LEARNING MADE EASY

4th Edition

C++ ALL-IN-ONE

for dummies®

A Wiley Brand

7 Books in one!

John Paul Mueller

Author of *Functional Programming For Dummies*

C++

ALL-IN-ONE

4th Edition

by John Paul Mueller

for dummies®
A Wiley Brand

C++ All-in-One For Dummies®, 4th Edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

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

Library of Congress Control Number: 2020949804

ISBN: 978-1-119-60174-6

ISBN 978-1-119-60175-3 (ebk); ISBN 978-1-119-60173-9 (ebk)

C++ All-in-One For Dummies®

To view this book's Cheat Sheet, simply go to www.dummies.com and search for “C++ All-in-One For Dummies Cheat Sheet” in the Search box.

Table of Contents

- Cover
- Title Page
- Copyright
- Introduction
  - About This Book
  - Foolish Assumptions
  - Icons Used in This Book
  - Beyond the Book
  - Where to Go from Here
- Book 1: Getting Started with C++
  - Chapter 1: Configuring Your Desktop System
    - Obtaining a Copy of C++ 20
    - Obtaining Code::Blocks
    - Installing Code::Blocks
    - Touring the Essential Code::Blocks Features

Using Other IDEs

Chapter 2: Configuring Your Mobile System

Obtaining CppDroid

Considering Other Alternatives

Touring the Essential CppDroid Features

Obtaining CppDroid Help

Chapter 3: Creating Your First C++ Application

Code::Blocks Creating a Project

Typing the Code

Starting with Main

Showing Information

Let Your Application Run Away

Chapter 4: Storing Data in C++

Putting Your Data Places: Variables

Manipulating Integer Variables

Characters

Strings

Making Decisions Using Conditional Operators

Telling the Truth with Boolean Variables

Reading from the Console

Chapter 5: Directing the Application Flow

Doing This or Doing That

Evaluating Conditions in C++

Including Evaluations in C++ Conditional Statements

Repeating Actions with Statements That Loop

Looping for

Looping while

Doing while

Breaking and continuing

Nesting loops

Chapter 6: Dividing Your Work with Functions

Dividing Your Work

Calling a Function

Writing Your Own Functions

Improving On the Basic Function

Calling All String Functions

Understanding main()

Chapter 7: Splitting Up Source Code Files

Creating Multiple Source Files

Sharing with Header Files

Sharing Variables among Source Files

Using the Mysterious Header Wrappers

Chapter 8: Referring to Your Data Through Pointers

Understanding the Changes in Pointers for C++ 20

Heaping and Stacking the Variables

Creating New Raw Pointers

Freeing Raw Pointers

Working with Smart Pointers

Passing Pointer Variables to Functions

Returning Pointer Variables from Functions

Book 2: Understanding Objects and Classes

Chapter 1: Working with Classes

Understanding Objects and Classes

Working with a Class

Starting and Ending with Constructors and Destructors

Building Hierarchies of Classes

Creating and Using Object Aliases

Chapter 2: Using Advanced C++ Features

Filling Your Code with Comments

Converting Types

Reading from the Console

Understanding Preprocessor Directives

Using Constants

Using Switch Statements

Supercharging enums with Classes

Working with Random Numbers

Storing Data in Arrays

Chapter 3: Planning and Building Objects

Recognizing Objects

Encapsulating Objects

Building Hierarchies

Chapter 4: Building with Design Patterns

Delving Into Pattern History

Introducing a Simple Pattern: the Singleton

Watching an Instance with an Observer

Mediating with a Pattern

Book 3: Understanding Functional Programming

Chapter 1: Considering Functional Programming

Understanding How Functional Programming Differs

Defining an Impure Language

Seeing Data as Immutable

Considering the Effects of State

Eliminating Side Effects

Understanding the Role of auto

Passing Functions to Functions

Using Lambda Expressions for Implementation

Chapter 2: Working with Lambda Expressions

Creating More Readable and Concise C++ Code

Defining the Essential Lambda Expression

Developing with Lambda Expressions

Chapter 3: Advanced Lambda Expressions

Considering the C++ 20 Lambda Extensions

Working in Unevaluated Contexts

Using Assignable Stateless Lambda Expressions

Dealing with Pack Expansions

Book 4: Fixing Problems

Chapter 1: Dealing with Bugs

It’s Not a Bug. It’s a Feature!

