Principle of Programming

5. Flow control (for loop)

Lecturers:

Sipan M. Hameed

www.sipan.dev

2024 - 2025

1. Table of Contents

5. C# for loop

In programming, it is often desired to execute certain block of statements for a specified number of times. A possible solution will be to type those statements for the required number of times. However, the number of repetitions may not be known in advance (during compile time) or maybe large enough (say 10000).

The best solution to such problem is loop. Loops are used in programming to repeatedly execute a certain block of statements until some condition is met.

In this article, we'll look at for loop in C#.

5.1 C# for loop

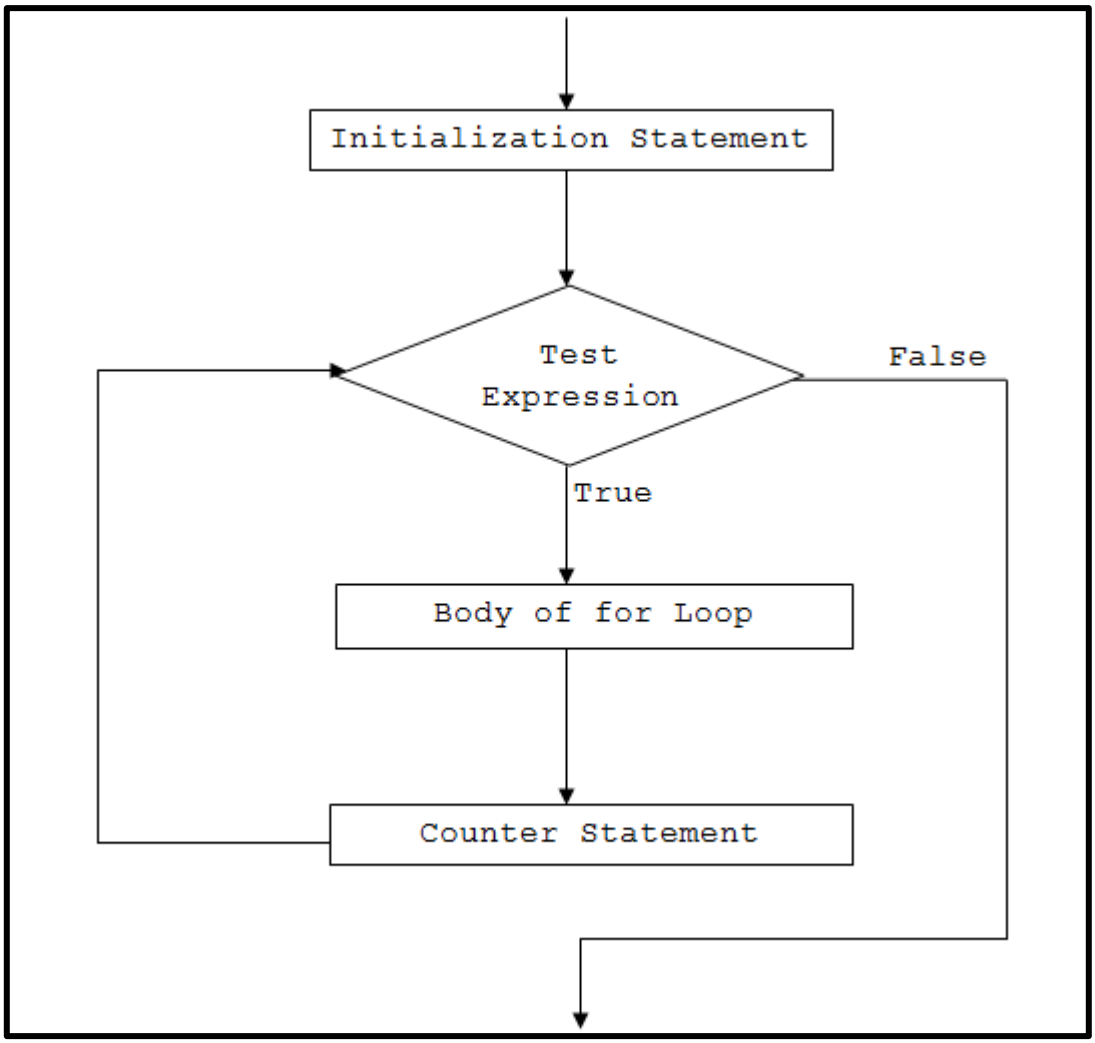
The **for** keyword is used to create for loop in C#. The syntax for **for loop** is:

