

# Contents

<b>Introduction</b>	<b>xxi</b>
<b>Who This Book Is For</b>	<b>xxi</b>
<b>What This Book Covers</b>	<b>xxi</b>
<b>How This Book Is Structured</b>	<b>xxii</b>
<b>What You Need to Use This Book</b>	<b>xxiii</b>
<b>Conventions</b>	<b>xxiii</b>
<b>Source Code</b>	<b>xxiv</b>
<b>Errata</b>	<b>xxiv</b>
<b>p2p.wrox.com</b>	<b>xxiv</b>
<b>Part I: Introduction to Professional C++</b>	
<b>Chapter 1: A Crash Course in C++</b>	<b>1</b>
<b>The Basics of C++</b>	<b>1</b>
The Obligatory Hello, World	2
Namespaces	4
Variables	6
Operators	8
Types	10
Conditionals	12
Loops	14
Arrays	15
Functions	16
Those Are the Basics	17
<b>Diving Deeper into C++</b>	<b>18</b>
Pointers and Dynamic Memory	18
Strings in C++	21
References	23
Exceptions	23
The Many Uses of const	25
<b>C++ as an Object-Oriented Language</b>	<b>26</b>
Declaring a Class	26
<b>Your First Useful C++ Program</b>	<b>29</b>
An Employee Records System	29
The Employee Class	29

# Contents

---

The Database Class	34
The User Interface	38
Evaluating the Program	41
<b>Summary</b>	<b>41</b>
<b>Chapter 2: Designing Professional C++ Programs</b>	<b>43</b>
<b>What Is Programming Design?</b>	<b>44</b>
<b>The Importance of Programming Design</b>	<b>44</b>
<b>What's Different about C++ Design?</b>	<b>46</b>
<b>Two Rules for C++ Design</b>	<b>47</b>
Abstraction	47
Reuse	49
<b>Designing a Chess Program</b>	<b>50</b>
Requirements	51
Design Steps	51
<b>Summary</b>	<b>56</b>
<b>Chapter 3: Designing with Objects</b>	<b>57</b>
<b>An Object-Oriented View of the World</b>	<b>57</b>
Am I Thinking Procedurally?	57
The Object-Oriented Philosophy	58
Living in a World of Objects	61
Object Relationships	63
Abstraction	73
<b>Summary</b>	<b>76</b>
<b>Chapter 4: Designing with Libraries and Patterns</b>	<b>77</b>
<b>Reusing Code</b>	<b>77</b>
A Note on Terminology	78
Deciding Whether or Not to Reuse Code	78
Strategies for Reusing Code	81
Bundling Third-Party Applications	85
Open-Source Libraries	86
The C++ Standard Library	87
<b>Designing with Patterns and Techniques</b>	<b>101</b>
Design Techniques	101
Design Patterns	102
<b>Summary</b>	<b>103</b>

---

<b>Chapter 5: Designing for Reuse</b>	<b>105</b>
<b>The Reuse Philosophy</b>	<b>106</b>
<b>How to Design Reusable Code</b>	<b>106</b>
Use Abstraction	107
Structure Your Code for Optimal Reuse	108
Design Usable Interfaces	112
Reconciling Generality and Ease of Use	117
<b>Summary</b>	<b>118</b>
<b>Chapter 6: Maximizing Software-Engineering Methods</b>	<b>119</b>
<b>The Need for Process</b>	<b>119</b>
<b>Software Life-Cycle Models</b>	<b>120</b>
The Stagewise and Waterfall Models	121
The Spiral Method	123
The Rational Unified Process	126
<b>Software-Engineering Methodologies</b>	<b>127</b>
Extreme Programming (XP)	128
Software Triage	132
<b>Building Your Own Process and Methodology</b>	<b>132</b>
Be Open to New Ideas	132
Bring New Ideas to the Table	132
Recognize What Works and What Doesn't Work	133
Don't Be a Renegade	133
<b>Summary</b>	<b>133</b>
<b>Part II: C++ Coding the Professional Way</b>	
<b>Chapter 7: Coding with Style</b>	<b>135</b>
<b>The Importance of Looking Good</b>	<b>135</b>
Thinking Ahead	135
Keeping It Clear	136
Elements of Good Style	136
<b>Documenting Your Code</b>	<b>136</b>
Reasons to Write Comments	136
Commenting Styles	140
Comments in This Book	145

