

FIFTH EDITION



Professional

C++

Marc Gregoire



# PROFESSIONAL C++

---

INTRODUCTION .....	xlvii
► PART I	INTRODUCTION TO PROFESSIONAL C++
CHAPTER 1	A Crash Course in C++ and the Standard Library..... 3
CHAPTER 2	Working with Strings and String Views ..... 87
CHAPTER 3	Coding with Style..... 111
► PART II	PROFESSIONAL C++ SOFTWARE DESIGN
CHAPTER 4	Designing Professional C++ Programs ..... 137
CHAPTER 5	Designing with Objects ..... 169
CHAPTER 6	Designing for Reuse..... 187
► PART III	C++ CODING THE PROFESSIONAL WAY
CHAPTER 7	Memory Management ..... 211
CHAPTER 8	Gaining Proficiency with Classes and Objects..... 249
CHAPTER 9	Mastering Classes and Objects ..... 283
CHAPTER 10	Discovering Inheritance Techniques..... 337
CHAPTER 11	Odds and Ends..... 397
CHAPTER 12	Writing Generic Code with Templates..... 421
CHAPTER 13	Demystifying C++ I/O ..... 465
CHAPTER 14	Handling Errors..... 495
CHAPTER 15	Overloading C++ Operators ..... 535
CHAPTER 16	Overview of the C++ Standard Library ..... 573
CHAPTER 17	Understanding Iterators and the Ranges Library..... 603
CHAPTER 18	Standard Library Containers ..... 627
CHAPTER 19	Function Pointers, Function Objects, and Lambda Expressions..... 699
CHAPTER 20	Mastering Standard Library Algorithms..... 725

*Continues*

CHAPTER 21	String Localization and Regular Expressions . . . . .	763
CHAPTER 22	Date and Time Utilities . . . . .	793
CHAPTER 23	Random Number Facilities . . . . .	809
CHAPTER 24	Additional Library Utilities . . . . .	821
 <b>► PART IV MASTERING ADVANCED FEATURES OF C++</b>		
CHAPTER 25	Customizing and Extending the Standard Library. . . . .	833
CHAPTER 26	Advanced Templates . . . . .	877
CHAPTER 27	Multithreaded Programming with C++ . . . . .	915
 <b>► PART V C++ SOFTWARE ENGINEERING</b>		
CHAPTER 28	Maximizing Software Engineering Methods . . . . .	971
CHAPTER 29	Writing Efficient C++ . . . . .	993
CHAPTER 30	Becoming Adept at Testing . . . . .	1021
CHAPTER 31	Conquering Debugging . . . . .	1045
CHAPTER 32	Incorporating Design Techniques and Frameworks. . . . .	1083
CHAPTER 33	Applying Design Patterns. . . . .	1105
CHAPTER 34	Developing Cross-Platform and Cross-Language Applications. . .	1137
 <b>► PART VI APPENDICES</b>		
APPENDIX A	C++ Interviews . . . . .	1165
APPENDIX B	Annotated Bibliography . . . . .	1191
APPENDIX C	Standard Library Header Files . . . . .	1203
APPENDIX D	Introduction to UML . . . . .	1213
INDEX . . . . .		1219



PROFESSIONAL

**C++**



PROFESSIONAL  
**C++**

**Fifth Edition**

Marc Gregoire



## Professional C++

Copyright © 2021 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada and the United Kingdom

ISBN: 978-1-119-69540-0

ISBN: 978-1-119-69550-9 (ebk)

ISBN: 978-1-119-69545-5 (ebk)

Manufactured in the United States of America

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 Sections 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, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions 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/permissions](http://www.wiley.com/go/permissions).

**Limit of Liability/Disclaimer of Warranty:** The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at [booksupport.wiley.com](http://booksupport.wiley.com). For more information about Wiley products, visit [www.wiley.com](http://www.wiley.com).

**Library of Congress Control Number:** 2020950208

**Trademarks:** Wiley, the Wiley logo, Wrox, the Wrox logo, Programmer to Programmer, and related trade dress 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.

