Loops

Loops in C# essentially allow you to execute a block of code repeatedly until a certain condition is met. C# provides the following four constructs:

1. **for loop**
2. **foreach loop**
3. **while loop**
4. **do/while loop**

1. The for loop construct

The for loop is for iterating a block of code a fixed number of times where you can test a specific condition before you do another iteration. The for loop construct syntax is as in the following:

Syntax

```
for (initializer; condition; iterator)
{
   Statements;
}
```

Here the initializer is evaluated first before the first loop is executed. The loop is executed until the condition expressions are false and the iterator is responsible for incrementing or decrementing the loop counter.

Example-

The following program typically writes the value 0 to 4 on the screen by using the for loop construct. We are declaring a local variable x and initializing it to 0 to use it as loop counter. Then we test whether or not it is less than 5 in the condition block and finally you increase the counter by one and walk through the process again as in the following:

```
int x;
for (x = 0; x < 5; x++)
 {
   Console.WriteLine(x);
   Console.WriteLine("\n");
 }
```

Output

0
1
2
3
4

Note-- we can declare the local counter variable in the for loop block also and x++ can be simply written as x=x+1. You can create a complex condition, endless loops and use goto, break and continue statements in the for loop.

Example-

The following program will terminate when the counter variable 8 is reached as in the following:

```
for (int i = 0; i < 10; i++)
   {
     if (i == 8)
      {
        break;
      }
      Console.WriteLine(i);
   }
```

Output

0
1
2
3
4
5
6
7

Example-

In some circumstance you want your loops to be executed indefinitely. C# allows you to write infinite loops using the following syntax:

```
for (; ;) {  }

for (; ;)
 {
 Console.WriteLine("printing");
}
```

Output

printing
printing
printing
printing
printing
.
.
.
.
printing

Example-

The for loop can be nested. That means we can implement an inner for loop in an outer loop. The inner loop executes once completely for each iteration of an outer loop. The following code displays prime numbers until 100 using a nested for loop.

```
static void Main(string[] args)
      {
         int i, j;
         bool isPrime = false;

         for (i = 2; i <= 100; i++)
         {
            for (j = 2; j < i; j++)
```

```
        {
            if (i % j == 0)
            {
                isPrime = true;
                break;
            }
        }
        if (isPrime == false)
            Console.Write("{0} ", j);
        else
            isPrime = false;
    }

    Console.ReadK     ey();
}
```

Output

All prime number upto 100

2. The foreach loop construct

The foreach loop construct allows you to iterate through each item into a collection without the necessary of testing the upper limit condition. This construct can also be applied on user-defined collections. **In the following you can traverse an array of string:**

Example-

```
string[] country = { "India", "USA", "Canada", "China" };
foreach (string x in country)
 {
  Console.WriteLine(x);
  Console.WriteLine("\n");
 }
```

Output-

India

USA

Canada
China

Here, the foreach loop steps through the array element one at a time. With each element, it places the value of the element in the string x variable and then performs an iteration of the loop.

3. The while loop construct

The while loop is mostly used to repeat a statement or a block of statements for a number of times that is not known before the loop begins. The syntax of the while loop is as in the following:

Syntax

```
while(condition)
   Statement;
```

The while loop does the expression evaluation first, then executes its body statements. A statement inside the loop body will set a Boolean flag to false on a certain iteration until the end of the loop as in the following:

Example-

```
bool status= false;
while (!status)
   {
         Console.WriteLine("status is false");
   }
```

Output

Example-

The following program checks whether the value of int x is less than 10 using a while loop and displays them:

```
int x = 0;
while (x<10)
  {
    Console.WriteLine("value of x is"+ x);
    x++;
  }
```

Output

value of x is 0
value of x is 1
value of x is 2
value of x is 3
value of x is 4
value of x is 5
value of x is 6
value of x is 7
value of x is 8
value of x is 9

4. The do/while loop constructs

The do/while is nearly similar to its previous construct while loop but has the slight difference that the condition is evaluated after the body of the loop has been executed. This loop is useful for the situation in which the statements must execute at least once. The syntax for the do/while loop is as in the following:

Syntax

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

Example-

The following program depicts the same objective as in a while loop but it executes the body statements first and later it checks the condition as in the following:

```
int x = 0;
do
  {
   Console.WriteLine("value of x is" + x);
   x++;
  } while (x < 10);
```

Output

```
value of x is 0
value of x is 1
value of x is 2
value of x is 3
value of x is 4
value of x is 5
value of x is 6
value of x is 7
value of x is 8
value of x is 9
```