A loop is another fundamental idea in programming. It provides you with a way to repeat one or more statements as many times as your application requires. You can employ a loop to handle any repetitive task, and for most programs of any consequence, loops are essential. Using a computer to calculate the company payroll, for example, would not be practicable without a loop.

C++ provides a number of ways to implement a loop, all of which have their own particular area of application. In this chapter, you’ll begin by looking at the theory behind loops and then get down to their practical uses, including, of course, how to write them. Along the way, you will learn the following:

- The principles behind the different kinds of loop
- How the while loop works
- What the merits of the do-while loop are
- How to use a for loop
- What the break statement does in a loop
- What the continue statement is used for in a loop
- How to construct nested loops

Understanding Loops

A loop is a mechanism that enables you to execute the same statement or block of statements repeatedly until a particular condition is met. The statements inside a loop are sometimes called iteration statements. A single execution of the statement or statement block that is within the loop is described as an iteration.

Two essential elements make up a loop: the statement or block of statements that forms the body of the loop that is to be executed repeatedly, and a loop condition of some kind that determines when to stop repeating the loop.
A loop condition can take a number of different forms to provide different ways of controlling the loop. For example, you might want to

- Execute the loop a given number of times.
- Execute the loop until a given value exceeds another value.
- Execute the loop until a particular character is entered from the keyboard.

You can set the loop condition to suit the circumstances. In the final analysis, however, loops come in two basic flavors; these are illustrated in Figure 5-1.

![Figure 5-1. The two basic loop flavors](image)

The difference between these two structures is evident at the point where you begin them. On the left, the loop condition is tested before the loop statements are executed; consequently, the loop statements are not executed if the test condition fails at the outset.

On the right, the test comes after the loop statements. The effect of this arrangement is that the loop statements are executed before the condition is tested for the first time, so this kind of loop always executes at least once.

The following are the three different kinds of loop in C++:

- The while loop
- The do-while loop
- The for loop