

Exploring C++

The Programmer's Introduction to C++



Ray Lischner

Exploring C++: The Programmer's Introduction to C++

Copyright © 2009 by Ray Lischner

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-59059-749-1

ISBN-10 (pbk): 1-59059-749-4

ISBN-13 (electronic): 978-1-4302-1895-1

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Matthew Moodie

Technical Reviewer: Francis Glassborow

Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell,

Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper,

Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Project Manager: Richard Dal Porto

Copy Editor: Octal Publishing, Inc.

Associate Production Director: Kari Brooks-Copony

Production Editor: Ellie Fountain

Compositor: Dina Quan

Proofreaders: Lisa Hamilton and Linda Seifert

Indexer: John Collin

Cover Designer: Kurt Krames

Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit <http://www.springeronline.com>.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit <http://www.apress.com>.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at <http://www.apress.com/info/bulksales>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at <http://www.apress.com>. You may need to answer questions pertaining to this book in order to successfully download the code.

Contents at a Glance

About the Author	xxi
About the Technical Reviewer	xxiii
Acknowledgments	xxv
Introduction	xxvii

PART 1 ■ ■ ■ The Basics

■ EXPLORATION 1	Honing Your Tools	3
■ EXPLORATION 2	Reading C++ Code	13
■ EXPLORATION 3	Integer Expressions	21
■ EXPLORATION 4	Strings	27
■ EXPLORATION 5	Simple Input	33
■ EXPLORATION 6	Error Messages	39
■ EXPLORATION 7	For Loops	45
■ EXPLORATION 8	Formatted Output	49
■ EXPLORATION 9	Arrays and Vectors	59
■ EXPLORATION 10	Increment and Decrement	71
■ EXPLORATION 11	Conditions and Logic	77
■ EXPLORATION 12	Compound Statements	85
■ EXPLORATION 13	Introduction to File I/O	95
■ EXPLORATION 14	The Map Data Structure	99
■ EXPLORATION 15	Type Synonyms	105
■ EXPLORATION 16	Characters	109
■ EXPLORATION 17	Character Categories	115
■ EXPLORATION 18	Case-Folding	123
■ EXPLORATION 19	Writing Functions	127
■ EXPLORATION 20	Function Arguments	137

EXPLORATION 21	Using Algorithms	145
EXPLORATION 22	Overloading Function Names	157
EXPLORATION 23	Big and Little Numbers	163
EXPLORATION 24	Very Big and Very Little Numbers	173
EXPLORATION 25	Documentation	181
EXPLORATION 26	Project 1: Body-Mass Index	189

PART 2 ■ ■ ■ Custom Types

EXPLORATION 27	Custom Types	195
EXPLORATION 28	Overloading Operators	203
EXPLORATION 29	Custom I/O Operators	213
EXPLORATION 30	Assignment and Initialization	219
EXPLORATION 31	Writing Classes	227
EXPLORATION 32	More About Member Functions	235
EXPLORATION 33	Access Levels	247
EXPLORATION 34	Introduction to Object-Oriented Programming	257
EXPLORATION 35	Inheritance	265
EXPLORATION 36	Virtual Functions	275
EXPLORATION 37	Classes and Types	283
EXPLORATION 38	Declarations and Definitions	293
EXPLORATION 39	Using Multiple Source Files	305
EXPLORATION 40	Function Objects	321
EXPLORATION 41	Useful Algorithms	331
EXPLORATION 42	Iterators	345
EXPLORATION 43	Exceptions	357
EXPLORATION 44	More Operators	373
EXPLORATION 45	Project 2: Fixed-Point Numbers	385

PART 3 ■ ■ ■ Generic Programming

EXPLORATION 46	Function Templates	393
EXPLORATION 47	Class Templates	405
EXPLORATION 48	Template Specialization	415

EXPLORATION 49	Partial Specialization	423
EXPLORATION 50	Names and Namespaces	429
EXPLORATION 51	Containers	449
EXPLORATION 52	International Characters	471
EXPLORATION 53	Locales and Facets	481
EXPLORATION 54	Text I/O	495
EXPLORATION 55	Project 3: Currency Type	509

