



Exploring C++20

The Programmer's Introduction to C++

—

Third Edition

—

Ray Lischner

Apress®

Exploring C++20

The Programmer's Introduction to C++

Third Edition



Ray Lischner

Apress®

Exploring C++20: The Programmer's Introduction to C++

Ray Lischner
Ellicott City, MD, USA

ISBN-13 (pbk): 978-1-4842-5960-3
<https://doi.org/10.1007/978-1-4842-5961-0>

ISBN-13 (electronic): 978-1-4842-5961-0

Copyright © 2020 by Ray Lischner

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

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

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Steve Anglin
Development Editor: Matthew Moodie
Coordinating Editor: Mark Powers

Cover designed by eStudioCalamar

Cover image by Barby Dalbosco on Unsplash (www.unsplash.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles 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 Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484259603. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

Table of Contents

About the Author	xxiii
About the Technical Reviewer	xxv
Acknowledgments	xxvii
Introduction	xxix
■ Part I: The Basics	1
■ Exploration 1: Honing Your Tools	3
C++ Versions	3
Ray's Recommendations	3
Clang and LLVM	4
GNU Compiler Collection	4
Microsoft Windows	4
Other Tools	4
Read the Documentation	4
Your First Program	5
■ Exploration 2: Reading C++ Code	13
Comments	14
Modules	14
Main Program	16
Variable Definitions	17
Statements	17
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	42
Unknown Name	42
Symbol Errors.....	42
Fun with Errors.....	43
■ Exploration 7: More Loops	45
Bounded Loops.....	45
Initialization	46
Condition	47
Postiteration	47
How a for Loop Works.....	47
Your Turn	47
■ Exploration 8: Formatted Output	49
The Problem	49
Field Width	50
Fill Character	51
std Prefix	52
Alignment	52
Exploring Formatting.....	53
Alternative Syntax	54
On Your Own.....	55
The format Function	57

■ Exploration 9: Arrays and Vectors	61
Vectors for Arrays.....	62
Vectors	62
Ranges and Algorithms	63
■ Exploration 10: Algorithms and Ranges	65
Algorithms.....	65
Output Iterators	66
■ Exploration 11: Increment and Decrement	71
Increment.....	71
Decrement.....	72
Member Types.....	73
Back to Iterators.....	73
■ Exploration 12: Conditions and Logic	77
I/O and bool.....	77
Boolean Type	78
Logic Operators	80
Old-Fashioned Syntax	81
Comparison Operators.....	81
■ Exploration 13: Compound Statements	85
Statements.....	85
Local Definitions and Scope.....	89
Definitions in for Loop Headers.....	92
■ Exploration 14: Introduction to File I/O	95
Reading Files.....	95
Writing Files	96

- **Exploration 15: The Map Data Structure** 99
 - Using Maps..... 100
 - Pairs 101
 - Searching in Maps..... 102
- **Exploration 16: Type Synonyms**..... 105
 - typedef and using Declarations..... 105
 - Common typedefs 107
- **Exploration 17: Characters** 109
 - Character Type 109
 - Character I/O 111
 - Newlines and Portability 112
 - Character Escapes 113
- **Exploration 18: Character Categories**..... 115
 - Character Sets..... 115
 - Character Categories..... 117
 - Locales 118
- **Exploration 19: Case-Folding**..... 123
 - Simple Cases..... 123
 - Harder Cases..... 124
- **Exploration 20: Writing Functions** 127
 - Functions..... 127
 - Function Call 129
 - Declarations and Definitions 129
 - Counting Words—Again 131
 - The main() Function 134

■ Exploration 21: Function Arguments	135
Argument Passing	135
Pass-by-Reference	138
const References	140
const_iterator	141
String Arguments	142
Multiple Output Parameters	142
■ Exploration 22: Using Ranges	143
Transforming Data	143
Predicates	148
Other Algorithms	149
■ Exploration 23: Using Iterators	153
Transforming Data	153
Sorting with Iterators	157
■ Exploration 24: Unnamed Functions	159
Lambdas	159
Naming an Unnamed Function	161
Capturing Local Variables	161
const Capture	163
Return Type	164
■ Exploration 25: Overloading Function Names	167
Overloading	167
bool is_alpha(char ch)	169
bool is_alpha(std::string_view str)	169
char to_lower(char ch)	169
std::string to_lower(std::string_view str)	170
char to_upper(char ch)	170
std::string to_upper(std::string_view str)	170