Make Your Application Features Look Like Features

Anticipating (Almost) Everything

Avoiding Mistakes, Plain and Simple

Chapter 2: Debugging an Application

Programming with Debuggers

Debugging with Different Tools

Debugging a Code::Blocks Application with Command-Line Arguments

Chapter 3: Stopping and Inspecting Your Code

Setting and Disabling Breakpoints

Watching, Inspecting, and Changing Variables

Chapter 4: Traveling About the Stack

Stacking Your Data

Debugging with Advanced Features

Book 5: Advanced Programming

Chapter 1: Working with Arrays, Pointers, and References

Building Up Arrays

Pointing with Pointers

Referring to References

Chapter 2: Creating Data Structures

Working with Data

Structuring Your Data

Naming Your Space

Chapter 3: Constructors, Destructors, and Exceptions

Constructing and Destructing Objects

Programming the Exceptions to the Rule

Chapter 4: Advanced Class Usage

Inherently Inheriting Correctly

Using Classes and Types within Classes

Chapter 5: Creating Classes with Templates

Templatizing a Class

Going Beyond the Basics

Parameterizing a Template

Typedefing a Template

Deriving Templates

Templatizing a Function

Chapter 6: Programming with the Standard Library

Architecting the Standard Library

Containing Your Classes

The Great Container Showdown

Copying Containers

Creating and Using Dynamic Arrays

Working with Unordered Data

Working with Ranges

Book 6: Reading and Writing Files

Chapter 1: Filing Information with the Streams Library

Seeing a Need for Streams

Programming with the Streams Library

Handling Errors When Opening a File

Flagging the ios Flags

Chapter 2: Writing with Output Streams

Inserting with the << Operator

Formatting Your Output

Chapter 3: Reading with Input Streams

Extracting with Operators

Encountering the End of File

Reading Various Types

Chapter 4: Building Directories and Contents

Manipulating Directories

Getting the Contents of a Directory

Copying Files

Moving and Renaming Files and Directories

Chapter 5: Streaming Your Own Classes

Streaming a Class for Text Formatting

Manipulating a Stream

Book 7: Advanced Standard Library Usage

Chapter 1: Exploring the Standard Library Further

Considering the Standard Library Categories

Parsing Strings Using a Hash

Obtaining Information Using a Random Access Iterator

Locating Values Using the Find Algorithm

Using the Random Number Generator

Working with Temporary Buffers

Chapter 2: Working with User-Defined Literals (UDLs)

Understanding the Need for UDLs

Working with the UDLs in the Standard Library

Creating Your Own UDLs

Chapter 3: Building Original Templates

Deciding When to Create a Template
Defining the Elements of a Good Template
Creating a Basic Math Template
Building a Structure Template
Developing a Class Template
Considering Template Specialization
Creating a Template Library
Using Your Template Library

Chapter 4: Investigating Boost

Considering the Standard Library Alternative
Understanding Boost
Obtaining and Installing Boost for Code::Blocks
Creating the Boost Tools
Using Boost.Build
Using Inspect
Understanding BoostBook
Using QuickBook
Using bcp
Using Wave
Building Your First Boost Application Using Date Time

Chapter 5: Boosting up a Step

Parsing Strings Using RegEx
Breaking Strings into Tokens Using Tokenizer
Performing Numeric Conversion
Creating Improved Loops Using Foreach
Accessing the Operating System Using Filesystem