PART 4 ■ ■ ■ Real Programming

EXPLORATION 56	Pointers	513
EXPLORATION 57	Dynamic Memory	527
EXPLORATION 58	Exception-Safety	545
EXPLORATION 59	Old-Fashioned Arrays	557
EXPLORATION 60	Smart Pointers	567
EXPLORATION 61	Working with Bits	581
EXPLORATION 62	Enumerations	599
EXPLORATION 63	Multiple Inheritance	615
EXPLORATION 64	Traits and Policies	629
EXPLORATION 65	Names and Templates	645
EXPLORATION 66	Overloaded Functions	655
EXPLORATION 67	Metaprogramming	669
EXPLORATION 68	Project 4: Calculator	679
INDEX	681

Contents

About the Author	xxi
About the Technical Reviewer	xxiii
Acknowledgments	xxv
Introduction	xxvii

PART 1 ■ ■ ■ The Basics

■ EXPLORATION 1 Honing Your Tools	3
Ray's Recommendations	3
Microsoft Windows	3
Macintosh OS 9 and Earlier	4
Everyone Else	4
Read the Documentation	4
Your First Program	5
■ EXPLORATION 2 Reading C++ Code	13
Comments	14
Headers	15
Main Program	16
Variable Definitions	17
Statements	18
Output	19
■ EXPLORATION 3 Integer Expressions	21
■ EXPLORATION 4 Strings	27
■ EXPLORATION 5 Simple Input	33

EXPLORATION 6 Error Messages	39
Misspelling	40
Bogus Character	40
Unknown Operator	41
Unknown Name	42
Symbol Errors	42
Fun with Errors	43
EXPLORATION 7 For Loops	45
Bounded Loops	45
Initialization	46
Condition	47
Postiteration	47
How a for Loop Works	47
Your Turn	48
EXPLORATION 8 Formatted Output	49
The Problem	49
Field Width	50
Padding	52
std Prefix	52
Alignment	53
Exploring Formatting	53
Alternative Syntax	54
On Your Own	55
EXPLORATION 9 Arrays and Vectors	59
Using Vectors for Arrays	60
Vectors	60
Iterators	62
Algorithms	63
Member Types	65
Using Iterators and Algorithms	66
EXPLORATION 10 Increment and Decrement	71
Increment	71
Decrement	72

EXPLORATION 11 Conditions and Logic	77
I/O and bool	77
Boolean Type	78
Logic Operators	80
Old-Fashioned Syntax	82
Comparison Operators	82
EXPLORATION 12 Compound Statements	85
Statements	85
Local Definitions and Scope	89
Definitions in for Loop Headers	92
EXPLORATION 13 Introduction to File I/O	95
Reading Files	95
Writing Files	96
EXPLORATION 14 The Map Data Structure	99
Using Maps	100
Iterators	101
Searching in Maps	103
EXPLORATION 15 Type Synonyms	105
typedef Declarations	105
Common typedefs	106
EXPLORATION 16 Characters	109
Character Type	109
Character I/O	111
Newlines and Portability	113
Character Escapes	113
EXPLORATION 17 Character Categories	115
Character Sets	115
Character Categories	117
Locales	119

EXPLORATION 18 Case-Folding	123
Simple Cases	123
Harder Cases	125
EXPLORATION 19 Writing Functions	127
Functions	127
Function Call	129
Declarations and Definitions	130
Counting Words, Again	132
The main() Function	134
EXPLORATION 20 Function Arguments	137
Argument Passing	137
Pass-by-Reference	140
const References	142
const_iterator	144
Output Parameters	144
EXPLORATION 21 Using Algorithms	145
Transforming Data	145
Predicates	150
Other Algorithms	152
EXPLORATION 22 Overloading Function Names	157
Overloading	157
EXPLORATION 23 Big and Little Numbers	163
The Long and Short of It	163
Long Integers	164
Short Integers	165
Integer Literals	166
Byte-Sized Integers	167
Type Casting	167
Integer Arithmetic	169
Overload Resolution	169

■ EXPLORATION 24 Very Big and Very Little Numbers	173
Floating-Point Numbers	173
Floating-Point Literals	174
Floating-Point Traits	176
Floating-Point I/O	177
■ EXPLORATION 25 Documentation	181
Doxygen	181
Structured Comments	181
Documentation Tags	182
Using Doxygen	187
■ EXPLORATION 26 Project 1: Body-Mass Index	189
Hints	190

