

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 57

CHAPTER 3 Coding with Style. 71

► **PART II PROFESSIONAL C++ SOFTWARE DESIGN**

CHAPTER 4 Designing Professional C++ Programs 95

CHAPTER 5 Designing with Objects 123

CHAPTER 6 Designing for Reuse. 143

► **PART III C++ CODING THE PROFESSIONAL WAY**

CHAPTER 7 Memory Management 163

CHAPTER 8 Gaining Proficiency with Classes and Objects 199

CHAPTER 9 Mastering Classes and Objects 231

CHAPTER 10 Discovering Inheritance Techniques 277

CHAPTER 11 C++ Quirks, Oddities, and Incidentals 333

CHAPTER 12 Writing Generic Code with Templates 373

CHAPTER 13 Demystifying C++ I/O 409

CHAPTER 14 Handling Errors 433

CHAPTER 15 Overloading C++ Operators 473

CHAPTER 16 Overview of the C++ Standard Library 507

CHAPTER 17 Understanding Containers and Iterators 535

CHAPTER 18 Mastering Standard Library Algorithms 607

CHAPTER 19 String Localization and Regular Expressions 663

CHAPTER 20 Additional Library Utilities 691

Continues

► **PART IV MASTERING ADVANCED FEATURES OF C++**

CHAPTER 21	Customizing and Extending the Standard Library	727
CHAPTER 22	Advanced Templates	775
CHAPTER 23	Multithreaded Programming with C++.	813

► **PART V C++ SOFTWARE ENGINEERING**

CHAPTER 24	Maximizing Software Engineering Methods.	859
CHAPTER 25	Writing Efficient C++	881
CHAPTER 26	Becoming Adept at Testing.	909
CHAPTER 27	Conquering Debugging.	933
CHAPTER 28	Incorporating Design Techniques and Frameworks.	971
CHAPTER 29	Applying Design Patterns	991
CHAPTER 30	Developing Cross-Platform and Cross-Language Applications	1017
APPENDIX A	C++ Interviews.	1039
APPENDIX B	Annotated Bibliography	1063
APPENDIX C	Standard Library Header Files.	1075
APPENDIX D	Introduction to UML	1083
INDEX	1087

PROFESSIONAL

C++

PROFESSIONAL
C++

Fourth Edition

Marc Gregoire



Professional C++, Fourth Edition

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2018 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-1-119-42130-6
ISBN: 978-1-119-42126-9 (ebk)
ISBN: 978-1-119-42122-1 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

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 <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 <http://booksupport.wiley.com>. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2017963243

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 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 master of science in engineering: 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 with 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 (www.nikonmetrology.com), a division of Nikon and a leading provider of precision optical instruments and metrology solutions for 3D geometric inspection.

His main expertise is in C/C++, and 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# and uses PHP for creating web pages.

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 (www.becpp.org), co-author of *C++ Standard Library Quick Reference* (Apress), technical editor for numerous books for several publishers, and a member on the CodeGuru forum (as Marc G). He maintains a blog at www.nuonsoft.com/blog/, and is passionate about traveling and gastronomic restaurants.

ABOUT THE TECHNICAL EDITOR

PETER VAN WEERT is a Belgian software engineer, whose main interests and expertise are in C++, programming languages, algorithms, and data structures.

He received his master of science in computer science from the University of Leuven, Belgium, *summa cum laude*, with congratulations of the Board of Examiners. In 2010, the same university awarded him a PhD for his research on the efficient compilation of rule-based programming languages (mainly Java). During his doctoral studies, he was a teaching assistant for courses on object-oriented analysis and design, Java programming, and declarative programming languages.

After his studies, Peter worked for Nikon Metrology on large-scale, industrial-application software in the area of 3D laser scanning and point cloud inspection. In 2017, he joined the software R&D unit of Nobel Biocare, which specializes in digital dentistry software. Throughout his professional career, Peter has mastered C++ software development, as well as the management, refactoring, and debugging of very large code bases. He also gained further proficiency in all aspects of the software development process, including the analysis of functional and technical requirements, and Agile- and Scrum-based project and team management.

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.

CREDITS

PROJECT EDITOR
Adaobi Obi Tulton

TECHNICAL EDITOR
Peter Van Weert

PRODUCTION EDITOR
Athiyappan Lalith Kumar

COPY EDITOR
Marylouise Wiack

**MANAGER OF CONTENT DEVELOPMENT
AND ASSEMBLY**
Mary Beth Wakefield

PRODUCTION MANAGER
Kathleen Wisor

MARKETING MANAGER
Christie Hilbrich

EXECUTIVE EDITOR
Jim Minatel

PROJECT COORDINATOR, COVER
Brent Savage

PROOFREADER
Nancy Bell

INDEXER
Johnna VanHoose Dinse

COVER DESIGNER
Wiley

COVER IMAGE
© ittipon/Shutterstock

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 new edition, Adaobi Obi Tulton, project editor, for managing this project, and Marylouise Wiack, copy editor, for improving readability and consistency and making sure the text is grammatically correct.

A very special thank you to my technical editor, Peter Van Weert, for his outstanding technical review. His many constructive comments and ideas have certainly made this book better.

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 toward 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.

CONTENTS

INTRODUCTION

xlvii

PART I: INTRODUCTION TO PROFESSIONAL C++

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

The Basics of C++	4
The Obligatory Hello, World	4
Comments	4
Preprocessor Directives	5
The main() Function	6
I/O Streams	7
Namespaces	8
Literals	10
Variables	10
Operators	13
Types	15
Enumerated Types	15
Structs	16
Conditional Statements	17
if/else Statements	17
switch Statements	18
The Conditional Operator	20
Logical Evaluation Operators	20
Functions	21
Function Return Type Deduction	22
Current Function's Name	23
C-Style Arrays	23
std::array	25
std::vector	25
Structured Bindings	26
Loops	26
The while Loop	26
The do/while Loop	27
The for Loop	27

The Range-Based for Loop	27
Initializer Lists	28
Those Are the Basics	28
Diving Deeper into C++	28
Strings in C++	29
Pointers and Dynamic Memory	29
The Stack and the Heap	29
Working with Pointers	30
Dynamically Allocated Arrays	31
Null Pointer Constant	32
Smart Pointers	33
The Many Uses of const	35
const Constants	35
const to Protect Parameters	35
References	35
Pass By Reference	36
Pass By const Reference	37
Exceptions	37
Type Inference	38
The auto Keyword	39
The decltype Keyword	40
C++ as an Object-Oriented Language	40
Defining Classes	40
Using Classes	43
Uniform Initialization	43
Direct List Initialization versus Copy List Initialization	45
The Standard Library	46
Your First Useful C++ Program	46
An Employee Records System	46
The Employee Class	47
Employee.h	47
Employee.cpp	48
EmployeeTest.cpp	50
The Database Class	50
Database.h	50
Database.cpp	51
DatabaseTest.cpp	52
The User Interface	53
Evaluating the Program	55
Summary	56

