.4	The RSA Cryptosystem	289
8.5	Divide-and-Conquer Algorithms	292
8.5.1	The Maximum Contiguous Subsequence Sum Problem	293
8.5.2	Analysis of a Basic Divide-and-Conquer Recurrence	297
8.5.3	A General Upper Bound for Divide-and-Conquer Running Times	301
8.6	Dynamic Programming	303
8.7	Backtracking	308
	Summary	310
	Objects of the Game	312
	Common Errors	313
	On the Internet	314
	Exercises	314
	References	319

Chapter 9 Sorting Algorithms 321

9.1	Why Is Sorting Important?	322
9.2	Preliminaries	323
9.3	Analysis of the Insertion Sort and Other Simple Sorts	324
9.4	Shellsort	326
9.4.1	Performance of Shellsort	328
9.5	Mergesort	330
9.5.1	Linear-Time Merging of Sorted Arrays	330
9.5.2	The Mergesort Algorithm	332
9.6	Quicksort	334
9.6.1	The Quicksort Algorithm	335
9.6.2	Analysis of Quicksort	337
9.6.3	Picking the Pivot	340
9.6.4	A Partitioning Strategy	342
9.6.5	Keys Equal to the Pivot	344
9.6.6	Median-of-Three Partitioning	345
9.6.7	Small Arrays	346
9.6.8	C++ Quicksort Routine	346

- 9.7 Quickselect 348
- 9.8 A Lower Bound for Sorting 349
- 9.9 Indirect Sorting 352
 - 9.9.1 Using Pointers to Reduce **Comparable** Copies to $2N$ 352
 - 9.9.2 Avoiding the Extra Array 353
- Summary 355
- Objects of the Game 356
- Common Errors 357
- On the Internet 357
- Exercises 358
- References 363

Chapter 10 Randomization 365

- 10.1 Why Do We Need Random Numbers? 365
- 10.2 Random Number Generators 366
- 10.3 Nonuniform Random Numbers 371
- 10.4 Generating a Random Permutation 373
- 10.5 Randomized Algorithms 375
- 10.6 Randomized Primality Testing 378
- Summary 380
- Objects of the Game 382
- Common Errors 383
- On the Internet 383
- Exercises 383
- References 386

Part III: Applications

Chapter 11 Fun and Games 389

- 11.1 Word Search Puzzles 389
 - 11.1.1 Theory 390
 - 11.1.2 C++ Implementation 391
- 11.2 The Game of Tic-Tac-Toe 395
 - 11.2.1 Alpha–Beta Pruning 397
 - 11.2.2 Transposition Tables 398
 - 11.2.3 Computer Chess 404
- Summary 405
- Objects of the Game 405

Common Errors	406
On the Internet	406
Exercises	406
References	408

Chapter 12 Stacks and Compilers 409

12.1	Balanced-Symbol Checker	409
12.1.1	Basic Algorithm	409
12.1.2	Implementation	411
12.2	A Simple Calculator	420
12.2.1	Postfix Machines	421
12.2.2	Infix to Postfix Conversion	422
12.2.3	Implementation	424
12.2.4	Expression Trees	432
	Summary	435
	Objects of the Game	435
	Common Errors	436
	On the Internet	436
	Exercises	436
	References	438

Chapter 13 Utilities 439

13.1	File Compression	439
13.1.1	Prefix Codes	440
13.1.2	Huffman's Algorithm	442
13.1.3	Implementation	445
13.2	A Cross-Reference Generator	461
13.2.1	Basic Ideas	461
13.2.2	C++ Implementation	462
	Summary	466
	Objects of the Game	466
	Common Errors	466
	On the Internet	467
	Exercises	467
	References	470

Chapter 14 Simulation 471

14.1	The Josephus Problem	471
14.1.1	The Simple Solution	473
14.1.2	A More Efficient Algorithm	473

- 14.2 Event-Driven Simulation 475
 - 14.2.1 Basic Ideas 477
 - 14.2.2 Example: A Modem Bank Simulation 478
- Summary 486
- Objects of the Game 486
- Common Errors 486
- On the Internet 486
- Exercises 486

