

Loops are used to repeat a block of code. Being able to have your program repeatedly execute a block of code is one of the most basic but useful tasks in programming. Or Loops are very useful when the # of lines needs to execute more than one time.

There are **3** types of loops in C# Language

- 1) **for** loop
- 2) **while** loop
- 3) **do while** loop
- 4) **foreach** loop

1) For Loop

for loop is used to execute a set of statements repeatedly until a particular condition is satisfied. we can say it an **open ended loop**. It is usually preferable when the number of iterations are known.

Syntax:

```
for (initialization; TestExpression; update value)
{
    Statement(s); Body of for loop
}
```

Explanation

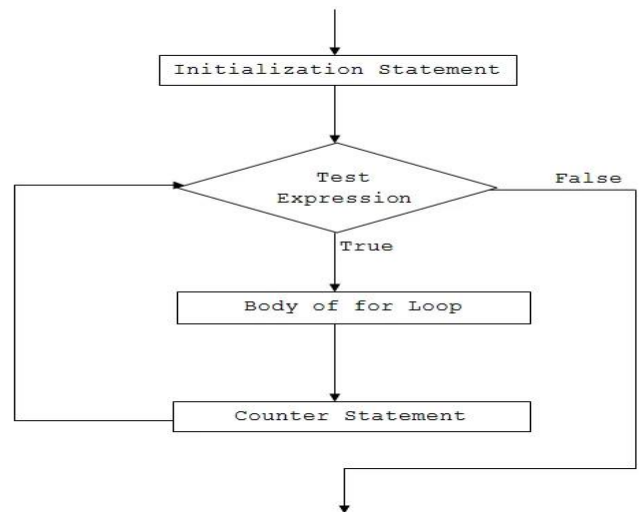
The initial expression is initialized only once at the beginning of the '**for**' loop. Then, the condition is checked by the program. If the condition is false, **for** loop is terminated. But, if condition is true then, the codes are executed and update expression is updated. Again, the condition is checked. If it is false, loop is terminated and if it is true, the same process repeats until condition is false.

Example

```
using System;

namespace loopclass1
{
    class Program
    {
        static void Main(string[] args)
        {
            int a;
            Console.WriteLine(" For loop example ");
            for (a = 1; a <= 5; a ++)  
                // if a is less than or equals to 50
            {
                Console.WriteLine(" a= " +a); // then it will print value of a, then update value by 5
            }
        }
    }
}
```

Flow Chart



```

C:\Windows\system32\cmd.exe
For loop example
a= 1
a= 2
a= 3
a= 4
a= 5
Press any key to continue . . .
    
```

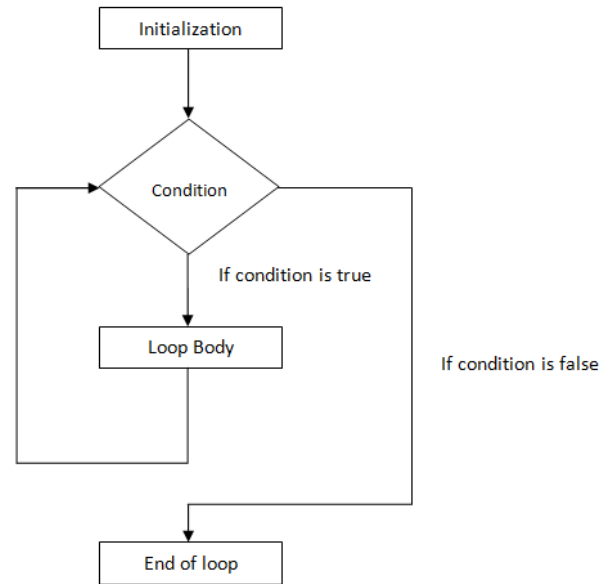
2) while loop

While loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The condition is evaluated, and if the condition is true, the code within the block is executed first. This repeats until the condition becomes false. Because **while** loop checks the condition before the block is executed, it is also known as a **pre-test loop**.

Syntax:

```

Initialization;
while(condition)
{
    Body of While loop
    Statement(s);
    Update_Value;
}
    
```



Explanation:

first checks whether A is TRUE, which it is, so then the {loop body} is entered, where it executes statement(s) that are in the body of loop, after execution of every statement once it checks the condition again if the condition remains TRUE it will execute the body again . Process repeating until the A remains true.

Example:

```

using System;

namespace loopclass2
{
    class Program
    {
        static void Main(string[] args)
        {
            int a=1;
            Console.WriteLine("While Loop Example.. ");
            while( a<=5) // if a is less than and equals to 5
            {
                Console.WriteLine(" a = "+ a ); // then it will print value of a, then update value by 1
                a ++;
            }
        }
    }
}
    
```

```

C:\Windows\system32\cmd.exe
While Loop Example..
a = 1
a = 2
a = 3
a = 4
a = 5
Press any key to continue . . .

