Preface

Welcome to ***C++20 for Programmers: An Objects-Natural Approach***. This book presents leading-edge computing technologies for software developers. It conforms to the C++20 standard (1,834 pages), which the ISO C++ Standards Committee approved in September 2020.[1,2]

The C++ programming language is popular for building high-performance business-critical and mission-critical computing systems—operating systems, real-time systems, embedded systems, game systems, banking systems, air-traffic-control systems, communications systems and more. This book is an introductory- through intermediate-level tutorial presentation of the C++20 version of C++, which is among the world's most popular programming languages,[3] and its associated standard libraries. We present a friendly, contemporary, code-intensive, case-study-oriented introduction to C++20. In this Preface, we explore the "soul of the book."

P.1 Modern C++

We focus on **Modern C++**, which includes the four most recent C++ standards—**C++20**, **C++17**, **C++14** and **C++11**, with a look toward key features anticipated for **C++23** and later. A common theme of this book is to focus on the new and improved ways to code in C++. We employ best practices, emphasizing current professional software-development Modern C++ idioms, and we focus on performance, security and software engineering issues.

Keep It Topical

"*Who dares to teach must never cease to learn.*"[4] (J. C. Dana)

To "take the pulse" of Modern C++, which changes the way developers write C++ programs, we read, browsed or watched approximately 6,000 current articles, research papers, white papers, documentation pieces, blog posts, forum posts and videos.

1. The final draft C++ standard is located at: `https://timsong-cpp.github.io/cppwp/n4861/`. This version is free. The published final version (ISO/IEC 14882:2020) may be purchased at `https://www.iso.org/standard/79358.html`.
2. Herb Sutter, "C++20 Approved, C++23 Meetings and Schedule Update," September 6, 2020. Accessed January 11, 2022. `https://herbsutter.com/2020/09/06/c20-approved-c23-meetings-and-schedule-update/`.
3. Tiobe Index for January 2022. Accessed January 7, 2022. `http://www.tiobe.com/tiobe-index`.
4. John Cotton Dana. From `https://www.bartleby.com/73/1799.html`: "In 1912 Dana, a Newark, New Jersey, librarian, was asked to supply a Latin quotation suitable for inscription on a new building at Newark State College (now Kean University), Union, New Jersey. Unable to find an appropriate quotation, Dana composed what became the college motto."—*The New York Times Book Review*, March 5, 1967, p. 55."

C++ Versions

As a developer, you might work on C++ legacy code or projects requiring specific C++ versions. So, we use margin icons like the “**20**” **icon** shown here to mark each mention of a Modern C++ language feature with the C++ version in which it first appeared. The icons help you see C++ evolving, often from programming with low-level details to easier-to-use, higher-level forms of expression. These trends help reduce development times, and enhance performance, security and system maintainability. 20

P.2 Target Audiences

C++20 for Programmers: An Objects-Natural Approach has several target audiences:

- C++ software developers who want to learn the latest C++20 features in the context of a full-language, professional-style tutorial,
- non-C++ software developers who are preparing to do a C++ project and want to learn the latest version of C++,
- software developers who learned C++ in college or used it professionally some time ago and want to refresh their C++ knowledge in the context of C++20, and
- professional C++ trainers developing C++20 courses.

P.3 Live-Code Approach and Getting the Code

At the heart of the book is the Deitel signature **live-code approach**. Rather than code snippets, we show C++ as it’s intended to be used in the context of hundreds of complete, working, real-world C++ programs with live outputs.

Read the **Before You Begin** section that follows this Preface to learn how to set up your **Windows**, **macOS** or **Linux** computer to run the 200+ code examples consisting of approximately 15,000 lines of code. All the source code is available free for download at

- `https://github.com/pdeitel/CPlusPlus20ForProgrammers`
- `https://www.deitel.com/books/c-plus-plus-20-for-programmers`
- `https://informit.com/title/9780136905691` (see Section P.8)

For your convenience, we provide the book’s examples in C++ source-code (`.cpp` and `.h`) files for use with integrated development environments and command-line compilers. See **Chapter 1’s Test-Drives** (Section 1.2) for information on compiling and running the code examples with our three preferred compilers. Execute each program in parallel with reading the text to make your learning experience “come alive.” If you encounter a problem, you can reach us at

```
deitel@deitel.com
```

P.4 Three Industrial-Strength Compilers

We tested the code examples on the latest versions of

- **Visual C++**® in Microsoft® Visual Studio® Community edition on Windows®,

- **Clang C++** (`clang++`) in Apple® Xcode® on macOS®, and in a Docker® container, and
- **GNU® C++** (`g++`) on Linux® and in the GNU Compiler Collection (GCC) Docker® container.

At the time of this writing, most C++20 features are fully implemented by all three compilers, some are implemented by a subset of the three and some are not yet implemented by any. We point out these differences as appropriate and will update our digital content as the compiler vendors implement the remaining C++20 features. We'll also post code updates to the **book's GitHub repository**: 20

```
https://github.com/pdeitel/CPlusPlus20ForProgrammers
```

and both code and text updates on the book's websites:

```
https://www.deitel.com/books/c-plus-plus-20-for-programmers
https://informit.com/title/9780136905691
```

P.5 Programming Wisdom and Key C++20 Features

Throughout the book, we use margin icons to call your attention to **software-development wisdom** and **C++20 modules** and **concepts** features:

- **Software engineering observations** highlight architectural and design issues for proper software construction, especially for larger systems. SE
- **Security best practices** help you strengthen your programs against attacks. Sec
- **Performance tips** highlight opportunities to make your programs run faster or minimize the amount of memory they occupy. Perf
- **Common programming errors** help reduce the likelihood that you'll make the same mistakes. Err
- **C++ Core Guidelines** recommendations (introduced in Section P.9). CG
- C++20's new **modules** features. Mod
- C++20's new **concepts** features. Concepts

P.6 "Objects-Natural" Learning Approach

In Chapter 9, we'll cover how to develop **custom C++20 classes**, then continue our treatment of object-oriented programming throughout the rest of the book.

What Is Objects Natural?

