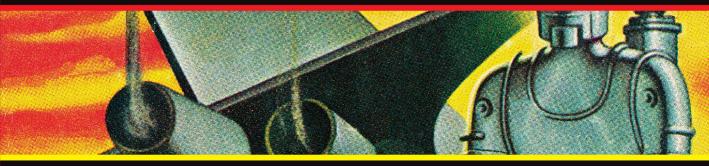


PROFESSIONAL BTH EDITION



MARC GREGOIRE



PROFESSIONAL

C++

Sixth Edition

Marc Gregoire

WILEY

Copyright © 2024 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey. Published simultaneously in Canada and the United Kingdom.

ISBNs: 9781394193172 (Paperback), 9781394193196 (ePDF), 9781394193189 (ePub)

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permission Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at www.wiley.com/go/permission.

Trademarks: WILEY and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at www.wiley.com.

Library of Congress Control Number: 2023948608

Cover image: © CSA-Printstock/Getty Images Cover design: Wiley Dedicated to my amazing parents and brother, whose continuous support and patience help me in tackling such a big project as writing this book.

ABOUT THE AUTHOR

MARC GREGOIRE is a software architect from Belgium. He graduated from the University of Leuven, Belgium, with a degree in "Burgerlijk ingenieur in de computer wetenschappen" (equivalent to a master of science in engineering in computer science). The year after, he received an advanced master's degree in artificial intelligence, *cum laude*, at the same university. After his studies, Marc started working for a software consultancy company called Ordina Belgium. As a consultant, he worked for Siemens and Nokia Siemens Networks on critical 2G and 3G software running on Solaris for telecom operators. This required working in international teams stretching from South America and the United States to Europe, the Middle East, Africa, and Asia. Now, Marc is a software project manager and software architect at Nikon Metrology (industry.nikon.com), a division of Nikon and a leading provider of precision optical instruments, X-ray machines, and metrology solutions for X-ray, CT, and 3-D geometric inspection.

His main expertise is C++. He has experience with developing C++ programs running 24/7 on Windows and Linux platforms: for example, KNX/EIB home automation software. In addition to C++, Marc also likes C#.

Since April 2007, he has received the annual Microsoft MVP (Most Valuable Professional) award for his Visual C++ expertise.

Marc is the founder of the Belgian C++ Users Group (becpp.org), co-author of C++ Standard Library Quick Reference 1st and 2nd editions (Apress 2016 and 2019), a technical editor for numerous books for several publishers, and a regular speaker at the CppCon C++ conference (cppcon.org). He maintains a blog at www.nuonsoft.com/blog and is passionate about traveling and gastronomic restaurants.

ABOUT THE TECHNICAL EDITORS

BRADLEY JONES has programmed in a variety of languages and tools ranging from C to Unity on platforms ranging from Windows to mobile and including the web as well as a little bit of virtual reality and embedded devices just for fun. In addition to programming, he has authored books on C, C++, C#, Windows, the web, and many more technical topics and a few nontechnical topics. Bradley is the owner of Lots of Software, LLC, and has been recognized in the industry as a community influencer as well as has been recognized as a Microsoft MVP, a CODiE Judge, an international technology speaker, a bestselling technical author, and more.

ARTHUR O'DWYER is a professional C++ trainer, software engineer, author, and WG21 committee member. He authored *Mastering the* C++17 *STL* (Packt Publishing, 2017), founded CppCon's "Back to Basics" track (2019), implemented libc++'s <memory_resource> header (2022), and is responsible for the simplified "implicit move" semantics in C++20 and C++23. He and his wife live in New York.

ACKNOWLEDGMENTS

I THANK THE JOHN WILEY & SONS editorial and production teams for their support. A special thankyou to Jim Minatel, executive editor at Wiley, for giving me a chance to write this sixth edition; Pete Gaughan, senior managing editor; Ashirvad Moses Thyagarajan, managing editor; Kathryn Hogan, PhD, project manager; Archana Pragash, content refinement specialist; and Kim Wimpsett, copyeditor.

A special thank you to technical editors Bradley Jones and Arthur O'Dwyer for checking the technical accuracy of the book. Their feedback and numerous contributions have strengthened this book and are greatly appreciated.