PART 2 ■ ■ ■ Custom Types

■ EXPLORATION 27 Custom Types	195
Defining a New Type	195
Member Functions	197
Constructors	200
Overloading Constructors	202
■ EXPLORATION 28 Overloading Operators	203
Comparing Rational Numbers	203
Arithmetic Operators	207
Math Functions	210
■ EXPLORATION 29 Custom I/O Operators	213
Input Operator	213
Output Operator	215
Error State	215

EXPLORATION 30 Assignment and Initialization	219
Assignment Operator	219
Constructors	221
Putting It All Together	222
EXPLORATION 31 Writing Classes	227
Anatomy of a Class	227
Member Functions	228
Constructor	230
EXPLORATION 32 More About Member Functions	235
Invoking the Default Constructor	235
Revisiting Project 1	238
const Member Functions	242
EXPLORATION 33 Access Levels	247
Public vs. Private	247
class vs. struct	250
Plain Old Data	251
Public or Private?	251
EXPLORATION 34 Introduction to Object-Oriented Programming	257
Books and Magazines	257
Classification	258
Inheritance	261
Liskov's Substitution Principle	262
Type Polymorphism	263
EXPLORATION 35 Inheritance	265
Deriving a Class	265
Destructors	268
Access Level	272
Programming Style	273

EXPLORATION 36 Virtual Functions	275
Type Polymorphism	275
Virtual Functions	278
References and Slices	280
Pure Virtual Functions	281
Virtual Destructors	282
EXPLORATION 37 Classes and Types	283
Classes vs. typedefs	283
Value Types	286
Copying	286
Assigning	287
Comparing	287
Resource Acquisition Is Initialization	290
EXPLORATION 38 Declarations and Definitions	293
Declaration vs. Definition	293
Inline Functions	295
Variable Declarations and Definitions	297
Static Variables	299
Static Data Members	301
Declarators	304
EXPLORATION 39 Using Multiple Source Files	305
Multiple Source Files	305
Declarations and Definitions	307
#include Files	309
Quotes and Brackets	310
Nested #include Directives	311
Include Guards	312
Documentation	313
extern Variables	316
Inline Functions	317
One-Definition Rule	317

EXPLORATION 40 Function Objects	321
The Function Call Operator	321
Function Objects	323
Recycling Member Functions	327
Generator Functor	328
EXPLORATION 41 Useful Algorithms	331
Searching	331
Linear Search Algorithms	331
Binary Search Algorithms	336
Comparing	339
Rearranging Data	341
Copying Data	343
Deleting Elements	343
Iterators	344
EXPLORATION 42 Iterators	345
Kinds of Iterators	345
Input Iterators	346
Output Iterators	346
Forward Iterators	347
Bidirectional Iterators	347
Random Access Iterators	347
Working with Iterators	348
const_iterator vs. const iterator	351
Error Messages	353
Specialized Iterators	354
The Importance of Being Iterator	355
EXPLORATION 43 Exceptions	357
Introducing Exceptions	357
Catching Exceptions	359
Throwing Exceptions	361
Program Stack	362
Standard Exceptions	366
I/O Exceptions	367
Custom Exceptions	368
Exceptional Advice	370

EXPLORATION 44 More Operators	373
Conditional Operator	373
Short-Circuit Operators	375
Comma Operator	376
Arithmetic Assignment Operators	378
Increment and Decrement	380
EXPLORATION 45 Project 2: Fixed-Point Numbers	385

PART 3 ■ ■ ■ Generic Programming

EXPLORATION 46 Function Templates	393
Generic Functions	393
Using Function Templates	394
Writing Function Templates	396
Template Parameters	398
Template Arguments	399
Declarations and Definitions	401
Member Function Templates	402
EXPLORATION 47 Class Templates	405
Parameterizing a Type	405
Parameterizing the rational Class	407
Using Class Templates	408
Overloaded Operators	410
Mixing Types	413
EXPLORATION 48 Template Specialization	415
Instantiation and Specialization	415
Custom Comparators	419
Specializing Function Templates	420
Traits	421

