Figure 5-5 The `main` module of Program 5-4

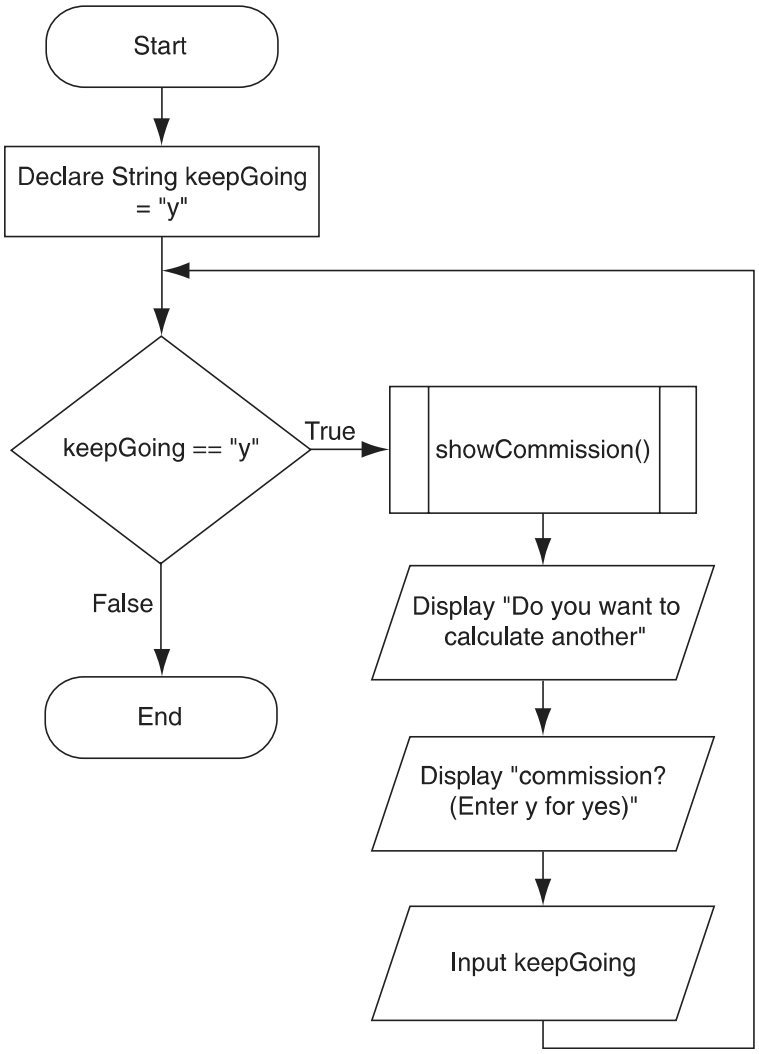
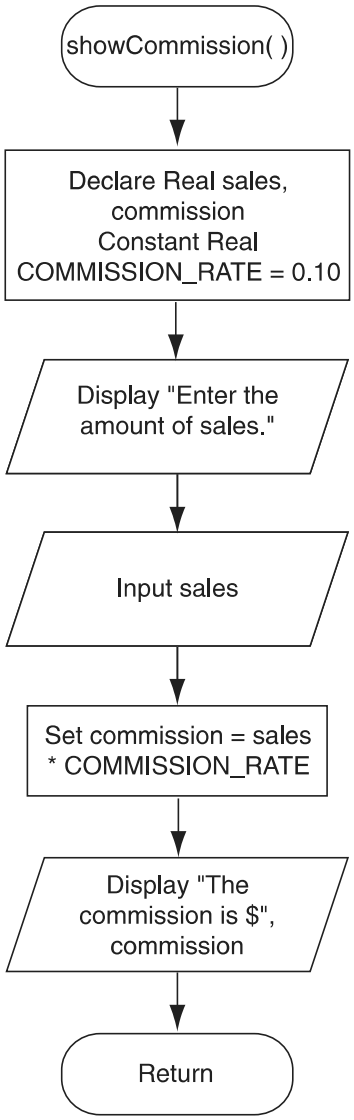


Figure 5-6 The `showCommission` module



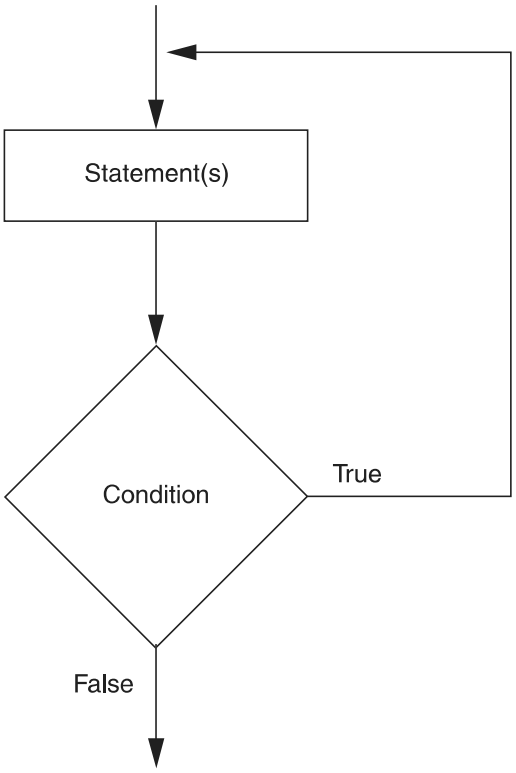
VideoNote

The `Do-While` Loop

The `Do-While` Loop

You have learned that the `While` loop is a pretest loop, which means it tests its condition before performing an iteration. The `Do-While` loop is a *posttest* loop. This means it performs an iteration before testing its condition. As a result, the `Do-While` loop always performs at least one iteration, even if its condition is false to begin with. The logic of a `Do-While` loop is shown in Figure 5-7.

Figure 5-7 The logic of a Do-While loop



In the flowchart, one or more statements are executed, and then a condition is tested. If the condition is true, 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 false, the program exits the loop.

Writing a Do-While Loop in Pseudocode

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

```
Do
   statement
   statement
   etc.
While condition
```

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

In the general format, the statements that appear in the lines between the Do and the While 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 loop starts over and the statements in the body are executed again. If the condition is false, however, the program exits the loop.

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

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