In the early chapters, you'll work with **preexisting classes that do significant things**. You'll quickly create objects of those classes and get them to "strut their stuff" with a minimal number of simple C++ statements. We call this the **"Objects-Natural Approach."**

Given the massive numbers of free, open-source class libraries created by the C++ community, **you'll be able to perform powerful tasks long before you study how to create your own custom C++ classes in Chapter 9**. This is one of the most compelling aspects of working with object-oriented languages, in general, and with a mature object-oriented language like C++, in particular.

Free Classes

We emphasize using the huge number of valuable free classes available in the C++ ecosystem. These typically come from:

- the C++ Standard Library,
- platform-specific libraries, such as those provided with Microsoft Windows, Apple macOS or various Linux versions,
- free third-party C++ libraries, often created by the open-source community, and
- fellow developers, such as those in your organization.

We encourage you to view lots of free, open-source C++ code examples (available on sites such as GitHub) for inspiration.

The Boost Project

Boost provides 168 open-source C++ libraries.[5] It also serves as a "breeding ground" for new capabilities that are eventually incorporated into the C++ standard libraries. Some that have been added to Modern C++ include multithreading, random-number generation, smart pointers, tuples, regular expressions, file systems and `string_views`.[6] The following StackOverflow answer lists Modern C++ libraries and language features that evolved from the Boost libraries:[7]

`https://stackoverflow.com/a/8852421`

Objects-Natural Case Studies

Chapter 1 reviews the basic concepts and terminology of object technology. In the early chapters, you'll then create and use objects of preexisting classes long before creating your own custom classes in Chapter 9 and in the remainder of the book. Our **objects-natural case studies** include:

- Section 2.7—**Creating and Using Objects of Standard-Library Class `string`**
- Section 3.12—**Arbitrary-Sized Integers**
- Section 4.13—**Using the `miniz-cpp` Library to Write and Read ZIP files**
- Section 5.20—**Lnfylun Lhqtomh Wjtz Qarcv: Qjwazkrplm xzz Xndmwwqhlz** (this is the encrypted title of our **private-key encryption case study**)
- Section 6.15—**C++ Standard Library Class Template `vector`**
- Section 7.10—**C++20 `span`s: Views of Contiguous Container Elements**
- Section 8.19—**Reading/Analyzing a CSV File Containing Titanic Disaster Data**
- Section 8.20—**Intro to Regular Expressions**
- Section 9.22—**Serializing Objects with JSON (JavaScript Object Notation)**

---

5. "Boost 1.78.0 Library Documentation." Accessed January 9, 2022. `https://www.boost.org/doc/libs/1_78_0/`.
6. "Boost C++ Libraries." Wikipedia. Wikimedia Foundation. Accessed January 9, 2022. `https://en.wikipedia.org/wiki/Boost_(C%2B%2B_libraries)`.
7. Kennytm, Answer to "Which Boost Features Overlap with C++11?" Accessed January 9, 2022. `https://stackoverflow.com/a/8852421`.

A perfect example of the objects-natural approach is using objects of existing classes, like **`array`** and **`vector`** (Chapter 6), without knowing how to write custom classes in general or how those classes are written in particular. Throughout the rest of the book, we use existing C++ standard library capabilities extensively.

P.7 A Tour of the Book

The full-color table of contents graphic inside the front cover shows the book's modular architecture. As you read this Tour of the Book, also refer to that graphic. Together, the graphic and this section will help you quickly "scope out" the book's coverage.

This Tour of the Book points out many of the book's key features. The early chapters establish a solid foundation in C++20 fundamentals. The mid-range to high-end chapters and the case studies ease you into Modern C++20-based software development. Throughout the book, we discuss C++20's programming models:

- procedural programming,
- functional-style programming,
- object-oriented programming,
- generic programming and
- template metaprogramming.

Part 1: Programming Fundamentals Quickstart

Chapter 1, Intro and Test-Driving Popular, Free C++ Compilers: This book is for professional software developers, so Chapter 1

- presents a brief introduction,
- discusses Moore's law, multi-core processors and why standardized concurrent programming is important in Modern C++, and
- provides a brief refresher on object orientation, introducing terminology used throughout the book.

Then we jump right in with **test-drives** demonstrating how to compile and execute C++ code with our three preferred free compilers:

- **Microsoft's Visual C++** in Visual Studio on Windows,
- **Apple's Xcode** on macOS and
- **GNU's `g++`** on Linux.

We tested the book's code examples using each, pointing out the few cases in which a compiler does not support a particular feature. Choose whichever program-development environment(s) you prefer. The book also will work well with other C++20 compilers.

We also demonstrate GNU `g++` in the GNU Compiler Collection Docker container and Clang C++ in a Docker container. This enables you to run the latest GNU `g++` and clang++ command-line compilers on Windows, macOS or Linux. See Section P.13, Docker, for more information on this important developer tool. See the Before You Begin section for installation instructions.

For Windows users, we point to Microsoft's step-by-step instructions that allow you to install Linux in Windows via the Windows Subsystem for Linux (WSL). This is another way to use the `g++` and `clang++` compilers on Windows.

Chapter 2, Intro to C++ Programming, presents C++ fundamentals and illustrates key language features, including input, output, fundamental data types, arithmetic operators and their precedence, and decision making. **Section 2.7's objects-natural case study** demonstrates **creating and using objects of standard-library class `string`**—without you having to know how to develop custom classes in general or how that large complex class is implemented in particular).

Chapter 3, Control Statements: Part 1, focuses on **control statements**. You'll use the `if` and `if…else` selection statements, the `while` iteration statement for counter-controlled and sentinel-controlled iteration, and the increment, decrement and assignment operators. **Section 3.12's objects-natural case study** demonstrates **using a third-party library to create arbitrary-sized integers**.

Chapter 4, Control Statements: Part 2, presents C++'s other **control statements**—`for`, `do…while`, `switch`, `break` and `continue`—and the logical operators. **Section 4.13's objects-natural case study** demonstrates **using the `miniz-cpp` library to write and read ZIP files** programmatically.