# Contents

---

<b>Decomposition</b>	<b>145</b>
Decomposition through Refactoring	147
Decomposition by Design	147
Decomposition in This Book	148
<b>Naming</b>	<b>148</b>
Choosing a Good Name	148
Naming Conventions	148
<b>Using Language Features with Style</b>	<b>151</b>
Use Constants	151
Take Advantage of const Variables	151
Use References Instead of Pointers	151
Use Custom Exceptions	152
<b>Formatting</b>	<b>152</b>
The Curly Brace Alignment Debate	153
Coming to Blows over Spaces and Parentheses	154
Spaces and Tabs	154
<b>Stylistic Challenges</b>	<b>155</b>
<b>Summary</b>	<b>155</b>
<b>Chapter 8: Gaining Proficiency with Classes and Objects</b>	<b>157</b>
<hr/>	
<b>Introducing the Spreadsheet Example</b>	<b>157</b>
<b>Writing Classes</b>	<b>158</b>
Class Definitions	158
Defining Methods	161
Using Objects	164
<b>Object Life Cycles</b>	<b>165</b>
Object Creation	165
Object Destruction	176
Assigning to Objects	177
Distinguishing Copying from Assignment	180
<b>Summary</b>	<b>182</b>
<b>Chapter 9: Mastering Classes and Objects</b>	<b>183</b>
<hr/>	
<b>Dynamic Memory Allocation in Objects</b>	<b>183</b>
The Spreadsheet Class	184
Freeing Memory with Destructors	186
Handling Copying and Assignment	186
<b>Different Kinds of Data Members</b>	<b>194</b>
Static Data Members	195
Const Data Members	196
Reference Data Members	198
Const Reference Data Members	199

---

<b>More about Methods</b>	<b>199</b>
Static Methods	199
Const Methods	200
Method Overloading	202
Default Parameters	203
Inline Methods	204
<b>Nested Classes</b>	<b>206</b>
<b>Friends</b>	<b>208</b>
<b>Operator Overloading</b>	<b>209</b>
Implementing Addition	209
Overloading Arithmetic Operators	212
Overloading Comparison Operators	215
Building Types with Operator Overloading	216
<b>Pointers to Methods and Members</b>	<b>217</b>
<b>Building Abstract Classes</b>	<b>218</b>
Using Interface and Implementation Classes	218
<b>Summary</b>	<b>221</b>
<b>Chapter 10: Discovering Inheritance Techniques</b>	<b>223</b>
<b>Building Classes with Inheritance</b>	<b>224</b>
Extending Classes	224
Overriding Methods	227
<b>Inheritance for Reuse</b>	<b>230</b>
The WeatherPrediction Class	230
Adding Functionality in a Subclass	231
Replacing Functionality in a Subclass	233
<b>Respect Your Parents</b>	<b>234</b>
Parent Constructors	234
Parent Destructors	235
Referring to Parent Data	237
Casting Up and Down	239
<b>Inheritance for Polymorphism</b>	<b>240</b>
Return of the Spreadsheet	240
Designing the Polymorphic Spreadsheet Cell	241
The Spreadsheet Cell Base Class	242
The Individual Subclasses	243
Leveraging Polymorphism	245
Future Considerations	246
<b>Multiple Inheritance</b>	<b>248</b>
Inheriting from Multiple Classes	248
Naming Collisions and Ambiguous Base Classes	249

