Chapter 5 Loops

In general, statements are executed sequentially: The first statement in a function is executed firstly, followed by the second, and so on. However, there may be a situation, when you need to execute a block of code many times.

顺序执行

A loop statement allows us to execute a statement or group of statements multiple times until a particular condition is satisfied.

循环语句

For example, let's say we want to show a message 10 times. Instead of writing the print statement 10 times, we can use a loop. That is just a simple example. We can achieve much more efficiency and sophistication in our programs by making effective use of loops.

精妙的作用

There are 3 types of loops in C/C++:

- for loop;
- while loop;
- do...while loop.

for 循环
while 循环
do…while 循环

5.1 for Loop

The syntax of for-loop is:

```
for (initialization; condition; update)
{
    // body of-loop
}
```

Here,

initialization—initializes variables and will be executed only once;

初始化；给循环变量赋初值

condition—if true, the body of for loop is executed;

if false, the for loop is terminated;

终止

update—updates the value of initialized variables.

更新循环变量的值

Figure 5-1 is the flowchart of a for-loop in C/C++.

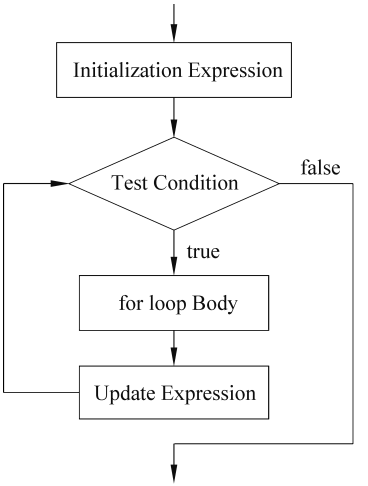

Figure 5-1 Flowchart of a for-loop in C/C++

Example: Printing Numbers From 1 to 5, the code is as below.

```
// C++ program: printing numbers from 1 to 5
#include<iostream>
using namespace std;
int main()
{
   int i=1;
   for(i=1; i<=5; i++)
      cout << i << " ";

   return 0;
}
```

The following program is written in C.

```
// C program: printing numbers from 1 to 5
#include<stdio.h>
int main()
{
   int i=1;
   for(i=1; i<=5; i++)
```

```
      printf("%d ", i);

   return 0;
}
```

The output is:

```
1 2 3 4 5
```

How does this program work?

Iteration	Value of i	i <= 5	Action
1st	i=1	true	1 printed, i is increased to 2
2nd	i=2	true	2 printed, i is increased to 3
3rd	i=3	true	3 printed, i is increased to 4
4th	i=4	true	4 printed, i is increased to 5
5th	i=5	true	5 printed, i is increased to 6
6th	i=6	false	Loop is terminated

Example: Find the sum of: 1+2+3+…+100.

```
int main()
{
   int i, sum=0;
   for(i=1; i<=100; i++)
   {
      sum = sum + i;
   }
   cout << "Sum = " << sum;

   return 0;
}
```

In the above example, we have two variables i and sum. The sum variable is initialized to 0, and used to hold the sum of the formula.

初始化为

Let us look at the for loop in more detail:

```
for (i = 1;  i <= 100;  i++)
```

Here,

i = 1: initializes the count variable;

i <= 100: runs the loop as long as i is less than or equal to 100;

i++: increase the i variable by 1 in each iteration.

When i becomes 101, the condition is false and sum will be equal to 1+2+3+⋯ +100.

5.2 while Loop

The syntax of the while loop is:

```
while(condition)
{
   // body of the loop
}
```

Here is how a while loop works:

If the condition evaluates to true, the code inside the while loop is executed.

Then the condition is evaluated again.

This process continues until the condition is false.

When the condition evaluates to false, the loop terminates.

Figure 5-2 is the flowchart of a while-Loop in C/C++.

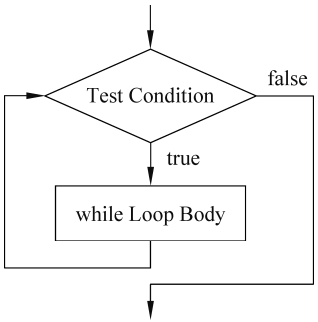

Figure 5-2 Flowchart of a while-Loop in C/C++

Example: Display numbers from 1 to 5.

```
// C++ Program to print numbers from 1 to 5
#include<iostream>
using namespace std;
int main()
{
   int i = 1;
   while (i <= 5)
   {
      cout << i << " ";
      ++i;
   }

   return 0;
}
```

Display numbers from 1 to 5 in C:

```
// C Program to print numbers from 1 to 5
#include<stdio.h>
int main()
{
   int i = 1;
   while (i <= 5)
   {
      printf("%d ",i);
      ++i;
   }

   return 0;
}
```

The output of the programs is as below.