■ Exploration 26: Big and Little Numbers	173
The Long and Short of It.....	173
Long Integers.....	174
Short Integers.....	174
Integer Literals	175
Byte-Sized Integers.....	177
Type Casting.....	177
Make Up Your Own Literals	179
Integer Arithmetic.....	180
Overload Resolution	180
■ Exploration 27: Very Big and Very Little Numbers	183
Floating-Point Numbers	183
Floating-Point Literals	184
Floating-Point Traits	185
Floating-Point I/O	186
■ Exploration 28: Documentation	191
Doxygen.....	191
Structured Comments	191
Documentation Tags and Markdown.....	192
@b word	192
@brief one-sentence-description.....	192
@c word	192
@em word	193
@file file name	193
@link entity text @endlink	193
@mainpage title	193
@p name	193
@par title.....	193
@param name description	193

@post precondition.....	193
@pre precondition.....	194
@return description.....	194
@see xref.....	194
@&, @@, @\, @%, @<.....	194
Using Doxygen.....	197
■ Exploration 29: Project 1: Body Mass Index.....	199
Hints.....	200
■ Part II: Custom Types.....	201
■ Exploration 30: Custom Types.....	203
Defining a New Type.....	203
Member Functions.....	204
Constructors.....	207
Overloading Constructors.....	208
■ Exploration 31: Overloading Operators.....	211
Comparing Rational Numbers.....	211
Arithmetic Operators.....	215
Math Functions.....	218
■ Exploration 32: Custom I/O Operators.....	221
Input Operator.....	221
Output Operator.....	222
Error State.....	223
■ Exploration 33: Assignment and Initialization.....	225
Assignment Operator.....	225
Constructors.....	226
Putting It All Together.....	227

■ Exploration 34: Writing Classes	233
Anatomy of a Class.....	233
Member Functions	234
Constructor.....	236
Defaulted and Deleted Constructors	240
■ Exploration 35: More About Member Functions	241
Revisiting Project 1	241
const Member Functions.....	245
■ Exploration 36: Access Levels	249
Public vs. Private.....	249
class vs. struct	252
Public or Private?	252
■ Exploration 37: Understanding Object-Oriented Programming	259
Books and Magazines	259
Classification.....	260
Inheritance	262
Liskov’s Substitution Principle	263
Type Polymorphism	264
■ Exploration 38: Inheritance	265
Deriving a Class.....	265
Member Functions	268
Destructors.....	268
Access Level	272
Programming Style.....	273
■ Exploration 39: Virtual Functions	275
Type Polymorphism	275
Virtual Functions	278
References and Slices.....	281

Pure Virtual Functions	282
Virtual Destructors	283
■ Exploration 40: Classes and Types	285
Classes vs. typedefs.....	285
Value Types.....	288
Copying.....	288
Assigning.....	288
Moving.....	289
Comparing	290
Resource Acquisition Is Initialization.....	293
■ Exploration 41: Declarations and Definitions	295
Declaration vs. Definition	295
inline Functions	297
Variable Declarations and Definitions	298
Static Variables.....	300
Static Data Members.....	302
Declarators	304
■ Exploration 42: Modules	305
Introducing Modules.....	305
Classes and Modules	306
Hiding the Implementation.....	309
Modules Exporting Modules	310
Compiling Modules.....	313
■ Exploration 43: Old-Fashioned “Modules”	315
Interfaces As Headers	315
Inline or Not Inline	317
Quotes and Brackets	317
Include Guards	318