Of course, the support and patience of my parents and my brother were very important in finishing this book. I would also like to express my sincere gratitude to my employer, Nikon Metrology, for supporting me during this project.

Finally, I thank you, the reader, for supporting me over all these years and across numerous editions with this approach to professional C++ software development.

-Marc Gregoire

CONTENTS

INTRODUCTION

PART I: INTRODUCTION TO PROFESSIONAL C++

CHAPTER 1: A CRASH COURSE IN C++ AND THE STANDARD LIBRARY 3

C++ Crash Course	4
The Obligatory "Hello, World" Program	4
Comments	5
Importing Modules	5
How the Compiler Processes Your Source Code	6
Preprocessor Directives	6
The main() Function	7
Printing Text	7
I/O Streams	7
Returning from a Function	9
Namespaces	9
Nested Namespace	11
Namespace Alias	11
Literals	11
Variables	12
Numerical Limits	15
Zero Initialization	15
Casting	16
Floating-Point Numbers	16
Operators	18
Enumerations	21
Old-Style Enumerations	22
Structs	23
Conditional Statements	24
if/else Statements	24
switch Statements	25
The Conditional Operator	27
Logical Evaluation Operators	27
Three-Way Comparisons	29
Functions	30

Function Return Type Deduction	31
Current Function's Name	32
Function Overloading	32
Attributes	32
[[nodiscard]]	33
[[maybe_unused]]	33
[[noreturn]]	34
[[deprecated]]	34
[[likely]] and [[unlikely]]	35
[[assume]]	35
C-Style Arrays	36
std::array	37
std::vector	38
std::pair	39
std::optional	40
Structured Bindings	41
Loops	41
The while Loop	41
The do/while Loop	42
The for Loop	42
The Range-Based for Loop	42
Initializer Lists	43
Strings in C++	43
C++ as an Object-Oriented Language	44
Defining Classes	44
Using Classes	47
Scope Resolution	47
Uniform Initialization	48
Designated Initializers	51
Pointers and Dynamic Memory	52
The Stack and the Free Store	52
Working with Pointers	53
Dynamically Allocated Arrays	54
Null Pointer Constant	55
The Use of const	56
const as a Qualifier for a Type	56
const Member Functions	58
References	59
Reference Variables	59
Reference Data Members	62
Reference Parameters	62

Reference Return Values	65
Deciding Between References and Pointers	65
const_cast()	69
Exceptions	70
Type Aliases	71
typedefs	72
Type Inference	72
The auto Keyword	72
The decltype Keyword	75
The Standard Library	75
Your First Bigger C++ Program	76
An Employee Records System	76
The Employee Class	76
Employee.cppm	76
Employee.cpp	78
EmployeeTest.cpp	79
The Database Class	80
Database.cppm	80
Database.cpp	81
DatabaseTest.cpp	82
The User Interface	82
Evaluating the Program	85
Summary	85
Exercises	85
CHAPTER 2: WORKING WITH STRINGS AND STRING VIEWS	87
Dynamic Strings	88
C-Style Strings	88
String Literals	90
Raw String Literals	90
The C++ std::string Class	92
What Is Wrong with C-Style Strings?	92
Using the std::string Class	92
std::string Literals	95
CTAD with std::vector and Strings	96
Numeric Conversions	96
High-Level Numeric Conversions	96
Low-Level Numeric Conversions	98
The std::string_view Class	100
std::string_view and Temporary Strings	102
std::string_view Literals	102

