had the first widely-used graphical interface. The Macintosh helped set the standard for other GUI operating systems.

Later, Microsoft and IBM created their own graphical operating systems. Figure 2-13 shows Microsoft's Windows 95 operating system. Windows and IBM's OS/2 do more than put a pretty face on the DOS prompt. They allow multiple programs to be run at the same time and provide resources that programs can share. They also make it easier to learn new programs, because each program has the same look and feel.

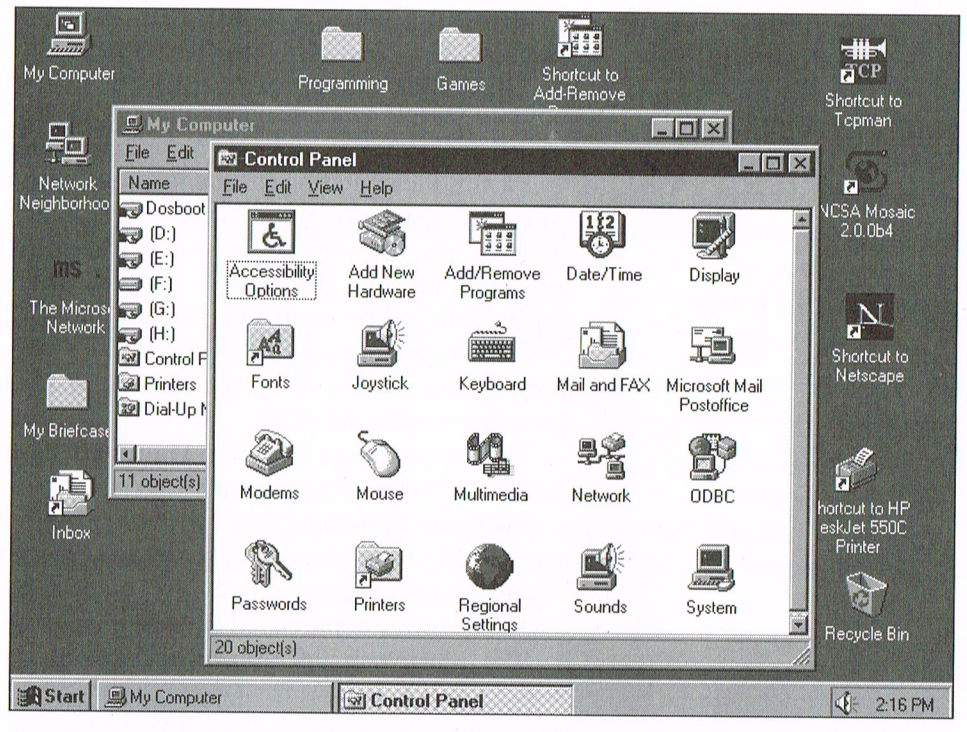

FIGURE 2-13
Microsoft Windows 95 is a graphical alternative to the DOS prompt.

SECTION 2.2 QUESTIONS

1. Give an example of a low-level programming language.
2. List three examples of high-level programming languages.
3. Describe the process involved when using a compiler to program a computer.
4. Describe one advantage that compiled programs have over interpreted programs.
5. List three operations managed by operating systems.

CHAPTER 2, SECTION 3

The Programming Process

Programmers are always tempted to immediately begin writing code to solve a problem. There is a better way. Sure, if you are writing a program to print your name on the screen a million times you might get by with just sitting down and keying in a program. But most programs are more complicated, and therefore a more structured and disciplined approach to programming is necessary.

Although there are different approaches used by different programmers, most good programmers follow five basic steps when developing programs:

1. Define the problem.
2. Develop an algorithm.
3. Code the program.
4. Test and debug the program.
5. Document and maintain the program.

DEFINING THE PROBLEM

Defining the problem to be solved requires an understanding of what the program is to accomplish.

For example, a program that calculates interest on a loan is fairly easy to define. Start by identifying the inputs and outputs. As input, the program needs the loan amount, the interest rate, and the number of months that the money is to be borrowed. A specific known formula can be applied to the data, and the amount of interest is the output.

Many programs are more difficult to define. Suppose you are defining a game program that involves characters in a maze. In your definition, the abilities of each character must be defined. In addition, the maze and how the characters interact with the maze and each other must also be defined. The list goes on and on.

Imagine how much there is to define before writing a program to handle airline reservations for a world-wide airline or the software that controls the launch of the space shuttle. Before any part of the program is written, the programmer must know exactly what the goal is.

Defining the problem does not take into consideration *how* the program will do the job, just *what* the job is. Exactly how a program accomplishes its work is addressed in the second step of the process.

DEVELOPING AN ALGORITHM

The second step in the programming process is to develop an algorithm. An ***algorithm*** is a set of sequential instructions that are followed to solve a problem. Algorithms have been commonly used for years. A recipe for baking a cake, instructions for assembling a bicycle, and directions to a shopping mall are all examples of algorithms. The directions to a mall, shown in Figure 2-14, are a set of steps that you execute sequentially.