Iteration	Value of i	i <= 5	Action
1st	i=1	true	1 printed, i is increased to 2
2nd	i=2	true	2 printed, i is increased to 3
3rd	i=3	true	3 printed, i is increased to 4
4th	i=4	true	4 printed, i is increased to 5
5th	i=5	true	5 printed, i is increased to 6
6th	i=6	false	Loop is terminated

5.3 do...while Loop

The do...while loop is a variant of the while loop with one important difference: the body of do...while loop is executed once before the condition is checked.

变形

Its syntax is:

```
do
{
  // body of loop;
}
while(condition);
```

Here, The body of the loop is executed at first. Then the condition is evaluated.

If the condition evaluates to true, the body of the loop inside the do statement is executed again.

The condition is evaluated once again.

If the condition evaluates to true, the body of the loop inside the do statement is executed again.

This process continues until the condition evaluates to false. Then the loop stops.

Figure 5-3 is the flowchart of a do…while loop.

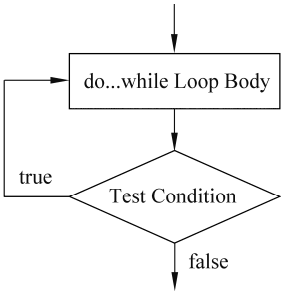

Figure 5-3 Flowchart of a do…while loop

Example: Display numbers from 1 to 5 using do…while loop.

```
#include<iostream>
using namespace std;
```

```
int main()
{
   int i = 1;
   do
   {
      cout << i << " ";
      ++i;
   }
   while (i <= 5);

   return 0;
}
```

The output is as below:

Iteration	Value of i	i <= 5	Action
	i=1	true	1 printed, i is increased to 2
1st	i=2	true	2 printed, i is increased to 3
2nd	i=3	true	3 printed, i is increased to 4
3rd	i=4	true	4 printed, i is increased to 5
4th	i=5	true	5 printed, i is increased to 6
5th	i=6	false	Loop is terminated

5.4 for Loops vs while Loops

A for loop is usually used when the number of iterations is known. For example,

```
// This loop is iterated 5 times
for (int i = 1; i <= 5; ++i)
{
  // body of the loop
}
```

Here, we know that the for-loop will be executed 5 times.

However, while and do...while loops are usually used when the number of iterations is unknown. For example,

```
while (condition)
{
   // body of the loop
}
```

5.5 Loop Control Statement “break” and “continue”

“break” and “continue” are two loop control statements in C and C++. They are only consisted of one keywork: break and continue.

循环控制语句

5.5.1 “Break” Statement

The break statement is a loop control statement which is used to terminate the loop.

终止循环

As soon as the break statement is encountered from within a loop, the loop iterations stop there and control jumps to the first statement after the loop immediately.

The syntax of the break statement is simple:

```
break;
```

Example : break with for loop

```
// program to print the value of i
#include<iostream>
using namespace std;
int main()
{
    for (int i = 1; i <= 5; i++)
    {
       // break condition
       if (i == 3)
       {
          break;
       }
       cout << i << endl;
   }
```

```
    return 0;
}
```

In the above program, the for loop is used to print the value of i in each iteration. Please notice the code:

```
if(i == 3)
{
    break;
}
```

This means, when i is equal to 3, the break statement executes and the loop is terminated. Hence, the output doesn't include values greater than or equal to 3.

The code below demonstrates the use of break statement in a while loop.

```
// Example: break with while loop
// program to find the sum of positive numbers
// if the user enters a negative number, break ends the loop
// the negative number entered is not added to sum

#include<iostream>
using namespace std;
int main()
{
    int number, sum = 0;
    while(true)
    {
        cout << "Enter a number: ";  //take input from the user
        cin >> number;
        if (number < 0)              //break condition
            break;
        sum += number;              //add all positive numbers
    }
    cout << "The sum is " << sum << endl; //display the sum
    return 0;
}
```

The code below shows the use of break in a nested for loop.

```
// C++ program to illustrate using break statement in nested loops
```

```
#include<iostream>
using namespace std;
int main()
{
    // nested for loops with break statement at inner loop
    for (int i = 0; i < 5; i++)    //outter loop
    {
       for (int j = 0; j < 5; j++)  //inner loop
       {
          if (j > 3)
             break;
          else
             cout << "*";
       }
       cout << endl;
   }
   return 0;
}
```

外层循环

内层循环

In the code above, we can clearly see that the inner loop is programmed to execute for 5 iterations. But when the value of j becomes greater than 3, the inner loop stops executing which restricts the number of iteration of the inner loop to 4 iterations only (j from 0 to 3). However, the iteration of outer loop is unaffected.

限制

未受影响

When the program runs, we can see on the screen:

```
****
****
****
****
****
```

5.5.2 “continue” Statement

continue 语句

Continue is also a loop control statement which is opposite to that of break statement, it terminates the current iteration, but executes the next iteration of the loop.