Chapter 5, Functions and an Intro to Function Templates, introduces custom functions. We demonstrate **simulation techniques** with **random-number generation**. The random-number generation function `rand` that C++ inherited from C does not have good statistical Sec properties and can be predictable.[8] This makes programs using `rand` less secure. We include 11 a treatment of C++11's **more secure library of random-number capabilities** that can produce nondeterministic random numbers—a set of random numbers that can't be predicted. Such random-number generators are used in simulations and security scenarios where predictability is undesirable. We also discuss passing information between functions, and recursion. **Section 5.20's objects-natural case study** demonstrates **private-key encryption**.

Part 2: Arrays, Pointers and Strings

Chapter 6, `array`s, `vector`s, Ranges and Functional-Style Programming, begins our early coverage of the C++ standard library's containers, iterators and algorithms. We present the C++ standard library's **`array` container** for representing lists and tables of values. You'll define and initialize `array`s, and access their elements. We discuss passing `array`s to functions, sorting and searching `array`s and manipulating multidimensional `array`s. We begin our introduction to **functional-style programming with lambda expressions** 20 (anonymous functions) and **C++20's Ranges**—one of C++20's "big four" features. **Section 6.15's objects-natural case study** demonstrates the **C++ standard library class template `vector`. This entire chapter is essentially a large objects-natural case study of both `array`s and `vector`s.** The code in this chapter is a good example of Modern C++ coding idioms.

8. Fred Long, "Do Not Use the `rand()` Function for Generating Pseudorandom Numbers." Last modified by Jill Britton on November 20, 2021. Accessed December 27, 2021. `https://wiki.sei.cmu.edu/confluence/display/c/MSC30-C.+Do+not+use+the+rand%28%29+function+for+generating+pseudorandom+numbers`.

Chapter 7, (Downplaying) Pointers in Modern C++, provides thorough coverage of pointers and the intimate relationship among built-in pointers, pointer-based arrays and pointer-based strings (also called C-strings), each of which C++ inherited from the C programming language. Pointers are powerful but challenging to work with and are error-prone. So, we point out Modern C++ features that **eliminate the need for most pointers** Sec and make your code more robust and secure, including **`array`s** and **`vector`s**, **C++20 `span`s** 20 and **C++17 `string_view`s**. We still cover built-in arrays because they remain useful in C++ 17 and so you'll be able to read legacy code. **In new development, you should favor Modern C++ capabilities. Section 7.10's objects-natural case study** demonstrates one such capability—**C++20 `span`s**. These enable you to view and manipulate elements of contiguous 20 containers, such as pointer-based arrays and standard library `array`s and `vector`s, without using pointers directly. This chapter again emphasizes Modern C++ coding idioms.

Chapter 8, `string`s, `string_view`s, Text Files, CSV Files and Regex, presents many of the standard library `string` class's features; shows how to write text to, and read text from, both plain text files and comma-separated values (CSV) files (popular for representing datasets); and introduces string pattern matching with the standard library's regular-expression (regex) capabilities. C++ offers *two* types of strings—**`string`** objects and **C-style pointer-based strings**. We use **`string`** class objects to make programs more robust and **eliminate many of the security problems of C strings. In new development, you should** Sec **favor `string` objects**. We also present C++17's **`string_view`s**—a lightweight, flexible 17 mechanism for passing any type of string to a function. This chapter presents **two objects-natural case studies:**

- Section 8.19 introduces **data analysis by reading and analyzing a CSV file containing the Titanic Disaster dataset**—a popular dataset for introducing data analytics to beginners.
- Section 8.20 introduces **regular-expression pattern matching** and **text replacement.**

Part 3: Object-Oriented Programming

Chapter 9, Custom Classes, begins our treatment of **object-oriented programming** as we **craft valuable custom classes**. C++ is extensible—each class you create becomes a new type you can use to create objects. **Section 9.22's objects-natural case study** uses the third-party library **`cereal`** to convert objects into **JavaScript Object Notation (JSON)** format—a process known as **serialization**—and to **recreate those objects from their JSON representation—**known as **deserialization**.

Chapter 10, OOP: Inheritance and Runtime Polymorphism, focuses on the relationships among classes in an inheritance hierarchy and the powerful runtime polymorphic processing capabilities that these relationships enable. An important aspect of this chapter is understanding how polymorphism works. A key feature of this chapter is its detailed diagram and explanation of how C++ typically implements polymorphism, `virtual` functions and dynamic binding "under the hood." You'll see that it uses an elegant pointer-based data structure. We present other mechanisms to achieve runtime polymorphism, including the **non-virtual interface idiom** (NVI) and **`std::variant`/`std::visit`**. We also discuss **programming to an interface, not an implementation**.

Chapter 11, Operator Overloading, Copy/Move Semantics and Smart Pointers, shows how to enable C++'s existing operators to work with custom class objects, and introduces smart pointers and **dynamic memory management**. Smart pointers help you avoid
Err dynamic memory management errors by providing additional functionality beyond that of built-in pointers. We discuss **`unique_ptr`** in this chapter and **`shared_ptr`** and **`weak_ptr`** in online Chapter 20. A key aspect of this chapter is crafting valuable classes. We begin with a **`string` class test-drive**, presenting an elegant use of operator overloading before you implement your own customized class with overloaded operators. Then, in one of the book's most important case studies, you'll build your own custom `MyArray` class using overloaded operators and other capabilities to **solve various problems with C++'s native pointer-based arrays.**[9] We introduce and implement the five **special member functions** you can define in each class—the **copy constructor**, **copy assignment operator**, **move constructor**, **move assignment operator** and **destructor**. We discuss **copy semantics** and
Perf **move semantics**, which enable a compiler to move resources from one object to another
20 to avoid costly unnecessary copies. We introduce **C++20's three-way comparison operator** (**<=>**; also called the "**spaceship operator**") and show how to implement custom conversion operators. In Chapter 15, you'll convert `MyArray` to a class template that can store elements of a specified type. You will have truly crafted valuable classes.

Chapter 12, Exceptions and a Look Forward to Contracts, continues our **exception-handling** discussion that began in Chapter 6. We discuss when to use exceptions, exception safety guarantees, exceptions in the context of constructors and destructors, handling
Err dynamic memory allocation failures and why some projects do not use exception handling. The chapter concludes with an introduction to **contracts**—a potential future C++ feature that we demonstrate via an experimental contracts implementation available on `godbolt.org`. **A goal of contracts is to make most functions `noexcept`—meaning they**
Perf **do not throw exceptions—which might enable the compiler to perform additional optimizations and eliminate the overhead and complexity of exception handling.**

Part 4: Standard Library Containers, Iterators and Algorithms

Chapter 13, Standard Library Containers and Iterators, begins our broader and deeper treatment of three key C++ standard library components:

- **containers** (templatized data structures),
- **iterators** (for accessing container elements) and
- **algorithms** (which use iterators to manipulate containers).