Index

About the Authors

Connect with Dummies

End User License Agreement

List of Tables

Book 1 Chapter 2

TABLE 2-1: CppDroid Features

Book 1 Chapter 3

TABLE 3-1 Navigation and Edit Commands

TABLE 3-2 Math Symbols

TABLE 3-3 Keyboard Shortcuts for Compiling and Running

Book 1 Chapter 4

TABLE 4-1 A Variable Has Three Aspects

TABLE 4-2 Examples of Bad Variable Names

TABLE 4-3 Doing Things with Addition

Book 1 Chapter 5

TABLE 5-1 Evaluating Numerical Conditions

TABLE 5-2 Choosing Your Loops

Book 2 Chapter 1

TABLE 1-1 Specifying Property Values for Instances of Class Pen

TABLE 1-2 Steps to Using Objects

Book 3 Chapter 2

TABLE 2-1 Elements of a Lambda Expression

Book 5 Chapter 2

TABLE 2-1: ANSI C++ Character Types

Book 6 Chapter 2

TABLE 2-1 ios_base Formatting Flags

TABLE 2-2 Using ANSI-Standard Manipulators and Demanipulators

Book 7 Chapter 2

TABLE 2-1 Standard Prefixes and Suffixes

List of Illustrations

Book 1 Chapter 1

FIGURE 1-1: The wizard asks you to select the configuration options to use.

FIGURE 1-2: The Ubuntu Software Center contains a list of the most popular soft...

FIGURE 1-3: The Developer Tools category contains an entry for Code::Blocks.

FIGURE 1-4: It's possible to obtain additional information about Code::Blocks i...

FIGURE 1-5: Associate Code::Blocks with your C++ files to make it easier to man...

FIGURE 1-6: Code::Blocks provides a tips dialog box that contains helpful infor...

FIGURE 1-7: Save your changes to disk.

FIGURE 1-8: Each example has a `.cbp` file associated with it that opens the exam...

FIGURE 1-9: Use the Start Here window to start a new session.

FIGURE 1-10: The Management window helps you manage your Code::Blocks projects.

FIGURE 1-11: Using the Logs & Others window to understand how your application ...

FIGURE 1-12: Set Code::Blocks to use the GCC compiler to run the examples in th...

Book 1 Chapter 2

FIGURE 2-1: Locating CppDroid in the Google Play Store.

FIGURE 2-2: Loading the CppDroid libraries for offline use.

FIGURE 2-3: Web-based IDEs tend to provide a very simple interface.

FIGURE 2-4: Accessing the basic CppDroid user interface features.

FIGURE 2-5: Use the menu to locate the CppDroid features and options.

FIGURE 2-6: Loading an example provides a quick way to see code in action.

FIGURE 2-7: Give permission to access your Google Drive.

FIGURE 2-8: The file is available for use with your local copy of CppDroid.

FIGURE 2-9: Define a new local project.

FIGURE 2-10: The list of actions in CppDroid is somewhat limited.

FIGURE 2-11: A list of Help sources for CppDroid.

FIGURE 2-12: The developer uses blog posts to help you find bug fixes.

FIGURE 2-13: Use an example to see how the static analysis feature works.

FIGURE 2-14: The tutorials take you through basic processes within CppDroid.

Book 1 Chapter 3

FIGURE 3-1: The New from Template dialog box lets you select a new project type...

FIGURE 3-2: Provide the name of your project for Code::Blocks.

FIGURE 3-3: Tell Code::Blocks where to place the Debug and Release versions of ...

FIGURE 3-4: Use the Code::Blocks IDE to interact with your project.

FIGURE 3-5: Execute your first application.

FIGURE 3-6: Code::Blocks tells you about errors in your application.

FIGURE 3-7: Configure the editor to use automatic indents.

FIGURE 3-8: The computer likes to use orders of operation.

FIGURE 3-9: A programmer calculator comes in handy when working with numbers.

FIGURE 3-10: You can see how overruns occur by doing the math.

Book 1 Chapter 5

FIGURE 5-1: Configure GCC to use the C++ 17 standard.