# Contents

---

<b>Interesting and Obscure Inheritance Issues</b>	<b>253</b>
Changing the Overridden Method's Characteristics	253
Special Cases in Overriding Methods	256
Copy Constructors and the Equals Operator	263
The Truth about Virtual	264
Runtime Type Facilities	267
Non-Public Inheritance	269
Virtual Base Classes	269
<b>Summary</b>	<b>270</b>
<b>Chapter 11: Writing Generic Code with Templates</b>	<b>271</b>
<b>Overview of Templates</b>	<b>272</b>
<b>Class Templates</b>	<b>273</b>
Writing a Class Template	273
How the Compiler Processes Templates	280
Distributing Template Code between Files	281
Template Parameters	282
Method Templates	285
Template Class Specialization	290
Subclassing Template Classes	293
Inheritance versus Specialization	295
<b>Function Templates</b>	<b>295</b>
Function Template Specialization	296
Function Template Overloading	297
Friend Function Templates of Class Templates	298
<b>Advanced Templates</b>	<b>299</b>
More about Template Parameters	299
Template Class Partial Specialization	307
Emulating Function Partial Specialization with Overloading	313
Template Recursion	314
<b>Summary</b>	<b>322</b>
<b>Chapter 12: Understanding C++ Quirks and Oddities</b>	<b>323</b>
<b>References</b>	<b>323</b>
Reference Variables	324
Reference Data Members	326
Reference Parameters	326
Reference Return Values	327
Deciding between References and Pointers	327

---

<b>Keyword Confusion</b>	<b>330</b>
The const Keyword	330
The static Keyword	333
Order of Initialization of Nonlocal Variables	336
<b>Types and Casts</b>	<b>337</b>
typedefs	337
Casts	338
<b>Scope Resolution</b>	<b>343</b>
<b>Header Files</b>	<b>343</b>
<b>C Utilities</b>	<b>345</b>
Variable-Length Argument Lists	345
Preprocessor Macros	347
<b>Summary</b>	<b>348</b>

## **Part III: Mastering Advanced Features of C++**

---

### **Chapter 13: Effective Memory Management** **349**

<b>Working with Dynamic Memory</b>	<b>349</b>
How to Picture Memory	350
Allocation and Deallocation	351
Arrays	353
Working with Pointers	360
<b>Array-Pointer Duality</b>	<b>362</b>
Arrays Are Pointers!	363
Not All Pointers Are Arrays!	364
<b>Dynamic Strings</b>	<b>365</b>
C-Style Strings	365
String Literals	366
The C++ string Class	367
<b>Low-Level Memory Operations</b>	<b>369</b>
Pointer Arithmetic	369
Custom Memory Management	370
Garbage Collection	370
Object Pools	371
Function Pointers	372
<b>Common Memory Pitfalls</b>	<b>374</b>
Underallocating Strings	374
Memory Leaks	374
Double-Deleting and Invalid Pointers	377
Accessing Out-of-Bounds Memory	378
<b>Summary</b>	<b>378</b>

## **Chapter 14: Demystifying C++ I/O** **379**

---

<b>Using Streams</b>	<b>379</b>
What Is a Stream, Anyway?	380
Stream Sources and Destinations	380
Output with Streams	380
Input with Streams	384
Input and Output with Objects	389
<b>String Streams</b>	<b>390</b>
<b>File Streams</b>	<b>392</b>
Jumping around with seek() and tell()	392
Linking Streams Together	395
<b>Bidirectional I/O</b>	<b>396</b>
<b>Internationalization</b>	<b>397</b>
Wide Characters	397
Non-Western Character Sets	398
Locales and Facets	398
<b>Summary</b>	<b>400</b>

## **Chapter 15: Handling Errors** **401**

---

<b>Errors and Exceptions</b>	<b>402</b>
What Are Exceptions, Anyway?	402
Why Exceptions in C++ Are a Good Thing	403
Why Exceptions in C++ Are a Bad Thing	404
Our Recommendation	404
<b>Exception Mechanics</b>	<b>404</b>
Throwing and Catching Exceptions	405
Exception Types	406
Throwing and Catching Multiple Exceptions	408
Uncaught Exceptions	411
Throw Lists	412
<b>Exceptions and Polymorphism</b>	<b>416</b>
The Standard Exception Hierarchy	416
Catching Exceptions in a Class Hierarchy	417
Writing Your Own Exception Classes	419
<b>Stack Unwinding and Cleanup</b>	<b>422</b>
Catch, Cleanup, and Rethrow	423
Use Smart Pointers	424

---

<b>Common Error-Handling Issues</b>	<b>424</b>
Memory Allocation Errors	424
Errors in Constructors	427
Errors in Destructors	428
<b>Putting It All Together</b>	<b>428</b>
<b>Summary</b>	<b>430</b>

