IN THIS CHAPTER

- Running a simple C++ program
- Working with C++ on Microsoft Windows, Apple macOS, and Linux
- Using an integrated development environment

Chapter 1
Compiling and Running Your First C++ Application

A C++ compiler translates your source files (which contain the text of your application's program code written in C++) into machine code that your computer can understand and run. Different C++ compilers run on the Microsoft Windows, Apple macOS, and Linux operating systems, but they all adhere to the international standard for C++, so if you write your program code in C++, you should be able to compile and run your application on any of the platforms.

This chapter has links to download and install the C++ compiler on the Windows, macOS, and Linux platforms. It demonstrates how to edit, compile, and run a simple C++ application on the command line (that is, in a terminal window) in each of the platforms.

Then it briefly looks at using an integrated development environment (IDE) and shows you the advantages an IDE offers to C++ developers. Many IDEs are available, so this chapter only gives an introduction to using one. The rest of this book assumes you're compiling and running programs on the command line.

Looking at a Simple C++ Program

Use any text editor (such as Windows Notepad, macOS TextEdit, or Linux vi) to create a plain-text file named `hello.cpp` containing the following C++ code:

```
#include <iostream>

using namespace std;

int main()
{
    for (int i = 1; i <= 5; i++)
    {
        cout << i << " Hello, world!" << endl;
    }
}
```

Using C++ on Microsoft Windows

On the Windows platform, the C++ compiler is a utility program called CL that you run on the command line in a special command window.

To install CL, follow these steps:

1. **Go to** `https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2022`**.**
2. **Scroll down to Build Tools for Visual Studio and click the Download button.**

   An installer downloads onto your computer.
3. **Run the installer to install CL and its tools.**

You must run CL in a special command window. Left-click the Start button at the bottom of the screen. Scroll down to the `Visual Studio 2022` folder, and click the down arrow to the right of the folder to reveal the folder's contents. Left-click Developer Command Prompt for VS 2022 to open a command window especially

set up to run CL. On the command line of that window, enter the following command on one line:

```
cl /EHsc /std:c++20 hello.cpp /link /out:hello.exe
```

This command compiles the source file and generates the executable file `hello.exe`. To run the executable file, enter the following command:

```
hello
```

In later chapters, an application may consist of multiple source files. Then, instead of specifying a single source file like `hello.cpp`, you would tell CL to compile all the source files in the directory whose names end with `.cpp`. For example:

```
cl /EHsc /std:c++20 *.cpp /link /out:books.exe
```

These example commands include the `/std:c++20` parameter to specify using the standard 2020 version of C++. You should be able compile this book's examples with earlier versions of C++ by specifying `/std:c++14` or `/std:c++17`.

The `/link` option says that the next option is meant for the *linkage editor*, a tool that combines the output of the compiler into an executable program. The `/out` parameter tells the linkage editor what to name the executable program.

`/EHsc` tells the compiler how to generate code to handle exceptions, which are runtime errors. Exception handling is briefly covered in Chapter 9.

TIP

For a walk-through on compiling and running a C++ program on the command line, go to `https://learn.microsoft.com/en-us/cpp/build/walkthrough-compiling-a-native-cpp-program-on-the-command-line`. For descriptions of the CL compiler options, go to `https://learn.microsoft.com/en-us/cpp/build/reference/compiler-options-listed-alphabetically`.

Using C++ on Apple macOS

On macOS, you can use the Clang C++ compiler. For instructions on how to install the compiler, go to `https://osxdaily.com/2023/05/02/how-install-gcc-mac`.

After installing the compiler, go to the folder where you stored your source file, `hello.cpp`, and enter the following command in a Terminal window:

```
g++ -std=c++0x -o hello hello.cpp
```

This command runs the C++ compiler to compile the source file and generates the executable file `hello`. To run the executable file, enter the following command:

```
./hello
```

In later chapters, an application may consist of multiple source files. Then, instead of specifying a single source file like `hello.cpp`, you would tell `g++` to compile all the source files in the directory whose names end with `.cpp`. For example:

```
g++ -std=c++20 -o books *.cpp
```

These example commands include the `-std:c++20` parameter to specify using the standard 2020 version of C++. You should be able compile this book's examples with versions as early as 2011 by specifying `-std=c++11` or `-std=c++0x`. The `-o` parameter gives the name of the executable program.