Book 1 Chapter 6

FIGURE 6-1: Dividing a process into a hierarchy.

FIGURE 6-2: You can think of a function as a machine.

FIGURE 6-3: Command-line apps often have switches and arguments.

Book 1 Chapter 7

FIGURE 7-1: The New from Template dialog box lets you select a new file type.

FIGURE 7-2: Provide the file information required by the wizard.

FIGURE 7-3: The Management window displays the files used to compile the projec...

FIGURE 7-4: Supply a project name for your new application.

FIGURE 7-5: The current directory doesn’t contain any code files.

FIGURE 7-6: Select the builds where the files are used.

FIGURE 7-7: The new project now contains references to the selected files.

FIGURE 7-8: Remove the references you no longer need.

Book 1 Chapter 8

FIGURE 8-1: The two stack frames used for the example code.

Book 2 Chapter 2

FIGURE 2-1: The formatted output shows the difference between truncating and ro...

FIGURE 2-2: Provide the compiler options you want to use to change the applicat...

Book 2 Chapter 3

FIGURE 3-1: The outer object in this picture is a mailbox container.

FIGURE 3-2: The smaller inner objects in this picture are the mailboxes.

FIGURE 3-3: You can draw a class by using a box divided into three horizontal s...

FIGURE 3-4: The arrows in this UML diagram point toward the base class.

Book 2 Chapter 4

FIGURE 4-1: A model of the hierarchy between classes.

FIGURE 4-2: A mediator certainly cleans things up!

Book 3 Chapter 1

FIGURE 1-1: Seeing errors generated as the result of immutability in a structur...

Book 4 Chapter 2

FIGURE 2-1: The main Code::Blocks window shows your source code.

FIGURE 2-2: Completing the run shows the application results.

FIGURE 2-3: Be sure to check the output to ensure that it’s what you expected.

FIGURE 2-4: The debugger shows the beginning and end of code blocks.

FIGURE 2-5: The Disassembly window displays the assembly language version of yo...

Book 4 Chapter 3

FIGURE 3-1: Code::Blocks displays any breakpoints you set using a red octagon.

FIGURE 3-2: Debug mode tells Code::Blocks to stop execution when it reaches the...

FIGURE 3-3: Use the Edit Breakpoint dialog box to enable and disable breakpoint...

FIGURE 3-4: Disabled breakpoints don’t stop application execution.

FIGURE 3-5: Use the Breakpoints window to access a number of breakpoints at one...

FIGURE 3-6: The Watches window shows the value of variables and objects.

FIGURE 3-7: Assigning a value to `i` changes its value in the Watches window.

FIGURE 3-8: Adding a watch to the window presents additional information.

Book 4 Chapter 4

FIGURE 4-1: The Call Stack window shows the function calls that led up to the c...

FIGURE 4-2: The Disassembly window shows the assembly code that results from th...

FIGURE 4-3: This Disassembly window shows the `CountCarSeats()` function code.

FIGURE 4-4: Small C++ code changes can result in large assembly-code changes.

FIGURE 4-5: Viewing the CPU registers can give you insight into how code intera...

Book 5 Chapter 1

FIGURE 1-1: Use the Select Target dialog box to add program arguments.

Book 5 Chapter 4

FIGURE 4-1: Using diamond inheritance can be hard.

Book 6 Chapter 1

FIGURE 1-1: Working with streams requires use of the appropriate headers and co...

Book 7 Chapter 3

FIGURE 3-1: Provide a description of your project for Code::Blocks.

FIGURE 3-2: Provide a description of your static library for Code::Blocks.

FIGURE 3-3: Change the compiler settings to meet your project needs.

FIGURE 3-4: Add new files using the New from Template dialog box.

FIGURE 3-5: Define the header requirements.

Book 7 Chapter 4

FIGURE 4-1: Use the Compiler Settings dialog box to configure Code::Blocks to u...

FIGURE 4-2: Add appropriate search directories for Boost header and library fil...

FIGURE 4-3: The Search Directories tab will display any compiler, linker, or re...

