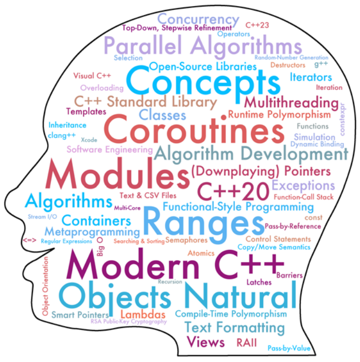

Before You Begin

Before using this book, please read this section to understand our conventions and set up your computer to compile and run our example programs. If there are changes to the instructions presented here, we'll post updates on the book's webpage:

```
https://deitel.com/cpphtp11
```

Font and Naming Conventions

We use fonts to distinguish application elements and C++ code elements from regular text:

- We use a **bold sans-serif font** for on-screen application elements, such as "the **File** menu."
- We use a `sans-serif font` for commands and C++ code elements, as in `sqrt(9)`.

Obtaining the Code Examples

Download the *C++ How to Program: An Objects-Natural Approach, 11/e* code examples from our GitHub repository at

```
https://github.com/pdeitel/CPlusPlusHowToProgram11e
```

If you're familiar with Git and GitHub, clone the repository to your system. If you're not a GitHub user, click the green **Code** button and select **Download ZIP** to download a ZIP file containing the code. To extract the ZIP file's contents:

- Windows: Right-click the ZIP file, select **Extract All...**, select your user account's `Documents` folder, then click **Extract**.
- macOS: Move the ZIP file to your user account's `Documents` folder, then double-click the ZIP file.
- Linux (varies by distribution): When you download the ZIP file on Ubuntu Linux, you can choose to open the file with the **Archive Manager** or save it. Choose **Archive Manager**, then click **Extract** in the window that appears. Select your user account's `Documents` folder, then click **Extract** again.

Throughout the book, our instructions assume the code examples reside in your user account's `Documents` folder in a subfolder named `examples`.

If you're not familiar with Git and GitHub but are interested in learning about these essential developer tools, check out

```
https://guides.github.com/activities/hello-world/
```

Compilers We Use

Ensure that you have a recent C++ compiler installed. We tested the book's code examples using the following free compilers:

- For Microsoft Windows, we used Microsoft Visual Studio Community edition, which includes the Visual C++ compiler and other Microsoft development tools.
- For Linux, we used the GNU C++ compiler (`g++`)[1]—part of the GNU Compiler Collection (GCC). Typically, a version of GNU C++ is pre-installed on most Linux systems. You might need to update the compiler to a more recent version. GNU C++ also can be installed on macOS and Windows systems.
- For macOS, we used both the GNU C++ compiler (`g++`) and the Apple Xcode[2] C++ compiler, which uses a version of the LLVM Clang C++ compiler (`clang++`).
- You can run the latest versions of GNU C++ (`g++`) and LLVM Clang C++ (`clang++`)[3] conveniently on Windows, macOS and Linux via Docker containers. See the "Docker and Docker Containers" section in this Before You Begin section.

At the time of this writing, Apple Xcode does not support several key C++20 features we use throughout this book, so we recommend using the most recent version of `g++`. When Xcode's C++20 support changes, we'll post updates at

```
https://deitel.com/cpphtp11
```

This Before You Begin describes installing the compilers and Docker. Section 1.11's test-drives demonstrate how to compile and run C++ programs using these compilers.

Installing Visual Studio Community Edition on Windows

If you are a Windows user, first ensure that your system meets the requirements for Microsoft Visual Studio Community edition at

```
https://docs.microsoft.com/en-us/visualstudio/releases/2022/system-
   requirements
```

Next, go to

```
https://visualstudio.microsoft.com/downloads/
```

Then perform the following installation steps:

1. Click **Free Download** under **Community**.
2. Depending on your web browser, you may see a pop-up at the bottom of your screen where you can click **Run** to start the installation process. If not, double-click the installer file in your **Downloads** folder when the download completes.
3. In the **User Account Control** dialog, click **Yes** to allow the installer to make changes to your system.
4. In the **Visual Studio Installer** dialog, click **Continue** to allow the installer to download the components it needs for you to configure your installation.

1. GNU C++ (`g++`) 13.1 at the time of this writing.
2. Xcode 14.3.1 at the time of this writing.
3. Clang C++ (`clang++`) 16 at the time of this writing.

5. For this book's examples, select the option **Desktop Development with C++**, which includes the Visual C++ compiler and the C++ standard libraries.
6. Click **Install**. The installation process can take a significant amount of time.

Installing Xcode on macOS

On macOS, perform the following steps to install Xcode:

1. Click the Apple menu and select **App Store…**, or click the **App Store** icon in the dock at the bottom of your Mac screen.
2. In the **App Store**'s **Search** field, type **Xcode**.
3. Click the **Get** button to install Xcode.