Nonstandard Strings	103
Formatting and Printing Strings	103
Format Strings	104
Argument Indices	105
Printing to Different Destinations	106
Compile-Time Verification of Format Strings	106
Non-Compile-Time Constant Format Strings	106
Handling Errors in Non-Compile-Time Constant Format Strings	107
Format Specifiers	107
width	108
[fill]align	108
sign	109
#	109
type	109
precision	110
0	111
L	111
Formatting Escaped Characters and Strings	111
Formatting Ranges	112
Support for Custom Types	114
Summary	117
Exercises	117
Exercises CHAPTER 3: CODING WITH STYLE	117 119
CHAPTER 3: CODING WITH STYLE	
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good	119
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead	119 119
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style	119 119 120
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead	119 119 120 120
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments	119 119 120 120 120
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments Commenting to Explain Usage	119 119 120 120 120 120 120
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments	119 119 120 120 120 120 120 120
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments Commenting to Explain Usage Commenting to Explain Complicated Code	119 119 120 120 120 120 120 120 122
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments Commenting to Explain Usage Commenting to Explain Complicated Code Commenting to Convey Meta-information	119 119 120 120 120 120 120 120 122 124
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments Commenting to Explain Usage Commenting to Explain Complicated Code Commenting to Convey Meta-information Copyright Comment	119 120 120 120 120 120 120 120 122 124 125
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments Commenting to Explain Usage Commenting to Explain Complicated Code Commenting to Convey Meta-information Copyright Comment Commenting Styles	119 120 120 120 120 120 120 120 122 124 125 125
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments Commenting to Explain Usage Commenting to Explain Complicated Code Commenting to Convey Meta-information Copyright Comment Commenting Styles Commenting Every Line	119 120 120 120 120 120 120 122 124 125 125 125
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments Commenting to Explain Usage Commenting to Explain Complicated Code Commenting to Convey Meta-information Copyright Comment Commenting Styles Commenting Every Line Prefix Comments	119 120 120 120 120 120 120 122 124 125 125 125 125 126
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments Commenting to Explain Usage Commenting to Explain Complicated Code Commenting to Convey Meta-information Copyright Comment Commenting Styles Commenting Every Line Prefix Comments Fixed-Format Comments	119 120 120 120 120 120 120 122 124 125 125 125 125 125 126 127 129 129
CHAPTER 3: CODING WITH STYLE The Importance of Looking Good Thinking Ahead Elements of Good Style Documenting Your Code Reasons to Write Comments Commenting to Explain Usage Commenting to Explain Complicated Code Commenting to Convey Meta-information Copyright Comment Commenting Styles Commenting Every Line Prefix Comments Fixed-Format Comments Ad Hoc Comments	119 120 120 120 120 120 120 122 124 125 125 125 125 125 126 127 129

Decomposition by Design	131
Decomposition in This Book	131
Naming	132
Choosing a Good Name	132
Naming Conventions	133
Counters	133
Prefixes	133
Hungarian Notation	134
Getters and Setters	134
Capitalization	134
Namespaced Constants	134
Using Language Features with Style	135
Use Constants	135
Use References Instead of Pointers	136
Use Custom Exceptions	136
Formatting	137
The Curly Brace Alignment Debate	137
Coming to Blows over Spaces and Parentheses	138
Spaces, Tabs, and Line Breaks	139
Stylistic Challenges	139
Summary	140
Exercises	140

PART II: PROFESSIONAL C++ SOFTWARE DESIGN

CHAPTER 4: DESIGNING PROFESSIONAL C++ PROGRAMS	145
What is Programming Design?	146
The Importance of Programming Design	147
Designing For C++	149
Two Rules for Your Own C++ Designs	150
Abstraction	150
Benefiting from Abstraction	150
Incorporating Abstraction in Your Design	151
Reuse	152
Writing Reusable Code	153
Reusing Designs	153
Reusing Existing Code	154
A Note on Terminology	155
Deciding Whether to Reuse Code or Write It Yourself	156
Advantages to Reusing Code	156

CONTENTS

Disadvantages to Reusing Code Putting It Together to Make a Decision	157 158
Understand the Capabilities and Limitations	158
Understand the Learning Cost	159
Understand the Performance	159
Understand Platform Limitations	162
Understand Licensing	162
Understand Support and Know Where to Find Help	162
Prototype	163
Open-Source Libraries	163
The C++ Standard Library	165
Designing a Chess Program	166
Requirements	166
Design Steps	167
Divide the Program into Subsystems	167
Choose Threading Models	169
Specify Class Hierarchies for Each Subsystem	170
Specify Classes, Data Structures, Algorithms, and Patterns	
for Each Subsystem	170
Specify Error Handling for Each Subsystem	173
Summary	174
Exercises	175
CHAPTER 5: DESIGNING WITH CLASSES	177
Am I Thinking Procedurally?	178
The Object-Oriented Philosophy	178
Classes	178
Components	179
Properties	179
Behaviors	180
Bringing It All Together	180
Living In a World of Classes	181
Over-Classification	182
Overly General Classes	182
Class Relationships	183
The Has-a Relationship	183
The Is-a Relationship (Inheritance)	184
Inheritance Techniques	185
Inheritance Techniques Polymorphism	185 186