CHAPTER 2: WORKING WITH STRINGS AND STRING VIEWS	57
Dynamic Strings	58
C-Style Strings	58
String Literals	60
Raw String Literals	60
The C++ <code>std::string</code> Class	62
What Is Wrong with C-Style Strings?	62
Using the <code>string</code> Class	62
<code>std::string</code> Literals	64
High-Level Numeric Conversions	64
Low-Level Numeric Conversions	65
The <code>std::string_view</code> Class	67
<code>std::string_view</code> Literals	69
Nonstandard Strings	69
Summary	69
CHAPTER 3: CODING WITH STYLE	71
The Importance of Looking Good	71
Thinking Ahead	72
Elements of Good Style	72
Documenting Your Code	72
Reasons to Write Comments	72
Commenting to Explain Usage	72
Commenting to Explain Complicated Code	74
Commenting to Convey Meta-information	75
Commenting Styles	77
Commenting Every Line	77
Prefix Comments	78
Fixed-Format Comments	79
Ad Hoc Comments	80
Self-Documenting Code	81
Decomposition	81
Decomposition through Refactoring	82
Decomposition by Design	83
Decomposition in This Book	83
Naming	83
Choosing a Good Name	83
Naming Conventions	84
Counters	84
Prefixes	84

Hungarian Notation	85
Getters and Setters	86
Capitalization	86
Namespaced Constants	86
Using Language Features with Style	86
Use Constants	87
Use References Instead of Pointers	87
Use Custom Exceptions	88
Formatting	88
The Curly Brace Alignment Debate	88
Coming to Blows over Spaces and Parentheses	89
Spaces and Tabs	90
Stylistic Challenges	90
Summary	91

PART II: PROFESSIONAL C++ SOFTWARE DESIGN

CHAPTER 4: DESIGNING PROFESSIONAL C++ PROGRAMS	95
What Is Programming Design?	96
The Importance of Programming Design	97
Designing for C++	99
Two Rules for C++ Design	100
Abstraction	100
Benefiting from Abstraction	100
Incorporating Abstraction in Your Design	101
Reuse	101
Writing Reusable Code	102
Reusing Designs	103
Reusing Existing Code	103
A Note on Terminology	104
Deciding Whether or Not to Reuse Code	105
Advantages to Reusing Code	105
Disadvantages to Reusing Code	105
Putting It Together to Make a Decision	106
Strategies for Reusing Code	107
Understand the Capabilities and Limitations	107
Understand the Performance	108
Understand Platform Limitations	110
Understand Licensing and Support	110
Know Where to Find Help	111
Prototype	111

Bundling Third-Party Applications	112
Open-Source Libraries	112
The Open-Source Movements	112
Finding and Using Open-Source Libraries	113
Guidelines for Using Open-Source Code	113
The C++ Standard Library	114
C Standard Library	114
Deciding Whether or Not to Use the Standard Library	114
Designing a Chess Program	114
Requirements	115
Design Steps	115
Divide the Program into Subsystems	115
Choose Threading Models	117
Specify Class Hierarchies for Each Subsystem	118
Specify Classes, Data Structures, Algorithms, and Patterns for Each Subsystem	118
Specify Error Handling for Each Subsystem	120
Summary	121
 CHAPTER 5: DESIGNING WITH OBJECTS	 123
<hr/>	
Am I Thinking Procedurally?	124
The Object-Oriented Philosophy	124
Classes	124
Components	125
Properties	125
Behaviors	126
Bringing It All Together	126
Living in a World of Objects	127
Over-Objectification	127
Overly General Objects	128
Object Relationships	129
The Has-A Relationship	129
The Is-A Relationship (Inheritance)	130
Inheritance Techniques	130
Polymorphism versus Code Reuse	131
The Fine Line between Has-A and Is-A	132
The Not-A Relationship	135
Hierarchies	136
Multiple Inheritance	137
Mixin Classes	138

Abstraction	138
Interface versus Implementation	138
Deciding on an Exposed Interface	139
Consider the Audience	139
Consider the Purpose	139
Consider the Future	141
Designing a Successful Abstraction	141
Summary	142
CHAPTER 6: DESIGNING FOR REUSE	143
The Reuse Philosophy	144
How to Design Reusable Code	144
Use Abstraction	145
Structure Your Code for Optimal Reuse	146
Avoid Combining Unrelated or Logically Separate Concepts	146
Use Templates for Generic Data Structures and Algorithms	148
Provide Appropriate Checks and Safeguards	150
Design for Extensibility	151
Design Usable Interfaces	153
Design Interfaces That Are Easy to Use	153
Design General-Purpose Interfaces	157
Reconciling Generality and Ease of Use	157
The SOLID Principles	158
Summary	159
PART III: C++ CODING THE PROFESSIONAL WAY	
CHAPTER 7: MEMORY MANAGEMENT	163
Working with Dynamic Memory	164
How to Picture Memory	164
Allocation and Deallocation	166
Using new and delete	166
What about My Good Friend malloc?	167
When Memory Allocation Fails	167
Arrays	168
Arrays of Basic Types	168
Arrays of Objects	170
Deleting Arrays	171
Multi-dimensional Arrays	172
Working with Pointers	175

A Mental Model for Pointers	175
Casting with Pointers	176
Array-Pointer Duality	177
Arrays Are Pointers!	177
Not All Pointers Are Arrays!	179
Low-Level Memory Operations	179
Pointer Arithmetic	179
Custom Memory Management	180
Garbage Collection	181
Object Pools	182
Smart Pointers	182
unique_ptr	183
Creating unique_ptrs	183
Using unique_ptrs	185
unique_ptr and C-Style Arrays	186
Custom Deleters	186
shared_ptr	186
Casting a shared_ptr	187
The Need for Reference Counting	188
Aliasing	189
weak_ptr	189
Move Semantics	190
enable_shared_from_this	191
The Old Deprecated/Removed auto_ptr	192
Common Memory Pitfalls	192
Underallocating Strings	192
Accessing Out-of-Bounds Memory	193
Memory Leaks	194
Finding and Fixing Memory Leaks in Windows with Visual C++	195
Finding and Fixing Memory Leaks in Linux with Valgrind	196
Double-Deleting and Invalid Pointers	197
Summary	197

