Use your operating system's scroll-back buffer to view the lines that have scrolled off the screen. Hopefully, the DOS prompt or terminal window that you're using has this feature.

What about typing 0 or a negative number? Try that now; run the program and type **–5**. You see something like this:

```
T-minus -5
Zero!
Blast off!
```

The `do while` loop is executed at least once, so the `-5` is displayed. Ack! It isn't one of the great boo-boos of modern programming history, but it's bound to startle the astronauts, who are expecting a leisurely though suspenseful takeoff sequence.

The way to guard against this faux pas is to write a special loop to ensure that the value that is typed is kosher. Yes, it's a kosher number loop. That is handled quite brilliantly by `do-while`.

The always kosher number-checking `do-while` loop

One thing you should do in all your programs is — and I have put this on a separate line for emphasis:

Check your input bounds!

This advice makes sense only if you know what *input bounds* are. Okay: They're the range of numbers or letters or whatever that your program is looking for. In COUNTDWN.C, they're numbers from 1 to 100. In some database programs, they're the "type 40 or fewer character" limits. Stuff like that.

You want to ensure that users cannot type a wrong, or "illegal," value — one that would screw up your program. You have to make sure that they type only 40 or fewer characters. Any more than that, and your program may die a strange death. You must guard against this situation — and you can, if you write your program correctly.

Traditionally, this type of defensive programming is known as *bulletproofing*. It protects the program from this type of error in advance. That's why you check everything the user types, to see whether it's kosher. If not, you can either ask politely for input again or just print a rude error message.