The Not-a Relationship Hierarchies Multiple Inheritance Mixin Classes Summary Exercises	190 191 192 193 194 194
CHAPTER 6: DESIGNING FOR REUSE	197
The Reuse Philosophy	198
How to Design Reusable Code	198
Use Abstraction	199
Structure Your Code for Optimal Reuse	200
Avoid Combining Unrelated or Logically Separate Concepts	201
Use Templates for Generic Data Structures and Algorithms	203
Provide Appropriate Checks and Safeguards	205
Design for Extensibility	206
Design Usable Interfaces	208
Consider the Audience	208
Consider the Purpose	209
Design Interfaces That Are Easy to Use	210
Design General-Purpose Interfaces	214
Reconciling Generality and Ease of Use	215
Designing a Successful Abstraction	216
The SOLID Principles	216
Summary	217
Exercises	217

PART III: C++ CODING THE PROFESSIONAL WAY **CHAPTER 7: MEMORY MANAGEMENT** 221 Working with Dynamic Memory 222 How to Picture Memory 222 Allocation and Deallocation 223 Using new and delete 223 What About My Good Friend malloc? 224 When Memory Allocation Fails 225 Arrays 225 Arrays of Primitive Types 226 Arrays of Objects 228

Deleting Arrays	228
Multidimensional Arrays	229
Working with Pointers	233
A Mental Model for Pointers	233
Casting with Pointers	234
Array-Pointer Duality	234
Arrays Decay to Pointers	234
Not All Pointers Are Arrays!	236
Low-Level Memory Operations	236
Pointer Arithmetic	236
Custom Memory Management	237
Garbage Collection	238
Object Pools	238
Common Memory Pitfalls	239
Underallocating Data Buffers and Out-of-Bounds Memory Access	239
Memory Leaks	240
Finding and Fixing Memory Leaks in Windows with Visual C++	241
Finding and Fixing Memory Leaks in Linux with Valgrind	243
Double-Deletion and Invalid Pointers	243
Smart Pointers	244
unique_ptr	245
Creating unique_ptrs	245
Using unique_ptrs	247
unique_ptr and C-Style Arrays	248
Custom Deleters	248
shared_ptr	249
Creating and Using shared_ptrs	249
The Need for Reference Counting	250
Casting a shared_ptr	251
Aliasing	252
weak_ptr	252
Passing to Functions	253
Returning from Functions	253
enable_shared_from_this	254
Interoperability of Smart Pointers with C-Style Functions	255
The Old and Removed auto_ptr	255
Summary	256
Exercises	256

CONTENTS

Introducing the Spreadsheet Example	260
Writing Classes	260
Class Definitions	260
Class Members	261
Access Control	261
Order of Declarations	262
In-Class Member Initializers	263
Defining Member Functions	263
Accessing Data Members	264
Calling Other Member Functions	264
Using Objects	265
Objects on the Stack	266
Objects on the Free Store	266
The this Pointer	267
Explicit Object Parameter	268
Understanding Object Life Cycles	269
Object Creation	269
Writing Constructors	270
Using Constructors	270
Providing Multiple Constructors	271
Default Constructors	272
Constructor Initializers aka Ctor-Initializers	276
Copy Constructors	279
Initializer-List Constructors	281
Delegating Constructors	283
Converting Constructors and Explicit Constructors	284
Summary of Compiler-Generated Constructors	285
Object Destruction	286
Assigning to Objects	288
Declaring an Assignment Operator	288
Defining an Assignment Operator	289
Explicitly Defaulted and Deleted Assignment Operator	290
Compiler-Generated Copy Constructor and Copy Assignment Operator	291
Distinguishing Copying from Assignment	291
Objects as Return Values	291
Copy Constructors and Object Members	292
Summary	293
Exercises	293

