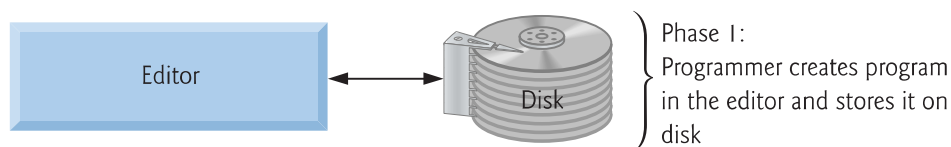


## 1.9 Typical C++ Development Environment

C++ systems generally consist of three parts: a program development environment, the language and the C++ Standard Library. C++ programs typically go through six phases: edit, preprocess, compile, link, load and execute. The following discussion explains a typical C++ program development environment.

### *Phase 1: Editing a Program*

Phase 1 consists of editing a file with an *editor program*, normally known simply as an *editor* (Fig. 1.6). You type a C++ program (typically referred to as **source code**) using the editor, make any necessary corrections and save the program on a secondary storage device, such as your hard drive. C++ source code filenames often end with the .cpp, .cxx, .cc or .C extensions (note that C is in uppercase) which indicate that a file contains C++ source code. See the documentation for your C++ compiler for more information on file-name extensions.



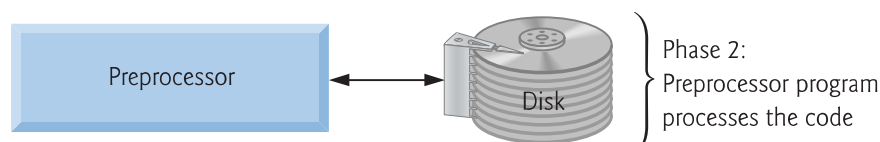
**Fig. 1.6** | Typical C++ development environment—editing phase.

Two editors widely used on Linux systems are vi and emacs. C++ software packages for Microsoft Windows such as Microsoft Visual C++ ([microsoft.com/express](http://microsoft.com/express)) have editors integrated into the programming environment. You can also use a simple text editor, such as Notepad in Windows, to write your C++ code.

For organizations that develop substantial information systems, **integrated development environments (IDEs)** are available from many major software suppliers. IDEs provide tools that support the software-development process, including editors for writing and editing programs and debuggers for locating **logic errors**—errors that cause programs to execute incorrectly. Popular IDEs include Microsoft® Visual Studio 2012 Express Edition, Dev C++, NetBeans, Eclipse, Apple’s Xcode and CodeLite.

### *Phase 2: Preprocessing a C++ Program*

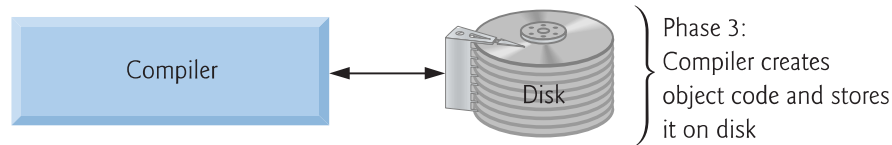
In Phase 2, you give the command to **compile** the program (Fig. 1.7). In a C++ system, a **preprocessor** program executes automatically before the compiler’s translation phase begins (so we call preprocessing Phase 2 and compiling Phase 3). The C++ preprocessor obeys commands called **preprocessing directives**, which indicate that certain manipulations are to be performed on the program before compilation. These manipulations usually include other text files to be compiled, and perform various text replacements. The most common preprocessing directives are discussed in the early chapters; a detailed discussion of preprocessor features appears in Appendix E, Preprocessor.



**Fig. 1.7** | Typical C++ development environment—preprocessor phase.

*Phase 3: Compiling a C++ Program*

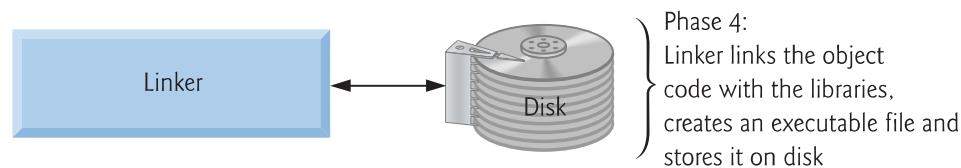
In Phase 3, the compiler translates the C++ program into machine-language code—also referred to as object code (Fig. 1.8).