Installing the GNU C++ (g++) 13 on macOS

On macOS perform the following steps to install GNU C++ on macOS:

1. In the **Finder**'s **Go** menu, select **Utilities**, then double-click **Terminal** to open a **Terminal** (command line) window.
2. Check if the `brew` command is installed by typing `brew` and pressing press *Enter* (or *return*). If macOS does not recognize the command, go to `https://brew.sh` and copy the installation command below **Install Homebrew**. Paste this command into the **Terminal** window, then press *Enter* (or *return*).
3. Type the following command, then press *Enter* (or *return*) to install the GNU Compiler Collection (GCC), which includes `g++`:

```
brew install gcc@13
```

Installing the GNU C++ (g++) 13 on Linux

There are many Linux distributions, and they often use different software upgrade techniques. Check your distribution's online documentation for instructions on how to upgrade GNU C++ to the latest version. You also can download GNU C++ for various platforms at

```
https://gcc.gnu.org/install/binaries.html
```

Docker and Docker Containers

Docker is a tool for packaging software into **containers** (also called **images**) that bundle everything required to execute that software across platforms, which is particularly useful for software packages with complicated setups and configurations. For many such packages, there are free preexisting Docker containers (often at `https://hub.docker.com`) that you can download and execute locally on your system. Docker is a great way to get started with new technologies quickly and to experiment with new compiler versions.

Installing Docker

To use a Docker container, you must first install Docker. Windows and macOS users should download and run the **Docker Desktop** installer from

```
https://www.docker.com/get-started
```

Then follow the on-screen instructions. Also, sign up for a **Docker Hub** account on this site, which gives you access to the many containers at `https://hub.docker.com`. Linux users should install **Docker Engine** from

```
https://docs.docker.com/engine/install/
```

Getting the GNU Compiler Collection Docker Container

The GNU team maintains official Docker containers at

```
https://hub.docker.com/_/gcc
```

Once Docker is installed and running, open a Command Prompt[4] (Windows), Terminal (macOS/Linux) or shell (Linux), then execute the command

```
docker pull gcc:latest
```

Docker downloads a container configured with the GNU Compiler Collection (GCC)'s most current version—13.1 at the time of this writing. In one of Section 1.11's test-drives, we'll demonstrate how to execute the container and use it to compile and run C++ programs.

Getting an LLVM Clang C++ Docker Container

Currently, the LLVM Clang team does not provide an official Docker container, but many working containers are available on `https://hub.docker.com`. For this book, we used a popular one from

```
https://hub.docker.com/r/teeks99/clang-ubuntu
```

Open a Command Prompt (Windows), Terminal (macOS/Linux) or shell (Linux), then execute the command

```
docker pull teeks99/clang-ubuntu:16
```

Docker downloads a container configured with LLVM Clang's most current version—16 at the time of this writing. In one of Section 1.11's test-drives, we'll demonstrate how to execute the container and use it to compile and run C++ programs.

Getting Your C++ Questions Answered

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

```
deitel@deitel.com
```

and

```
https://deitel.com/contact-us
```

We'll respond promptly.

The web is loaded with programming information. An invaluable resource for nonprogrammers and programmers alike is the website

```
https://stackoverflow.com
```

on which you can

- search for answers to common programming questions,
- search for error messages to see what causes them,

4. Windows users should choose **Run as administrator** when opening the Command Prompt.

- ask programming questions to get answers from programmers worldwide and
- gain valuable insights about programming in general.

For live C++ discussions, check out the Slack channel cpplang:

```
https://cpplang-inviter.cppalliance.org
```

and the Discord server `#include<C++>`:

```
https://www.includecpp.org/discord/
```

Online C++ Documentation

For C++ standard library documentation, visit

```
https://cppreference.com
```

Also, be sure to check out the C++ FAQ at

```
https://isocpp.org/faq
```

Static Code Analysis Tools

We used the following static code analyzers to check our code examples 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:

- **`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

You can install `clang-tidy` on Linux with the following commands:

```
sudo apt-get update -y
sudo apt-get install -y clang-tidy
```

You can install `cppcheck` for various operating-system platforms by following the instructions at `https://cppcheck.sourceforge.io/`.

For Visual C++, once you learn how to create a project in Section 1.11's test-drives, you can configure Microsoft's C++ Core Guidelines static code analysis tools as follows:

1. Right-click your project name in the **Solution Explorer** and select **Properties**.
2. In the dialog that appears, select **Code Analysis > General** in the left column, then set **Enable Code Analysis on Build** to **Yes** in the right column.
3. Next, select **Code Analysis > Microsoft** in the left column. Then, in the right column, you can select a subset of the analysis rules from the drop-down list. We used the option **<Choose multiple rule sets...>** to select all the rules that begin with **C++ Core Check**. Click **Save As...**, give your custom rule set a name, click **Save**, then click **Apply**.