FIGURE 4-4: The example requires the use of a special library.

FIGURE 4-5: The needed file appears in the Link Libraries list.

FIGURE 4-6: The example displays the result of the search.

FIGURE 4-7: Inspect normally outputs its reports as HTML.

FIGURE 4-8: The bcp utility can output some nice-looking reports about Boost mo...

FIGURE 4-9: Select the Boost library directory.

FIGURE 4-10: Make sure you set the environment to use Boost.

Book 7 Chapter 5

FIGURE 5-1: Select the library you want to add.

FIGURE 5-2: Add the library to the application.

FIGURE 5-3: Add appropriate search directories for Boost header and library fil...

FIGURE 5-4: The search location for any compiler, linker, or resource compiler.

FIGURE 5-5: Using the Filesystem library requires the System library as well.

Introduction

There are many general-purpose programming languages today, but few can claim to be the language of the millennium. C++ can make that claim, and for good reason:

- » It's powerful. You can write almost any program in it.
- » It's fast, and it's fully compiled. That's a good thing.
- » It's easy to use — if you have this book.
- » It's object oriented. If you're not sure what that is, don't worry. You can find out about it by reading this very book you're holding.
- » It supports functional programming techniques, which makes modeling math problems considerably easier and makes parallel processing easier. This book covers functional programming techniques, too.
- » It's portable. Versions are available for nearly every computer.
- » It's standardized. The American National Standards Institute (ANSI) and the International Standards Organization (ISO) both approve an official version.
- » It's continually updated to meet the changing challenges of the computer community.
- » It's popular. More people are using C++ because so many other people use it.

Sure, some people criticize C++. But most of these people don't truly understand C++ or are just having a bad day. Or both.

About This Book

This book is a hands-on, roll-up-your-sleeves experience that gives you the opportunity to truly learn C++. This edition starts out by helping you get a great C++ installation in place. A lot of readers wrote to tell me that they simply couldn’t get C++ to work for them, and I listened by adding configuration instructions in Book 1, Chapter 1. You can find instructions for working with the Mac, Linux, and Windows throughout the book. The examples are also tested to work on all three platforms.

C++ All-in-One For Dummies, 4th Edition, is devoted to working with C++ wherever you want to use it. Book 1, Chapter 2 even includes techniques for writing C++ code on your mobile device, although writing a complex application on your smartphone would be understandably difficult because of the small device size.

At the very beginning, I start you out from square one. I don’t assume any programming experience whatsoever. Everybody has to start somewhere. You can start here. Not to brag, but you are in the hands of a highly successful C++ developer who has shown thousands of people how to program, many of whom also started out from square one.

You already know C++? This book is great for you, too, because although I start discussing C++ from the beginning, I cover the important aspects of the language in depth. Even if you’ve used C++ in the past, this book gets you up to speed with the latest in C++ 14 and above innovations, including C++ 20 additions. Plus, this edition of the book focuses on all the latest programming strategies while removing some of the less used functionality of the past.

If you’re interested in using the time-tested Object Oriented Programming (OOP) techniques that C++ developers have used for years, then Book 2 is where you want to look. You start with a view of classes, but eventually move into more advanced topics, including the use of programming patterns in Book 2 Chapter 4.

One of the most exciting additions to this edition is the use of functional programming techniques, which you can find in Book 3. Functional programming has become extremely popular because it makes modeling math problems significantly easier, and many people use functional programming techniques to solve modern data science problems. More important, functional programming can be a lot easier than earlier programming paradigms.

Every application out there has a bug or two. If you doubt this statement, just try to find one that is bug free —you won’t. Book 4 includes all sorts of techniques you can use to make your application as bug free as possible before it leaves your machine and then help you find the bugs that others graciously point out later.

Book 5 is all about moving you from generalized programming strategies into the advanced strategies used by modern developers. It starts with a look at standardized structures for working with classes in a safe manner. The minibook takes you through

- Simple structures, such as arrays
- More advanced data management
- The use of constructors, destructors, and exceptions
- Templatized programming
- Use of the Standard Library (originally called the Standard Template Library or STL).