CHAPTER 8: GAINING PROFICIENCY WITH CLASSES AND OBJECTS 199

Introducing the Spreadsheet Example	200
Writing Classes	200
Class Definitions	200
Class Members	201
Access Control	201
Order of Declarations	203
In-Class Member Initializers	203

Defining Methods	203
Accessing Data Members	204
Calling Other Methods	204
The this Pointer	206
Using Objects	207
Objects on the Stack	207
Objects on the Heap	207
Object Life Cycles	208
Object Creation	208
Writing Constructors	209
Using Constructors	210
Providing Multiple Constructors	211
Default Constructors	212
Constructor Initializers	215
Copy Constructors	218
Initializer-List Constructors	220
Delegating Constructors	222
Summary of Compiler-Generated Constructors	222
Object Destruction	224
Assigning to Objects	225
Declaring an Assignment Operator	225
Defining an Assignment Operator	226
Explicitly Defaulted and Deleted Assignment Operator	227
Compiler-Generated Copy Constructor and Copy Assignment Operator	228
Distinguishing Copying from Assignment	228
Objects as Return Values	228
Copy Constructors and Object Members	229
Summary	230
CHAPTER 9: MASTERING CLASSES AND OBJECTS	231
Friends	232
Dynamic Memory Allocation in Objects	233
The Spreadsheet Class	233
Freeing Memory with Destructors	235
Handling Copying and Assignment	236
The Spreadsheet Copy Constructor	239
The Spreadsheet Assignment Operator	240
Disallowing Assignment and Pass-By-Value	242
Handling Moving with Move Semantics	243
Rvalue References	243
Implementing Move Semantics	245

Testing the Spreadsheet Move Operations	248
Implementing a Swap Function with Move Semantics	250
Rule of Zero	250
More about Methods	251
static Methods	251
const Methods	251
mutable Data Members	253
Method Overloading	253
Overloading Based on const	254
Explicitly Deleting Overloads	255
Inline Methods	255
Default Arguments	257
Different Kinds of Data Members	258
static Data Members	258
Inline Variables	259
Accessing static Data Members within Class Methods	259
Accessing static Data Members Outside Methods	260
const static Data Members	260
Reference Data Members	261
const Reference Data Members	262
Nested Classes	263
Enumerated Types inside Classes	264
Operator Overloading	265
Example: Implementing Addition for SpreadsheetCells	265
First Attempt: The add Method	265
Second Attempt: Overloaded operator+ as a Method	266
Third Attempt: Global operator+	268
Overloading Arithmetic Operators	269
Overloading the Arithmetic Shorthand Operators	269
Overloading Comparison Operators	270
Building Types with Operator Overloading	271
Building Stable Interfaces	272
Using Interface and Implementation Classes	272
Summary	275
CHAPTER 10: DISCOVERING INHERITANCE TECHNIQUES	277
Building Classes with Inheritance	278
Extending Classes	278
A Client's View of Inheritance	279
A Derived Class's View of Inheritance	280
Preventing Inheritance	281

Overriding Methods	281
How I Learned to Stop Worrying and Make Everything virtual	281
Syntax for Overriding a Method	282
A Client's View of Overridden Methods	283
The override Keyword	284
The Truth about virtual	286
Preventing Overriding	290
Inheritance for Reuse	291
The WeatherPrediction Class	291
Adding Functionality in a Derived Class	292
Replacing Functionality in a Derived Class	293
Respect Your Parents	294
Parent Constructors	294
Parent Destructors	296
Referring to Parent Names	297
Casting Up and Down	299
Inheritance for Polymorphism	301
Return of the Spreadsheet	301
Designing the Polymorphic Spreadsheet Cell	301
The SpreadsheetCell Base Class	302
A First Attempt	302
Pure Virtual Methods and Abstract Base Classes	303
The Individual Derived Classes	304
StringSpreadsheetCell Class Definition	304
StringSpreadsheetCell Implementation	304
DoubleSpreadsheetCell Class Definition and Implementation	305
Leveraging Polymorphism	306
Future Considerations	306
Multiple Inheritance	308
Inheriting from Multiple Classes	308
Naming Collisions and Ambiguous Base Classes	309
Name Ambiguity	309
Ambiguous Base Classes	311
Uses for Multiple Inheritance	312
Interesting and Obscure Inheritance Issues	312
Changing the Overridden Method's Characteristics	313
Changing the Method Return Type	313
Changing the Method Parameters	315
Inherited Constructors	316
Special Cases in Overriding Methods	320

The Base Class Method Is static	320
The Base Class Method Is Overloaded	321
The Base Class Method Is private or protected	322
The Base Class Method Has Default Arguments	324
The Base Class Method Has a Different Access Level	325
Copy Constructors and Assignment Operators in Derived Classes	327
Run-Time Type Facilities	329
Non-public Inheritance	331
Virtual Base Classes	331
Summary	332
CHAPTER 11: C++ QUIRKS, ODDITIES, AND INCIDENTALS	333
References	334
Reference Variables	334
Modifying References	335
References to Pointers and Pointers to References	336
Reference Data Members	336
Reference Parameters	336
References from Pointers	337
Pass-by-Reference versus Pass-by-Value	337
Reference Return Values	338
Rvalue References	338
Deciding between References and Pointers	339
Keyword Confusion	343
The const Keyword	343
const Variables and Parameters	343
const Methods	345
The constexpr Keyword	346
The static Keyword	347
static Data Members and Methods	347
static Linkage	347
static Variables in Functions	350
Order of Initialization of Nonlocal Variables	351
Order of Destruction of Nonlocal Variables	351
Types and Casts	351
Type Aliases	352
Type Aliases for Function Pointers	353
Type Aliases for Pointers to Methods and Data Members	355
typedefs	356

Casts	357
const_cast()	357
static_cast()	358
reinterpret_cast()	359
dynamic_cast()	360
Summary of Casts	361
Scope Resolution	362
Attributes	363
[[noreturn]]	363
[[deprecated]]	364
[[fallthrough]]	364
[[nodiscard]]	364
[[maybe_unused]]	365
Vendor-Specific Attributes	365
User-Defined Literals	365
Standard User-Defined Literals	367
Header Files	367
C Utilities	369
Variable-Length Argument Lists	369
Accessing the Arguments	370
Why You Shouldn't Use C-Style Variable-Length Argument Lists	371
Preprocessor Macros	371
Summary	372
CHAPTER 12: WRITING GENERIC CODE WITH TEMPLATES	373
Overview of Templates	374
Class Templates	375
Writing a Class Template	375
Coding without Templates	375
A Template Grid Class	378
Using the Grid Template	382
Angle Brackets	383
How the Compiler Processes Templates	383
Selective Instantiation	384
Template Requirements on Types	384
Distributing Template Code between Files	384
Template Definitions in Header Files	384
Template Definitions in Source Files	385
Template Parameters	386
Non-type Template Parameters	387