CHAPTER 8: GAINING PROFICIENCY WITH CLASSES AND OBJECTS 259

CHAPTER 9: MASTERING CLASSES AND OBJECTS	295
Friends	296
Dynamic Memory Allocation in Objects	297
The Spreadsheet Class	297
Freeing Memory with Destructors	300
Handling Copying and Assignment	301
The Spreadsheet Copy Constructor	303
The Spreadsheet Assignment Operator	303
Disallowing Assignment and Pass-by-Value	306
Handling Moving with Move Semantics	307
Rvalue References	307
Decay Сору	310
Implementing Move Semantics	310
Testing the Spreadsheet Move Operations	314
Implementing a Swap Function with Move Semantics	316
Using std::move() in Return Statements	317
Optimal Way to Pass Arguments to Functions	318
Rule of Zero	319
More About Member Functions	320
static Member Functions	320
const Member Functions	321
mutable Data Members	322
Member Function Overloading	323
Overloading Based on const	323
Explicitly Deleting Overloads	325
Ref-Qualified Member Functions	325
Inline Member Functions	327
Default Arguments	329
Constexpr and Consteval	330
The constexpr Keyword	330
The consteval Keyword	331
constexpr and consteval Classes	332
Different Kinds of Data Members	333
static Data Members	333
Inline Variables	334
Accessing static Data Members from within Class	
Member Functions	334
constexpr static Data Members	335
Accessing static Data Members from Outside	
Class Member Functions	336
Reference Data Members	336

Nested Classes	338
Enumerations Inside Classes	339
Operator Overloading	339
Example: Implementing Addition for SpreadsheetCells	340
First Attempt: The add Member Function	340
Second Attempt: Overloaded operator+ as a Member Function	341
Third Attempt: Global operator+	342
Overloading Arithmetic Operators	343
Overloading the Arithmetic Shorthand Operators	344
Overloading Comparison Operators	345
Overloading Comparison Operators Before C++20	345
Overloading Comparison Operators Since C++20	347
Compiler-Generated Comparison Operators	348
Building Stable Interfaces	350
Using Interface and Implementation Classes	350
Summary	354
Exercises	354
CHAPTER 10: DISCOVERING INHERITANCE TECHNIQUES	357
Building Classes with Inheritance	358
Extending Classes A Client's View of Inheritance	358
	359
A Derived Class's View of Inheritance	360 362
Preventing Inheritance	362 362
Overriding Member Functions	362 362
The virtual Keyword	
Syntax for Overriding a Member Function A Client's View of Overridden Member Functions	363 363
	363 365
The override Keyword The Truth about virtual	365
	300 370
Preventing Overriding Inheritance For Reuse	370
The WeatherPrediction Class	370
Adding Functionality in a Derived Class	371
Replacing Functionality in a Derived Class	373
Respect Your Parents	373
Parent Constructors	373
Parent Destructors	375
virtual Member Function Calls within Constructors and Destructor	376
Referring to Parent Names	377
Casting Up and Down	379

Inheritance for Polymorphism	380
Return of the Spreadsheet	380
Designing the Polymorphic Spreadsheet Cell	381
The SpreadsheetCell Base Class	382
A First Attempt	382
Pure virtual Member Functions and Abstract Base Classes	382
The Individual Derived Classes	383
StringSpreadsheetCell Class Definition	383
StringSpreadsheetCell Implementation	384
DoubleSpreadsheetCell Class Definition and Implementation	384
Leveraging Polymorphism	385
Future Considerations	386
Providing Implementations for Pure virtual Member Functions	388
Multiple Inheritance	388
Inheriting from Multiple Classes	389
Naming Collisions and Ambiguous Base Classes	390
Name Ambiguity	390
Ambiguous Base Classes	391
Uses for Multiple Inheritance	392
Interesting and Obscure Inheritance Issues	392
Changing the Overridden Member Function's Return Type	393
Adding Overloads of virtual Base Class Member	
Functions to Derived Classes	396
Inherited Constructors	396
Hiding of Inherited Constructors	397
Inherited Constructors and Multiple Inheritance	398
Initialization of Data Members	399
Special Cases in Overriding Member Functions	400
The Base Class Member Function Is static	400
The Base Class Member Function Is Overloaded	401
The Base Class Member Function Is private	403
The Base Class Member Function Has Default Arguments	404
The Base Class Member Function Has a Different	105
Access Specification	405
Copy Constructors and Assignment Operators in Derived Classes	407
Run-Time Type Facilities	408
Non-public Inheritance	410
Virtual Base Classes	411
Casts	414
static_cast()	414

