The *while* Loop

The other two functions in our library are more difficult. Both of them require you to learn about a new kind of loop bounds, called **limit bounds**. Loops that do some processing and then check the results against a boundary condition are **limit loops**.

To write a limit loop, use the `while` loop, which executes a statement repeatedly **while its condition remains true**. The general form of the while loop looks like this:

```
while (condition)
{
    statements;
}
```

A `while` loop first evaluates the condition. If `false`, the loop terminates and the program continues with the next statement after the loop body. If `true`, the actions in the body are run, after which control **returns to the loop condition**. One pass through the body constitutes a **cycle or iteration of the loop**.

1. The test is performed **before every cycle of the loop**, including the first. If the test is `false` initially, the body of the loop is **not executed at all**. That's why this is known as a **guarded** loop.
2. The test is performed **only** at the beginning of a loop cycle. If the condition **becomes** `false` at some point during the loop, the **program won't notice** that fact until it completes the entire cycle. When the program evaluates the test condition again, if it is still `false`, only then does the loop terminate.