Default Values for Type Parameters	389
Template Parameter Deduction for Constructors	389
Method Templates	391
Method Templates with Non-type Parameters	393
Class Template Specialization	395
Deriving from Class Templates	397
Inheritance versus Specialization	399
Alias Templates	399
Function Templates	400
Function Template Specialization	401
Function Template Overloading	402
Function Template Overloading and Specialization Together	403
Friend Function Templates of Class Templates	403
More on Template Parameter Deduction	404
Return Type of Function Templates	405
Variable Templates	407
Summary	407
 CHAPTER 13: DEMYSTIFYING C++ I/O	 409
Using Streams	410
What Is a Stream, Anyway?	410
Stream Sources and Destinations	411
Output with Streams	411
Output Basics	412
Methods of Output Streams	412
Handling Output Errors	414
Output Manipulators	415
Input with Streams	417
Input Basics	417
Handling Input Errors	418
Input Methods	419
Input Manipulators	423
Input and Output with Objects	423
String Streams	425
File Streams	426
Text Mode versus Binary Mode	427
Jumping around with seek() and tell()	428
Linking Streams Together	430
Bidirectional I/O	431
Summary	432

CHAPTER 14: HANDLING ERRORS	433
Errors and Exceptions	434
What Are Exceptions, Anyway?	434
Why Exceptions in C++ Are a Good Thing	434
Recommendation	436
Exception Mechanics	436
Throwing and Catching Exceptions	437
Exception Types	439
Catching Exception Objects by const Reference	440
Throwing and Catching Multiple Exceptions	441
Matching and const	443
Matching Any Exception	443
Uncaught Exceptions	444
noexcept	445
Throw Lists (Deprecated/Removed)	446
Exceptions and Polymorphism	446
The Standard Exception Hierarchy	446
Catching Exceptions in a Class Hierarchy	448
Writing Your Own Exception Classes	449
Nested Exceptions	452
Rethrowing Exceptions	454
Stack Unwinding and Cleanup	456
Use Smart Pointers	457
Catch, Cleanup, and Rethrow	458
Common Error-Handling Issues	459
Memory Allocation Errors	459
Non-throwing new	460
Customizing Memory Allocation Failure Behavior	460
Errors in Constructors	462
Function-Try-Blocks for Constructors	464
Errors in Destructors	467
Putting It All Together	468
Summary	472
CHAPTER 15: OVERLOADING C++ OPERATORS	473
Overview of Operator Overloading	474
Why Overload Operators?	474
Limitations to Operator Overloading	474
Choices in Operator Overloading	475
Method or Global Function	475

Choosing Argument Types	476
Choosing Return Types	477
Choosing Behavior	477
Operators You Shouldn't Overload	477
Summary of Overloadable Operators	478
Rvalue References	481
Relational Operators	482
Overloading the Arithmetic Operators	483
Overloading Unary Minus and Unary Plus	483
Overloading Increment and Decrement	483
Overloading the Bitwise and Binary Logical Operators	484
Overloading the Insertion and Extraction Operators	485
Overloading the Subscripting Operator	486
Providing Read-Only Access with operator[]	489
Non-integral Array Indices	490
Overloading the Function Call Operator	491
Overloading the Dereferencing Operators	492
Implementing operator*	494
Implementing operator->	494
What in the World Are operator.* and operator->*?	495
Writing Conversion Operators	496
Solving Ambiguity Problems with Explicit Conversion Operators	497
Conversions for Boolean Expressions	498
Overloading the Memory Allocation and Deallocation Operators	500
How new and delete Really Work	500
The New-Expression and operator new	501
The Delete-Expression and operator delete	501
Overloading operator new and operator delete	501
Explicitly Deleting/Defaulting operator new and operator delete	504
Overloading operator new and operator delete with Extra Parameters	504
Overloading operator delete with Size of Memory as Parameter	505
Summary	506
CHAPTER 16: OVERVIEW OF THE C++ STANDARD LIBRARY	507
Coding Principles	508
Use of Templates	508
Use of Operator Overloading	509
Overview of the C++ Standard Library	509
Strings	509
Regular Expressions	510
I/O Streams	510

Smart Pointers	510
Exceptions	510
Mathematical Utilities	511
Time Utilities	512
Random Numbers	512
Initializer Lists	512
Pair and Tuple	512
optional, variant, and any	512
Function Objects	513
Filesystem	513
Multithreading	513
Type Traits	513
Standard Integer Types	514
Containers	514
vector	514
list	515
forward_list	515
deque	516
array	516
queue	516
priority_queue	516
stack	517
set and multiset	517
map and multimap	518
Unordered Associative Containers/Hash Tables	518
bitset	519
Summary of Standard Library Containers	519
Algorithms	522
Non-modifying Sequence Algorithms	523
Modifying Sequence Algorithms	525
Operational Algorithms	527
Swap and Exchange Algorithms	527
Partition Algorithms	527
Sorting Algorithms	528
Binary Search Algorithms	529
Set Algorithms	529
Heap Algorithms	529
Minimum/Maximum Algorithms	530
Numerical Processing Algorithms	530

Permutation Algorithms	532
Choosing an Algorithm	532
What's Missing from the Standard Library	532
Summary	533
CHAPTER 17: UNDERSTANDING CONTAINERS AND ITERATORS	535
Containers Overview	536
Requirements on Elements	537
Exceptions and Error Checking	539
Iterators	539
Sequential Containers	542
vector	542
vector Overview	542
vector Details	544
vector Example: A Round-Robin Class	556
The vector<bool> Specialization	561
deque	562
list	562
Accessing Elements	562
Iterators	562
Adding and Removing Elements	563
list Size	563
Special list Operations	563
list Example: Determining Enrollment	565
forward_list	566
array	568
Container Adaptors	569
queue	570
queue Operations	570
queue Example: A Network Packet Buffer	570
priority_queue	572
priority_queue Operations	573
priority_queue Example: An Error Correlator	573
stack	575
stack Operations	575
stack Example: Revised Error Correlator	575
Ordered Associative Containers	576
The pair Utility Class	576

map	577
Constructing maps	577
Inserting Elements	578
map Iterators	580
Looking Up Elements	581
Removing Elements	582
Nodes	582
map Example: Bank Account	583
multimap	585
multimap Example: Buddy Lists	586
set	589
set Example: Access Control List	589
multiset	590
Unordered Associative Containers or Hash Tables	591
Hash Functions	591
unordered_map	593
unordered_map Example: Phone Book	596
unordered_multimap	597
unordered_set/unordered_multiset	598
Other Containers	598
Standard C-Style Arrays	598
Strings	599
Streams	600
bitset	600
bitset Basics	600
Bitwise Operators	601
bitset Example: Representing Cable Channels	601
Summary	605
CHAPTER 18: MASTERING STANDARD LIBRARY ALGORITHMS	607
<hr/>	
Overview of Algorithms	608
The find and find_if Algorithms	608
The accumulate Algorithm	611
Move Semantics with Algorithms	612
std::function	612
Lambda Expressions	614
Syntax	614
Generic Lambda Expressions	617
Lambda Capture Expressions	618
Lambda Expressions as Return Type	618
Lambda Expressions as Parameters	619

