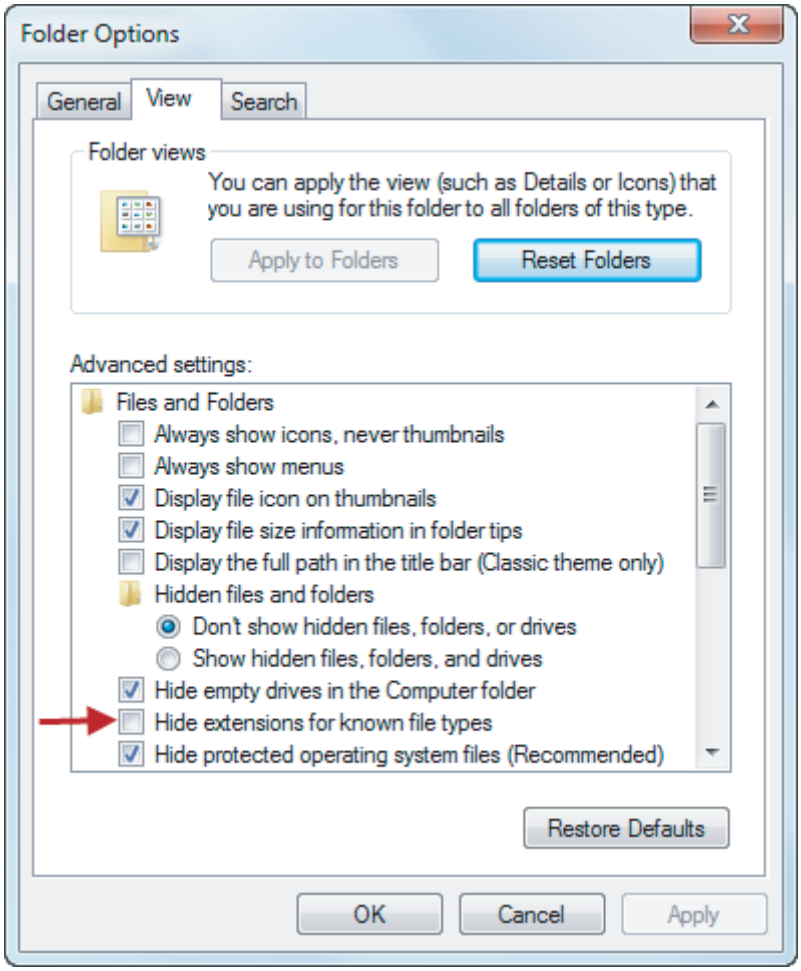

FIGURE 1.3 **Folder Options dialog box.**

2. Click on the View tab in the dialog box. (A dialog box similar to the one in Fig. 1.3 will appear.)
3. If there is a check mark in the box next to "Hide extensions for known file types," click on the box to remove the check mark.
4. Click on the *OK* button to close the Folder Options dialog box.

1.3 Program Development Cycle

We learned in Section 1.1 that hardware refers to the machinery in a computer system (such as the monitor, keyboard, and CPU) and software refers to a collection of instructions, called a **program**, that directs the hardware. Programs are written to solve problems or perform tasks on a computer. Programmers translate the solutions or tasks into a language the computer can understand. As we write programs, we must keep in mind that the computer will do only what we instruct it to do. Because of this, we must be very careful and thorough with our instructions. ***Note:*** A program is also known as a **project**, **application,** or **solution**.

Performing a Task on the Computer

The first step in writing instructions to carry out a task is to determine what the **output** should be—that is, exactly what the task should produce. The second step is to identify the data, or

input, necessary to obtain the output. The last step is to determine how to **process** the input to obtain the desired output—that is, to determine what formulas or ways of doing things can be used to obtain the output.

This problem-solving approach is the same as that used to solve word problems in an algebra class. For example, consider the following algebra problem:

How fast is a car moving if it travels 50 miles in 2 hours?

The first step is to determine the type of answer requested. The answer should be a number giving the speed in miles per hour (the output). (*Speed* is also called *velocity*.) The information needed to obtain the answer is the distance and time the car has traveled (the input). The formula

$$\text{speed} = \text{distance/time}$$

is used to process the distance traveled and the time elapsed in order to determine the speed. That is,

$$\begin{aligned}\text{speed} &= 50 \text{ miles}/2 \text{ hours}\\ &= 25 \text{ miles/hour}\end{aligned}$$

A pictorial representation of this problem-solving process is

We determine what we want as output, get the needed input, and process the input to produce the desired output.

