This program has two modules: `main`, which executes when the program runs, and `showRetail`, which calculates and displays an item's retail price. In the `main` module, a `Do-While` loop appears in lines 5 through 12. In line 7, the loop calls the `showRetail` module. Then, in line 10 the user is prompted "Do you have another item? (Enter y for yes.)" In line 11, the user's input is stored in the `doAnother` variable. In line 12, the following statement is the end of the `Do-While` loop:

```
While doAnother == "y" OR doAnother == "Y"
```

Notice that we are using the logical `OR` operator to test a compound Boolean expression. The expression on the left side of the `OR` operator will be true if `doAnother` is equal to lowercase `"y"`. The expression on the right side of the `OR` operator will be true if `doAnother` is equal to uppercase `"Y"`. If either of these subexpressions is true, the loop will iterate. This is a simple way to make a case insensitive comparison, which means that it does not matter whether the user enters uppercase or lowercase letters.

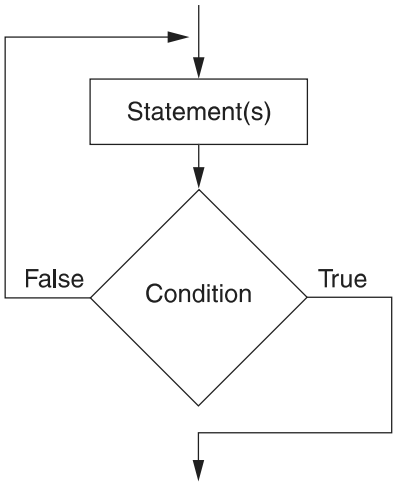
The `Do-Until` Loop

Both the `While` and the `Do-While` loops iterate as long as a condition is true. Sometimes, however, it is more convenient to write a loop that iterates *until* a condition is true—that is, a loop that iterates as long as a condition is false, and then stops when the condition becomes true.

For example, consider a machine in an automobile factory that paints cars as they move down the assembly line. When there are no more cars to paint, the machine stops. If you were programming such a machine, you would want to design a loop that causes the machine to paint cars until there are no more cars on the assembly line.

A loop that iterates until a condition is true is known as a `Do-Until` loop. Figure 5-10 shows the general logic of a `Do-Until` loop.

Figure 5-10 The logic of a `Do-Until` loop



Notice that the Do-Until loop is a posttest loop. First, one or more statements are executed, and then a condition is tested. If the condition is false, the program's execution flows back to the point just above the first statement in the body of the loop, and this process repeats. If the condition is true, the program exits the loop.

Writing a Do-Until Loop in Pseudocode

In pseudocode, we will use the Do-Until statement to write a Do-Until loop. Here is the general format of the Do-Until statement:

```
Do
   statement
   statement
   etc.
Until condition
```

These statements are the body of the loop. They are always performed once, and then repeated until the condition is true.

In the general format, the statements that appear in the lines between the Do and the Until clauses are the body of the loop. The *condition* that appears after the While clause is a Boolean expression. When the loop executes, the statements in the body of the loop are executed, and then the *condition* is tested. If the *condition* is true, the program exits the loop. If the *condition* is false, the loop starts over and the statements in the body are executed again.

As shown in the general format, you should use the following conventions when you write a Do–Until statement:

- Make sure the Do clause and the Until clause are aligned.
- Indent the statements in the body of the loop.

The pseudocode in Program 5-7 shows an example of the Do-Until loop. The loop in lines 6 through 16 repeatedly asks the user to enter a password until the string "prospero" is entered. Figure 5-11 shows a flowchart for the program.

Program 5-7

```
 1 // Declare a variable to hold the password.
 2 Declare String password
 3
 4 // Repeatedly ask the user to enter a password
 5 // until the correct one is entered.
 6 Do
 7    // Prompt the user to enter the password.
 8    Display "Enter the password."
 9    Input password
10
11    // Display an error message if the wrong
12    // password was entered.
13    If password != "prospero" Then
14       Display "Sorry, try again."
15    End If
16 Until password == "prospero"
17
18 // Indicate that the password is confirmed.
19 Display "Password confirmed."
```