We'll discuss **containers**, **container adaptors** and **near containers**. You'll see that the C++ standard library provides commonly used data structures, so you do not need to create your own—the vast majority of your data structures needs can be fulfilled by reusing these standard library capabilities. We demonstrate most standard library containers and introduce how iterators enable algorithms to be applied to various container types. You'll see that different containers support different kinds of iterators. We continue showing how
20 **C++20 Ranges** can simplify your code.

9. In industrial-strength systems, you'll use standard library classes for this, but this example enables us to demonstrate many key Modern C++ concepts.

Chapter 14, Standard Library Algorithms and C++20 Ranges & Views, presents many of the standard library's **115 algorithms**, focusing on common container manipulations, including filling containers with values, generating values, comparing elements or entire containers, removing elements, replacing elements, mathematical operations, searching, sorting, swapping, copying, merging, set operations, determining boundaries, and calculating minimums and maximums. We discuss minimum iterator requirements so you can determine which containers can be used with each algorithm. We begin discussing **C++20 Concepts**— another of C++20's "big four" features. The algorithms in **C++20's `std::ranges` namespace** use **C++20 Concepts** to specify their requirements. We continue our discussion of C++'s functional-style programming features with **C++20 Ranges and Views**. 20 20 20

Part 5: Advanced Topics

Chapter 15, Templates, C++20 Concepts and Metaprogramming, discusses **generic programming with templates**, which have been in C++ since the 1998 C++ standard was released. The importance of **Templates** has increased with each new C++ release. **A major Modern C++ theme is to do more at compile-time for better type checking and better runtime performance**—anything resolved at compile-time avoids runtime overhead and makes systems faster. As you'll see, templates and especially **template metaprogramming** are the keys to powerful **compile-time operations**. In this chapter, we'll take a deeper look at **templates**, showing how to develop custom class templates and exploring C++20 concepts. You'll create your own concepts, convert Chapter 11's `MyArray` case study to a class template with its own **iterators**, and work with **variadic templates** that can receive any number of template arguments. We'll introduce how to work with **C++ metaprogramming**. Perf 20

Chapter 16, C++20 Modules, presents another of C++20's "big four" features. **Modules** are a new way to organize your code, precisely control which declarations you expose to client code and encapsulate implementation details. Modules help developers be more productive, especially as they build, maintain and evolve large software systems. Modules help such systems build faster and make them more scalable. C++ creator Bjarne Stroustrup says, "*Modules offer a historic opportunity to improve code hygiene and compile times for C++ (bringing C++ into the 21st century).*"[10] You'll see that even in small systems, modules offer immediate benefits in every program by eliminating the need for the C++ preprocessor. We would have liked to integrate modules in our programs but, at the time of this writing, our key compilers are still missing various modules capabilities. Mod Perf SE

Chapter 17, Parallel Algorithms and Concurrency: A High-Level View, is one of the most important chapters in the book, presenting C++'s features for building applications that create and manage **multiple tasks**. This can significantly improve program performance and responsiveness. We show how to use **C++17's prepackaged parallel algorithms** to create **multithreaded programs** that will run faster (often much faster) on today's **multi-core computer architectures**. For example, we sort 100 million values using a sequential sort, then a parallel sort. We use C++'s **`<chrono>` library** features to profile the performance improvement we get on today's popular multi-core systems, as we employ an increasing number of cores. You'll see that the parallel sort runs 6.76 times faster than the Perf 17

10. Bjarne Stroustrup, "Modules and Macros." February 11, 2018. Accessed January 9, 2022. `http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0955r0.pdf`.

sequential sort on our Windows 10 64-bit computer using an 8-core Intel processor. We discuss the **producer–consumer relationship** and demonstrate various ways to implement 20 it using low-level and high-level C++ concurrency primitives, including C++20's new latch, barrier and semaphore capabilities. We emphasize that concurrent programming is difficult to get right and that you should aim **to use the higher-level concurrency features whenever possible. Lower-level features like semaphores and atomics can be used to implement higher-level features like latches.**

20 **Chapter 18, C++20 Coroutines**, presents **coroutines**—the last of C++20's "big four" features. **A coroutine is a function that can suspend its execution and be resumed later by another part of the program**. The mechanisms supporting this are handled entirely by code that's written for you by the compiler. You'll see that a function containing any of the keywords **`co_await`**, **`co_yield`** or **`co_return`** is a **coroutine** and that **coroutines enable you to do concurrent programming with a simple sequential-like coding style**. Coroutines require sophisticated infrastructure, which you can write yourself, but doing so is SE complex, tedious and error-prone. Instead, most experts agree that **you should use high-level coroutine support libraries**, which is the approach we demonstrate. The open-source community has created several experimental libraries for developing coroutines quickly 23 and conveniently—we use two in our presentation. C++23 is expected to have standard library support for coroutines.

Appendices

Appendix A, Operator Precedence Chart, lists C++'s operators in highest-to-lowest precedence order.

Appendix B, Character Set, shows characters and their corresponding numeric codes.

P.8 How to Get the Online Chapters and Appendices

We provide several **online chapters and appendices** on `informit.com`. Perform the following steps to register your copy of *C++20 for Programmers: An Objects-Natural Approach* on `informit.com` and access this online content:

1. Go to **`https://informit.com/register`** and sign in with an existing account or create a new one.
2. Under **Register a Product**, enter the ISBN **9780136905691**, then click **Submit**.
3. In your account page's **My Registered Products** section, click the **Access Bonus Content** link under *C++20 for Programmers: An Objects-Natural Approach*.

This will take you to the book's online content page.

Online Chapters

20 **Chapter 19, Stream I/O; C++20 Text Formatting: A Deeper Look**, discusses standard C++ input/output capabilities and legacy formatting features of the `<iomanip>` library. We include these formatting features primarily for programmers who might encounter them in **legacy C++ code**. We also present **C++20's new text-formatting features** in more depth.

