The code relies upon the fact that an empty test condition in a for loop is treated as being true. Neither of these examples is easy to read, and neither should be used as a general model of writing a loop. The functionality of the first example is more clearly expressed in a do while loop:

```cpp
do {
    I++;  // do something;
    while (30 < I);
}
```

Similarly, the second example can be expressed more clearly as a while loop:

```cpp
while (I < 30)
{
   // do something
    I++;
}
```

In general, writing clear, easily understood code is a more useful goal than the demonstration of the ability to exploit obscure features of the language.

## Loops and Text Input

Now that you've seen how loops work, let's look at one of the most common and important tasks assigned to loops: reading text character-by-character from a file or from the keyboard. For example, you might want to write a program that counts the number of characters, lines, and words in the input. Traditionally, C++, like C, uses the while loop for this sort of task. We'll investigate now how that is done. If you already know C, don't skim through this part too fast. Although the C++ while loop is the same as C's, C++'s I/O facilities are different. This can give the C++ loop a somewhat different look. In fact, the cin object supports three distinct modes of single-character input, each with a different user