## **Part IV: Ensuring Bug-Free Code**

---

### **Chapter 16: Overloading C++ Operators** **431**

---

<b>Overview of Operator Overloading</b>	<b>432</b>
Why Overload Operators?	432
Limitations to Operator Overloading	432
Choices in Operator Overloading	433
Operators You Shouldn't Overload	435
Summary of Overloadable Operators	435
<b>Overloading the Arithmetic Operators</b>	<b>438</b>
Overloading Unary Minus and Unary Plus	438
Overloading Increment and Decrement	439
<b>Overloading the Bitwise and Binary Logical Operators</b>	<b>441</b>
<b>Overloading the Insertion and Extraction Operators</b>	<b>441</b>
<b>Overloading the Subscripting Operator</b>	<b>443</b>
Providing Read-Only Access with operator[]	446
Non-Integral Array Indices	447
<b>Overloading the Function Call Operator</b>	<b>448</b>
<b>Overloading the Dereferencing Operators</b>	<b>449</b>
Implementing operator*	451
Implementing operator->	452
What in the World Is operator->* ?	452
<b>Writing Conversion Operators</b>	<b>453</b>
Ambiguity Problems with Conversion Operators	454
Conversions for Boolean Expressions	455
<b>Overloading the Memory Allocation and Deallocation Operators</b>	<b>457</b>
How new and delete Really Work	457
Overloading operator new and operator delete	459
Overloading operator new and operator delete with Extra Parameters	461
<b>Summary</b>	<b>463</b>

# Contents

---

## **Chapter 17: Writing Efficient C++** **465**

---

<b>Overview of Performance and Efficiency</b>	<b>465</b>
Two Approaches to Efficiency	466
Two Kinds of Programs	466
Is C++ an Inefficient Language?	466
<b>Language-Level Efficiency</b>	<b>467</b>
Handle Objects Efficiently	467
Don't Overuse Costly Language Features	471
Use Inline Methods and Functions	472
<b>Design-Level Efficiency</b>	<b>472</b>
Cache as Much as Possible	472
Use Object Pools	473
Use Thread Pools	479
<b>Profiling</b>	<b>479</b>
Profiling Example with gprof	479
<b>Summary</b>	<b>488</b>

## **Chapter 18: Developing Cross-Platform and Cross-Language Applications** **489**

---

<b>Cross-Platform Development</b>	<b>489</b>
Architecture Issues	490
Implementation Issues	492
Platform-Specific Features	493
<b>Cross-Language Development</b>	<b>494</b>
Mixing C and C++	494
Shifting Paradigms	495
Linking with C Code	498
Mixing Java and C++ with JNI	499
Mixing C++ with Perl and Shell Scripts	501
Mixing C++ with Assembly Code	504
<b>Summary</b>	<b>505</b>

## **Chapter 19: Becoming Adept at Testing** **507**

---

<b>Quality Control</b>	<b>507</b>
Whose Responsibility Is Testing?	508
The Life Cycle of a Bug	508
Bug-Tracking Tools	509
<b>Unit Testing</b>	<b>510</b>
Approaches to Unit Testing	511
The Unit Testing Process	512
Unit Testing in Action	515

---

<b>Higher-Level Testing</b>	<b>523</b>
Integration Tests	523
System Tests	525
Regression Tests	525
<b>Tips for Successful Testing</b>	<b>526</b>
<b>Summary</b>	<b>526</b>
<b>Chapter 20: Conquering Debugging</b>	<b>527</b>
<b>The Fundamental Law of Debugging</b>	<b>527</b>
<b>Bug Taxonomies</b>	<b>528</b>
<b>Avoiding Bugs</b>	<b>528</b>
<b>Planning for Bugs</b>	<b>528</b>
Error Logging	528
Debug Traces	530
Asserts	540
<b>Debugging Techniques</b>	<b>541</b>
Reproducing Bugs	541
Debugging Reproducible Bugs	542
Debugging Nonreproducible Bugs	543
Debugging Memory Problems	544
Debugging Multithreaded Programs	547
Debugging Example: Article Citations	548
Lessons from the ArticleCitations Example	559
<b>Summary</b>	<b>559</b>
<b>Chapter 21: Delving into the STL: Containers and Iterators</b>	<b>561</b>
<b>Containers Overview</b>	<b>562</b>
Requirements on Elements	562
Exceptions and Error Checking	563
Iterators	564
<b>Sequential Containers</b>	<b>565</b>
Vector	566
The vector<bool> Specialization	583
deque	584
list	584
<b>Container Adapters</b>	<b>588</b>
queue	588
priority_queue	591
stack	594