EXPLORATION 49 Partial Specialization	423
Degenerate Pairs	423
Partial Specialization	424
Partially Specializing Function Templates	425
Value Template Parameters	425
EXPLORATION 50 Names and Namespaces	429
Namespaces	429
Nested Namespaces	432
Global Namespace	434
The std Namespace	435
Using Namespaces	435
The using Directive	436
The using Declaration	438
The using Declaration in a Class	441
Unnamed Namespaces	442
Name Lookup	443
EXPLORATION 51 Containers	449
Properties of Containers	449
Technical Report 1	450
Member Types	451
What Can Go Into a Container	452
Inserting and Erasing	454
Inserting in a Sequence Container	454
Erasing From a Sequence Container	454
Inserting in an Associative Container	455
Erasing From an Associative Container	457
Exceptions	457
Iterators and References	458
Sequence Containers	460
The array Class Template	462
The deque Class Template	463
The list Class Template	464
The vector Class Template	465
Associative Containers	466

■ EXPLORATION 52 International Characters	471
Why Wide?	471
Using Wide Characters	471
Wide Strings	472
Wide Character I/O	475
Multi-Byte Character Sets	476
Unicode	477
Universal Character Names	479
■ EXPLORATION 53 Locales and Facets	481
The Problem	481
Locales to the Rescue	482
Locales and I/O	483
Facets	483
Character Categories	487
Collation Order	492
■ EXPLORATION 54 Text I/O	495
File Modes	495
String Streams	496
Text Conversion	503
Boost Lexical Cast	507
■ EXPLORATION 55 Project 3: Currency Type	509

PART 4 ■ ■ ■ Real Programming

■ EXPLORATION 56 Pointers	513
The Problem	513
The Solution	522
Addresses vs. Pointers	523
Dependency Graphs	524

EXPLORATION 57 Dynamic Memory	527
Allocating Memory	527
Freeing Memory	528
Pointer to Nothing	528
Implementing Standard Containers	530
Adding Variables	532
Special Member Functions	540
EXPLORATION 58 Exception-Safety	545
Memory Leaks	545
Exceptions and Dynamic Memory	547
Automatically Deleting Pointers	549
What You Can't Do with <code>auto_ptr</code>	552
Exceptions and Constructors	552
EXPLORATION 59 Old-Fashioned Arrays	557
C-Style Arrays	557
Array Limitations	558
Dynamically Allocating an Array	559
Multi-Dimensional Arrays	561
C-Style Strings	561
Command-Line Arguments	562
Pointer Arithmetic	564
EXPLORATION 60 Smart Pointers	567
Revisiting <code>auto_ptr</code>	567
Copyable Smart Pointers	569
Smart Arrays	571
Pimpls	572
Iterators	580
EXPLORATION 61 Working with Bits	581
Integer As a Set of Bits	581
Bit Masks	583
Shifting Bits	584

Safe Shifting with Unsigned Types	585
Signed and Unsigned Types	586
Unsigned Literals	587
Type Conversions	587
Overflow	592
Introducing Bitfields	593
Portability	594
The bitset Class Template	594
EXPLORATION 62 Enumerations	599
Ideal Enumerations	599
Enumerations As Bitmasks	600
Simulating Enumerations	602
Enumerating Computer Languages	602
Comparing Languages	603
Assignment	604
Strings and Languages	605
Initializing	610
Reading and Writing Languages	611
Using the Simulated Enumeration	611
Revisiting Projects	613
EXPLORATION 63 Multiple Inheritance	615
Multiple Base Classes	615
Virtual Base Classes	618
Java-Like Interfaces	620
Interfaces vs. Templates	622
Mix-Ins	624
Friends to the Rescue	625
Protected Access Level	627
EXPLORATION 64 Traits and Policies	629
Case Study: Iterators	629
Iterator Traits	634
Case Study: char_traits	635
Policy-Based Programming	636

EXPLORATION 65 Names and Templates	645
Problems with Qualified Names	645
Problems with Unqualified Names	647
EXPLORATION 66 Overloaded Functions	655
Review of Overloaded Functions	655
Overload Resolution	658
Candidate Functions	659
Viable Functions	660
Best Viable Function	661
Default Arguments	665
EXPLORATION 67 Metaprogramming	669
Compile-Time Programming	669
Template Specialization	669
Partial Specialization	671
EXPLORATION 68 Project 4: Calculator	679
INDEX	681

About the Author

■ **RAY LISCHNER** is the author of *C++ in a Nutshell* and other books. He has been programming for over three decades, using languages as diverse as Algol, APL, Bash, C, C++, COBOL, csh, DCL, Delphi, Eiffel, Fortran, Haskell, Icon, Java, LISP, Pascal, Perl, PHP, PL/I, Python, Ruby, Scheme, Smalltalk, and a variety of assemblers.