Chapter 15 Graphs and Paths 489

- 15.1 Definitions 489
 - 15.1.1 Representation 491
- 15.2 Unweighted Shortest-Path Problem 503
 - 15.2.1 Theory 504
 - 15.2.2 C++ Implementation 509
- 15.3 Positive-Weighted, Shortest-Path Problem 509
 - 15.3.1 Theory: Dijkstra's Algorithm 509
 - 15.3.2 C++ Implementation 513
- 15.4 Negative-Weighted, Shortest-Path Problem 514
 - 15.4.1 Theory 514
 - 15.4.2 C++ Implementation 517
- 15.5 Path Problems in Acyclic Graphs 517
 - 15.5.1 Topological Sorting 517
 - 15.5.2 Theory of the Acyclic Shortest-Path Algorithm 520
 - 15.5.3 C++ Implementation 522
 - 15.5.4 An Application: Critical-Path Analysis 522
- Summary 526
- Objects of the Game 527
- Common Errors 528
- On the Internet 529
- Exercises 529
- References 533

Part IV: Implementations

Chapter 16 Stacks and Queues 537

- 16.1 Dynamic Array Implementations 537
 - 16.1.1 Stacks 538
 - 16.1.2 Queues 541

16.2	Linked List Implementations	548
16.2.1	Stacks	548
16.2.2	Queues	553
16.3	Comparison of the Two Methods	557
16.4	The STL Stack and Queue Adapters	558
16.5	Double-Ended Queues	558
	Summary	559
	Objects of the Game	561
	Common Errors	561
	On the Internet	561
	Exercises	562

Chapter 17 Linked Lists 565

17.1	Basic Ideas	565
17.1.1	Header Nodes	567
17.1.2	Iterator Classes	569
17.2	C++ Implementation	570
17.3	Doubly Linked Lists and Circularly Linked Lists	579
17.4	Sorted Linked Lists	582
17.5	Implementing the STL list Class	582
	Summary	597
	Objects of the Game	597
	Common Errors	598
	On the Internet	598
	Exercises	599

Chapter 18 Trees 605

18.1	General Trees	605
18.1.1	Definitions	605
18.1.2	Implementation	607
18.1.3	An Application: File Systems	608
18.2	Binary Trees	611
18.3	Recursion and Trees	619
18.4	Tree Traversal: Iterator Classes	622
18.4.1	Postorder Traversal	624
18.4.2	Inorder Traversal	630
18.4.3	Preorder Traversal	630
18.4.4	Level-Order Traversals	630

Summary	633
Objects of the Game	636
Common Errors	637
On the Internet	637
Exercises	637

Chapter 19 Binary Search Trees 641

19.1 Basic Ideas	641
19.1.1 The Operations	642
19.1.2 C++ Implementation	644
19.2 Order Statistics	652
19.2.1 C++ Implementation	653
19.3 Analysis of Binary Search Tree Operations	657
19.4 AVL Trees	661
19.4.1 Properties	662
19.4.2 Single Rotation	664
19.4.3 Double Rotation	667
19.4.4 Summary of AVL Insertion	670
19.5 Red–Black Trees	670
19.5.1 Bottom–Up Insertion	672
19.5.2 Top–Down Red–Black Trees	674
19.5.3 C++ Implementation	676
19.5.4 Top–Down Deletion	680
19.6 AA-Trees	685
19.6.1 Insertion	686
19.6.2 Deletion	688
19.6.3 C++ Implementation	690
19.7 Implementing the STL set and map Classes	693
19.8 B-Trees	707
Summary	714
Objects of the Game	715
Common Errors	717
On the Internet	717
Exercises	718
References	721

Chapter 20 Hash Tables 725

20.1 Basic Ideas	725
20.2 Hash Function	727
20.3 Linear Probing	729
20.3.1 Naive Analysis of Linear Probing	731

20.3.2	What Really Happens: Primary Clustering	732
20.3.3	Analysis of the find Operation	733
20.4	Quadratic Probing	735
20.4.1	C++ Implementation	739
20.4.2	Analysis of Quadratic Probing	745
20.5	Separate Chaining Hashing	746
20.6	Hash Tables Versus Binary Search Trees	746
20.7	Hashing Applications	747
	Summary	747
	Objects of the Game	748
	Common Errors	749
	On the Internet	749
	Exercises	749
	References	752
Chapter 21	A Priority Queue: The Binary Heap	755
21.1	Basic Ideas	755
21.1.1	Structure Property	756
21.1.2	Heap-Order Property	758
21.1.3	Allowed Operations	759
21.2	Implementation of the Basic Operations	761
21.2.1	The insert Operation	762
21.2.2	The deleteMin Operation	763
21.3	The buildHeap Operation: Linear-Time Heap Construction	766
21.4	STL priority_queue Implementation	771
21.5	Advanced Operations: decreaseKey and merge	773
21.6	Internal Sorting: Heapsort	773
21.7	External Sorting	778
21.7.1	Why We Need New Algorithms	778
21.7.2	Model for External Sorting	778
21.7.3	The Simple Algorithm	779
21.7.4	Multiway Merge	781
21.7.5	Polyphase Merge	782
21.7.6	Replacement Selection	783
	Summary	785
	Objects of the Game	785
	Common Errors	786
	On the Internet	787
	Exercises	787
	References	791