Using C++ on Linux

On Linux, you can use the GNU C++ compiler. For instructions on how to install the compiler, go to `www.cherryservers.com/blog/how-to-install-gcc-on-ubuntu`.

After installing the compiler, go to the folder where you stored your source file, `hello.cpp`. Enter the following command in a Terminal window:

```
g++ -std=c++20 -o hello hello.cpp
```

This command runs the C++ compiler to compile the source file and generates the application `hello`. To run the executable file, enter the following command:

```
./hello
```

In later chapters, an application may consist of multiple source files. Then, instead of specifying a single source file, like `hello.cpp`, you would tell `g++` to compile all the source files in the directory whose names end with `.cpp`. For example:

```
g++ -std=c++20 -o books *.cpp
```

These example commands include the `-std:c++20` or `-std=c++2a` parameter to specify using the standard 2020 version of C++. You should be able compile this book's examples with versions as early as 2011 by specifying `-std=c++11` or `-std=c++0x`). The `-o` parameter gives the name of the executable program.

Working with an Integrated Development Environment

An IDE is an application that combines a syntax-aware code editor, a compiler, a debugger, and often other tools. Many IDEs are cross-platform — they have versions that run on different platforms. Others only run on particular platforms. Some IDEs for C++ include the following:

- CLion (cross-platform; `www.jetbrains.com/clion`)
- Code::Blocks (cross-platform; `www.codeblocks.org`)
- Eclipse CDT (cross-platform; `https://projects.eclipse.org/projects/tools.cdt`)
- Microsoft Visual C++ (`https://visualstudio.microsoft.com/downloads`)
- Visual Studio (cross-platform; `https://visualstudio.microsoft.com`)
- Xcode (Apple macOS; `https://developer.apple.com/xcode`)

An IDE's syntax-aware code editor makes it easy to edit C++ code. It can use color-coding to highlight different source code components, such as keywords, and flag syntax errors as you type the code.

The debugger has features to help you get your code to run correctly. You can set breakpoints at statements to cause your program to temporarily pause at those statements during execution. When your program is paused at a breakpoint, you can examine the current values of variables. You can *single-step* (execute one statement at a time while you monitor the values of variables), or you can resume normal program execution.

Figure 1-1 shows debugging `hello.cpp` in Eclipse CDT. The program is paused at the `cout` statement. You can see that the current value of variable `i` is `4`.

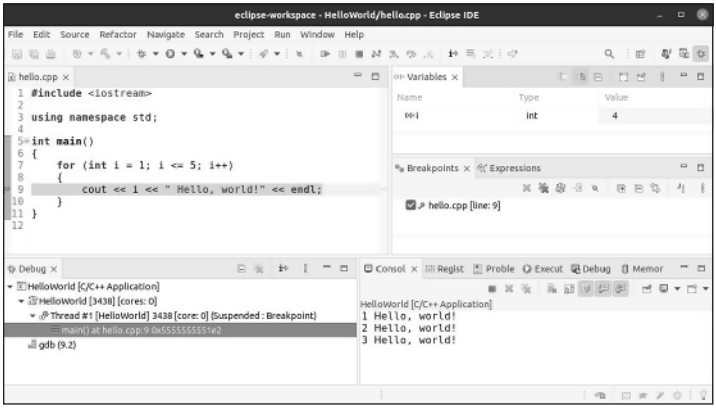

FIGURE 1-1: Using the Eclipse CDT integrated development environment (IDE).

Using an IDE greatly simplifies C++ application development, but how to use an IDE is beyond the scope of this book. If you're interested in learning more, you can find online tutorials for any IDE you want to use.