Everyone needs to work with files at some point. You use local, network, and Internet files today on a regular basis. Book 6 is all about working with files in various ways. This book includes topics on working with data streams as well.

The Standard Library is immense and there are entire books written about its use. *C++ All-in-One For Dummies,* 4th Edition, focuses on providing you with a really good overview that you can use to drill down into more detailed topics later. Besides looking at the Standard Library in more detail, you discover how to work with User Defined Literals (UDLs) and how to create your own templates. This book also delves into the Boost library, which is the library that has added more to Standard Library than just about any other source. Check out Book 7, Chapters 4 and 5 to learn about Boost. If you use C++ and don't use Boost, you're really missing out!

C++ is standardized, and you can use the information in this book on many different platforms. I wrote the samples using Mac OS X, SUSE Linux (some of the beta readers used other flavors of Linux), and Windows systems (with some testing on my ASUS tablet as well). In order to make this happen, I used a compiler called *Code::Blocks* that runs on almost every computer (Windows, Linux, and Macintosh) and CppDroid for my tablet. It doesn't matter which device you're using!

To make absorbing the concepts easy, this book uses the following conventions:

» Text that you're meant to type just as it appears in the book is in **bold**. The exception is when you're working through a step list: Because each step is bold, the text to type is not bold.

- Web addresses and programming code appear in `monofont`. If you’re reading a digital version of this book on a device connected to the Internet, you can click or tap the web address to visit that website, like this: `https://www.dummies.com`.
- When you need to type command sequences, you see them separated by a special arrow, like this: File⇒New File. In this example, you go to the File menu first and then select the New File entry on that menu.
- When you see words in *italics* as part of a typing sequence, you need to replace that value with something that works for you. For example, if you see “Type ***Your Name*** and press Enter,” you need to replace *Your Name* with your actual name.

Foolish Assumptions

This book is designed for novice and professional alike. You can either read this book from cover to cover, or you can look up topics and treat the book as a reference guide — whichever works best for you. Keep it on your shelf, and have it ready to grab when you need to look something up. However, I’ve made some assumptions about your level of knowledge when I put the book together. The most important of these assumptions is that you already know how to use your device and work with the operating system that supports it. You also need to know how to perform tasks like downloading files and installing applications. A familiarity with the Internet is also required, and you need to know how to interact with it moderately well to locate the resources you need to work with the book. Finally, you must know how to work with archives, such as the ZIP file format.

Icons Used in This Book

As you read this book, you see icons in the margins that indicate material of interest (or not, as the case may be). This section briefly describes each icon in this book.

TIP Tips are nice because they help you save time or perform some task without a lot of extra work. The tips in this book are time-saving techniques or pointers to resources that you should try so that you can get the maximum benefit from C++. Most important, many of these tips will help you make sense of the overwhelming quantity of libraries and tools that C++ developers have created over the years.

WARNING I don't want to sound like an angry parent or some kind of maniac, but you should avoid doing anything that's marked with a Warning icon. Otherwise, you might find that your application fails to work as expected, you get incorrect answers from seemingly bulletproof code, or (in the worst-case scenario) you lose data. Given where C++ appears, you might also send the next rocket off to Mars prematurely, make someone's thermostat misbehave, or cause nationwide power outages. Really, warnings are for everyone!

TECHNICAL STUFF Whenever you see this icon, think advanced tip or technique. You might find these tidbits of useful information just too boring for words, or they could contain the solution you need to get a program running. Skip these bits of information whenever you like.

REMEMBER If you don't get anything else out of a particular chapter or section, remember the material marked by this icon. This text usually contains an essential process or a bit of information that you must know to work with C++, or to perform development tasks successfully.

