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 ISBN-13 (electronic): 978-1-4842-5961-0
https://doi.org/10.1007/978-1-4842-5961-0

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

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