idiom
```
while (getchar() != '\n')    /* skips rest of line */
  ;
```

The resulting loop is a well-known C idiom that's cryptic but worth learning.

`getchar` is useful in loops that skip characters as well as loops that search for characters. Consider the following statement, which uses `getchar` to skip an indefinite number of blank characters:

idiom
```
while ((ch = getchar()) == ' ')    /* skips blanks */
  ;
```

When the loop terminates, `ch` will contain the first nonblank character that `getchar` encountered.

Be careful if you mix `getchar` and `scanf` in the same program. `scanf` has a tendency to leave behind characters that it has "peeked" at but not read, including the new-line character. Consider what happens if we try to read a number first, then a character:

```
printf("Enter an integer: ");
scanf("%d", &i);
printf("Enter a command: ");
command = getchar();
```

The call of `scanf` will leave behind any characters that weren't consumed during the reading of `i`, including (but not limited to) the new-line character. `getchar` will fetch the first leftover character, which wasn't what we had in mind.

PROGRAM

Determining the Length of a Message

To illustrate how characters are read, let's write a program that calculates the length of a message. After the user enters the message, the program displays the length:

```
Enter a message: Brevity is the soul of wit.
Your message was 27 character(s) long.
```

The length includes spaces and punctuation, but not the new-line character at the end of the message.

We'll need a loop whose body reads a character and increments a counter. The loop will terminate as soon as a new-line character turns up. We could use either `scanf` or `getchar` to read characters; most C programmers would choose `getchar`. Using a straightforward `while` loop, we might end up with the following program.