# Contents

---

<b>Associative Containers</b>	<b>595</b>
The pair Utility Class	595
map	596
multimap	604
set	608
multiset	610
<b>Other Containers</b>	<b>611</b>
Arrays as STL Containers	611
Strings as STL Containers	612
Streams as STL Containers	613
bitset	613
<b>Summary</b>	<b>618</b>

## Part V: Using Libraries and Patterns

### **Chapter 22: Mastering STL Algorithms and Function Objects** **619**

---

<b>Overview of Algorithms</b>	<b>620</b>
The find() and find_if() Algorithms	620
The accumulate() Algorithms	623
<b>Function Objects</b>	<b>624</b>
Arithmetic Function Objects	624
Comparison Function Objects	625
Logical Function Objects	627
Function Object Adapters	627
Writing Your Own Function Objects	630
<b>Algorithm Details</b>	<b>631</b>
Utility Algorithms	632
Nonmodifying Algorithms	633
Modifying Algorithms	639
Sorting Algorithms	643
Set Algorithms	646
<b>Algorithms and Function Objects Example: Auditing Voter Registrations</b>	<b>648</b>
The Voter Registration Audit Problem Statement	648
The auditVoterRolls() Function	648
The getDuplicates() Function	649
The RemoveNames Functor	650
The NameInList Functor	651
Testing the auditVoterRolls() Function	652
<b>Summary</b>	<b>653</b>

---

<b>Chapter 23: Customizing and Extending the STL</b>	<b>655</b>
<b>Allocators</b>	<b>656</b>
<b>Iterator Adapters</b>	<b>656</b>
Reverse Iterators	656
Stream Iterators	657
Insert Iterators	658
<b>Extending the STL</b>	<b>660</b>
Why Extend the STL?	660
Writing an STL Algorithm	660
Writing an STL Container	662
<b>Summary</b>	<b>691</b>
<b>Chapter 24: Exploring Distributed Objects</b>	<b>693</b>
<b>The Appeal of Distributed Computing</b>	<b>693</b>
Distribution for Scalability	693
Distribution for Reliability	694
Distribution for Centrality	694
Distributed Content	695
Distributed versus Networked	695
<b>Distributed Objects</b>	<b>696</b>
Serialization and Marshalling	696
Remote Procedure Calls	700
<b>CORBA</b>	<b>702</b>
Interface Definition Language	702
Implementing the Class	704
Using the Objects	706
<b>XML</b>	<b>709</b>
A Crash Course in XML	709
XML as a Distributed Object Technology	712
Generating and Parsing XML in C++	712
XML Validation	721
Building a Distributed Object with XML	723
SOAP (Simple Object Access Protocol)	726
<b>Summary</b>	<b>728</b>
<b>Chapter 25: Incorporating Techniques and Frameworks</b>	<b>729</b>
<b>“I Can Never Remember How to . . .”</b>	<b>730</b>
. . . Write a Class	730
. . . Subclass an Existing Class	731