Chapter 20, Other Topics, presents miscellaneous C++ topics and looks forward to new 23 features expected in C++23 and beyond.

Online Appendices

Appendix C, Number Systems, overviews the binary, octal, decimal and hexadecimal number systems.

Appendix D, Preprocessor, discusses additional features of the C++ preprocessor. Template metaprogramming (Chapter 15) and C++20 Modules (Chapter 16) obviate many of 20
this appendix's features.

Appendix E, Bit Manipulation, discusses bitwise operators for manipulating the individual bits of integral operands and bit fields for compactly representing integer data.

Web-Based Materials on `deitel.com`

Our `deitel.com` web page for the book

`https://deitel.com/c-plus-plus-20-for-programmers`

contains the following additional resources:

- Links to our **GitHub repository** containing the book's downloadable C++ source code
- Blog posts—`https://deitel.com/blog`
- Book updates

For more information about downloading the examples and setting up your C++ development environment, see the **Before You Begin** section.

P.9 C++ Core Guidelines

CG The **C++ Core Guidelines** (approximately 500 printed pages)

`https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines`

are recommendations "to help people use modern C++ effectively."[11] They're edited by Bjarne Stroustrup (C++'s creator) and Herb Sutter (Convener of the ISO C++ Standards Committee). According to the overview:

> "*The guidelines are focused on relatively high-level issues, such as interfaces, resource management, memory management, and concurrency. Such rules affect application architecture and library design. Following the rules will lead to code that is statically type safe, has no resource leaks, and catches many more programming logic errors than is common in code today. And it will run fast—you can afford to do things right.*"[12]

SE Err Perf

Throughout this book, we adhere to these guidelines as appropriate. You'll want to pay close attention to their wisdom. We point out many **C++ Core Guidelines** recommendations with a **CG icon**. There are hundreds of core guidelines divided into scores of categories and subcategories. Though this might seem overwhelming, static code analysis tools (Section P.10) can check your code against the guidelines. CG

11. C++ Core Guidelines, "Abstract." Accessed January 9, 2020. `https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#S-abstract`.
12. C++ Core Guidelines, "Abstract."

Guidelines Support Library

The C++ Core Guidelines often refer to capabilities of the **Guidelines Support Library (GSL)**, which implements helper classes and functions to support various recommendations.[13] Microsoft provides an open-source GSL implementation on GitHub at

`https://github.com/Microsoft/GSL`

We use GSL features in a few examples in the early chapters. Some GSL features have since been incorporated into the C++ standard library.

P.10 Industrial-Strength Static Code Analysis Tools

Err Sec **Static code analysis tools** let you quickly check your code for **common errors** and **security problems** and provide insights for code improvement. Using these tools is like having world-class experts checking your code. To help us adhere to the C++ Core Guidelines and improve our code in general, we used the following static-code analyzers:

- **`clang-tidy`**—`https://clang.llvm.org/extra/clang-tidy/`
- **`cppcheck`**—`https://cppcheck.sourceforge.io/`
- **Microsoft's C++ Core Guidelines static code analysis tools**, which are built into Visual Studio's static code analyzer

We used these three tools on the book's code examples to check for

- adherence to the C++ Core Guidelines,
- adherence to coding standards,
- adherence to modern C++ idioms,
- possible security problems,
- common bugs,
- possible performance issues,
- code readability
- and more.

Err We also used the compiler flag `-Wall` in the GNU `g++` and Clang C++ compilers to enable all compiler warnings. **With a few exceptions for warnings beyond this book's scope, we ensure that our programs compile without warning messages.** See the **Before You Begin** section for static analysis tool configuration information.

P.11 Teaching Approach

Sec *C++20 for Programmers: An Objects-Natural Approach* contains a rich collection of live-code examples. We stress program clarity and concentrate on building well-engineered software.

13. C++ Core Guidelines, "GSL: Guidelines Support Library." Accessed January 9, 2022. `https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#S-gsl`.

Using Fonts for Emphasis

We place the key terms and the index's page reference for each defining occurrence in **bold text** for easier reference. C++ code uses a `fixed-width font` (e.g., `x = 5`). We place on-screen components in the **bold Helvetica** font (e.g., the **File** menu).

Syntax Coloring

For readability, we syntax color all the code. In our e-books, our syntax-coloring conventions are as follows:

```
comments appear in green
keywords appear in dark blue
constants and literal values appear in light blue
errors appear in red
all other code appears in black
```

Objectives and Outline

Each chapter begins with objectives that tell you what to expect.

Tables and Illustrations

Abundant tables and line drawings are included.

Programming Tips and Key Features

We call out programming tips and key features with icons in margins (see Section P.5).

Index

For convenient reference, we've included an extensive index, with defining occurrences of key terms highlighted with a **bold** page number.

P.12 Developer Resources

StackOverflow

StackOverflow is one of the most popular developer-oriented, question-and-answer sites. Many problems programmers encounter have already been discussed here, so it's a great place to find solutions to those problems and post questions about new ones. Many of our Google searches for various, often complex, issues throughout our writing effort returned StackOverflow answers as their first results.

GitHub

"The best way to prepare [to be a programmer] is to write programs, and to study great programs that other people have written. In my case, I went to the garbage cans at the Computer Science Center and fished out listings of their operating systems."[14]—William Gates

GitHub is an excellent venue for finding free, open-source code to incorporate into your projects—and for you to contribute your code to the open-source community if you like. Fifty million developers use GitHub.[15] The site hosts over 200 million repositories for

14. William Gates, quoted in *Programmers at Work: Interviews with 19 Programmers Who Shaped the Computer Industry* by Susan Lammers. Microsoft Press, 1986, p. 83.
15. "GitHub." Accessed January 7, 2022. `https://github.com/`.

code written in an enormous number of programming languages[16]—developers contributed to 61+ million repositories in the last year.[17] **GitHub** is a crucial element of the professional software developer's arsenal with **version-control tools** that help developer teams manage public open-source projects and private projects.

