The format for using the basic `printf` function is

```
printf("text");
```

`printf` is always written in lowercase. It's a must. It's followed by parentheses, which contain a quoted string of text, *text* (see the example). It's `printf()`'s job to display that text on the screen.

In the C language, `printf()`is a complete statement. A semicolon always follows the last parenthesis. (Okay, you may see an exception, but it's not worth fussing over at this point in the game.)

- Although *text* is enclosed in double quotes, they aren't part of the message that `printf()` puts up on the screen.
- You have to follow special rules about the text you can display, all of which are covered in Chapter 24.
- The format shown in the preceding example is simplified. A more advanced format for `printf()` appears later in this chapter.

Printing funky text

Ladies and gentlemen, I give you the following:

```
Ta da!  I am a text string.
```

It's a simple collection of text, numbers, letters, and other characters — but it's not a string of text. Nope. For those characters to be considered as a unit, they must be neatly enclosed in double quotes:

```
"Ta da!  I am a text string."
```

Now you have a string of text, but that's still nothing unless the computer can manipulate it. For manipulation, you need to wrap up the string in the bun-like parentheses:

```
("Ta da!  I am a text string.")
```

Furthermore, you need an engine — a *function* — to manipulate the string. Put `printf` on one side and a semicolon on the other:

```
printf("Ta da!  I am a text string.");
```

And, you have a hot dog of a C command to display the simple collection of text, numbers, letters, and other characters on the screen. Neat and tidy.