reinterpret_cast()	415
dynamic_cast()	416
std::bit_cast()	417
Summary of Casts	418
Summary	418
Exercises	419
CHAPTER 11: MODULES, HEADER FILES, AND	
MISCELLANEOUS TOPICS	421
Modules	422
Unmodularizing Code	423
Standard Named Modules	423
Module Interface Files	423
Module Implementation Files	425
Splitting Interface from Implementation	426
Visibility vs. Reachability	427
Submodules	428
Module Partitions	429
Implementation Partitions	431
Private Module Fragment	432
Header Units	433
Importable Standard Library Headers	434
Preprocessor Directives	436
Preprocessor Macros	437
Linkage	438
Internal Linkage	439
The extern Keyword	440
Header Files	441
One Definition Rule (ODR)	441
Duplicate Definitions	442
Circular Dependencies	442
Querying Existence of Headers	443
Module Import Declarations	443
Feature-Test Macros for Core Language Features	444
The Static Keyword	445
static Data Members and Member Functions	445
static Variables in Functions	445
Order of Initialization of Nonlocal Variables	446
Order of Destruction of Nonlocal Variables	446

C-Style Variable-Length Argument Lists	447
Accessing the Arguments	448
Why You Shouldn't Use C-Style Variable-Length Argument Lists	448
Summary	449
Exercises	449
CHAPTER 12: WRITING GENERIC CODE WITH TEMPLATES	451
Overview of Templates	452
Class Templates	453
Writing a Class Template	453
Coding Without Templates	453
A Template Grid Class	456
Using the Grid Template	460
How the Compiler Processes Templates	461
Selective/Implicit Instantiation	462
Explicit Instantiation	462
Template Requirements on Types	462
Distributing Template Code Between Files	463
Member Function Definitions in Same File as Class	
Template Definition	463
Member Function Definitions in Separate File	463
Template Parameters	464
Non-type Template Parameters	464
Default Values for Template Parameters	466
Class Template Argument Deduction	467
Member Function Templates	468
Member Function Templates with Non-type Template	
Parameters	471
Using Member Function Templates with Explicit	473
Object Parameters to Avoid Code Duplication	473
Class Template Specialization	
Deriving from Class Templates	477
Inheritance vs. Specialization	478
Alias Templates	479
	479
Function Overloads vs. Function Template	481
Function Template Overloading	481
Function Templates as Friends of Class Templates	482
More on Template Type Parameter Deduction	484
Return Type of Function Templates	484
Abbreviated Function Template Syntax	486

Variable Templates	487
Concepts	487
Syntax	488
Constraints Expression	488
Requires Expressions	489
Combining Concept Expressions	491
Predefined Standard Concepts	491
Type-Constrained auto	492
Type Constraints and Function Templates	493
Constraint Subsumption	495
Type Constraints and Class Templates	495
Type Constraints and Class Member Functions	496
Constraint-Based Class Template Specialization and	
Function Template Overloading	496
Best Practices	497
Summary	498
Exercises	498
CHAPTER 13: DEMYSTIFYING C++ I/O	501
Using Streams	502
What Is a Stream, Anyway?	502
Stream Sources and Destinations	504
Output with Streams	504
Output Basics	504
Member Functions of Output Streams	505
Handling Output Errors	506
Output Manipulators	508
Input with Streams	510
Input Basics	510
Handling Input Errors	511
Input Member Functions	512
Input Manipulators	516
Input and Output with Objects	517
Custom Manipulators	519
String Streams	519
Span-Based Streams	521
File Streams	522
Text Mode vs. Binary Mode	523
Jumping Around with seek() and tell()	523
Linking Streams Together	526
Read an Entire File	526

