A: Let's look at the following example:

```
printf("Enter a number: ");
scanf("%d", &i);
```

Suppose that the user enters a valid number, followed by nonnumeric characters:

```
Enter a number: 23foo
```

In this case, scanf reads the 2 and the 3, storing 23 in i. The remaining characters (foo) are left to be read by the next call of scanf (or some other input function). On the other hand, suppose that the input is invalid from the beginning:

```
Enter a number: foo
```

In this case, the value of i is undefined and foo is left for the next scanf.

What can we do about this sad state of affairs? Later, we'll see how to test whether a call of scanf has succeeded. If the call fails, we can have the program either terminate or try to recover, perhaps by discarding the offending input and asking the user to try again. (Ways to discard bad input are discussed in the Q&A section at the end of Chapter 22.)

detecting errors in scanf ➤22.3

**Q: I don't understand how scanf can "put back" characters and read them again later. [p. 44]**

A: As it turns out, programs don't read user input as it is typed. Instead, input is stored in a hidden buffer, to which scanf has access. It's easy for scanf to put characters back into the buffer for subsequent reading. Chapter 22 discusses input buffering in more detail.

**Q: What does scanf do if the user puts punctuation marks (commas, for example) between numbers?**

A: Let's look at a simple example. Suppose that we try to read a pair of integers using scanf:

```
printf("Enter two numbers: ");
scanf("%d%d", &i, &j);
```

If the user enters

```
4,28
```

scanf will read the 4 and store it in i. As it searches for the beginning of the second number, scanf encounters the comma. Since numbers can't begin with a comma, scanf returns immediately. The comma and the second number are left for the next call of scanf.

Of course, we can easily solve the problem by adding a comma to the format string if we're sure that the numbers will *always* be separated by a comma:

```
printf("Enter two numbers, separated by a comma: ");
scanf("%d,%d", &i, &j);
```