In the years after he graduated from Caltech (in the mid-1980s), Ray worked as a software developer on both coasts of the United States, with stops in between. He has worked at companies big and small: from start-ups to Fortune 500. Not so very long ago, he decided to escape from the corporate rat race. Due to a minor error in timing, he quit before he figured out how to pay for such trivialities as food and shelter. Undaunted, he persevered and soon discovered writing as a means to keep the creditors at bay.

Ray has always enjoyed teaching. While his wife completed her Ph.D. in physics, he occupied his time teaching computer science at Oregon State University. Dissatisfied with the traditional introductory computer science curriculum, he revamped the first programming course and introduced novel lecture and teaching techniques. He pioneered interactive teaching labs—the genesis of this book.

Today, Ray lives in Maryland with his wife and two children. Ray has returned to full-time work as a software developer at Proteus Technologies, where he is the resident C++ expert. Writing has become a part-time endeavor. When he isn't working, Ray has a variety of other part-time, unpaid jobs: chef, taxi driver, house cleaner, arbitrator, soccer coach, banker, tooth fairy, story reader, and chief enforcer of the domestic nocturnal preparation procedure, just to name a few.

The best way to contact Ray is via email to exploring@cpphelp.com.

About the Technical Reviewer

■ **FRANCIS GLASSBOROW** read Mathematics at Merton College, Oxford. He spent 25 years teaching mathematics and computing studies to teenagers. Ill health forced him to retire from teaching. In 1989 he became editor of C Vu (ACCU's principal journal) which job/office he held until 2002. He was chair of ACCU throughout the 1990s. He was a regular columnist for .EXE magazine (a UK developers magazine) from 1991 until it ceased publication in August 2000. He was chair of the annual ACCU conference for seven years. He has been an active member of the BSI panels for C and C++ since 1991 and is a regular member of the BSI delegations to SC22/WG21 (ISO C++) and SC22/WG14 (ISO C) and is frequently HoD for these meetings. He is the author of *You Can Do It!*, an introduction for novice programmers, and *You Can Program in C++*, an introduction to C++ for those who can already program.

Acknowledgments

This book has been more difficult and time consuming than any of my previous books. I appreciate beyond words the patience of Apress and the staff who helped bring this project to fruition: particularly Matthew Moodie and Richard Dal Porto.

I am especially grateful to my Technical Reviewer, Francis Glassborow, whose deep knowledge of C++ has saved me from mistakes big and small. If any technical errors remain, no doubt I introduced them too late in the editing cycle for Francis to spot them.

Most of all, I thank my wife, Cheryl, whose support and encouragement sustained me when I lacked the strength to carry on. I also thank my children who put up with days and evenings without me while I finished this book. I love you all.

Finally, I thank the scientists and doctors who have worked miracles in the treatment of rheumatoid arthritis, permitting me to continue to work, write, and play.

Introduction

Hi, there. Thank you for reading my book, *Exploring C++*. My name is Ray, and I'll be your author today. And tomorrow. And the day after that. We'll be together for quite a while, so why don't you pull up a chair and get comfortable. My job is to help you learn C++. To do that, I have written a series of lessons, called *explorations*. Each exploration is an interactive exercise that helps you learn C++ one step at a time. Your job is to complete the explorations, and in so doing, learn C++.

No doubt you have already leafed through the book a little bit. If not, do so now. Notice that this book is different from most books. Most programming books are little more than written lectures. The author tells you stuff and expects you to read the stuff, learn it, and understand it.

This book is different. I don't see much point in lecturing at you. That's not how people learn best. You learn programming by reading, modifying, and writing programs. To that end, I've organized this book so that you spend as much time as possible reading, modifying, and writing programs.

How to Use This Book

Each exploration in this book is a mixture of text and interactive exercises. The exercises are unlike anything you've seen in other books. Instead of multiple choice, fill-in-the-blank, or simple Q&A exercises, my lessons are interactive explorations of key C++ features. Early in the book, I will give you complete programs to work with. As you learn more C++, you will modify and extend programs. Pretty soon, you will write entire programs on your own.

By "interactive," I mean that I ask questions and you answer them. I do my best to respond to your answers throughout the lesson text. It sounds crazy, but by answering the questions, you will be learning C++. To help ensure you answer the questions, I leave space in this book for you to write your answers. I'm giving you permission to write in this book (unless you are borrowing the book from a library or friend). In fact, I encourage you to write all your answers in the book. Only by answering the questions will you learn the material properly.