*Dedicated to my wonderful parents and my brother,  
who are always there for me. Their support and  
patience helped me in finishing 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 architect at Nikon Metrology ([nikonmetrology.com](http://nikonmetrology.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/C++, specifically Microsoft VC++ and the MFC framework. He has experience in developing C++ programs running 24/7 on Windows and Linux platforms: for example, KNX/EIB home automation software. In addition to C/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](http://becpp.org)), co-author of *C++ Standard Library Quick Reference* 1<sup>st</sup> and 2<sup>nd</sup> editions (Apress), a technical editor for numerous books for several publishers, and a regular speaker at the CppCon C++ conference. He maintains a blog at [www.nuonsoft.com/blog/](http://www.nuonsoft.com/blog/) and is passionate about traveling and gastronomic restaurants.



# ABOUT THE TECHNICAL EDITORS

**PETER VAN WEERT** is a Belgian software engineer whose main interests and expertise are application software development, programming languages, algorithms, and data structures.

He received his master of science degree in computer science *summa cum laude* with congratulations from the Board of Examiners from the University of Leuven. In 2010, he completed his PhD thesis on the design and efficient compilation of rule-based programming languages at the research group for declarative programming languages and artificial intelligence. During his doctoral studies he was a teaching assistant for object-oriented programming (Java), software analysis and design, and declarative programming.

Peter then joined Nikon Metrology, where he worked on large-scale, industrial application software in the area of 3-D laser scanning and point cloud inspection for over six years. Today, Peter is senior C++ engineer and Scrum team leader at Medicim, the R&D unit for digital dentistry software of Envista Holdings. At Medicim, he codevelops a suite of applications for dental professionals, capable of capturing patient data from a wide range of hardware, with advanced diagnostic functionality and support for implant planning and prosthetic design.

Common themes in his professional career include advanced desktop application development, mastering and refactoring of code bases of millions of lines of C++ code, high-performant, real-time processing of 3-D data, concurrency, algorithms and data structures, interfacing with cutting-edge hardware, and leading agile development teams.

Peter is a regular speaker at, and board member of, the Belgian C++ Users Group. He also co-authored two books: *C++ Standard Library Quick Reference* and *Beginning C++* (5th edition), both published by Apress.

**OCKERT J. DU PREEZ** is a self-taught developer who started learning programming in the days of QBasic. He has written hundreds of developer articles over the years detailing his programming quests and adventures. His articles can be found on CodeGuru ([codeguru.com](http://codeguru.com)), Developer.com ([developer.com](http://developer.com)), DevX ([devx.com](http://devx.com)), and Database Journal ([databasejournal.com](http://databasejournal.com)). Software development is his second love, just after his wife and child.

He knows a broad spectrum of development languages including C++, C#, VB.NET, JavaScript, and HTML. He has written the books *Visual Studio 2019 In-Depth* (BpB Publications) and *JavaScript for Gurus* (BpB Publications).

He was a Microsoft Most Valuable Professional for .NET (2008–2017).



# ACKNOWLEDGMENTS

**I THANK THE JOHN WILEY & SONS AND WROX PRESS** editorial and production teams for their support. Especially, thank you to Jim Minatel, executive editor at Wiley, for giving me a chance to write this fifth edition; Kelly Talbot, project editor, for managing this project; and Kim Wimpsett, copy editor, for improving readability and consistency and making sure the text is grammatically correct.

Thanks to technical editor Hannes Du Preez for checking the technical accuracy of the book. His contributions in strengthening this book are greatly appreciated.

A very special thank you to technical editor Peter Van Weert for his outstanding contributions. His considerable advice and insights have truly elevated this book to a higher level.

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 trying this approach to professional C++ software development.

—MARC GREGOIRE





# CONTENTS

INTRODUCTION

xlvii

## 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
Preprocessor Directives	5
The main() Function	8
I/O Streams	8
Namespaces	9
Nested Namespace	11
Namespace Alias	11
Literals	11
Variables	12
Numerical Limits	14
Zero Initialization	15
Casting	15
Floating-Point Numbers	16
Operators	16
Enumerated Types	19
Old-Style Enumerated Types	21
Structs	22
Conditional Statements	23
if/else Statements	23
switch Statements	24
The Conditional Operator	25
Logical Evaluation Operators	26
Three-Way Comparisons	27
Functions	28
Function Return Type Deduction	30
Current Function’s Name	30
Function Overloading	30