In the chapters that follow we discuss how to write programs to carry out the preceding operations. But first we look at the general process of writing programs.

Program Planning

A baking recipe provides a good example of a plan. The ingredients and the amounts are determined by what is to be baked. That is, the *output* determines the *input* and the *processing*. The recipe, or plan, reduces the number of mistakes you might make if you tried to bake with no plan at all. Although it's difficult to imagine an architect building a bridge or a factory without a detailed plan, many programmers (particularly students in their first programming course) try to write programs without first making a careful plan. The more complicated the problem, the more complex the plan may be. You will spend much less time working on a program if you devise a carefully thought out step-by-step plan and test it before actually writing the program.

Many programmers plan their programs using a sequence of steps, referred to as the **program development cycle**. The following step-by-step process will enable you to use your time efficiently and help you design error-free programs that produce the desired output.

1. ***Analyze:*** Define the problem.

   Be sure you understand what the program should do—that is, what the output should be. Have a clear idea of what data (or input) are given and the relationship between the input and the desired output.

2. ***Design:*** Plan the solution to the problem.

   Find a logical sequence of precise steps that solve the problem. Such a sequence of steps is called an **algorithm**. Every detail, including obvious steps, should appear in the algorithm. In

the next section, we discuss three popular methods used to develop the logic plan: flowcharts, pseudocode, and top-down charts. These tools help the programmer break a problem into a sequence of small tasks the computer can perform to solve the problem. Planning also involves using representative data to test the logic of the algorithm by hand to ensure that it is correct.

3. ***Design the interface:*** Select the objects (text boxes, buttons, etc.).

   Determine how the input will be obtained and how the output will be displayed. Then create objects to receive the input and display the output. Also, create appropriate buttons and menus to allow the user to control the program.

4. ***Code:*** Translate the algorithm into a programming language.

   Coding is the technical word for writing the program. During this stage, the program is written in Visual Basic and entered into the computer. The programmer uses the algorithm devised in Step 2 along with a knowledge of Visual Basic.

5. ***Test and debug:*** Locate and remove any errors in the program.

   Testing is the process of finding errors in a program, and **debugging** is the process of correcting errors that are found. (An error in a program is called a **bug**.) As the program is typed, Visual Basic points out certain kinds of program errors. Other kinds of errors will be detected by Visual Basic when the program is executed; however, many errors due to typing mistakes, flaws in the algorithm, or incorrect use of the Visual Basic language rules can be uncovered and corrected only by careful detective work. An example of such an error would be using addition when multiplication was the proper operation.

6. ***Complete the documentation:*** Organize all the material that describes the program.

   Documentation is intended to allow another person, or the programmer at a later date, to understand the program. Internal documentation (comments) consists of statements in the program that are not executed but point out the purposes of various parts of the program. Documentation might also consist of a detailed description of what the program does and how to use it (for instance, what type of input is expected). For commercial programs, documentation includes an instruction manual and on-line help. Other types of documentation are the flowchart, pseudocode, and hierarchy chart that were used to construct the program. Although documentation is listed as the last step in the program development cycle, it should take place as the program is being coded.

1.4 Programming Tools

This section discusses some specific algorithms and describes three tools used to convert algorithms into computer programs: flowcharts, pseudocode, and hierarchy charts.

You use algorithms every day to make decisions and perform tasks. For instance, whenever you mail a letter, you must decide how much postage to put on the envelope. One rule of thumb is to use one stamp for every five sheets of paper or fraction thereof. Suppose a friend asks you to determine the number of stamps to place on an envelope. The following algorithm will accomplish the task.

1. Request the number of sheets of paper; call it Sheets. *(input)*
2. Divide Sheets by 5. *(processing)*
3. Round the quotient up to the next highest whole number; call it Stamps. *(processing)*
4. Reply with the number Stamps. *(output)*

The preceding algorithm takes the number of sheets (Sheets) as input, processes the data, and produces the number of stamps needed (Stamps) as output. We can test the algorithm for a letter with 16 sheets of paper.