Part V: Advanced Data Structures

Chapter 22 Splay Trees 795

- 22.1 Self-Adjustment and Amortized Analysis 795
 - 22.1.1 Amortized Time Bounds 797
 - 22.1.2 A Simple Self-Adjusting Strategy (That Does Not Work) 797
- 22.2 The Basic Bottom–Up Splay Tree 799
- 22.3 Basic Splay Tree Operations 802
- 22.4 Analysis of Bottom–Up Splaying 803
 - 22.4.1 Proof of the Splaying Bound 806
- 22.5 Top–Down Splay Trees 809
- 22.6 Implementation of Top–Down Splay Trees 812
- 22.7 Comparison of the Splay Tree with Other Search Trees 818
- Summary 819
- Objects of the Game 819
- Common Errors 820
- On the Internet 820
- Exercises 820
- References 822

Chapter 23 Merging Priority Queues 823

- 23.1 The Skew Heap 823
 - 23.1.1 Merging Is Fundamental 823
 - 23.1.2 Simplistic Merging of Heap-Ordered Trees 824
 - 23.1.3 The Skew Heap: A Simple Modification 825
 - 23.1.4 Analysis of the Skew Heap 826
- 23.2 The Pairing Heap 828
 - 23.2.1 Pairing Heap Operations 829
 - 23.2.2 Implementation of the Pairing Heap 830
 - 23.2.3 Application: Dijkstra’s Shortest Weighted Path Algorithm 836
- Summary 840
- Objects of the Game 840
- Common Errors 841
- On the Internet 841
- Exercises 841
- References 842

Chapter 24 The Disjoint Set Class 845

- 24.1 Equivalence Relations 845
- 24.2 Dynamic Equivalence and Two Applications 846

24.2.1	Application: Generating Mazes	847
24.2.2	Application: Minimum Spanning Trees	850
24.2.3	Application: The Nearest Common Ancestor Problem	853
24.3	The Quick-Find Algorithm	857
24.4	The Quick-Union Algorithm	858
24.4.1	Smart Union Algorithms	860
24.4.2	Path Compression	862
24.5	C++ Implementation	863
24.6	Worst Case for Union-by-Rank and Path Compression	865
24.6.1	Analysis of the Union/Find Algorithm	866
	Summary	873
	Objects of the Game	873
	Common Errors	875
	On the Internet	875
	Exercises	875
	References	877

Appendices

Appendix A Miscellaneous C++ Details A-3

A.1	None of the Compilers Implement the Standard	A-3
A.2	Unusual C++ Operators	A-4
A.2.1	Autoincrement and Autodecrement Operators	A-4
A.2.2	Type Conversions	A-5
A.2.3	Bitwise Operators	A-6
A.2.4	The Conditional Operator	A-8
A.3	Command-Line Arguments	A-8
A.4	Input and Output	A-9
A.4.1	Basic Stream Operations	A-9
A.4.2	Sequential Files	A-13
A.4.3	String Streams	A-13
A.5	Namespaces	A-15
A.6	New C++ Features	A-17
	Common C++ Errors	A-17

Appendix B Operators A-21**Appendix C Some Library Routines A-23**

- C.1 Routines Declared in `<ctype.h>` and `<cctype>` A-23
- C.2 Constants Declared in `<limits.h>` and `<climits>` A-24
- C.3 Routines Declared in `<math.h>` and `<cmath>` A-25
- C.4 Routines Declared in `<stdlib.h>` and `<cstdlib>` A-26

Appendix D Primitive Arrays in C++ A-27

- D.1 Primitive Arrays A-27
 - D.1.1 The C++ Implementation: An Array Name Is a Pointer A-28
 - D.1.2 Multidimensional Arrays A-31
 - D.1.3 The `char *` Type, `const` Pointers, and Constant Strings A-31
- D.2 Dynamic Allocation of Arrays: `new []` and `delete []` A-35
- D.3 Pointer Arithmetic, Pointer Hopping, and Primitive Iteration A-41
 - D.3.1 Implications of the Precedence of `*`, `&`, and `[]` A-41
 - D.3.2 What Pointer Arithmetic Means A-42
 - D.3.3 A Pointer-Hopping Example A-44
 - D.3.4 Is Pointer Hopping Worthwhile? A-45
- Common C++ Errors A-47
- On the Internet A-47

Index I-1