Attributes	30
[[nodiscard]]	31
[[maybe_unused]]	31
[[noreturn]]	32
[[deprecated]]	32
[[likely]] and [[unlikely]]	33
C-Style Arrays	33
std::array	35
std::vector	36
std::pair	36
std::optional	37
Structured Bindings	38
Loops	38
The while Loop	38
The do/while Loop	39
The for Loop	39
The Range-Based for Loop	39
Initializer Lists	40
Strings in C++	40
C++ as an Object-Oriented Language	41
Defining Classes	41
Using Classes	44
Scope Resolution	44
Uniform Initialization	45
Designated Initializers	48
Pointers and Dynamic Memory	49
The Stack and the Free Store	49
Working with Pointers	50
Dynamically Allocated Arrays	51
Null Pointer Constant	52
The Use of const	53
const as a Qualifier for a Type	53
const Methods	55
The constexpr Keyword	56
The consteval Keyword	57
References	58
Reference Variables	58
Reference Data Members	61
Reference Parameters	61
Reference Return Values	64
Deciding Between References and Pointers	64

const_cast()	68
Exceptions	69
Type Aliases	70
typedefs	71
Type Inference	72
The auto Keyword	72
The decltype Keyword	75
The Standard Library	75
<b>Your First Bigger C++ Program</b>	<b>75</b>
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
<b>Summary</b>	<b>85</b>
<b>Exercises</b>	<b>85</b>
 <b>CHAPTER 2: WORKING WITH STRINGS AND STRING VIEWS</b>	 <b>87</b>
<b>Dynamic Strings</b>	<b>88</b>
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 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	97
The std::string_view Class	100
std::string_view and Temporary Strings	102
std::string_view Literals	102
Nonstandard Strings	102
<b>String Formatting</b>	<b>103</b>
Format Specifiers	104

width	104
[fill]align	105
sign	105
#	105
type	106
precision	107
0	107
Format Specifier Errors	107
Support for Custom Types	107
<b>Summary</b>	<b>110</b>
<b>Exercises</b>	<b>110</b>

---

<b>CHAPTER 3: CODING WITH STYLE</b>	<b>111</b>
-------------------------------------	------------

---

<b>The Importance of Looking Good</b>	<b>111</b>
Thinking Ahead	112
Elements of Good Style	112
<b>Documenting Your Code</b>	<b>112</b>
Reasons to Write Comments	112
Commenting to Explain Usage	112
Commenting to Explain Complicated Code	115
Commenting to Convey Meta-information	116
Commenting Styles	117
Commenting Every Line	117
Prefix Comments	118
Fixed-Format Comments	119
Ad Hoc Comments	120
Self-Documenting Code	122
<b>Decomposition</b>	<b>122</b>
Decomposition Through Refactoring	123
Decomposition by Design	124
Decomposition in This Book	124
<b>Naming</b>	<b>124</b>
Choosing a Good Name	124
Naming Conventions	125
Counters	125
Prefixes	126
Hungarian Notation	126
Getters and Setters	127
Capitalization	127
Namespaced Constants	127
<b>Using Language Features with Style</b>	<b>127</b>
Use Constants	128

Use References Instead of Pointers	128
Use Custom Exceptions	129
Formatting	129
The Curly Brace Alignment Debate	130
Coming to Blows over Spaces and Parentheses	131
Spaces, Tabs, and Line Breaks	131
Stylistic Challenges	132
Summary	132
Exercises	133

## PART II: PROFESSIONAL C++ SOFTWARE DESIGN

<b>CHAPTER 4: DESIGNING PROFESSIONAL C++ PROGRAMS</b>	<b>137</b>
What Is Programming Design?	138
The Importance of Programming Design	139
Designing for C++	141
Two Rules for Your Own C++ Designs	142
Abstraction	142
Benefiting from Abstraction	142
Incorporating Abstraction in Your Design	143
Reuse	144
Writing Reusable Code	144
Reusing Designs	145
Reusing Existing Code	146
A Note on Terminology	146
Deciding Whether to Reuse Code or Write it Yourself	147
Advantages to Reusing Code	147
Disadvantages to Reusing Code	148
Putting It Together to Make a Decision	149
Guidelines for Choosing a Library to Reuse	149
Understand the Capabilities and Limitations	149
Understand the Learning Cost	150
Understand the Performance	150
Understand Platform Limitations	153
Understand Licensing	153
Understand Support and Know Where to Find Help	154
Prototype	154
Open-Source Libraries	155
The C++ Standard Library	157

