body is executed again and then the expression is evaluated once more. Execution of the do statement terminates when the controlling expression has the value 0 *after* the loop body has been executed.

Let's rewrite the countdown example of Section 6.1, using a do statement this time:

```
i = 10;
do {
  printf("T minus %d and counting\n", i);
  --i;
} while (i > 0);
```

When the do statement is executed, the loop body is first executed, causing the message `T minus 10 and counting` to be printed and `i` to be decremented. The condition `i > 0` is now tested. Since 9 is greater than 0, the loop body is executed a second time. This process continues until the message `T minus 1 and counting` is printed and `i` becomes 0. The test `i > 0` now fails, causing the loop to terminate. As this example shows, the do statement is often indistinguishable from the `while` statement. The difference between the two is that the body of a do statement is always executed at least once; the body of a `while` statement is skipped entirely if the controlling expression is 0 initially.

Incidentally, it's a good idea to use braces in *all* do statements, whether or not they're needed, because a do statement without braces can easily be mistaken for a `while` statement:

```
do
  printf("T minus %d and counting\n", i--);
while (i > 0);
```

A careless reader might think that the word `while` was the beginning of a `while` statement.

PROGRAM Calculating the Number of Digits in an Integer

Although the `while` statement appears in C programs much more often than the do statement, the latter is handy for loops that must execute at least once. To illustrate this point, let's write a program that calculates the number of digits in an integer entered by the user:

```
Enter a nonnegative integer: 60
The number has 2 digit(s).
```

Our strategy will be to divide the user's input by 10 repeatedly until it becomes 0; the number of divisions performed is the number of digits. Clearly we'll need some kind of loop, since we don't know how many divisions it will take to reach 0. But should we use a `while` statement or a do statement? The do statement turns out to be more attractive, because every integer—even 0—has at least *one* digit. Here's the program: