

BRIEF CONTENTS

Preface xiv

CHAPTER 1	
Classes in C++	1
CHAPTER 2	
Storage Structures for Container Classes	35
CHAPTER 3	
Introduction to Software Engineering	63
CHAPTER 4	
Recursion	93
CHAPTER 5	
Vectors and Deques	163
CHAPTER 6	
Lists	205
CHAPTER 7	
Queues and Stacks	253
CHAPTER 8	
Binary Trees and Binary Search Trees	307
CHAPTER 9	
AVL Trees	353
CHAPTER 10	
Red-Black Trees	391
CHAPTER 11	
Priority Queues and Heaps	439
CHAPTER 12	
Sorting	477

CHAPTER	13	
	Searching and the Hash Classes	513
<hr/>		
CHAPTER	14	
	Graphs, Trees, and Networks	563
<hr/>		
APPENDIX	1	
	Mathematical Background	619
<hr/>		
APPENDIX	2	
	The string Class	633
<hr/>		
APPENDIX	3	
	Polymorphism	647
<hr/>		

References 651

Index 653

CONTENTS

Preface xiv

CHAPTER 1 Classes in C++ 1

1.1	Classes	2
1.1.1	Method Interfaces	2
1.1.2	Objects	3
1.1.3	Data Abstraction	6
1.1.4	Constructors	8
1.1.5	An Employee Class	10
1.1.6	Definition of the Employee Class	15
	Lab 1: The Company Project	17
1.1.7	Inheritance	17
1.1.8	Protected Access	18
1.1.9	The HourlyEmployee Class	20
	Lab 2: More Details of Inheritance	23
1.1.10	Operator Overloading	25
1.1.11	Friends	26
	Lab 3: Overloading operator= and operator>>	27
1.1.12	Information Hiding	28
	Summary	28
	Exercises	29
	Programming Project 1.1: A Sequence Class	33

CHAPTER 2 Storage Structures for Container Classes 35

2.1	Pointers	36
2.1.1	The Heap versus the Stack	37
2.1.2	Reference Parameters	38
2.1.3	Pointer Fields	39

2.1.4	Arrays and Pointers	39
	Lab 4: Pointer-Variable Assignments versus Dynamic-Variable Assignments	40
2.1.5	Deallocation of Dynamic Variables	40

2.2	Arrays	41
2.3	Container Classes	42
2.3.1	Storage Structures for Container Classes	44
2.3.2	Linked Structures	44
2.3.3	Iterators	48
2.3.4	Design and Implementation of the Iterator Class	50
	Lab 5: Defining the Other Iterator Operators	52
2.3.5	The pop_front Method	52
2.3.6	Destructors	53
	Lab 6: Overloading operator=	54
2.3.7	Generic Algorithms	54
	Lab 7: More on Generic Algorithms	58
2.3.8	Data Structures and the Standard Template Library	58
	Summary	59
	Exercises	60
	Programming Project 2.1: Extending the Linked Class	62

CHAPTER 3 Introduction to Software Engineering 63

3.1	The Software Developmental Life Cycle	64
3.2	Problem Analysis	64
3.2.1	System Tests	66
3.3	Program Design	67
3.3.1	Method Interfaces and Fields	67
3.3.2	Dependency Diagrams	68

3.4	Program Implementation	71
3.4.1	Method Validation	71
	Lab 8: Drivers	72
3.4.2	Is Correctness Feasible?	73
3.4.3	Estimating the Efficiency of Methods	74
3.4.4	Big-O Notation	74
3.4.5	Getting Big-O Estimates Quickly	77
3.4.6	Trade-Offs	81
3.4.7	Run-Time Analysis	83
3.4.8	Randomness	84
	Lab 9: Timing and Randomness	86
3.4.9	Casting	86
3.5	Program Maintenance	87
	Summary	88
	Exercises	88
	Programming Project 3.1: Further Expansion of the Linked Class	91

CHAPTER 4

Recursion 93

4.1	Introduction	94
4.2	Factorials	94
4.2.1	Execution Frames	96
4.3	Decimal-to-Binary	99
	Lab 10: Fibonacci Numbers	102
4.4	Towers of Hanoi	103
4.4.1	A Recurrence Relation	110
4.5	Backtracking	112
4.5.1	An A-Maze-ing Application	117
4.6	Binary Search	125
	Lab 11: Iterative Binary Search	134
4.7	Generating Permutations	135
4.7.1	Estimating the Time and Space Requirements	142
4.8	Indirect Recursion	144
4.9	The Cost of Recursion	145
	Summary	146

	Exercises	147
	Programming Project 4.1: An Iterative Version of Towers of Hanoi	154
	Programming Project 4.2: The Eight-Queens Problem	156
	Programming Project 4.3: A Knight's Tour	158

CHAPTER 5

Vectors and Deques 163

5.1	The Standard Template Library	164
5.2	Vectors	165
5.2.1	Method Interfaces for the vector Class	166
5.2.2	Vector Iterators	174
5.2.3	Comparison of Vectors to Other Containers	176
5.2.4	Possible Fields of the vector Class	177
5.2.5	An Implementation of the vector Class	177
	Lab 12: More Implementation Details of the vector Class	184
5.3	A Vector Application: High-Precision Arithmetic	184
5.3.1	Design of the very_long_int Class	185
5.3.2	An Implementation of the very_long_int Class	187
	Lab 13: Expanding the very_long_int Class	190
5.4	Deque	190
5.4.1	Fields and Implementation of the deque Class	192
	Lab 14: More Details of Hewlett-Packard's deque Class	198
5.5	A Deque Application: Very Long Integers	198
	Summary	199
	Exercises	199
	Programming Project 5.1: Extending the very_long_int Class	203
	Programming Project 5.2: An Alternative Implementation of the deque Class	204

CHAPTER 6

Lists 205

- 6.1 Lists 206
 - 6.1.1 Method Interfaces for the list Class 207
 - 6.1.2 Iterator Interfaces 211
 - 6.1.3 Differences between List Methods and Vector or Deque Methods 213
 - 6.1.4 Fields and Implementation of the list Class 214
 - 6.1.5 Storage of list Nodes 221
- Lab 15:** More Implementation Details for the list Class 224
- Lab 16:** Timing the Sequence Containers 224
- Lab 17:** Iterators, Part 2 225
- 6.1.6 Alternative Implementations of the list Class 226
- 6.2 List Application: A Line Editor 228
 - 6.2.1 Design of the Editor Class 232
 - 6.2.2 Implementation of the Editor Class 234
- Summary 240
- Exercises 241
- Programming Project 6.1: Extending the Editor Class 244
- Programming Project 6.2: An Alternate Design and Implementation of the list Class 251

CHAPTER 7

Queues and Stacks 253

- 7.1 Queues 254
 - 7.1.1 Method Interfaces for the queue Class 255
 - 7.1.2 Using the queue Class 257
 - 7.1.3 Container Adaptors 259
 - 7.1.4 A Contiguous Design 260
- 7.2 Computer Simulation 261
- 7.3 A Queue Application: A Simulated Car Wash 264
 - 7.3.1 Program Design 266
 - 7.3.2 Implementation of the carWash Class 268
 - 7.3.3 Analysis of the carWash Methods 272

7.3.4 Randomizing the Arrival Times 272

Lab 18: Randomizing the Arrival Times 274

- 7.4 Stacks 274
 - 7.4.1 Method Interfaces for the stack Class 274
 - 7.4.2 Using the stack Class 275
 - 7.4.3 The stack Class is a Container Adaptor 276
- 7.5 Stack Application 1: How Recursion Is Implemented 277
- 7.6 Stack Application 2: Converting Infix to Postfix 285
 - 7.6.1 Postfix Notation 286
 - 7.6.2 Transition Matrix 289
 - 7.6.3 Tokens 290
- Lab 19:** Converting from Infix to Postfix 291
- 7.6.4 Prefix Notation 292
- Summary 295
- Exercises 295
- Programming Project 7.1: Extending the Car Wash Application 298
- Programming Project 7.2: Evaluating a Condition 300
- Programming Project 7.3: An Iterative Maze Search 304
- Programming Project 7.4: An Alternate Design of the queue Class 305

CHAPTER 8

Binary Trees and Binary Search Trees 307

- 8.1 Definition and Properties 308
 - 8.1.1 The Binary Tree Theorem 314
 - 8.1.2 External Path Length 317
 - 8.1.3 Traversals of a Binary Tree 318
- 8.2 Binary Search Trees 324
 - 8.2.1 The BinSearchTree Class 325
 - 8.2.2 The Iterator Class for the BinSearchTree Class 327
 - 8.2.3 Fields and Implementation of the BinSearchTree Class 329

- 8.2.4 *Recursive Methods?* 334
- 8.2.5 *BinSearchTree Iterators* 342

Lab 20: *The Average Height of a BinSearchTree* 345

- Summary 345
- Exercises 346
- Programming Project 8.1: Alternative Implementation of the BinSearchTree Class 350

CHAPTER 9

AVL Trees 353

- 9.1 *Balanced Binary Search Trees* 354
- 9.2 *Rotations* 354
- 9.3 *AVL Trees* 358
 - 9.3.1 *The Height of an AVL Tree* 360
 - 9.3.2 *Function Objects* 361
 - Lab 21:** *More on Function Objects* 364
 - 9.3.3 *The AVLTree Class* 364
 - 9.3.4 *The fixAfterinsert Method* 367
 - 9.3.5 *Correctness of the insert Method* 377
- 9.4 *AVL Tree Application: A Simple Spell-Checker* 380
- Summary 383
- Exercises 384
- Programming Project 9.1: The erase Method in the AVLTree Class 388
- Programming Project 9.2: Enhancing the SpellChecker Project 389

CHAPTER 10

Red-Black Trees 391

- 10.1 *Red-Black Trees* 392
 - 10.1.1 *The Height of a Red-Black Tree* 394
 - 10.1.2 *Hewlett-Packard's rb_tree Class* 399
 - 10.1.3 *The insert Method in the rb_tree Class* 402

Lab 22: *A Red-Black Tree Insertion with All Three Cases* 408

10.1.4 *The erase Method* 408

Lab 23: *A Call to erase in Which All Four Cases Apply* 421

- 10.2 *The Standard Template Library's Associative Containers* 422
 - 10.2.1 *The set Class* 422
- 10.3 *Set Application: Spell-Checker, Revisited* 425
 - 10.3.1 *The multiset Class* 425
 - Lab 24:** *More on the set and multiset Classes* 427
 - 10.3.2 *The map Class* 427
 - 10.3.3 *The multimap Class* 431
 - Lab 25:** *More on the map and multimap Classes* 431
- Summary 432
- Exercises 432
- Programming Project 10.1: A Simple Thesaurus 436
- Programming Project 10.2: Building a Concordance 437

CHAPTER 11

Priority Queues and Heaps 439

- 11.1 *Introduction* 440
 - 11.1.1 *The priority_queue Class* 440
 - 11.1.2 *Fields and Implementation of the priority_queue Class* 443
 - 11.1.3 *Heaps* 444
 - Lab 26:** *Incorporating Fairness in Priority Queues* 454
 - 11.1.4 *Alternative Designs and Implementations of the priority_queue Class* 454
- 11.2 *Application of Priority Queues: Huffman Codes* 456
 - 11.2.1 *Design of the huffman Class* 461
 - 11.2.2 *Implementation of the huffman Class* 463
- Summary 469
- Exercises 469