Forward Declarations	318
extern Variables.....	319
One-Definition Rule	320
■ Exploration 44: Function Objects.....	323
The Function Call Operator.....	323
Function Objects	324
■ Exploration 45: Useful Algorithms.....	327
Ranges and Iterators	327
Searching	327
Linear Search Algorithms	328
Binary Search Algorithms	332
Comparing.....	335
Rearranging Data	337
Copying Data	338
Deleting Elements	339
Iterators.....	340
■ Exploration 46: More About Iterators.....	341
Kinds of Iterators	341
Input Iterators	342
Output Iterators	342
Forward Iterators.....	342
Bidirectional Iterators	342
Random Access Iterators.....	343
Contiguous Iterators	343
Working with Iterators.....	344
const_iterator vs. const iterator	346
Error Messages	348
Specialized Iterators.....	349

■ Exploration 47: Ranges, Views, and Adaptors	351
Ranges	351
Range Views	352
Range Pipelines	353
Range Adaptors	354
The drop View	354
The filter View	354
The join View	354
The keys View	355
The reverse View	355
The transform View	355
The take View	355
The values View	356
■ Exploration 48: Exceptions	357
Introducing Exceptions	357
Catching Exceptions	359
Throwing Exceptions	361
Program Stack	362
Standard Exceptions	366
I/O Exceptions	367
Custom Exceptions	368
When a Function Doesn't Throw Exceptions	370
System Errors	371
Exceptional Advice	372
■ Exploration 49: More Operators	373
Conditional Operator	373
Short-Circuit Operators	375
Comma Operator	375

Arithmetic Assignment Operators 378

Increment and Decrement..... 379

■ **Exploration 50: Project 2: Fixed-Point Numbers..... 383**

 The fixed Class 383

 value_type 383

 places 383

 places10 383

 fixed() 383

 fixed(value_type integer, value_type fraction)..... 383

 fixed(double val) 384

 to_string()..... 384

 round()..... 384

 integer()..... 384

 fraction()..... 384

■ **Part III: Generic Programming..... 389**

■ **Exploration 51: Function Templates 391**

 Generic Functions 391

 Using Function Templates 392

 Writing Function Templates..... 394

 Template Parameters 395

 Template Arguments 396

 Abbreviated Function Templates 398

 Declarations and Definitions 399

 Member Function Templates 399

■ **Exploration 52: Class Templates..... 401**

 Parameterizing a Type 401

 Parameterizing the rational Class 402

 Using Class Templates..... 404

Overloaded Operators.....	405
Mixing Types.....	407
Template Variables	408
■ Exploration 53: Template Specialization	409
Instantiation and Specialization	409
Custom Comparators.....	410
Specializing Function Templates	412
Traits.....	413
■ Exploration 54: Partial Template Specialization	415
Degenerate Pairs.....	415
Partial Specialization.....	416
Partially Specializing Function Templates	418
Value Template Parameters.....	418
■ Exploration 55: Template Constraints.....	421
Constraining a Function Template	421
Constraining a Class Template	424
Standard Concepts	425
std::equality_comparable<T>	425
std::floating_point<T>.....	425
std::integral<T>	425
predicate<T>.....	425
std::strict_weak_order<T>	425
Iterator Concepts.....	425
std::bidirection_iterator<I>	425
std::contiguous_iterator<I>	426
std::forward_iterator<I>.....	426
std::indirectly_readable<I>.....	426
std::indirectly_writable<I>.....	426
std::input_iterator<I>	426

std::input_or_output_iterator<I>	426
std::output_iterator<I>	426
std::permutable<I>	426
std::random_access_iterator<I>	426
std::sortable<I>	426
Range Concepts	427
std::ranges::bidirectional_range<R>	427
std::ranges::contiguous_range<R>	427
std::ranges::forward_range<R>	427
std::ranges::input_range<R>	427
std::ranges::output_range<R>	427
std::ranges::random_access_range<R>	427
std::ranges::range<R>	427
std::ranges::sized_range<R>	427
std::ranges::view<R>	428
Writing Your Own Concept.....	428
■ Exploration 56: Names and Namespaces	431
Namespaces.....	431
Nested Namespaces	434
Global Namespace	436
The std Namespace.....	436
Using Namespaces.....	437
Namespace Alias	437
The using Directive	437
The using Declaration.....	439
The using Declaration in a Class	442
Name Lookup	443

