CHAPTER 10

Program Files and Preprocessing Directives

In this chapter, you'll look into subjects that relate to how multiple program files and header files interact, and to how you can manage and control the contents of your program files.

I have yet to discuss the major topic, classes. You'll begin to define your own data types in the next chapter, and this chapter will (in part) form a gentle introduction to classes. The material of this chapter has implications for how you define your data types; I'll discuss those implications as I come to them.

In this chapter you will learn the following:

- The details of how header files and program files interrelate
- What a translation unit is
- What linkage is and why it is important
- What namespaces are and how you create and use them
- What preprocessing is, and how to use the preprocessing directives that are available
- The basic ideas in debugging, and what debugging help you can get from preprocessing and the standard library

Working with Program Files

Way back in Chapter 1, I talked about how your C++ program will, in general, consist of multiple files. Let's recap a little. Two basic kinds of files are involved:

header files are commonly identified with the file name extension .h (note that some older systems use .hpp). These files contain type definitions and other code that is used in one or more source files in a program.
source files usually have the extension .cpp although .c, .cxx, and other extensions may also be used. These files contain the code that compiles to machine instructions—primarily function definitions. Any required header files are added to a source file by means of #include directives.

As you have seen, the standard headers for library functions in ANSI C++ (for example, <iostream>) have no extension, so it's always obvious whether or not an #include directive is for a standard header. You use a special notation in an #include directive where a standard library header name is enclosed between angled brackets:

```
#include <iostream>
```

**NOTE** You also need to remember that you shouldn't add spaces between the angled brackets, otherwise the header name won't be recognized.

Of course, you may come across other kinds of files that support the environment in which you are programming (defining resources of one kind or another, perhaps) but the .h and .cpp files are the ones that contain all your C++ code. This is illustrated in Figure 10-1.

*Figure 10-1. Your C++ program consists of .cpp and .h files.*