Sometimes, the questions have no right answer. I pose the question to make you ponder it, perhaps to look at a familiar topic from a new perspective. Other times, the question has an unambiguous, correct answer. I always give the answer in the subsequent text, so don't skip ahead! Write your answer before you continue reading. Then and only then can you check your answer. Some questions are tricky or require information that I have not yet presented. In such cases, I expect your answer to be wrong, but that's okay. Don't worry. I won't be grading you. (If you are using this book as part of a formal class, your teacher should grade this book's exercises solely on whether you complete them, and never on whether your answer was correct. The teacher will have other exercises, quizzes, and tests to assess your progress in

the class.) And no fair looking ahead and writing down the “correct” answer. You don’t learn anything that way.

Ready? Let’s practice.

What is your most important task when reading this book?

This question does not have a single correct answer, but it does have a number of demonstrably wrong answers. I hope you wrote something similar to, “Completing every exercise” or “Understanding all the material.” Another good answer is, “Having fun.”

The Book’s Organization

C++ is a complicated language. To write even the most trivial program requires an understanding of many disparate aspects of the language. The language does not lend itself to neat compartmentalization into broad topics, such as functions, classes, statements, or expressions. This book, therefore, does not attempt such an organization. Instead, you learn C++ in small increments: a little bit of this, a little bit of that, some more of this, and pretty soon you will have accumulated enough knowledge to start writing nontrivial programs.

Roughly speaking, the book starts with basic expressions, declarations, and statements that are sufficient to work with simple programs. You learn how to use the standard library early in the book. Next, you learn to write your own functions, to write your own classes, to write your own templates, and then to write fairly sophisticated programs.

You won’t be an expert, however, when you finish this book. You will need much more practice, more exposure to the breadth and depth of the language and library, and more practice. You will also need more practice. And some more. You get the idea.

Who Should Read This Book

Read this book if you want to learn C++ and you already know at least one other programming language. You don’t need to know a specific language or technology, however. In particular, you don’t need to know C, nor do you need to know anything about object-oriented programming.

The C programming language influenced the design of many other languages, from PHP to Perl to AWK to C#, not to mention C++. As a result, many programmers who do not know C or C++ nonetheless find many language constructs hauntingly familiar. You might even feel confident enough to skip sections of this book that seem to cover old ground. Don’t do that! From the start, the lessons present language features that are unique to C++. In a few, isolated cases, I will tell you when it is safe to skip a section, and only that section. Even when a language feature is familiar, it might have subtle issues that are unique to C++.

The trap is most perilous for C programmers because C++ bears the greatest superficial similarity with C. C programmers, therefore, have the most to overcome. By design, many C programs are also valid C++ programs, leading the unwary C programmer into the trap of thinking that good C programs are also good C++ programs. In fact, C and C++ are distinct

languages, each with their own idioms and idiosyncrasies. To become an effective C++ programmer, you must learn the C++ way of programming. C programmers need to break some of their established habits and learn to avoid certain C features (such as arrays) in favor of better C++ idioms. The structure of this book helps you get started thinking in terms of C++, not C.

Projects

This book also contains four projects. The projects are opportunities to apply what you have learned. Each project is a realistic endeavor, based on the amount of C++ covered up to that point. I encourage you to try every project. Design your project using your favorite software design techniques. Remember to write test cases in addition to the source code. Do your best to make the code clean and readable, in addition to correct. After you are confident that your solution is finished, download the files from the book's web site, and compare your solution with mine.

Work Together

You can use this book alone, teaching yourself C++, or a teacher might adopt this book as a textbook for a formal course. You can also work with a partner. It's more fun to work with friends, and you'll learn more and faster by working together. Each of you needs your own copy of the book. Read the lessons and do the work on your own. If you have questions, discuss them with your partner, but answer the exercises on your own. Then compare answers with your partner. If your answers are different, discuss your reasoning. See if you can agree on a single answer before proceeding.

Work on the projects together. Maybe you can divide the work into two (or more) modules. Maybe one person codes and the other person checks. Maybe you'll practice some form of pair programming. Do whatever works best for you, but make sure you understand every line of code in the project. If you have asymmetric roles, be sure to swap roles for each project. Give everyone a chance to do everything.