Examples with Standard Library Algorithms	619
count_if	619
generate	620
Function Objects	620
Arithmetic Function Objects	621
Transparent Operator Functors	622
Comparison Function Objects	622
Logical Function Objects	623
Bitwise Function Objects	624
Adaptor Function Objects	624
Binders	624
Negators	626
Calling Member Functions	628
Invokers	629
Writing Your Own Function Objects	629
Algorithm Details	630
Iterators	631
Non-modifying Sequence Algorithms	631
Search Algorithms	631
Specialized Searchers	633
Comparison Algorithms	634
Counting Algorithms	636
Modifying Sequence Algorithms	636
transform	637
copy	638
move	640
replace	641
remove	641
unique	643
sample	643
reverse	644
shuffle	644
Operational Algorithms	644
for_each	644
for_each_n	646
Swap and Exchange Algorithms	646
swap	646
exchange	647
Partition Algorithms	647
Sorting Algorithms	649
Binary Search Algorithms	649

Set Algorithms	650
Minimum/Maximum Algorithms	653
Parallel Algorithms	655
Numerical Processing Algorithms	655
inner_product	656
iota	656
gcd and lcm	656
reduce	656
transform_reduce	657
Scan Algorithms	657
Algorithms Example: Auditing Voter Registrations	657
The Voter Registration Audit Problem Statement	658
The auditVoterRolls Function	658
The getDuplicates Function	659
Testing the auditVoterRolls Function	660
Summary	661
 CHAPTER 19: STRING LOCALIZATION AND REGULAR EXPRESSIONS	 663
Localization	663
Localizing String Literals	664
Wide Characters	664
Non-Western Character Sets	665
Conversions	667
Locales and Facets	668
Using Locales	668
Character Classification	669
Character Conversion	670
Using Facets	670
Regular Expressions	671
ECMAScript Syntax	672
Anchors	673
Wildcards	673
Alternation	673
Grouping	673
Repetition	673
Precedence	674
Character Set Matches	674
Word Boundaries	676
Back References	677

Lookahead	677
Regular Expressions and Raw String Literals	677
The regex Library	678
regex_match()	679
regex_match() Example	680
regex_search()	682
regex_search() Example	683
regex_iterator	683
regex_iterator Example	684
regex_token_iterator	685
regex_token_iterator Examples	685
regex_replace()	687
regex_replace() Examples	688
Summary	690
CHAPTER 20: ADDITIONAL LIBRARY UTILITIES	691
Ratios	691
The Chrono Library	694
Duration	694
Clock	698
Time Point	700
Random Number Generation	702
Random Number Engines	703
Random Number Engine Adaptors	705
Predefined Engines and Engine Adaptors	705
Generating Random Numbers	706
Random Number Distributions	708
optional	711
variant	712
any	713
Tuples	714
Decompose Tuples	717
Structured Bindings	717
tie	717
Concatenation	718
Comparisons	718
make_from_tuple	719
apply	719
Filesystem Support Library	720
Path	720

Directory Entry	721
Helper Functions	721
Directory Iteration	722
Summary	723

PART IV: MASTERING ADVANCED FEATURES OF C++

CHAPTER 21: CUSTOMIZING AND EXTENDING THE STANDARD LIBRARY **727**

Allocators	728
Stream Iterators	729
Output Stream Iterator	729
Input Stream Iterator	730
Iterator Adaptors	730
Reverse Iterators	730
Insert Iterators	731
Move Iterators	733
Extending the Standard Library	735
Why Extend the Standard Library?	735
Writing a Standard Library Algorithm	735
find_all()	735
Iterator Traits	737
Writing a Standard Library Container	737
A Basic Hash Map	738
Making hash_map a Standard Library Container	747
Note on Allocators	760
Note on Reversible Containers	760
Making hash_map an Unordered Associative Container	760
Note on Sequential Containers	773
Summary	773

CHAPTER 22: ADVANCED TEMPLATES **775**

More about Template Parameters	776
More about Template Type Parameters	776
Introducing Template Template Parameters	778
More about Non-type Template Parameters	780
Class Template Partial Specialization	782
Emulating Function Partial Specialization with Overloading	786
Template Recursion	787
An N-Dimensional Grid: First Attempt	788
A Real N-Dimensional Grid	789

Variadic Templates	792
Type-Safe Variable-Length Argument Lists	792
Variable Number of Mixin Classes	795
Folding Expressions	796
Metaprogramming	797
Factorial at Compile Time	798
Loop Unrolling	799
Printing Tuples	800
constexpr if	802
Using a Compile-Time Integer Sequence with Folding	803
Type Traits	803
Using Type Categories	805
Using Type Relations	807
Using enable_if	808
Using constexpr if to Simplify enable_if Constructs	810
Logical Operator Traits	811
Metaprogramming Conclusion	811
Summary	812
 CHAPTER 23: MULTITHREADED PROGRAMMING WITH C++	 813
Introduction	814
Race Conditions	815
Tearing	817
Deadlocks	817
False-Sharing	818
Threads	819
Thread with Function Pointer	819
Thread with Function Object	820
Thread with Lambda	822
Thread with Member Function	823
Thread Local Storage	823
Cancelling Threads	824
Retrieving Results from Threads	824
Copying and Rethrowing Exceptions	824
Atomic Operations Library	827
Atomic Type Example	828
Atomic Operations	830
Mutual Exclusion	831
Mutex Classes	831
Non-timed Mutex Classes	832
Timed Mutex Classes	832