Designing a Chess Program	157
Requirements	158
Design Steps	158
Divide the Program into Subsystems	158
Choose Threading Models	160
Specify Class Hierarchies for Each Subsystem	161
Specify Classes, Data Structures, Algorithms, and Patterns for Each Subsystem	162
Specify Error Handling for Each Subsystem	165
Summary	166
Exercises	166
<b>CHAPTER 5: DESIGNING WITH OBJECTS</b>	<b>169</b>
Am I Thinking Procedurally?	170
The Object-Oriented Philosophy	170
Classes	170
Components	171
Properties	171
Behaviors	172
Bringing It All Together	172
Living in a World of Classes	173
Over-Classification	173
Overly General Classes	174
Class Relationships	175
The Has-a Relationship	175
The Is-a Relationship (Inheritance)	176
Inheritance Techniques	177
Polymorphism	178
The Fine Line Between Has-a and Is-a	178
The Not-a Relationship	181
Hierarchies	182
Multiple Inheritance	183
Mixin Classes	184
Summary	185
Exercises	185
<b>CHAPTER 6: DESIGNING FOR REUSE</b>	<b>187</b>
The Reuse Philosophy	188
How to Design Reusable Code	189
Use Abstraction	189
Structure Your Code for Optimal Reuse	191
Avoid Combining Unrelated or Logically Separate Concepts	191



Use Templates for Generic Data Structures and Algorithms	193
Provide Appropriate Checks and Safeguards	195
Design for Extensibility	196
Design Usable Interfaces	198
Consider the Audience	198
Consider the Purpose	199
Design Interfaces That Are Easy to Use	200
Design General-Purpose Interfaces	204
Reconciling Generality and Ease of Use	205
Designing a Successful Abstraction	205
The SOLID Principles	206
Summary	207
Exercises	207

## PART III: C++ CODING THE PROFESSIONAL WAY

### CHAPTER 7: MEMORY MANAGEMENT 211

Working with Dynamic Memory	212
How to Picture Memory	212
Allocation and Deallocation	213
Using new and delete	213
What About My Good Friend malloc?	214
When Memory Allocation Fails	215
Arrays	215
Arrays of Primitive Types	215
Arrays of Objects	218
Deleting Arrays	218
Multidimensional Arrays	219
Working with Pointers	223
A Mental Model for Pointers	223
Casting with Pointers	224
Array-Pointer Duality	224
Arrays Are Pointers!	224
Not All Pointers Are Arrays!	226
Low-Level Memory Operations	227
Pointer Arithmetic	227
Custom Memory Management	228
Garbage Collection	228
Object Pools	229
Common Memory Pitfalls	229
Underallocating Data Buffers and Out-of-Bounds Memory Access	229

Memory Leaks	231
Finding and Fixing Memory Leaks in Windows with Visual C++	232
Finding and Fixing Memory Leaks in Linux with Valgrind	233
Double-Deletion and Invalid Pointers	234
<b>Smart Pointers</b>	<b>234</b>
unique_ptr	235
Creating unique_ptrs	236
Using unique_ptrs	237
unique_ptr and C-Style Arrays	238
Custom Deleters	239
shared_ptr	239
Creating and Using shared_ptrs	239
The Need for Reference Counting	241
Casting a shared_ptr	242
Aliasing	242
weak_ptr	243
Passing to Functions	244
Returning from Functions	244
enable_shared_from_this	244
The Old and Removed auto_ptr	245
<b>Summary</b>	<b>246</b>
<b>Exercises</b>	<b>246</b>

---

## **CHAPTER 8: GAINING PROFICIENCY WITH CLASSES AND OBJECTS** **249**

---

Introducing the Spreadsheet Example	250
<b>Writing Classes</b>	<b>250</b>
Class Definitions	250
Class Members	251
Access Control	251
Order of Declarations	252
In-Class Member Initializers	253
Defining Methods	253
Accessing Data Members	254
Calling Other Methods	254
The this Pointer	255
Using Objects	257
Objects on the Stack	257
Objects on the Free Store	257
<b>Understanding Object Life Cycles</b>	<b>258</b>
Object Creation	258
Writing Constructors	259
Using Constructors	260

