Microsoft

ASP.NET

Module 6

Repetition/Loop Statement

I. Preparations

At the end of this module students will:

- construct program to execute multiple times.
- construct solution to complex problems using looping/repetition structure appropriately;

II. Presentation

Another crucial programming construct is the looping statement or loop. Using a loop, you can repeat a certain section of your source based on the boundaries. Similar to a conditional statement, the loop starts running as soon as the condition is satisfied and ends when it is not. You will get knowledge and comprehension of how to use the loop statement from this chapter.

There are four (4) structure of loop

- do-while loop
- while loop
- for loop

The *do-while( )* statement

A do-while loop is a type of loop in programming languages which executes a block of code at least once, and then repeatedly executes the block, or not, depending on a given boolean condition at the end of the block. The condition is evaluated after executing the code block, so the code block will always be executed at least once.

Syntax:

```
initialization;
do {
        code to be executed;
        update;
} while (condition);
```

The do-while loop starts with the do keyword followed by a code block and a boolean expression with the while keyword. The do while loop stops execution exits when a boolean condition evaluates to false. Because the while(condition) specified at the end of the block, it certainly executes the code block at least once.

Example:

```
int i = 0;
          do
          {
             Response.Write("i = " + i + "<br/>");
             i++;
          } while (i <= 10);
Output:

          i = 1
          i = 2
          i = 3
          i = 4
          i = 5
          i = 6
          i = 7
          i = 8
          i = 9
          i = 10
```

Specify initialization out of the loop and increment/decrement counter inside do while loop. Use break or return to exit from the do while loop.

```
int i = 1;
          do
          {
             Response.Write("i = " + i + "<br/>");
             i++;
             if (i > 5)
                break;
          } while (i <= 10)

Output:

          i = 1
          i = 2
          i = 3
          i = 4
          i = 5
```

The *while loop* Statement

A **while** loop is a control flow statement that allows code to be executed repeatedly based on a given condition. The while loop can be thought of as a repeating if statement. The while loop will keep iterating until the defined condition evaluates to false.

The **while** keyword is used to create while loop in C#. The syntax for while loop is:

Syntax:

```
initialization;
while (condition) {
        code to be executed; true block
        update;
}
```

How while loop works?

1. C# while loop consists of a test-expression.
2. If the test-expression is evaluated to true,
   - statements inside the while loop are executed.
   - after execution, the test-expression is evaluated again.
3. If the test-expression is evaluated to false, the while loop terminates.

```
int i = 1;
        while (i <= 10)
        {
           Response.Write("i = " + i + "<br/>");
           i++;
        }
Output:

        i = 1
        i = 2
        i = 3
        i = 4
        i = 5
        i = 6
        i = 7
        i = 8
        i = 9
        i = 10
```

Initially the value of i is 1.

When the program reaches the while loop statement,

- the test expression i <=10 is evaluated. Since i is 1 and 1 <= 10 is true, it executes the body of the while loop. Here, the line is printed on the screen with Iteration 1, and the value of i is increased by 1 to become 2.
- Now, the test expression (i <=10) is evaluated again. This time too, the expression returns true (2 <= 10), so the line is printed on the screen and the value of i is now incremented to 3..
- This goes and the while loop executes until i becomes 11. At this point, the test-expression will evaluate to false and hence the loop terminates.

The *for loop* Statement

A for loop is a type of loop in programming which allows you to repeat a certain set of instructions a fixed number of times. It is most commonly used to iterate over a sequence, such as a list or array, and execute the same set of instructions for every item in the sequence.

Syntax:

```
for (initialization;condition;update){
        code to be executed; true block
}
```

How for loop works?

1. C# for loop has three statements: initialization, condition and iterator.
2. The initialization statement is executed at first and only once. Here, the variable is usually declared and initialized.
3. Then, the condition is evaluated. The condition is a boolean expression, i.e. it returns either true or false.
4. If the condition is evaluated to true:
    - The statements inside the for loop are executed.

- Then, the iterator statement is executed which usually changes the value of the initialized variable.
- Again the condition is evaluated.
- The process continues until the condition is evaluated to false.

5. If the condition is evaluated to false, the for loop terminates.

III. Practice

1. Input two positive integers. Compute and display the product of the two numbers without using multiplication operator (*).
2. Input integer N. Compute and display the sum of squares from 1 to N. If the value of N is 4, then the result is 30, because 1^2 + 2^2 + 3^2 + 4^2 = 1 + 4 + 9 + 16 = 30.

IV. Performance

1. Input two positive integers. Compute and display the quotient and the remainder without using any division (/) operator or modulo operator.
2. Input positive integer. Determine if the number is:
   a. Prime number or not
   b. Perfect number or not
   c. Strong number
   d. Factorial
   e. Palindrome or not