Locks	833
lock_guard	833
unique_lock	834
shared_lock	835
Acquiring Multiple Locks at Once	835
scoped_lock	835
std::call_once	836
Examples Using Mutual Exclusion Objects	837
Thread-Safe Writing to Streams	837
Using Timed Locks	838
Double-Checked Locking	839
Condition Variables	840
Spurious Wake-Ups	841
Using Condition Variables	841
Futures	843
std::promise and std::future	843
std::packaged_task	844
std::async	845
Exception Handling	846
std::shared_future	847
Example: Multithreaded Logger Class	848
Thread Pools	853
Threading Design and Best Practices	853
Summary	855

PART V: C++ SOFTWARE ENGINEERING

CHAPTER 24: MAXIMIZING SOFTWARE ENGINEERING METHODS 859

The Need for Process	860
Software Life Cycle Models	861
The Waterfall Model	861
Benefits of the Waterfall Model	862
Drawbacks of the Waterfall Model	862
Sashimi Model	863
Spiral-Like Models	863
Benefits of a Spiral-Like Model	864
Drawbacks of a Spiral-Like Model	866
Agile	866
Software Engineering Methodologies	867
The Unified Process	867
The Rational Unified Process	868

RUP as a Product	868
RUP as a Process	869
RUP in Practice	869
Scrum	869
Roles	870
The Process	870
Benefits of Scrum	871
Drawbacks of Scrum	872
Extreme Programming	872
XP in Theory	872
XP in Practice	876
Software Triage	876
Building Your Own Process and Methodology	877
Be Open to New Ideas	877
Bring New Ideas to the Table	877
Recognize What Works and What Doesn't Work	877
Don't Be a Renegade	878
Source Code Control	878
Summary	880
CHAPTER 25: WRITING EFFICIENT C++	881
Overview of Performance and Efficiency	882
Two Approaches to Efficiency	882
Two Kinds of Programs	882
Is C++ an Inefficient Language?	882
Language-Level Efficiency	883
Handle Objects Efficiently	884
Pass-by-Reference	884
Return-by-Reference	886
Catch Exceptions by Reference	886
Use Move Semantics	886
Avoid Creating Temporary Objects	886
The Return-Value Optimization	887
Pre-allocate Memory	888
Use Inline Methods and Functions	888
Design-Level Efficiency	889
Cache Where Necessary	889
Use Object Pools	890
An Object Pool Implementation	891
Using the Object Pool	893

Profiling	894
Profiling Example with gprof	895
First Design Attempt	895
Profiling the First Design Attempt	898
Second Design Attempt	900
Profiling the Second Design Attempt	901
Profiling Example with Visual C++ 2017	902
Summary	907

CHAPTER 26: BECOMING ADEPT AT TESTING	909
--	------------

Quality Control	910
Whose Responsibility Is Testing?	910
The Life Cycle of a Bug	910
Bug-Tracking Tools	912
Unit Testing	913
Approaches to Unit Testing	914
The Unit Testing Process	915
Define the Granularity of Your Tests	915
Brainstorm the Individual Tests	917
Create Sample Data and Results	918
Write the Tests	918
Run the Tests	919
Unit Testing in Action	919
Introducing the Microsoft Visual C++ Testing Framework	920
Writing the First Test	921
Building and Running Tests	922
Negative Tests	923
Adding the Real Tests	923
Debugging Tests	927
Basking in the Glorious Light of Unit Test Results	927
Higher-Level Testing	927
Integration Tests	928
Sample Integration Tests	928
Methods of Integration Testing	929
System Tests	929
Regression Tests	930
Tips for Successful Testing	930
Summary	931

CHAPTER 27: CONQUERING DEBUGGING	933
The Fundamental Law of Debugging	934
Bug Taxonomies	934
Avoiding Bugs	934
Planning for Bugs	935
Error Logging	935
Debug Traces	937
Debug Mode	937
Ring Buffers	942
Assertions	945
Crash Dumps	946
Static Assertions	947
Debugging Techniques	948
Reproducing Bugs	948
Debugging Reproducible Bugs	949
Debugging Nonreproducible Bugs	950
Debugging Regressions	951
Debugging Memory Problems	951
Categories of Memory Errors	952
Tips for Debugging Memory Errors	954
Debugging Multithreaded Programs	956
Debugging Example: Article Citations	957
Buggy Implementation of an ArticleCitations Class	957
Testing the ArticleCitations class	960
Lessons from the ArticleCitations Example	969
Summary	969
CHAPTER 28: INCORPORATING DESIGN TECHNIQUES AND FRAMEWORKS	971
“I Can Never Remember How to...”	972
...Write a Class	972
...Derive from an Existing Class	974
...Use the Copy-and-Swap Idiom	975
...Throw and Catch Exceptions	976
...Read from a File	976
...Write to a File	977
...Write a Template Class	977
There Must Be a Better Way	979
Resource Acquisition Is Initialization	979
Double Dispatch	981

Attempt #1: Brute Force	982
Attempt #2: Single Polymorphism with Overloading	983
Attempt #3: Double Dispatch	984
Mixin Classes	985
Designing a Mixin Class	986
Implementing a Mixin Class	987
Using a Mixin Class	988
Object-Oriented Frameworks	988
Working with Frameworks	988
The Model-View-Controller Paradigm	989
Summary	990

CHAPTER 29: APPLYING DESIGN PATTERNS **991**

The Iterator Pattern	992
The Singleton Pattern	993
Example: A Logging Mechanism	993
Implementation of a Singleton	994
Using a Singleton	997
The Abstract Factory Pattern	997
Example: A Car Factory Simulation	998
Implementation of a Factory	999
Using a Factory	1002
Other Uses of Factories	1003
The Proxy Pattern	1004
Example: Hiding Network Connectivity Issues	1004
Implementation of a Proxy	1004
Using a Proxy	1005
The Adaptor Pattern	1006
Example: Adapting a Logger Class	1006
Implementation of an Adaptor	1007
Using an Adaptor	1008
The Decorator Pattern	1008
Example: Defining Styles in Web Pages	1008
Implementation of a Decorator	1009
Using a Decorator	1010
The Chain of Responsibility Pattern	1010
Example: Event Handling	1011
Implementation of a Chain of Responsibility	1011
Chain of Responsibility without Hierarchy	1012
The Observer Pattern	1014
Implementation of an Observer	1014