■ Exploration 57: Containers	449
Properties of Containers.....	449
Member Types	450
value_type	450
key_type	451
reference	451
const_reference	451
iterator	451
const_iterator	451
size_type	451
What Can Go into a Container	451
Inserting and Erasing	453
Inserting in a Sequence Container	453
Erasing from a Sequence Container	454
Inserting in an Associative Container	454
Erasing from an Associative Container	456
Exceptions	456
Iterators and References.....	457
Sequence Containers	459
The array Class Template	460
The deque Class Template.....	461
The list Class Template.....	462
The vector Class Template.....	463
Associative Containers.....	464
■ Exploration 58: Locales and Facets	469
The Problem	469
Locales to the Rescue	470
Locales and I/O.....	471
Facets.....	471
Character Categories.....	472
Collation Order	478

- **Exploration 59: International Characters** 481
 - Why Wide?..... 481
 - Using Wide Characters 481
 - Wide Strings 482
 - Wide Character I/O 484
 - Multi-byte Character Sets 485
 - Unicode 486
 - Universal Character Names..... 488
 - Unicode Difficulties 488
- **Exploration 60: Text I/O** 491
 - File Modes..... 491
 - String Streams 492
 - Text Conversion 498
- **Exploration 61: Project 3: Currency Type** 507
- **Part IV: Real Programming** 509
- **Exploration 62: Pointers** 511
 - A Programming Problem 511
 - The Solution 519
 - Addresses vs. Pointers 521
 - Dependency Graphs 521
- **Exploration 63: Regular Expressions**..... 525
 - Parsing with Regular Expressions..... 525
- **Exploration 64: Moving Data with Rvalue References**..... 539
 - Copying vs. Moving 539
 - Lvalues, Rvalues, and More..... 541
 - Implementing Move..... 543
 - Rvalue or Lvalue? 544
 - Special Member Functions..... 546

■ Exploration 65: Smart Pointers	549
Pointers and Iterators.....	549
More About <code>unique_ptr</code>	550
Copyable Smart Pointers.....	550
Smart Arrays.....	552
Pimpls	552
Dumb Arrays.....	558
■ Exploration 66: Files and File Names	561
Portable File Names	561
Working with Files.....	563
Errors.....	565
Navigating Directories.....	567
■ Exploration 67: Working with Bits	571
Integer As a Set of Bits.....	571
Bitmasks	573
Shifting Bits.....	574
Safe Shifting with Unsigned Types.....	575
Signed and Unsigned Types.....	576
Unsigned Literals.....	577
Type Conversions.....	577
Overflow	581
Rotating Integers.....	582
Introducing Bitfields	582
Portability	583
The <code>bitset</code> Class Template	584

■ Exploration 68: Enumerations	587
Scoped Enumerations	587
Unscoped Enumerations	589
Strings and Enumerations	590
Spaceships	592
Revisiting Projects	593
■ Exploration 69: Multiple Inheritance	597
Multiple Base Classes	597
Virtual Base Classes	600
Java-Like Interfaces	602
Interfaces vs. Templates	603
Mix-Ins	604
Protected Access Level	606
■ Exploration 70: Concepts, Traits, and Policies	607
Case Study: Iterators	607
Type Traits	610
Case Study: char_traits	611
Policy-Based Programming	613
■ Exploration 71: Names, Namespaces, and Templates	619
Common Rules	619
Name Lookup in Templates	620
Three Kinds of Name Lookup	620
Member Access Operators	621
Qualified Name Lookup	622
Unqualified Name Lookup	625
Argument-Dependent Lookup	626

