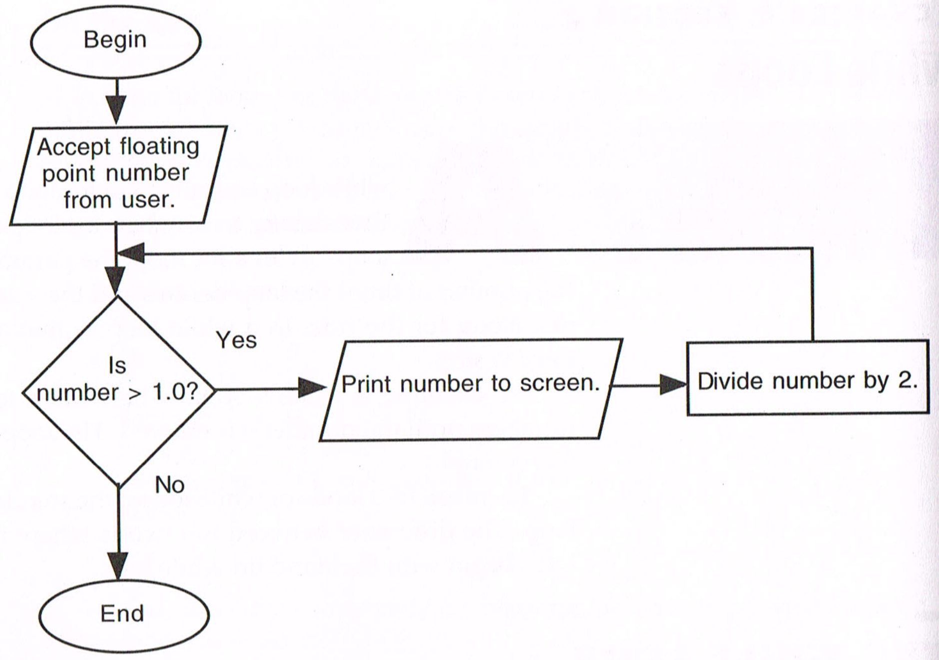

F I G U R E 8 - 6
A while loop tests the control expression before the loop begins.

EXERCISE 8-5

USING A while LOOP

1. Enter the program shown in Figure 8-5 into a blank editor screen.
2. Save the source file as *WHILE1.CPP*.
3. Compile and run the program. Run the program several times. Try the following numbers as input: 8, 21, 8650, 1, 2.1, 0.5.
4. Close the source file.

In order for a while loop to come to an end, the statements in the loop must change a variable used in the control expression. The result of the control expression must be false for a loop to stop. Otherwise, iterations continue indefinitely in what is called an ***infinite loop.*** In the program you compiled in Exercise 8-5, the statement `num = num / 2;` divides the number by two each time the loop repeats. Even if the user enters a large value, the loop will eventually end when the number becomes less than 1.

A while loop can be used to replace any for loop. So why have a for loop in the language? Because sometimes a for loop offers a better solution. Figure 8-7 shows two programs that produce the same output. The program using the for loop is better in this case because the counter variable is initialized, tested, and incremented in the same statement. In a while loop, a counter variable must be initialized and incremented in separate statements.

THE do while LOOP

The last iteration structure in C++ is the do while loop. A ***do while loop*** repeats a statement or group of statements as long as a control expression is true at the end of the loop. Because the control expression is tested at the end of the loop, a do while loop is executed at least one time. Figure 8-8 shows an example of a do while loop.

To help illustrate the difference between a while and a do while loop, compare the two flow charts in Figure 8-9. Use a while loop when you need to test the control expression before the loop is executed the first time. Use a do while loop when the statements in the loop need to be executed at least once.

```
#include <iostream.h>

main()
{
 int i;
 for(i = 1; i <= 3; i++)
   cout << i << '\n';
 return 0;
}
```

```
#include <iostream.h>

main()
{
 int i;
 i = 1;
 while(i <= 3)
  {
   cout << i << '\n';
   i++;
  }
 return 0;
}
```

F I G U R E 8 - 7
Although both of these programs produce the same output, the for loop gives a more efficient solution.

EXERCISE 8-6

USING A do while LOOP

1. Enter the program from Figure 8-8 into a blank editor screen.
2. Save the source file as *DOWHILE.CPP*.
3. Compile and run the program. Enter several numbers greater than 0 to cause the loop to repeat. Enter 0 to end the program.
4. Close the source file.

```
#include <iostream.h>

main()
{
 float num, squared;
 do
  {
   cout << "Enter a number (Enter 0 to quit): ";
   cin >> num;
   squared = num * num;
   cout << num << " squared is " << squared << '\n';
  }
 while (num != 0);
 return 0;
}
```

F I G U R E 8 - 8
In a do while loop, the control expression is tested at the end of the loop.