Implementation of an Observable	1015
Using an Observer	1016
Summary	1016
CHAPTER 30: DEVELOPING CROSS-PLATFORM AND CROSS- LANGUAGE APPLICATIONS	1017
<hr/>	
Cross-Platform Development	1018
Architecture Issues	1018
Size of Integers	1018
Binary Compatibility	1019
Address Sizes	1020
Byte Order	1020
Implementation Issues	1021
Compiler Quirks and Extensions	1021
Library Implementations	1022
Platform-Specific Features	1022
Cross-Language Development	1024
Mixing C and C++	1024
Shifting Paradigms	1024
Linking with C Code	1028
Calling C++ Code from C#	1030
Calling C++ Code from Java with JNI	1031
Calling Scripts from C++ Code	1033
Calling C++ Code from Scripts	1034
A Practical Example: Encrypting Passwords	1034
Calling Assembly Code from C++	1036
Summary	1038
APPENDIX A: C++ INTERVIEWS	1039
<hr/>	
APPENDIX B: ANNOTATED BIBLIOGRAPHY	1063
<hr/>	
APPENDIX C: STANDARD LIBRARY HEADER FILES	1075
<hr/>	
APPENDIX D: INTRODUCTION TO UML	1083
<hr/>	
INDEX	1087

INTRODUCTION

For many years, C++ has served as the de facto language for writing fast, powerful, and enterprise-class object-oriented programs. As popular as C++ has become, the language is surprisingly difficult to grasp in full. There are simple, but powerful, techniques that professional C++ programmers use that don't show up in traditional texts, and there are useful parts of C++ that remain a mystery even to experienced C++ programmers.

Too often, programming books focus on the syntax of the language instead of its real-world use. The typical C++ text introduces a major part of the language in each chapter, explaining the syntax and providing an example. *Professional C++* does not follow this pattern. Instead of giving you just the nuts and bolts of the language with little practical context, this book will teach you how to use C++ in the real world. It will show you the little-known features that will make your life easier, and the programming techniques that separate novices from professional programmers.

WHO THIS BOOK IS FOR

Even if you have used the language for years, you might still be unfamiliar with the more-advanced features of C++, or you might not be using the full capabilities of the language. Perhaps you write competent C++ code, but would like to learn more about design and good programming style in C++. Or maybe you're relatively new to C++, but want to learn the “right” way to program from the start. This book will meet those needs and bring your C++ skills to the professional level.

Because this book focuses on advancing from basic or intermediate knowledge of C++ to becoming a professional C++ programmer, it assumes that you have some knowledge of the language. Chapter 1 covers the basics of C++ as a refresher, but it is not a substitute for actual training and use of the language. If you are just starting with C++, but you have significant experience in another programming language such as C, Java, or C#, you should be able to pick up most of what you need from Chapter 1.

In any case, you should have a solid foundation in programming fundamentals. You should know about loops, functions, and variables. You should know how to structure a program, and you should be familiar with fundamental techniques such as recursion. You should have some knowledge of common data structures such as queues, and useful algorithms such as sorting and searching. You don't need to know about object-oriented programming just yet—that is covered in Chapter 5.

You will also need to be familiar with the compiler you will be using to develop your code. Two compilers, Microsoft Visual C++ and GCC, are introduced later in this introduction. For other compilers, refer to the documentation that came with your compiler.

WHAT THIS BOOK COVERS

Professional C++ uses an approach to C++ programming that will both increase the quality of your code and improve your programming efficiency. You will find discussions on new C++17 features throughout this fourth edition. These features are not just isolated to a few chapters or sections; instead, examples have been updated to use new features when appropriate.

Professional C++ teaches you more than just the syntax and language features of C++. It also emphasizes programming methodologies, reusable design patterns, and good programming style. The *Professional C++* methodology incorporates the entire software development process, from designing and writing code, to debugging, and working in groups. This approach will enable you to master the C++ language and its idiosyncrasies, as well as take advantage of its powerful capabilities for large-scale software development.

Imagine users who have learned all of the syntax of C++ without seeing a single example of its use. They know just enough to be dangerous! Without examples, they might assume that all code should go in the `main()` function of the program, or that all variables should be global—practices that are generally not considered hallmarks of good programming.

Professional C++ programmers understand the correct way to use the language, in addition to the syntax. They recognize the importance of good design, the theories of object-oriented programming, and the best ways to use existing libraries. They have also developed an arsenal of useful code and reusable ideas.

By reading and understanding this book, you will become a professional C++ programmer. You will expand your knowledge of C++ to cover lesser-known and often misunderstood language features. You will gain an appreciation for object-oriented design, and acquire top-notch debugging skills. Perhaps most important, you will finish this book armed with a wealth of reusable ideas that you can actually apply to your daily work.

There are many good reasons to make the effort to be a professional C++ programmer, as opposed to a programmer who knows C++. Understanding the true workings of the language will improve the quality of your code. Learning about different programming methodologies and processes will help you to work better with your team. Discovering reusable libraries and common design patterns will improve your daily efficiency and help you stop reinventing the wheel. All of these lessons will make you a better programmer and a more valuable employee. While this book can't guarantee you a promotion, it certainly won't hurt.

HOW THIS BOOK IS STRUCTURED

This book is made up of five parts.

Part I, “Introduction to Professional C++,” begins with a crash course in C++ basics to ensure a foundation of C++ knowledge. Following the crash course, Part I goes deeper into working with strings and string views because strings are used extensively in most examples throughout the book. The last chapter of Part I explores how to write *readable* C++ code.

Part II, “Professional C++ Software Design,” discusses C++ design methodologies. You will read about the importance of design, the object-oriented methodology, and the importance of code reuse.

Part III, “C++ Coding the Professional Way,” provides a technical tour of C++ from the professional point of view. You will read about the best ways to manage memory in C++, how to create reusable classes, and how to leverage important language features such as inheritance. You will also learn about the unusual and quirky parts of the language, techniques for input and output, error handling, string localization, and how to work with regular expressions. You will read about how to implement operator overloading, and how to write templates. This part also explains the C++ Standard Library, including containers, iterators, and algorithms. You will also read about some additional libraries that are available in the standard, such as the libraries to work with time, random numbers, and the filesystem.

Part IV, “Mastering Advanced Features of C++,” demonstrates how you can get the most out of C++. This part of the book exposes the mysteries of C++ and describes how to use some of its more-advanced features. You will read about how to customize and extend the C++ Standard Library to your needs, advanced details on template programming, including template metaprogramming, and how to use multithreading to take advantage of multiprocessor and multicore systems.

Part V, “C++ Software Engineering,” focuses on writing enterprise-quality software. You’ll read about the engineering practices being used by programming organizations today; how to write efficient C++ code; software testing concepts, such as unit testing and regression testing; techniques used to debug C++ programs; how to incorporate design techniques, frameworks, and conceptual object-oriented design patterns into your own code; and solutions for cross-language and cross-platform code.

The book concludes with a useful chapter-by-chapter guide to succeeding in a C++ technical interview, an annotated bibliography, a summary of the C++ header files available in the standard, and a brief introduction to the Unified Modeling Language (UML).