Programming Project 11.1: Decoding a
Message 473

CHAPTER 12

Sorting 477

12.1 Introduction 478
 12.1.1 Insertion Sort 478
 12.2 How Fast Can We Sort? 481
 12.3 Fast Sorts 483
 12.3.1 Tree Sort 483
 12.3.2 Heap Sort 485
 12.3.3 Merge Sort 487
 12.3.4 Quick Sort 493
 12.3.5 Divide-and-Conquer Algorithms 500
 Lab 27: Run-Times for Sorting Algorithms 500
 Summary 501
 Exercises 501
 Programming Project 12.1: Sorting a File 509

CHAPTER 13

Searching and the Hash Classes 513

13.1 A Framework to Analyze Searching 514
 13.2 Review of Searching 514
 13.2.1 Sequential Search 514
 13.2.2 Binary Search 515
 13.2.3 Red-Black Trees 517
 13.3 The hash_map Class 517
 13.3.1 Fields in the hash_map Class 519
 13.3.2 Hashing 519
 13.3.3 Chaining 522
 13.3.4 Fields and Implementation of the iterator
 Class 530
 13.3.5 Implementation of the hash_map
 Class 532
 13.3.6 Analysis of Chained Hashing 535

 13.3.7 The value_type Class 537

 13.3.8 Applications 538

Lab 28: Timing a hash_map 539

13.4 The hash_set Class 540
 13.5 Open-Address Hashing 540
 13.5.1 The erase Method 543
 13.5.2 Primary Clustering 549
 13.5.3 Double Hashing 550
 13.5.4 Analysis of Open-Address Hashing 554
 Summary 557
 Exercises 558
 Programming Project 13.1: A Run-Time
 Comparison of Chaining and Double Hashing
 in Building a Symbol Table 561

CHAPTER 14

Graphs, Trees, and Networks 563

14.1 Undirected Graphs 564
 14.2 Directed Graphs 567
 14.3 Trees 568
 14.4 Networks 569
 14.5 Graph Algorithms 571
 14.5.1 Iterators 571
 14.5.2 Connectedness 578
 14.5.3 Finding a Minimum Spanning Tree 579
 14.5.4 Finding the Shortest Path through a
 Network 584
 14.6 Developing a Network Class 588
 14.7 The network Class 589
 14.7.1 Fields in the network Class 591
 14.7.2 Implementation of the network
 Class 593
 14.7.3 Implementation of Edge-Related
 Methods 594
 14.7.4 Implementation of Global Methods 596
 14.7.5 The get_minimum_spanning_tree
 Method 599
 14.7.6 The get_shortest_path Method 601

14.7.7	<i>Time Estimates for the Network Methods</i>	604
Lab 29:	<i>The Traveling Salesperson Problem</i>	604
14.7.8	<i>An Alternative Design and Implementation of the network Class</i>	605
14.8	Backtracking through a Network	607
	Summary	610
	Exercises	610
	Programming Project 14.1: Completing the Adjacency-Matrix Implementation	614
	Programming Project 14.2: Backtracking through a Network	615

APPENDIX 1 **Mathematical Background 619**

A1.1	Introduction	619
A1.2	Functions and Sequences	619
A1.3	Sums and Products	620
A1.4	Logarithms	621
A1.5	Mathematical Induction	623
A1.6	Induction and Recursion	630

APPENDIX 2 **The string Class 633**

A2.1	Introduction	633
A2.2	Declaration of the string Class	633
A2.2.1	<i>Constructors</i>	634
A2.2.2	<i>Operators</i>	635
A2.2.3	<i>Nonmember Functions</i>	635
A2.2.4	<i>Methods That Are Neither Constructors nor Operators</i>	638
A2.2.5	<i>A String Processing Program</i>	642
A2.3	Fields and Implementation of the string Class	644

APPENDIX 3 **Polymorphism 647**

A3.1	Introduction	647
A3.2	The Importance of Polymorphism	648
A3.3	Dynamic Binding	649

References 651

Index 653