**Fig. 1.8** | Typical C++ development environment—compilation phase.

*Phase 4: Linking*

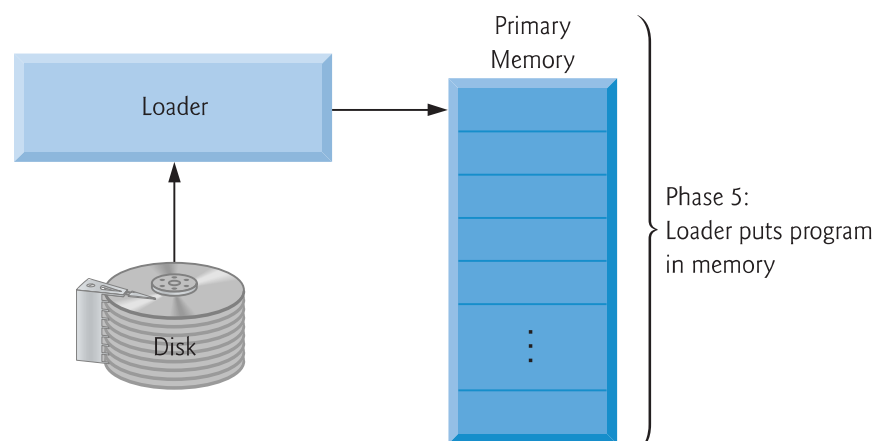
Phase 4 is called **linking**. C++ programs typically contain references to functions and data defined elsewhere, such as in the standard libraries or in the private libraries of groups of programmers working on a particular project (Fig. 1.9). The object code produced by the C++ compiler typically contains “holes” due to these missing parts. A **linker** links the object code with the code for the missing functions to produce an **executable program** (with no missing pieces). If the program compiles and links correctly, an executable image is produced.



**Fig. 1.9** | Typical C++ development environment—linking phase.

*Phase 5: Loading*

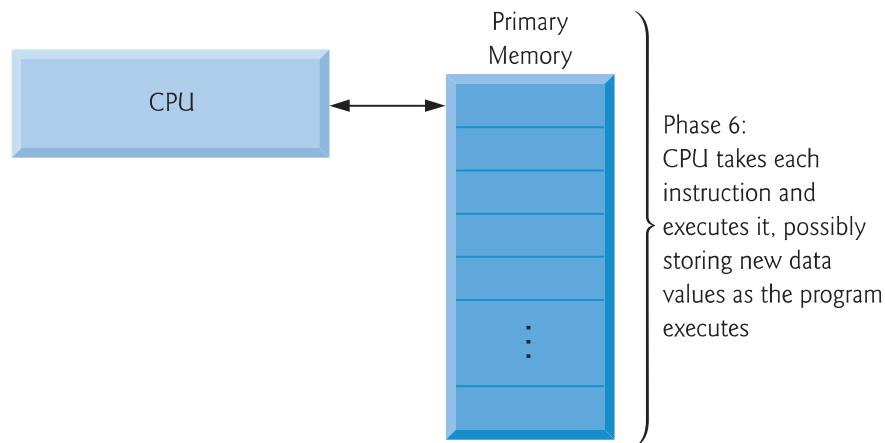
Phase 5 is called **loading**. Before a program can be executed, it must first be placed in memory (Fig. 1.10). This is done by the **loader**, which takes the executable image from disk and transfers it to memory. Additional components from shared libraries that support the program are also loaded.



**Fig. 1.10** | Typical C++ development environment—loading phase.

*Phase 6: Execution*

Finally, the computer, under the control of its CPU, **executes** the program one instruction at a time (Fig. 1.11). Some modern computer architectures can execute several instructions in parallel.



**Fig. 1.11** | Typical C++ development environment—execution phase.

*Problems That May Occur at Execution Time*

Programs might not work on the first try. Each of the preceding phases can fail because of various errors that we'll discuss throughout this book. For example, an executing program might try to divide by zero (an illegal operation for integer arithmetic in C++). This would cause the C++ program to display an error message. If this occurred, you'd have to return to the edit phase, make the necessary corrections and proceed through the remaining phases again to determine that the corrections fixed the problem(s). [Note: Most programs in C++ input or output data. Certain C++ functions take their input from `cin` (the **standard input stream**; pronounced "see-in"), which is normally the keyboard, but `cin` can be redirected to another device. Data is often output to `cout` (the **standard output stream**; pronounced "see-out"), which is normally the computer screen, but `cout` can be redirected to another device. When we say that a program prints a result, we normally mean that the result is displayed on a screen. Data may be output to other devices, such as disks and hard-copy printers. There is also a **standard error stream** referred to as **cerr**. The `cerr` stream (normally connected to the screen) is used for displaying error messages.

**Common Programming Error 1.1**

Errors such as division by zero occur as a program runs, so they're called **runtime errors** or **execution-time errors**. **Fatal runtime errors** cause programs to terminate immediately without having successfully performed their jobs. **Nonfatal runtime errors** allow programs to run to completion, often producing incorrect results.

## 1.10 Test-Driving a C++ Application

In this section, you'll run and interact with your first C++ application. You'll begin by running an entertaining guess-the-number game, which picks a number from 1 to 1000 and prompts you to guess it. If your guess is correct, the game ends. If your guess is not correct,