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

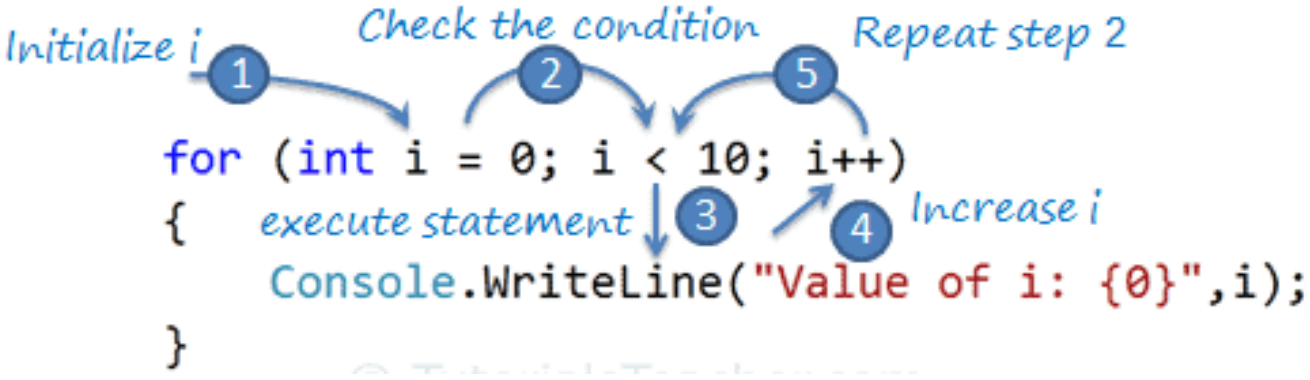
5.2 How for loop works?

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`:
    1. The statements inside the for loop are executed.
    2. Then, the `iterator` statement is executed which usually changes the value of the initialized variable.
    3. Again, the `condition` is evaluated.
    4. The process continues until the `condition` is evaluated to `false`.
5. If the `condition` is evaluated to `false`, the for loop terminates.

5.3 or Loop Flowchart



The below figure illustrates the execution steps of the `for` loop.



for Loop Execution Steps

5.4 Example 1: C# for Loop

```
using System;

namespace Loop
{
      class ForLoop
      {
            public static void Main(string[] args)
            {
                  for (int i=1; i<=5; i++)
                  {
                        Console.WriteLine("C# For Loop: Iteration {0}", i);
                  }
            }
      }
}
```

When we run the program, the output will be:

```
C# For Loop: Iteration 1
C# For Loop: Iteration 2
C# For Loop: Iteration 3
C# For Loop: Iteration 4
C# For Loop: Iteration 5
```

In this program,

- initialization statement is int i=1
- condition statement is i<=5
- iterator statement is i++

When the program runs,

- First, the variable i is declared and initialized to 1.
- Then, the condition (i<=5) is evaluated.
- Since, the condition returns true, the program then executes the body of the for loop. It prints the given line with Iteration 1 (Iteration simply means repetition).
- Now, the iterator (i++) is evaluated. This increments the value of i to 2.
- The condition (i<=5) is evaluated again and at the end, the value of i is incremented by 1. The condition will evaluate to true for the first 5 times.
- When the value of i will be 6 and the condition will be false, hence the loop will terminate.

5.5 Example 2: for loop to compute sum of first n natural numbers

```
using System;
namespace Loop
{
      class ForLoop
      {
            public static void Main(string[] args)
            {
                  int n = 5,sum = 0;

                  for (int i=1; i<=n; i++)
                  {
                        // sum = sum + i;
                        sum += i;
                  }

                  Console.WriteLine("Sum of first {0} natural numbers = {1}",
n, sum);
            }
      }
}
```

When we run the program, the output will be:

Sum of first 5 natural numbers = 15

Here, the value of sum and n are initialized to 0 and 5 respectively. The iteration variable i is initialized to 1 and incremented on each iteration.

Inside the for loop, value of sum is incremented by i i.e. sum = sum + i. The for loop continues until i is less than or equal to n (user's input).Let's see what happens in the given program on each iteration.

Initially, i = 1, sum = 0 and n = 3

For loop execution steps

Iteration	Value of i	i<=5	Value of sum
1	1	true	0+1 = 1
2	2	true	1+2 = 3
3	3	true	3+3 = 6
4	4	true	6+4 = 10
5	5	true	10+5 = 15
6	6	false	Loop terminates

So, the final value of sum will be 15 when n = 5.

5.6 Multiple expressions inside a for loop

We can also use multiple expressions inside a for loop. It means we can have more than one initialization and/or iterator statements within a for loop. Let's see the example below.