There is a massive C++ open-source community. On GitHub, there are over 41,000[18] C++ code repositories. You can check out other people's C++ code on GitHub and even build upon it if you like. This is a great way to learn and is a natural extension of our live-code teaching approach.[19]

In 2018, Microsoft purchased **GitHub** for $7.5 billion. As a software developer, you're almost certainly using GitHub regularly. According to Microsoft's CEO, Satya Nadella, the company bought GitHub to "*empower every developer to build, innovate and solve the world's most pressing challenges.*"[20]

We encourage you to study and execute lots of developers' open-source C++ code on GitHub and to contribute your own.

P.13 Docker

We use **Docker**—a tool for packaging software into **containers** that bundle everything required to execute that software conveniently and portably across platforms. Some software packages require complicated setup and configuration. For many of these, you can download free preexisting Docker containers, avoiding complex installation issues. You can simply execute software locally on your desktop or notebook computers, making Docker a great way to help you get started with new technologies quickly, conveniently and economically.

We show how to install and execute Docker containers preconfigured with

- the GNU Compiler Collection (GCC), which includes the `g++` compiler, and
- the latest version of Clang's **`clang++`** compiler.

Each can run in **Docker** on **Windows**, **macOS** and **Linux**.

Docker also helps with **reproducibility**. Custom Docker containers can be configured with the software and libraries you use. This would enable others to recreate the environment you used, then reproduce your work, and will help you reproduce your own results. Reproducibility is especially important in the sciences and medicine—for example, when researchers want to prove and extend the work in published articles.

P.14 Some Key C++ Documentation and Resources

The book includes over 900 citations to videos, blog posts, articles and online documentation we studied while writing the manuscript. You may want to access some of these resources to investigate more advanced features and idioms. The website **`cppreference.com`** has become the defacto C++ documentation site. We reference it frequently so

16. "Where the World Builds Software." Accessed January 7, 2022. `https://github.com/about`.
17. "The 2021 State of the Octoverse." Accessed January 7, 2022. `https://octoverse.github.com`.
18. "C++." Accessed January 7, 2022. `https://github.com/topics/cpp`.
19. Students will need to become familiar with the variety of open-source licenses for software on GitHub.
20. "Microsoft to Acquire GitHub for $7.5 Billion." Accessed January 7, 2022. `https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/`.

you can get more details about the standard C++ classes and functions we use throughout the book. We also frequently reference the final draft of the **C++20 standard document**, which is available for free on GitHub at

`https://timsong-cpp.github.io/cppwp/n4861/`

You may also find the following C++ resources helpful as you work through the book.

Documentation

- C++20 standard document final draft adopted by the C++ Standard Committee: 20
  `https://timsong-cpp.github.io/cppwp/n4861/`
- C++ Reference at `cppreference.com`:
  `https://cppreference.com/`
- Microsoft's C++ language documentation:
  `https://docs.microsoft.com/en-us/cpp/cpp/`
- The GNU C++ Standard Library Reference Manual:
  `https://gcc.gnu.org/onlinedocs/libstdc++/manual/index.html`

Blogs

- Sutter's Mill Blog—Herb Sutter on software development:
  `https://herbsutter.com/`
- Microsoft's C++ Team Blog:
  `https://devblogs.microsoft.com/cppblog`
- Marius Bancila's Blog:
  `https://mariusbancila.ro/blog/`
- Jonathan Boccara's Blog:
  `https://www.fluentcpp.com/`
- Bartlomiej Filipek's Blog:
  `https://www.cppstories.com/`
- Rainer Grimm's Blog:
  `http://modernescpp.com/`
- Arthur O'Dwyer's Blog:
  `https://quuxplusone.github.io/blog/`

Additional Resources

- Bjarne Stroustrup's website:
  `https://stroustrup.com/`
- Standard C++ Foundation website:
  `https://isocpp.org/`
- C++ Standard Committee website:
  `http://www.open-std.org/jtc1/sc22/wg21/`

P.15 Getting Your Questions Answered

Popular C++ and general programming online forums include

- `https://stackoverflow.com`
- `https://www.reddit.com/r/cpp/`
- `https://groups.google.com/g/comp.lang.c++`
- `https://www.dreamincode.net/forums/forum/15-c-and-c/`

For a list of other valuable sites, see

`https://www.geeksforgeeks.org/stuck-in-programming-get-the-solution-from-these-10-best-websites/`

Sec Also, vendors often provide forums for their tools and libraries. Many libraries are managed and maintained at `github.com`. Some library maintainers provide support through the **Issues** tab on a given library's GitHub page.

Communicating with the Authors

As you read the book, if you have questions, we're easy to reach at

`deitel@deitel.com`

We'll respond promptly.

P.16 Join the Deitel & Associates, Inc. Social Media Communities

Join the Deitel social media communities on

- LinkedIn®—`https://bit.ly/DeitelLinkedIn`
- YouTube®—`https://youtube.com/DeitelTV`
- Twitter®—`https://twitter.com/deitel`
- Facebook®—`https://facebook.com/DeitelFan`

P.17 Deitel Pearson Products on O'Reilly Online Learning

If you're at a company or college, your organization might have an **O'Reilly Online Learning** subscription, giving you free access to all of Deitel's Pearson e-books and LiveLessons videos hosted on the site, as well as Paul Deitel's live, one-day Full Throttle training courses, offered on a continuing basis. Individuals may sign up for a **10-day free trial** at

`https://learning.oreilly.com/register/`

For a list of all our current products and courses on O'Reilly Online Learning, visit

`https://deitel.com/LearnWithDeitel`

Textbooks and Professional Books

Each Deitel e-book on O'Reilly Online Learning is presented in full color, extensively indexed and text searchable. As we write our professional books, they're posted on

O'Reilly Online Learning for early "rough cut" access, then replaced with the book's final content once published. The final e-book for ***C++20 for Programmers: An Objects-Natural Approach*** is available to O'Reilly subscribers at 20

```
https://learning.oreilly.com/library/view/c-20-for-programmers/
   9780136905776
```

Asynchronous LiveLessons Video Products

Learn hands-on with Paul Deitel as he presents compelling, leading-edge computing technologies in C++, Java, Python and Python Data Science/AI (and more coming). Access to our ***C++20 Fundamentals LiveLessons*** videos is available to O'Reilly subscribers at

```
https://learning.oreilly.com/videos/c-20-fundamentals-parts/
   9780136875185
```

These videos are ideal for self-paced learning. At the time of this writing, we're still recording this product. Additional videos will be posted as they become available during Q1 and Q2 of 2022. The final video product will contain 50–60 hours of video—approximately the equivalent of two college semester courses.