Bidirectional I/O	527
Filesystem Support Library	528
Path	528
Directory Entry	530
Helper Functions	530
Directory Iteration	530
Summary	531
Exercises	532
CHAPTER 14: HANDLING ERRORS	533
Errors and Exceptions	534
What Are Exceptions, Anyway?	534
Why Exceptions in C++ Are a Good Thing	535
Recommendation	536
Exception Mechanics	536
Throwing and Catching Exceptions	537
Exception Types	540
Catching Exception Objects as Reference-to-const	541
Throwing and Catching Multiple Exceptions	541
Matching and const	543
Matching Any Exception	543
Uncaught Exceptions	544
noexcept Specifier	546
noexcept(expression) Specifier	546
noexcept(expression) Operator	546
Throw Lists	547
Exceptions and Polymorphism	547
The Standard Exception Hierarchy	547
Catching Exceptions in a Class Hierarchy	549
Writing Your Own Exception Classes	550
Nested Exceptions	553
Rethrowing Exceptions	555
Stack Unwinding and Cleanup	556
Use Smart Pointers	558
Catch, Cleanup, and Rethrow	558
Source Location	559
Source Location for Logging	560
Automatically Embed a Source Location in Custom Exceptions	560
Stack Trace	561

The Stack Trace Library	561
Automatically Embed a Stack Trace in Custom Exceptions	563
Common Error-Handling Issues	564
Memory Allocation Errors	565
Non-throwing new	565
Customizing Memory Allocation Failure Behavior	566
Errors in Constructors	567
Function-Try-Blocks for Constructors	569
Errors in Destructors	572
Exception Safety Guarantees	573
Summary	573
Exercises	573
CHAPTER 15: OVERLOADING C++ OPERATORS	577
Overview of Operator Overloading	578
Why Overload Operators?	578
Limitations to Operator Overloading	578
Choices in Operator Overloading	579
Member Function or Global Function	579
Choosing Argument Types	580
Choosing Return Types	581
Choosing Behavior	581
Operators You Shouldn't Overload	581
Summary of Overloadable Operators	582
Rvalue References	586
Precedence and Associativity	587
Relational Operators	588
Alternative Notation	589
Overloading The Arithmetic Operators	589
Overloading Unary Minus and Unary Plus	589
Overloading Increment and Decrement	590
Overloading the Bitwise and Binary Logical Operators	591
Overloading the Insertion and Extraction Operators	591
Overloading the Subscripting Operator	593
Providing Read-Only Access with operator[]	596
Multidimensional Subscripting Operator	598
Non-integral Array Indices	599
static Subscripting Operator	599
Overloading the Function Call Operator	600
static Function Call Operator	601

Overloading the Dereferencing Operators	602
Implementing operator*	603
Implementing operator->	604
What in the World Are operator.* and operator->*?	604
Writing Conversion Operators	605
Operator auto	606
Solving Ambiguity Problems with Explicit Conversion Operators	606
Conversions for Boolean Expressions	607
Overloading the Memory Allocation and Deallocation Operators	609
How new and delete Really Work	609
The New-Expression and operator new	609
The Delete-Expression and operator delete	610
Overloading operator new and operator delete	610
Explicitly Deleting or Defaulting operator new and operator delete	613
Overloading operator new and operator delete with Extra Parameters	613
Overloading operator delete with Size of Memory as Parameter	614
Overloading User-Defined Literal Operators	615
Standard Library Literals	615
User-Defined Literals	616
Cooked-Mode Literal Operator	616
Raw-Mode Literal Operator	617
Summary	618
Exercises	618
CHAPTER 16: OVERVIEW OF THE C++ STANDARD LIBRARY	619
Coding Principles	620
Use of Templates	621
Use of Operator Overloading	621
Overview of the C++ Standard Library	621
Strings	621
Regular Expressions	622
I/O Streams	622
Smart Pointers	622
Exceptions	623
Standard Integer Types	623
Numerics Library	623
Integer Comparisons	624
Bit Manipulation	624
Time and Date Utilities	625
Random Numbers	625

