- If you're writing a C program that requires input, you must create a place to store it. For text input, that place is a string variable, which you create by using the `char` keyword.
- Variables are officially introduced in Chapter 8 in this book. For now, consider the string variable that `scanf()` uses as merely a storage chamber for text you type.
- The formatting codes used by `scanf()` are identical to those used by `printf()`. In real life, you use them mostly with `printf()` because there are better ways to read the keyboard than to use `scanf()`. Refer to Table 24-2 in Chapter 24 for a list of the formatting percent-sign placeholder codes.
- Forgetting to stick the `&` in front of `scanf()`'s variable is a common mistake. Not doing so leads to some wonderful *null pointer assignment* errors that you may relish in the years to come. As a weird quirk, however, the ampersand is optional when you're dealing with string variables. Go figure.

The miracle of `scanf()`

Consider the following pointless program, COLOR.C, which uses two string variables, `name` and `color`. It asks for your name and then your favorite color. The final `printf()` statement then displays what you enter.

```
#include <stdio.h>

int main()
{
   char name[20];
   char color[20];

   printf("What is your name?");
   scanf("%s",name);
   printf("What is your favorite color?");
   scanf("%s",color);
   printf("%s's favorite color is %s\n",name,color);
   return(0);
}
```

Enter this source code into your editor. Save this file to disk as COLOR.C. Compile.

If you get any errors, double-check your source code and reedit the file. A common mistake: forgetting that there are two commas in the final `printf()` statement.