Live Full-Throttle Training Courses

Paul Deitel's live **Full-Throttle training courses** at O'Reilly Online Learning

```
https://deitel.com/LearnWithDeitel
```

are one-full-day, presentation-only, fast-paced, code-intensive introductions to Python, Python Data Science/AI, Java, C++20 Fundamentals and the C++20 Standard Library. 20 These courses are for experienced developers and software project managers preparing for projects using other languages. After taking a Full-Throttle course, participants often watch the corresponding *LiveLessons* video course, which has many more hours of classroom-paced learning.

P.18 Live Instructor-Led Training with Paul Deitel

Paul Deitel has been teaching programming languages to developer audiences for three decades. He presents a variety of one- to five-day C++, Python and Java corporate training courses, and teaches Python with an Introduction to Data Science for the UCLA Anderson School of Management's Master of Science in Business Analytics (MSBA) program. His courses can be delivered worldwide on-site or virtually. Please contact **`deitel@deitel.com`** for a proposal customized to meet your company's or academic program's needs.

P.19 College Textbook Version of *C++20 for Programmers*

Our college textbook, ***C++ How to Program, Eleventh Edition***, will be available in three digital formats:

- **Online e-book** offered through popular e-book providers.
- Interactive **Pearson eText** (see below).
- Interactive **Pearson Revel** with assessment (see below).

All of these textbook versions include standard **"How to Program" features** such as:

- A chapter introducing hardware, software and Internet concepts.

- An introduction to programming for novices.
- End-of-section programming and non-programming **Checkpoint self-review exercises with answers**.
- **End-of-chapter exercises**.

Deitel Pearson eTexts and Revels include:

- **Videos** in which Paul Deitel discusses the material in the book's core chapters.
- Interactive programming and non-programming **Checkpoint self-review exercises with answers**.
- **Flashcards** and other learning tools.

In addition, **Pearson Revels** include interactive programming and non-programming automatically graded exercises, as well as instructor course-management tools, such as a grade book.

Supplements available to qualified college instructors teaching from the textbook include:

- **Instructor solutions manual** with solutions to most of the end-of-chapter exercises.
- **Test-item file** with four-part, code-based and non-code-based multiple-choice questions with answers.
- Customizable **PowerPoint lecture slides**.

Please write to **`deitel@deitel.com`** for more information.

P.20 Acknowledgments

We'd like to thank Barbara Deitel for long hours devoted to Internet research on this project. We're fortunate to have worked with the dedicated team of publishing professionals at Pearson. We appreciate the efforts and 27-year mentorship of our friend and colleague Mark L. Taub, Vice President of the Pearson IT Professional Group. Mark and his team publish our professional books and LiveLessons video products, and sponsor our live online training seminars, offered through the O'Reilly Online Learning service:

`https://learning.oreilly.com/`

Charvi Arora recruited the book's reviewers and managed the review process. Julie Nahil managed the book's production. Chuti Prasertsith designed the cover.

Reviewers

We were fortunate on this project to have 10 distinguished professionals review the manuscript. Most of the reviewers are either on the ISO C++ Standards Committee, have served on it or have a working relationship with it. Many have contributed features to the language. They helped us make a better book—any remaining flaws are our own.

- Andreas Fertig, Independent C++ Trainer and Consultant, Creator of `cppinsights.io`, Author of *Programming with C++20* 20
- Marc Gregoire, Software Architect, Nikon Metrology, Microsoft Visual C++ MVP and author of *Professional C++, 5/e* (which is up-to-date with C++20) 20

- Dr. Daisy Hollman, ISO C++ Standards Committee Member
- Danny Kalev, Ph.D. and Certified System Analyst and Software Engineer, Former ISO C++ Standards Committee Member
- Dietmar Kühl, Senior Software Developer, Bloomberg L.P., ISO C++ Standard Committee Member
- Inbal Levi, SolarEdge Technologies, ISO C++ Foundation director, ISO C++ SG9 (Ranges) chair, ISO C++ Standards Committee member
- Arthur O'Dwyer, C++ trainer, Chair of CppCon's Back to Basics track, author of several accepted C++17/20/23 proposals and the book *Mastering the C++17 STL* 17 20 23
- Saar Raz, Senior Software Engineer, Swimm.io and Implementor of C++20 Concepts in Clang 20
- José Antonio González Seco, Parliament of Andalusia
- Anthony Williams, Member of the British Standards Institution C++ Standards Panel, Director of Just Software Solutions Ltd., Author of C++ *Concurrency in Action, 2/e* (Anthony is the author or co-author of many C++ Standard Committee papers that led to C++'s standardized concurrency features)

Arthur O'Dwyer
We'd like to call out the extraordinary efforts Arthur O'Dwyer put into reviewing our manuscript. While working through his comments, we learned a great deal about C++'s subtleties and especially Modern C++ coding idioms. In addition to carefully marking each chapter PDF we sent him, Arthur provided a separate comprehensive document explaining his comments in detail, often rewriting code and providing external resources that offered additional insights. As we applied all the reviewers' comments, we always looked forward to what Arthur had to say, especially regarding the more challenging issues. He's a busy professional, yet he was generous with his time and always constructive. He insisted that we "get it right" and worked hard to help us do that. Arthur teaches C++ to professionals. He taught us a much about how to do C++ right.

GitHub
Thanks to GitHub for making it easy for us to share our code and keep it up-to-date, and for providing the tools that enable 73+ million developers to contribute to 200 million+ code repositories.[21] These tools support the massive open-source communities that provide libraries for today's popular programming languages, making it easier for developers to create powerful applications and avoid "reinventing the wheel."

Matt Godbolt and Compiler Explorer
Thanks to Matt Godbolt, creator of **Compiler Explorer** at `https://godbolt.org`, which enables you to compile and run programs in many programming languages. Through this site, you can test your code

- on most popular C++ compilers—including our three preferred compilers—and
- across many released, developmental and experimental compiler versions.