Providing Multiple Constructors	260
Default Constructors	261
Constructor Initializers	265
Copy Constructors	269
Initializer-List Constructors	271
Delegating Constructors	273
Converting Constructors and Explicit Constructors	273
Summary of Compiler-Generated Constructors	275
Object Destruction	276
Assigning to Objects	277
Declaring an Assignment Operator	278
Defining an Assignment Operator	278
Explicitly Defaulted and Deleted Assignment Operator	280
Compiler-Generated Copy Constructor and Copy Assignment Operator	280
Distinguishing Copying from Assignment	280
Objects as Return Values	280
Copy Constructors and Object Members	281
Summary	282
Exercises	282
<b>CHAPTER 9: MASTERING CLASSES AND OBJECTS</b>	<b>283</b>
Friends	284
Dynamic Memory Allocation in Objects	285
The Spreadsheet Class	285
Freeing Memory with Destructors	288
Handling Copying and Assignment	289
The Spreadsheet Copy Constructor	291
The Spreadsheet Assignment Operator	291
Disallowing Assignment and Pass-by-Value	294
Handling Moving with Move Semantics	295
Rvalue References	295
Implementing Move Semantics	297
Testing the Spreadsheet Move Operations	301
Implementing a Swap Function with Move Semantics	303
Using <code>std::move()</code> in Return Statements	303
Optimal Way to Pass Arguments to Functions	304
Rule of Zero	305
More About Methods	306
static Methods	306
const Methods	307
mutable Data Members	308

Method Overloading	308
Overloading Based on const	309
Explicitly Deleting Overloads	310
Ref-Qualified Methods	310
Inline Methods	311
Default Arguments	313
<b>Different Kinds of Data Members</b>	<b>314</b>
static Data Members	314
Inline Variables	314
Accessing static Data Members within Class Methods	315
Accessing static Data Members Outside Methods	316
const static Data Members	316
Reference Data Members	317
<b>Nested Classes</b>	<b>318</b>
<b>Enumerated Types Inside Classes</b>	<b>319</b>
<b>Operator Overloading</b>	<b>320</b>
Example: Implementing Addition for SpreadsheetCells	320
First Attempt: The add Method	320
Second Attempt: Overloaded operator+ as a Method	321
Third Attempt: Global operator+	322
Overloading Arithmetic Operators	324
Overloading the Arithmetic Shorthand Operators	324
Overloading Comparison Operators	325
Compiler-Generated Comparison Operators	328
Building Types with Operator Overloading	330
<b>Building Stable Interfaces</b>	<b>330</b>
Using Interface and Implementation Classes	330
<b>Summary</b>	<b>334</b>
<b>Exercises</b>	<b>335</b>
 <b>CHAPTER 10: DISCOVERING INHERITANCE TECHNIQUES</b>	 <b>337</b>
<b>Building Classes with Inheritance</b>	<b>338</b>
Extending Classes	338
A Client's View of Inheritance	339
A Derived Class's View of Inheritance	340
Preventing Inheritance	341
Overriding Methods	342
The virtual Keyword	342
Syntax for Overriding a Method	342
A Client's View of Overridden Methods	343
The override Keyword	344

The Truth About virtual	346
Preventing Overriding	350
<b>Inheritance for Reuse</b>	<b>350</b>
The WeatherPrediction Class	350
Adding Functionality in a Derived Class	351
Replacing Functionality in a Derived Class	352
<b>Respect Your Parents</b>	<b>353</b>
Parent Constructors	353
Parent Destructors	355
Referring to Parent Names	356
Casting Up and Down	358
<b>Inheritance for Polymorphism</b>	<b>360</b>
Return of the Spreadsheet	360
Designing the Polymorphic Spreadsheet Cell	360
The SpreadsheetCell Base Class	361
A First Attempt	361
Pure Virtual Methods and Abstract Base Classes	362
The Individual Derived Classes	363
StringSpreadsheetCell Class Definition	363
StringSpreadsheetCell Implementation	363
DoubleSpreadsheetCell Class Definition and Implementation	364
Leveraging Polymorphism	364
Future Considerations	365
<b>Multiple Inheritance</b>	<b>367</b>
Inheriting from Multiple Classes	367
Naming Collisions and Ambiguous Base Classes	368
Name Ambiguity	368
Ambiguous Base Classes	369
Uses for Multiple Inheritance	371
<b>Interesting and Obscure Inheritance Issues</b>	<b>371</b>
Changing the Overridden Method's Return Type	371
Adding Overloads of Virtual Base Class Methods to Derived Classes	373
Inherited Constructors	374
Hiding of Inherited Constructors	375
Inherited Constructors and Multiple Inheritance	376
Initialization of Data Members	377
Special Cases in Overriding Methods	378
The Base Class Method Is static	378
The Base Class Method Is Overloaded	379
The Base Class Method Is private	380
The Base Class Method Has Default Arguments	382