Initializer Lists	626
Pair and Tuple	626
Vocabulary Types	626
Function Objects	627
Filesystem	627
Multithreading	627
Type Traits	627
Standard Library Feature-Test Macros	627
<version></version>	629
Source Location	629
Stack Trace	629
Containers	629
Sequential Containers	630
Sequential Views	632
Container Adapters	632
Ordered Associative Containers	634
Unordered Associative Containers/Hash Tables	635
Flat Associative Container Adapters	635
bitset	636
Summary of Standard Library Containers	636
Algorithms	639
Non-modifying Sequence Algorithms	640
Modifying Sequence Algorithms	642
Operational Algorithms	643
Swap Algorithms	644
Partitioning Algorithms	644
Sorting Algorithms	645
Binary Search Algorithms	645
Set Algorithms on Sorted Sequences	645
Other Algorithms on Sorted Sequences	646
Heap Algorithms	646
Minimum/Maximum Algorithms	646
Numerical Processing Algorithms	647
Permutation Algorithms	648
Choosing an Algorithm	648
Ranges Library	649
What's Missing from the Standard Library	650
Summary	650
Exercises	650

CHAPTER 17: UNDERSTANDING ITERATORS AND THE RANGES LIBRARY	653
Iterators	654
Getting Iterators for Containers	656
Iterator Traits	658
Examples	659
Function Dispatching Using Iterator Traits	660
Stream Iterators	661
Output Stream Iterator: ostream_iterator	662
Input Stream Iterator: istream_iterator	663
Input Stream Iterator: istreambuf_iterator	663
Iterator Adapters	663
Insert Iterators	664
Reverse Iterators	665
Move Iterators	666
Ranges	668
Constrained Algorithms	669
Projection	670
Views	671
Modifying Elements Through a View	677
Mapping Elements	677
Range Factories	678
Input Streams as Views	679
Converting a Range into a Container	680
Summary	681
Exercises	681
CHAPTER 18: STANDARD LIBRARY CONTAINERS	683
Containers Overview	684
Requirements on Elements	685
Exceptions and Error Checking	687
Sequential Containers	687
vector	687
vector Overview	687
vector Details	690
Move Semantics	703
vector Example: A Round-Robin Class	704
The vector <bool> Specialization</bool>	709
deque	709
list	710

Accessing Elements	710
Iterators	711
Adding and Removing Elements	711
list Size	711
Special list Operations	711
list Example: Determining Enrollment	713
forward_list	714
array	717
Sequential Views	718
span	718
ndspan	720
Container Adapters	722
queue	722
queue Operations	722
queue Example: A Network Packet Buffer	723
priority_queue	725
priority_queue Operations	725
priority_queue Example: An Error Correlator	726
stack	727
stack Operations	728
stack Example: Revised Error Correlator	728
Associative Containers	728
Ordered Associative Containers	728
The pair Utility Class	729
map	729
multimap	738
set	742
multiset	744
Unordered Associative Containers Or Hash Tables	744
Hash Functions	744
unordered_map	746
unordered_multimap	750
unordered_set/unordered_multiset	751
Flat Set and Flat Map Associative Container Adapters	751
Performance of Associative Containers	752
Other Containers	752
Standard C-Style Arrays	752
Strings	753
Streams	754
bitset	754
bitset Basics	755

Bitwise Operators	755
bitset Example: Representing Cable Channels	756
Summary	759
Exercises	759
CHAPTER 19: FUNCTION POINTERS, FUNCTION OBJECTS, AND LAMBDA EXPRESSIONS	761
Function Pointers	762
findMatches() Using Function Pointers	762
findMatches() As a Function Template	764
Windows DLLs and Function Pointers	765
Pointers to Member Functions (And Data Members)	765
Function Objects	767
Writing Your First Function Object	767
Function Objects in the Standard Library	767
Arithmetic Function Objects	768
Comparison Function Objects	769
Logical Function Objects	771
Bitwise Function Objects	771
Adapter Function Objects	771
Polymorphic Function Wrappers	775
std::function	775
std::move_only_function	776
Lambda Expressions	777
Syntax	777
Lambda Expressions as Parameters	783
Generic Lambda Expressions	783
Lambda Capture Expressions	784
Templated Lambda Expressions	785
Lambda Expressions as Return Type	785
Lambda Expressions in Unevaluated Contexts	786
Default Construction, Copying, and Assigning	786
Recursive Lambda Expressions	787
Invokers	787
Summary	788
Exercises	788
CHAPTER 20: MASTERING STANDARD LIBRARY ALGORITHMS	791
Overview of Algorithms	792
The find and find_if Algorithms	793
The accumulate Algorithm	795