For More Information

This book cannot teach you everything you need to know about C++. No single book can. After you finish this book, I encourage you to continue to read and write C++ programs, and to seek out other sources of information. To help guide you, this book has a dedicated web site, <http://cpphelp.com/exploring/>. The web site has links to other books, other web sites, mailing lists, newsgroups, FAQs, compilers, other tools, and more. You can also download all the source code for this book, so you can save yourself some typing.

Why Explorations?

In case you were wondering about the unusual nature of this book, rest assured that, "though this be madness, yet there is method in't."

The method is an approach to teaching and writing that I developed while I was teaching computer science at Oregon State University. I wanted to improve the quality of my teaching,

so I investigated research into learning and knowledge, especially scientific knowledge, and in particular, computer programming.

To summarize several decades of research: everyone constructs mental models of the world. We acquire knowledge by adding information to our models. The new information must always be in concert with the model. Sometimes, however, new information contradicts the model. In that case, we must adjust our models to accommodate the new information. Our brains are always at work, always taking in new information, always adjusting our mental models to fit.

As a result of this research, the emphasis in the classroom has shifted from teachers to students. In the past, teachers considered students to be empty vessels, waiting to be filled from the fount of the teacher's knowledge and wisdom. Students were passive recipients of information. Now we know better. Students are not passive, but active. Even when their outward appearance suggests otherwise, their brains are always at work, always absorbing new information and fitting that information into their mental models. The teacher's responsibility has changed from being the source of all wisdom to being an indirect manager of mental models. The teacher cannot manage those models directly, but can only create classroom situations in which students have the opportunity to adjust their own models.

Although the research has focused on teachers, the same applies to authors.

In other words, I cannot teach you C++, but I can create explorations that enable you to learn C++. Explorations are not the only way to apply research to learning and writing, but they are a technique that I have refined over several years of teaching and have found successful. Explorations work because

- They force you to participate actively in the learning process. It's too easy to read a book passively. The questions force you to confront new ideas and to fit them into your mental model. If you skip the questions, you might also skip a crucial addition to your model.
- They are small, so your model grows in easy steps. If you try to grasp too much new information at once, you are likely to incorporate incorrect information into your model. The longer that misinformation festers, the harder it will be to correct. I want to make sure your model is as accurate as possible at all times.
- They build on what you know. I don't toss out new concepts with the vain hope that you will automatically grasp them. Instead, I tie new concepts to old ones. I do my best to ensure that every concept has a strong anchor in your existing mental model.
- They help you learn by doing. Instead of spending the better part of a chapter reading how someone else solves a problem, you spend as much time as possible working hands-on with a program: modifying existing programs and writing new programs.

C++ is a complicated language, and learning C++ is not easy. In any group of C++ programmers, even simple questions can often provoke varied responses. Most C++ programmers' mental models of the language are not merely incomplete, but are flawed, sometimes in fundamental ways. My hope is that I can provide you with a solid foundation in C++, so that you can write interesting and correct programs, and most importantly, so that you can continue to learn and enjoy C++ for many years to come.

The C++ Standard

This book covers the current standard, namely, ISO/IEC 14882:2003 (E), *Programming languages — C++*. The 2003 edition of the standard is a bug-fix edition, containing corrections to and clarifications of the original 1998 edition. Most modern compilers do a decent job of conforming to the standard.

The standardization committee has also issued an addendum to the standard, adding regular expressions, mathematical functions, and a lot more. This addendum is an optional extension to the standard library called Technical Report 1, or TR1. Because it is optional, vendors are not required to implement it. Most vendors provide at least part of the library. A few implement TR1 in its entirety. You do not need TR1 support to use this book, but I point out a few cases where TR1 makes your life a little easier.

By issuing TR1 and having thousands of C++ developers use it, the standardization committee gained valuable practical experience to feed back into the next major revision of the C++ standard. Work on the next revision is underway as I write this. Depending on when you read this, their work may be complete. You may even have a compiler and library that conforms to the new release of the standard, which will likely be labeled ISO/IEC 14882:2010 (E).

Even if you have a brand new compiler, this book still has value. Many of the new features are advanced so they don't affect this book. Other planned features impact C++ programmers of all levels and abilities. I point out the proposed changes throughout this book, but keep my focus on the tools that are available and in widespread use today.