This book is not a reference of every single class, method, and function available in C++. The book *C++ Standard Library Quick Reference* by Peter Van Weert and Marc Gregoire¹ is a condensed reference to all essential data structures, algorithms, and functions provided by the C++ Standard Library. Appendix B lists a couple more references. Two excellent online references are:

- www.cppreference.com

You can use this reference online, or download an offline version for use when you are not connected to the Internet.

- www.cplusplus.com/reference/

When I refer to a “Standard Library Reference” in this book, I am referring to one of these detailed C++ references.

¹Apress, 2016. ISBN: 978-1-4842-1875-4.

WHAT YOU NEED TO USE THIS BOOK

All you need to use this book is a computer with a C++ compiler. This book focuses only on parts of C++ that have been standardized, and not on vendor-specific compiler extensions.

Note that this book includes new features introduced with the C++17 standard. At the time of this writing, some compilers are not yet fully C++17 compliant.

You can use whichever C++ compiler you like. If you don't have a C++ compiler yet, you can download one for free. There are a lot of choices. For example, for Windows, you can download Microsoft Visual Studio 2017 Community Edition, which is free and includes Visual C++. For Linux, you can use GCC or Clang, which are also free.

The following two sections briefly explain how to use Visual C++ and GCC. Refer to the documentation that came with your compiler for more details.

Microsoft Visual C++

First, you need to create a project. Start Visual C++ and click File ⇨ New ⇨ Project. In the project template tree on the left, select Visual C++ ⇨ Win32 (or Windows Desktop). Then select the Win32 Console Application (or Windows Console Application) template in the list in the middle of the window. At the bottom, specify a name for the project and a location where to save it, and click OK. A wizard opens². In this wizard, click Next, select Console Application, Empty Project, and click Finish.

Once your new project is loaded, you can see a list of project files in the Solution Explorer. If this docking window is not visible, go to View ⇨ Solution Explorer. You can add new files or existing files to a project by right-clicking the project name in the Solution Explorer and then clicking Add ⇨ New Item or Add ⇨ Existing Item.

Use Build ⇨ Build Solution to compile your code. When it compiles without errors, you can run it with Debug ⇨ Start Debugging.

If your program exits before you have a chance to view the output, use Debug ⇨ Start without Debugging. This adds a pause to the end of the program so you can view the output.

At the time of this writing, Visual C++ 2017 does not yet automatically enable C++17 features. To enable C++17 features, in the Solution Explorer window, right-click your project and click Properties. In the properties window, go to Configuration Properties ⇨ C/C++ ⇨ Language, and set the C++ Language Standard option to “ISO C++17 Standard” or “ISO C++ Latest Draft Standard,” whichever is available in your version of Visual C++. These options are only accessible if your project contains at least one .cpp file.

Visual C++ supports so-called precompiled headers, a topic outside the scope of this book. In general, I recommend using precompiled headers if your compiler supports them. However, the source

²Depending on your version of VC++ 2017, you might not see any wizard. Instead, a new project will be created automatically containing four files: stdafx.h, stdafx.cpp, targetver.h, and <projectname>.cpp. If that is the case, and you want to compile source code files from the downloadable source archive for this book, then you have to select those files in the Solution Explorer (View ⇨ Solution Explorer) and delete them.

code files in the downloadable source code archive do not use precompiled headers, so you have to disable that feature for them to compile without errors. In the Solution Explorer window, right-click your project and click Properties. In the properties window, go to Configuration Properties ⇄ C/C++ ⇄ Precompiled Headers, and set the Precompiled Header option to “Not Using Precompiled Headers.”

GCC

Create your source code files with any text editor you prefer and save them to a directory. To compile your code, open a terminal and run the following command, specifying all your .cpp files that you want to compile:

```
gcc -lstdc++ -std=c++17 -o <executable_name> <source1.cpp> [source2.cpp ...]
```

The `-std=c++17` option is required to tell GCC to enable C++17 support.

For example, you can compile the `AirlineTicket` example from Chapter 1 by changing to the directory containing the code and running the following command:

```
gcc -lstdc++ -std=c++17 -o AirlineTicket AirlineTicket.cpp AirlineTicketTest.cpp
```

When it compiles without errors, you can run it as follows:

```
./AirlineTicket
```

CONVENTIONS

To help you get the most from the text and keep track of what’s happening, a number of conventions are used throughout this book.

WARNING *Boxes like this one hold important, not-to-be-forgotten information that is directly relevant to the surrounding text.*

NOTE *Tips, hints, tricks, and asides to the current discussion are placed in boxes like this one.*

As for styles in the text:

Important words are *highlighted* when they are introduced.

Keyboard strokes are shown like this: Ctrl+A.

Filenames and code within the text are shown like so: `monkey.cpp`.

URLs are shown like this: `www.wrox.com`.

Code is presented in three different ways:

```
// Comments in code are shown like this.
```

In code examples, new and important code is highlighted like this.

Code that's less important in the present context or that has been shown before is formatted like this.

Paragraphs or sections that are specific to the C++17 standard have a little C++17 icon on the left, just as this paragraph does. C++11 and C++14 features are not marked with any icon.

SOURCE CODE

As you work through the examples in this book, you may choose either to type in all the code manually, or to use the source code files that accompany the book. However, I suggest you type in all the code manually because it greatly benefits the learning process and your memory. All of the source code used in this book is available for download at www.wiley.com/go/proc++4e.

Alternatively, you can go to the main Wrox code download page at www.wrox.com/dynamic/books/download.aspx to see the code that is available for this book and all other Wrox books.

NOTE *Because many books have similar titles, you may find it easiest to search by ISBN; for this book, the ISBN is 978-1-119-42130-6.*

Once you've downloaded the code, just decompress it with your favorite decompression tool.

ERRATA

At Wrox, we make every effort to ensure that there are no errors in the text or in the code of our books. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, such as a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata, you may save another reader hours of frustration, and at the same time you will be helping us provide even higher-quality information.

To find the errata page for this book, go to www.wrox.com and locate the title by using the Search box or one of the title lists. Then, on the book details page, click the Book Errata link. On this page you can view all errata that has been submitted for this book and posted by Wrox editors. A complete book list, including links to each book's errata, is also available at www.wrox.com/misc-pages/booklist.shtml.

If you don't spot "your" error on the Book Errata page, go to www.wrox.com/contact/techsupport.shtml and complete the form there to send us the error you have found. We'll check the information and, if appropriate, post a message to the book's errata page and fix the problem in subsequent editions of the book.