Type a character, such as the **$** (dollar sign). Press Enter after typing the character.

Uh-oh! What happened? You see something like this:

```
Type another character:'$' is greater than '
'$
```

```
What? What? What?
```

You bet! A problem. Right in the middle of the chapter that talks about comparing single-character variables, I have to totally pick up the cat and talk about something else vital in C programming: properly reading single characters from the keyboard.

The problem with `getchar()`

Despite what it says in the brochure, `getchar()` *does not* read a single character from the keyboard. That's what the `getchar()` function *returns,* but it's not what the function does.

Internally, `getchar()` reads what's called *standard input*. It grabs the first character you type and stores it, but afterward it sits and waits for you to type something that signals "the end."

Two characters signal the end of input: The Enter key is one. The second is the EOF, or end-of-file, character. In Windows, it's the Ctrl+Z character. In the Unix-like operating systems, it's Ctrl+D.

Run the GREATER program, from the preceding section. When it asks for input, type **123** and then press the Enter key. Here's what you see for output:

```
Which character is greater?
Type a single character:123
Type another character:'2' is greater than '1'
```

When you're first asked to type a single character, you provide standard input for the program: the string `123` plus the press of the Enter key.

The `getchar()` function reads standard input. First, it reads the `1` and places it in the variable *`a`* (Line 9 in GREATER.C). Then, the program uses a second `getchar()` function to read again from standard input — but it has already been supplied; `2` is read into the variable *`b`* (refer to Line 11).