Errors can be made during coding that can prevent the program from successfully compiling and linking. So part of the coding step involves resolving errors that prevent the program from running.

A common error is called a ***syntax error.*** A syntax error occurs when you key a command or some other part of the program incorrectly. Computers must be told exactly what to do. If someone leaves you a note that says "Lock the back *dore* before you leave," you will be able to figure out what the instruction is. When the computer recognizes a syntax error, the programmer is notified immediately. Everything has to be just right or the computer will not accept it.

There are other errors that the computer may detect when compiling. Most of them are easily resolved. When all of those errors are resolved, the program will compile, link, and be ready to run. Even if a program runs, it may still fail to do its job correctly. That is where the next step of the programming process comes in.

On the Net

There are different kinds of programming errors. It is important to understand that a program which runs is not necessarily error-free. Learn more about errors and bugs at http://www.ProgramCPP.com. See topic 2.3.1.

TESTING AND DEBUGGING

Testing and debugging is an important step that is too often ignored. Programs typically fail to operate 100% correctly the first time they are compiled or interpreted. Logic errors and other hidden problems called ***bugs*** must be located. Software must be tested and "debugged" to make sure the output is correct and reliable.

Why Bugs?

Back when computers used vacuum tubes, the heat and light generated by the tubes sometimes attracted bugs such as moths. The bugs sometimes caused short circuits, resulting in the need to "debug" the computer.

One way to test a program is to provide input for which the results are known. For example, a program that converts meters to feet can be easily tested by giving the program input values for which you know the output. Carefully select a wide variety of inputs. Use values that are larger or smaller than those which are typical. Use zero and negative numbers as inputs when allowable.

You should also test every part of a program. Make sure you provide input that tests every line of your code. Test each part of the program repeatedly to make sure that consistent results are obtained.

A type of error that can cause your program to stop running (or ***crash***) is called a ***run-time error.*** A run-time error occurs when a program gives the computer an instruction that it is incapable of executing. A run-time error may lead to a program "crash." For example, if your program tries to divide a number by zero, a run-time error will occur on most systems. A run-time error could also occur if the system runs out of memory while your program is running.

You will experience lots of bugs and errors as a programmer. They are a part of every programmer's day. Even the best programmers spend lots of time testing and debugging. Throughout this book you will be warned of possible pitfalls and bugs so that you can avoid as many as possible. But the best way to learn how to avoid bugs is to experience them.

DOCUMENTING AND MAINTAINING THE PROGRAM

This fifth step applies mostly to programs used in the real world. But since you may someday write such programs, you should be aware of this step as well. Programmers must document their work so that they and other programmers can make changes or updates later. Documentation may also have to be written for the program's users.

The Importance of Documentation

You should document your programs while you are programming and avoid saving the task for last. The time to write documentation for a program is while you are programming. By the time you finish the programming, you may have already forgotten some of what you did.

You may also be less likely to write proper documentation once a program is complete. You may think your time is better spent on another project. You will be pleased to have the documentation when it is needed.

DOCUMENTATION IN THE PROGRAM

Documentation that is included in the program itself is very important. Virtually all programming languages allow comments to be included in the source code. The comments are ignored by the interpreter or the compiler. Therefore the programmer can include notes and explanations that will make the program easier for people to read. You will learn how to use comments in your source code in the next chapter.

DOCUMENTATION OUTSIDE OF THE PROGRAM

Many times a program is complex enough that documents should be written that explain how the programming problem was solved. This documentation might be diagrams, flowcharts, or descriptions.

DOCUMENTATION FOR THE USER

You have probably already been exposed to user documentation. Programs that are to be used by more than a few people usually include user documentation that explains the functions of the software.

PROGRAM MAINTENANCE

Maintenance is an important part of the programming process. Most programs are written to help with a task. As the task changes, the program must also change. Users are likely to request additions and changes be made to the program. Maintaining a program is an important part of the process because it keeps the programmer's work up-to-date and in use.