Beyond the Book

If you want to email me, please do! Make sure you send your book-specific requests to:

```
John@JohnMuellerBooks.com
```

I get a lot of email from readers, so sometimes it takes me a while to answer. I try very hard to answer every book-specific question I receive, though, so I highly recommend contacting me with your questions. I want to ensure that your book experience is the best one possible. The blog category at `http://blog.johnmuellerbooks.com/categories/263/c-all-in-one-for-dummies.aspx` contains a wealth of additional information about this book. You can check out the website at `http://www.johnmuellerbooks.com/`.

This book isn't the end of your C++ programming experience — it's really just the beginning. I provide online content to make this book more flexible and better able to meet your needs. That way, as I receive email from you, I can address questions and tell you how updates to either Code::Blocks or the C++ language affect book content. You can also access other cool materials:

- **Cheat Sheet:** You remember using crib notes in school to make a better mark on a test, don't you? You do? Well, a cheat sheet is sort of like that. It provides you with some special notes on things you can do with C++ that not every other developer knows. You can find the cheat sheet for this book at `www.dummies.com` and typing **C++ All-in-One For Dummies, 4th Edition** in the search field. It contains really neat information like the top ten mistakes developers make when working with C++, a list of header files that you use in most applications, and some of the C++ syntax that gives most developers problems.
- **Updates:** Sometimes changes happen. For example, I might not have seen an upcoming change when I looked into my crystal ball during the writing of this book. In the past, such a situation simply meant that the book would become outdated and less useful, but you can now find updates to the book at `www.dummies.com`. In addition to these updates, check out the blog posts with answers to reader questions and demonstrations of useful book-related techniques at `http://blog.johnmuellerbooks.com/`.
- **Companion files:** Hey! Who really wants to type all the code in the book? Most readers would prefer to spend their time actually working through coding examples rather than typing. Fortunately for you, the

source code is available for download, so all you need to do is read the book to learn C++ coding techniques. Each of the book examples even tells you precisely which example project to use. You can find these files by visiting `www.dummies.com/go/caiofd4e`.

Just in case you're worried about Code::Blocks, you can find complete download and installation instructions for it in Book 1, Chapter 1. Don't worry about which platform you use. This chapter includes instructions for Mac OS X, Linux, and Windows.

Where to Go from Here

If you're just starting your C++ adventure, I highly recommend starting at either Book 1, Chapter 1 (for desktop developers) or Book 1, Chapter 2 (for mobile developers). You really do need to create a solid foundation before you can tackle the code in this book. If you're in a hurry and already have a C++ installation, you can always try starting with Book 1, Chapter 3.

Readers with a little more experience, who already know some C++ basics, can skip some of these introductory chapters, but you definitely don't want to skip Book 1, Chapter 8 because it contains a lot of pointer-related changes in current versions of C++. If you skip this chapter, you may find later that you have a hard time following the example code in the book because the newer examples use these pointer features.

An advanced reader with some idea of the current changes in C++ 20 could possibly skip Book 1, but scanning Book 2 is a good idea because there are some OOP changes you definitely want to know about. However, even for advanced readers, skipping Book 3 is a bad idea because modern development really is moving toward functional programming techniques.

Book 1
Getting Started with C++

Contents at a Glance

Chapter 1: Configuring Your Desktop System

- Obtaining a Copy of C++ 20
- Obtaining Code::Blocks
- Installing Code::Blocks
- Touring the Essential Code::Blocks Features
- Using Other IDEs

Chapter 2: Configuring Your Mobile System

- Obtaining CppDroid
- Considering Other Alternatives
- Touring the Essential CppDroid Features
- Obtaining CppDroid Help

Chapter 3: Creating Your First C++ Application

- Code::Blocks Creating a Project
- Typing the Code
- Starting with Main
- Showing Information
- Let Your Application Run Away

Chapter 4: Storing Data in C++

- Putting Your Data Places: Variables
- Manipulating Integer Variables
- Characters
- Strings
- Making Decisions Using Conditional Operators
- Telling the Truth with Boolean Variables
- Reading from the Console