# Contents

---

. . . Throw and Catch Exceptions	732
. . . Read from a File	733
. . . Write to a File	734
. . . Write a Template Class	734
<b>There Must Be a Better Way</b>	<b>736</b>
Smart Pointers with Reference Counting	736
Double Dispatch	741
Mix-In Classes	747
<b>Object-Oriented Frameworks</b>	<b>750</b>
Working with Frameworks	750
The Model-View-Controller Paradigm	750
<b>Summary</b>	<b>752</b>
<b>Chapter 26: Applying Design Patterns</b>	<b>753</b>
<b>The Singleton Pattern</b>	<b>754</b>
Example: A Logging Mechanism	754
Implementation of a Singleton	754
Using a Singleton	759
<b>The Factory Pattern</b>	<b>760</b>
Example: A Car Factory Simulation	760
Implementation of a Factory	762
Using a Factory	764
Other Uses of Factories	766
<b>The Proxy Pattern</b>	<b>766</b>
Example: Hiding Network Connectivity Issues	766
Implementation of a Proxy	767
Using a Proxy	767
<b>The Adapter Pattern</b>	<b>768</b>
Example: Adapting an XML Library	768
Implementation of an Adapter	768
Using an Adapter	772
<b>The Decorator Pattern</b>	<b>773</b>
Example: Defining Styles in Web Pages	773
Implementation of a Decorator	774
Using a Decorator	775
<b>The Chain of Responsibility Pattern</b>	<b>776</b>
Example: Event Handling	776
Implementation of a Chain of Responsibility	777
Using a Chain of Responsibility	778

<b>The Observer Pattern</b>	<b>778</b>
Example: Event Handling	778
Implementation of an Observer	778
Using an Observer	780
<b>Summary</b>	<b>781</b>
<hr/> <b>Appendix A: C++ Interviews</b>	<hr/> <b>783</b>
<b>Chapter 1: A Crash Course in C++</b>	<b>783</b>
<b>Chapter 2: Designing Professional C++ Programs</b>	<b>784</b>
<b>Chapter 3: Designing with Objects</b>	<b>785</b>
<b>Chapter 4: Designing with Libraries and Patterns</b>	<b>786</b>
<b>Chapter 5: Designing for Reuse</b>	<b>787</b>
<b>Chapter 6: Maximizing Software Engineering Methods</b>	<b>787</b>
<b>Chapter 7: Coding with Style</b>	<b>788</b>
<b>Chapters 8 and 9: Classes and Objects</b>	<b>789</b>
<b>Chapter 10: Discovering Inheritance Techniques</b>	<b>792</b>
<b>Chapter 11: Writing Generic Code with Templates</b>	<b>793</b>
<b>Chapter 12: Understanding C++ Quirks and Oddities</b>	<b>793</b>
<b>Chapter 13: Effective Memory Management</b>	<b>794</b>
<b>Chapter 14: Demystifying C++ I/O</b>	<b>795</b>
<b>Chapter 15: Handling Errors</b>	<b>796</b>
<b>Chapter 16: Overloading C++ Operators</b>	<b>796</b>
<b>Chapter 17: Writing Efficient C++</b>	<b>797</b>
<b>Chapter 18: Developing Cross-Platform and Cross-Language Applications</b>	<b>798</b>
<b>Chapter 19: Becoming Adept at Testing</b>	<b>798</b>
<b>Chapter 20: Conquering Debugging</b>	<b>799</b>
<b>Chapters 21, 22, and 23: The Standard Template Library</b>	<b>799</b>
<b>Chapter 24: Exploring Distributed Objects</b>	<b>800</b>
<b>Chapter 25: Incorporating Techniques and Frameworks</b>	<b>801</b>
<b>Chapter 26: Applying Design Patterns</b>	<b>801</b>
<hr/> <b>Appendix B: Annotated Bibliography</b>	<hr/> <b>803</b>
<b>C++</b>	<b>803</b>
Beginning C++	803
General C++	804
I/O Streams	805
The C++ Standard Library	805
C++ Templates	806

# Contents

---

<b>C</b>	<b>806</b>
<b>Integrating C++ and Other Languages</b>	<b>806</b>
<b>Algorithms and Data Structures</b>	<b>807</b>
<b>Open-Source Software</b>	<b>807</b>
<b>Software-Engineering Methodology</b>	<b>807</b>
<b>Programming Style</b>	<b>808</b>
<b>Computer Architecture</b>	<b>809</b>
<b>Efficiency</b>	<b>809</b>
<b>Testing</b>	<b>809</b>
<b>Debugging</b>	<b>809</b>
<b>Distributed Objects</b>	<b>810</b>
CORBA	810
XML and SOAP	810
<b>Design Patterns</b>	<b>811</b>
<b>Index</b>	<b>813</b>

---