21. "Where the World Builds Software." Accessed January 7, 2022. `https://github.com/about`.

For example, we used an experimental `g++` compiler version to demonstrate **contracts** (Chapter 12, Exceptions and a Look Forward to Contracts), which we hope to see standardized in a future C++ language version. Several of our reviewers used `godbolt.org` to demonstrate suggested changes to us, helping us improve the book.

Dietmar Kühl

We would like to thank Dietmar Kühl, Senior Software Developer at Bloomberg L.P. and an ISO C++ Committee member, for sharing with us his views on inheritance and static and dynamic polymorphism. His insights helped us shape our presentations of these topics in Chapters 10 and 15.

Rainer Grimm

Our thanks to Rainer Grimm (`http://modernescpp.com/`), among the Modern C++ community's most prolific bloggers. As we got deeper into C++20, our Google searches frequently pointed us to his writings. Rainer Grimm is a professional C++ trainer who offers courses in German and English. He is the author of several C++ books, including *C++20: Get the Details*, *Concurrency with Modern C++*, *The C++ Standard Library, 3/e* and *C++ Core Guidelines Explained*. He is already blogging about features likely to appear in C++23.

Brian Goetz

We were privileged to have as a reviewer on one of our other books—*Java How to Program, 10/e*—Brian Goetz, Oracle Java Language Architect and co-author of *Java Concurrency in Practice*. He provided us with many insights and constructive comments, especially on

- inheritance hierarchy design, which influenced our design decisions for several examples in **Chapter 10, OOP: Inheritance and Runtime Polymorphism**, and
- Java concurrency, which influenced our approach to C++20 concurrency in **Chapter 17, Parallel Algorithms and Concurrency: A High-Level View**.

Open-Source Contributors and Bloggers

A special note of thanks to the technically oriented people worldwide who contribute to the open-source movement and blog about their work online, and to their organizations that encourage the proliferation of such open software and information.

Google Search

Thanks to Google, whose search engine answers our constant stream of queries, each in a fraction of a second, at any time day or night—and at no charge. It's the single best productivity enhancement tool we've added to our research process in the last 20 years.

Grammarly

We now use the paid version of **Grammarly** on all our manuscripts. They describe their tools as helping you "compose bold, clear, mistake-free writing" with their "AI-powered writing assistant."[22] They also say, "Using a variety of innovative approaches—including advanced machine learning and deep learning—we consistently break new ground in nat-

22. "Grammarly." Accessed January 15, 2022. `https://www.grammarly.com`.

ural language processing (NLP) research to deliver unrivaled assistance."[23] Grammarly provides free tools that you can integrate into several popular web browsers, Microsoft® Office 365™ and Google Docs™. They also offer more powerful premium and business tools. You can view their free and paid plans at

```
https://www.grammarly.com/plans
```

As you read the book and work through the code examples, we'd appreciate your comments, criticisms, corrections and suggestions for improvement. Please send all correspondence, including questions, to

```
deitel@deitel.com
```

We'll respond promptly.

Welcome to the exciting world of C++20 programming. We've enjoyed writing 11 editions of our academic and professional C++ content over the last 30 years. We hope you have an informative, challenging and entertaining learning experience with ***C++20 for Programmers: An Objects-Natural Approach*** and enjoy this look at leading-edge, Modern C++ software development. 20

Paul Deitel
Harvey Deitel

About the Authors

Paul J. Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is an MIT graduate with 42 years in computing. Paul is one of the world's most experienced programming-languages trainers, having taught professional courses to software developers since 1992. He has delivered hundreds of programming courses to academic, industry, government and military clients of Deitel & Associates, Inc. internationally, including UCLA, Cisco, IBM, Siemens, Sun Microsystems (now Oracle), Dell, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, Puma, iRobot and many more. He and his co-author, Dr. Harvey M. Deitel, are among the world's best-selling programming-language textbook, professional book, video and interactive multimedia e-learning authors, and virtual- and live-training presenters.

Dr. Harvey M. Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 61 years of experience in computing. Dr. Deitel earned B.S. and M.S. degrees in Electrical Engineering from MIT and a Ph.D. in Mathematics from Boston University—he studied computing in each of these programs before they spun off Computer Science departments. He has extensive industry and college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates in 1991 with his son, Paul. The Deitels' publications have earned international recognition, with more than 100 translations published in Japanese, German, Russian, Spanish, French, Polish, Italian, Simplified Chinese, Traditional Chinese, Korean, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of programming courses to academic, corporate, government and military clients.

23. "Our Mission." Accessed January 15, 2022. `https://www.grammarly.com/about`.

About Deitel® & Associates, Inc.

Deitel & Associates, Inc., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring and corporate-training organization, specializing in computer programming languages, object technology, mobile app development and Internet and web software technology. The company's training clients include some of the world's largest companies, government agencies, branches of the military, and academic institutions. The company offers instructor-led training courses delivered virtually and live at client sites worldwide, and virtually for Pearson Education on O'Reilly Online Learning (`https://learning.oreilly.com`), formerly called Safari Books Online.

Through its 47-year publishing partnership with Pearson, Deitel & Associates, Inc., publishes leading-edge programming professional books and college textbooks in print and e-book formats, LiveLessons video courses, O'Reilly Online Learning live training courses and Revel™ interactive multimedia college courses.

To contact Deitel & Associates, Inc. and the authors, or to request a proposal for virtual or on-site, instructor-led training worldwide, write to

`deitel@deitel.com`

To learn more about Deitel virtual and on-site corporate training, visit

`https://deitel.com/training`

Individuals wishing to purchase Deitel books can do so at

`https://amazon.com`
`https://www.barnesandnoble.com/`

Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Pearson. For corporate and government sales, send an email to

`corpsales@pearsoned.com`

Deitel e-books are available in various formats from

`https://www.amazon.com/`	`https://www.vitalsource.com/`
`https://www.barnesandnoble.com/`	`https://www.redshelf.com/`
`https://www.informit.com/`	`https://www.chegg.com/`

To register for a free 10-day trial to O'Reilly Online Learning, visit

`https://learning.oreilly.com/register/`