

Join the discussion @ p2p.wrox.com



FOURTH 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 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