■ Exploration 72: Overloaded Functions and Operators	629
Type Conversion	629
Review of Overloaded Functions	630
Overload Resolution	633
Ranking Functions	634
List Initialization	635
Tie-Breakers	636
Default Arguments.....	639
■ Exploration 73: Programming at Compile Time	643
Compile-Time Functions	643
Compile-Time Variables.....	645
Variable-Length Template Argument Lists.....	646
User-Defined Literals.....	648
Types As Values	648
Conditional Types	650
Substitution Failure Is Not An Error (SFINAE)	652
■ Exploration 74: Project 4: Calculator	655
■ Index	657

About the Author

All the world is paged,
And all the men and women merely programs:
They have their exits and their segfaults;
And one man in his time plays many games,
His acts being seven ages. At first, the newbie,
Mewling and puking in BASIC terms.
And then the whining school-boy, with his packages,
And JavaServer Faces, creeping like snail
Downloading from the web. And then the l0v3r,
Sighing like heat sink fan, with an unmerged commit
Made to his github project. Then a hacker,
Full of strange oaths and bearded like a guru,
Jealous in honor, sudden and quick in quarrel,
Seeking the flamebait reputation
Even on lkml. And then the team lead,
In fair round belly with cappuccino drowned,
With eyes severe and beard of two days' cut,
Full of wise saws and modern design patterns;
And so he plays his part. The sixth age shifts
Into the lean and sandal'd manager,
With bifocals on nose and balding pate,
His COBOL code, well saved, a world too wide
For his shrunk shank; and his big noisy voice,
Turning again toward childish errors, buffer
Overruns in his code. Last scene of all,
That ends this strange eventful history,
Is second childishness and mere oblivion,
Sans mouse, sans keyboard, sans debugger, sans everything.

—By William Shakespeare, edited by Ray Lischner

Ray started writing programs before he had access to a computer, and over the subsequent four decades, he progressed steadily through the ages of programming. He currently lives with his family, where he does his best to retard the inexorable descent into the seventh age.

About the Technical Reviewer

Michael Thomas has worked in software development for more than 20 years as an individual contributor, team lead, program manager, and vice president of engineering. Michael has more than 10 years of experience working with mobile devices. His current focus is in the medical sector, using mobile devices to accelerate information transfer between patients and health-care providers.

Acknowledgments

Writing a book suits life during a pandemic. I sit at home, isolated from the world, crafting prose today and code tomorrow. Meanwhile, editors and reviewers fix the prose and critique the code. I never meet them in person. I trust that they, too, work in safety and isolation during this time of global pandemic. But before I issue the common thanks and acknowledgments to those who worked specifically on this book, I must first thank the unsung heroes of the pandemic who are keeping us alive, who are keeping us fed, and who help maintain our safety and isolation. And so I thank the many people who are unable to work in blissful solitude as they grow, pick, pack, truck, stock, prepare, and deliver our food and the supplies of everyday life. I thank the health-care workers who risk their lives daily in care of our loved ones. And I thank Michael Thomas for his technical review, my editor Mark Powers, and the staff at Apress for turning my humble bits and bytes into a finished product.

*We therefore have great cause of thankfulness,
And shall forget the office of our hand
Sooner than quittance of desert and merit,
According to the weight and worthiness.*

—William Shakespeare, *The Life of Henry the Fifth*, I.i

Introduction

Hi, there. Thank you for reading my book, *Exploring C++ 20*. 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 website 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 website, <https://cpphelp.com/exploring/>. The website has links to other books, other websites, 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 14882:2020 (E), *Programming languages — C++*. The 2020 edition of the standard is the all-new, improved, standard, typically referred to as C++ 20. This book reflects new idioms, new language patterns, and new code. All the exercises have been tested on modern compilers, but not always successfully. Most modern compilers do a decent job of conforming to the standard, but it takes time. The standardization committee approved the final draft in February of 2020. I am writing this in May while we all wait for the International Organization for Standardization (ISO) to accept that final draft as standard 14882:2020. Meanwhile, compiler writers implement different features at different rates, and each provider makes different choices as to which features to implement first. As I write this introduction, no compiler fully implements C++ 20. The book's website will have up-to-date details as vendors release updates to their compilers.

PART I



The Basics