The Base Class Method Has a Different Access Specification	383
Copy Constructors and Assignment Operators in Derived Classes	385
Run-Time Type Facilities	386
Non-public Inheritance	388
Virtual Base Classes	389
<b>Casts</b>	<b>390</b>
static_cast()	390
reinterpret_cast()	391
std::bit_cast()	392
dynamic_cast()	393
Summary of Casts	394
<b>Summary</b>	<b>394</b>
<b>Exercises</b>	<b>395</b>
 <b>CHAPTER 11: ODDS AND ENDS</b>	 <b>397</b>
<b>Modules</b>	<b>397</b>
Module Interface Files	399
Module Implementation Files	401
Splitting Interface from Implementation	402
Visibility vs. Reachability	403
Submodules	404
Module Partitions	405
Implementation Partitions	407
Header Units	408
<b>Header Files</b>	<b>408</b>
Duplicate Definitions	409
Circular Dependencies	409
Querying Existence of Headers	410
<b>Feature Test Macros for Core Language Features</b>	<b>410</b>
<b>The static Keyword</b>	<b>411</b>
static Data Members and Methods	411
static Linkage	411
The extern Keyword	413
static Variables in Functions	414
Order of Initialization of Nonlocal Variables	415
Order of Destruction of Nonlocal Variables	415



<b>C Utilities</b>	<b>415</b>
Variable-Length Argument Lists	415
Accessing the Arguments	416
Why You Shouldn't Use C-Style Variable-Length Argument Lists	417
Preprocessor Macros	417
Summary	419
Exercises	419
 <b>CHAPTER 12: WRITING GENERIC CODE WITH TEMPLATES</b>	 <b>421</b>
<b>Overview of Templates</b>	<b>422</b>
<b>Class Templates</b>	<b>422</b>
Writing a Class Template	423
Coding Without Templates	423
A Template Grid Class	426
Using the Grid Template	430
How the Compiler Processes Templates	431
Selective Instantiation	431
Template Requirements on Types	432
Distributing Template Code Between Files	432
Method Definitions in Same File as Class Template Definition	433
Method Definitions in Separate File	433
Template Parameters	433
Non-type Template Parameters	434
Default Values for Type Parameters	436
Class Template Argument Deduction	436
Method Templates	438
Method Templates with Non-type Parameters	440
Class Template Specialization	442
Deriving from Class Templates	445
Inheritance vs. Specialization	446
Alias Templates	447
<b>Function Templates</b>	<b>447</b>
Function Template Overloading	449
Friend Function Templates of Class Templates	449
More on Template Parameter Deduction	451
Return Type of Function Templates	451
Abbreviated Function Template Syntax	453

<b>Variable Templates</b>	<b>454</b>
<b>Concepts</b>	<b>454</b>
Syntax	455
Constraints Expression	455
Requires Expressions	455
Combining Concept Expressions	457
Predefined Standard Concepts	457
Type-Constrained auto	458
Type Constraints and Function Templates	458
Constraint Subsumption	460
Type Constraints and Class Templates	461
Type Constraints and Class Methods	461
Type Constraints and Template Specialization	462
<b>Summary</b>	<b>463</b>
<b>Exercises</b>	<b>463</b>
 <b>CHAPTER 13: DEMYSTIFYING C++ I/O</b>	 <b>465</b>
<b>Using Streams</b>	<b>466</b>
What Is a Stream, Anyway?	466
Stream Sources and Destinations	467
Output with Streams	468
Output Basics	468
Methods of Output Streams	469
Handling Output Errors	470
Output Manipulators	471
Input with Streams	473
Input Basics	473
Handling Input Errors	475
Input Methods	476
Input Manipulators	480
Input and Output with Objects	481
Custom Manipulators	482
<b>String Streams</b>	<b>482</b>
<b>File Streams</b>	<b>484</b>
Text Mode vs. Binary Mode	485
Jumping Around with seek() and tell()	485
Linking Streams Together	487
<b>Bidirectional I/O</b>	<b>488</b>
<b>Filesystem Support Library</b>	<b>490</b>
Path	490
Directory Entry	491