Example 3: for loop with multiple initialization and iterator expressions

```
using System;

namespace Loop
{
      class ForLoop
      {
            public static void Main(string[] args)
            {
                  for (int i=0, j=0; i+j<=5; i++, j++)
                  {
                        Console.WriteLine("i = {0} and j = {1}", i,j);
                  }
            }
      }
}
```

When we run the program, the output will be:

```
i = 0 and j = 0
i = 1 and j = 1
i = 2 and j = 2
```

In this program, we have declared and initialized two variables: i and j in the initialization statement. Also, we have two expressions in the iterator part. That means both *i* and *j* are incremented by 1 on each iteration.

5.7 Infinite for loop

If the condition in a for loop is always true, for loop will run forever. This is called infinite for loop.

Example 5: Infinite for loop

```
using System;

namespace Loop
{
      class ForLoop
      {
            public static void Main(string[] args)
            {
                  for (int i=1 ; i>0; i++)
                  {
                        Console.WriteLine("C# For Loop: Iteration {0}", i);
                  }
            }
      }
}
```

Here, i is initialized to 1 and the condition is i>0. On each iteration we are incrementing the value of i by 1, so the condition will never be false. This will cause the loop to execute infinitely.

We can also create an infinite loop by replacing the condition with a blank. For example,

```
for ( ; ; )
{
      // body of for loop
}
```

or

```
for (initialization ; ; iterator)
{
      // body of for loop
}
```

5.8 Practical Examples

5.8.1 Example: counter control: increment and decrement

```
using System;
public class ex51
{
    static void Main()
    {
        int x = 0;
        x++;
        Console.WriteLine("x++  = {0}",x);

        x = 0;
        x--;
        Console.WriteLine("x--  = {0}", x);

        x = 0;
        x = x + 5;
        Console.WriteLine("x = x + 5  = {0}", x);

        x = 0;
        x += 5;
        Console.WriteLine("x += 5  = {0}", x);

        x = 0;
        x -= 5;
        Console.WriteLine("x -= 5  = {0}", x);

    }
}
```

Output:

```
x++  = 1
x--  = -1
x = x + 5  = 5
x += 5  = 5
x -= 5  = -5
```

5.8.2 Example: math pattern-1

Print the following number pattern:

```
1
2
3
4
5
…
…
…
99
100
```

Code:

```
using System;
public class ex52
{
    static void Main()
    {
        for (int i = 1; i <= 100; i++)
        {
            Console.WriteLine(i);
        }

    }
}
```

5.8.3 Example: math pattern-2

Print the following number pattern:

```
1 ,3, 5, 7, 9, 11 , 13 , … … … , 97, 97
```

Code:

```
using System;
public class ex53
{
    static void Main()
    {
        for (int i = 1; i <= 99; i+=2)
        {
            Console.Write("{0} , ",i);
        }

    }
}
```

5.8.4 Example: math pattern-3

Print the following number pattern:

```
0 , 5 , 10 , 15 , 20 , …  …  …, 100
```

Code:

```
using System;
public class ex54
{
    static void Main()
    {
        for (int i = 0; i <= 100; i+=5)
        {
            Console.Write("{0} , ", i);
        }

    }
}
```

5.8.5 Example: math pattern-4

sum the following number pattern:

```
1 + 2 + 3 + 4 + 5 + 6 + … … … 99 + 100
```

Code:

```
using System;
public class ex55
{
    static void Main()
    {
        int s = 0;
        for (int i = 0; i <= 100; i++)
        {
            s = s + i;
        }

        Console.Write("sum = {0} , ", s);
    }
}
```

Output:

```
sum = 5050
```

5.8.6 Example: math pattern-5

sum the following number pattern:

```
0 + 2 + 4 + 6 + 8 + … … … + 98 + 100
```

Code:

```
using System;
public class ex56
{
    static void Main()
    {
        int s = 0;
        for (int i = 0; i <= 100; i+=2)
        {
            s = s + i;
        }

        Console.WriteLine("sum = {0}  ", s);
    }
}
```

Output:

```
sum = 2550
```

5.8.7 Example: math pattern-6

multiply the following number pattern:

```
1 * 2 * 3 * 4 … … … * 10
```

Code:

```
using System;
public class ex57
{
    static void Main()
    {
        int s = 1;
        for (int i = 1; i <= 10; i++)
        {
            s = s * i;
        }

        Console.WriteLine("sum = {0}  ", s);
    }
}
```

Output:

```
multi = 3628800
```

5.8.8 Example: math equation

write a program for the following math equation:

```
z = x + 2x + 3x + 4x + … … … + 10x
```

code:

```
using System;
public class ex58
{
    static void Main()
    {
        double z = 0;
        double x = Convert.ToDouble(Console.ReadLine());
        for (int i = 1; i <= 10; i++)
        {
            z = z + (i * x) ;
        }

        Console.WriteLine("z = {0}  ", z);
    }
}
```

Output:

```
2.3
z = 126.49999999999999
```