```

do...while loop

In C#, do...while loop is very similar to while loop. Only difference between these two loops is that, in while loops, condition is checked at first but, in do...while loop code is executed at first then the condition is checked. So, the code is executed at least once in do...while loops that's why it is also called **Post Test** loop.

Syntax: Initialization;

```

do
{
    Statement(s);
    Update value;
} while (condition);

```

Notice, there is semicolon in the end of while (); in do...while loop.

Explanation:

At first codes inside body of do is executed. Then, the test expression is checked. If it is true, code/s inside body of do are executed again and the process continues until test expression becomes false (zero).

Example:

```

using System;

namespace loopclass3
{
    class Program
    {
        static void Main(string[] args)
        {
            int a;
            Console.WriteLine("do while loop result\n ");
            a = 1;
            do
            {
                Console.WriteLine(" a = "+a); // then it will print value of a, then update value by 1
                a ++; // increment or update by 5 in variable a
            } while (a <= 5); // if a is less than and equals to 5
        }
    }
}

```

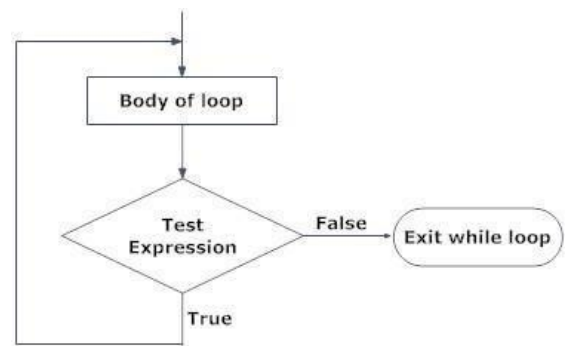
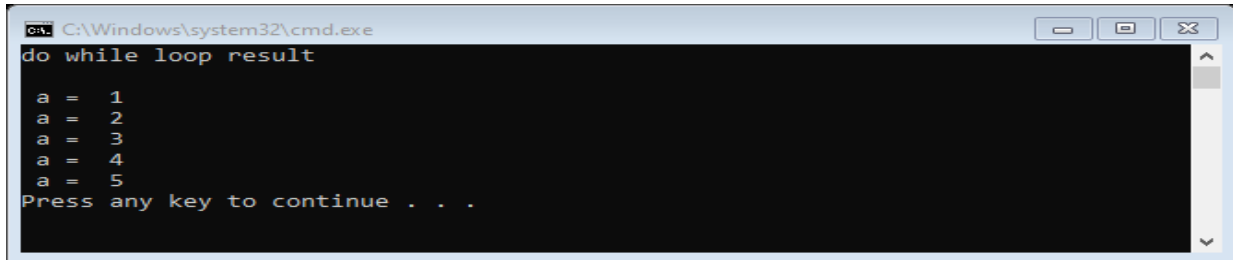


Figure: Flowchart of do...while loop

```

    }
}
}

```



```

C:\Windows\system32\cmd.exe
do while loop result
a = 1
a = 2
a = 3
a = 4
a = 5
Press any key to continue . . .

```

Nested Loop

Like nested if statement, if loop uses within a loop then this terminology is called “Nested loop”

Syntax:

```

for (initialization; condition; increment/decrement) // Outer Loop
{
    // Outer loop start here
    for (initialization; condition; increment/decrement) // Inner Loop
    {
        // Inner loop start here
        Statement's; // execute this statement
    }
}

```

Example:

```

using System;

namespace LoopClass4
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b;
            Console.WriteLine("Nested Loop ");
            for (a = 1; a <= 5; a++)
            {
                for (b = 1; b <= a; b++)
                {
                    Console.Write(a);
                }
                Console.WriteLine("");
            }
        }
    }
}

```

A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window displays the output of a C# program. The output consists of the text "Nested Loop" followed by five lines of numbers: "1", "22", "333", "4444", and "55555". The final line of output is "Press any key to continue . . .". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

break keyword

The **break** is a keyword in C# which is used to the program control exit from the loop. The break statement is used inside loops or switch statement. The break statement breaks the loop one by one, in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops.

continue keyword.

Sometimes a situation arises where we want to take the control to the beginning of the loop (**for** example **for**, **while**, **do while** etc.) skipping the rest statements inside the loop which have not yet been executed. The keywords **continue** allow us to **do** this. When the keywords **continue** executed inside a loop the control automatically passes to the beginning of loop. Continue is usually associated with the if.

In the following example, the lists of odd numbers between 1 to 10 have printed. In the **while** loop we test the remainder (here $x \% 2$) of every number, if the remainder is 0 then it becomes an even number and to avoid printing of even numbers continue statement is immediately used and the control passes to the beginning of the loop.

Example

```
using System;
```

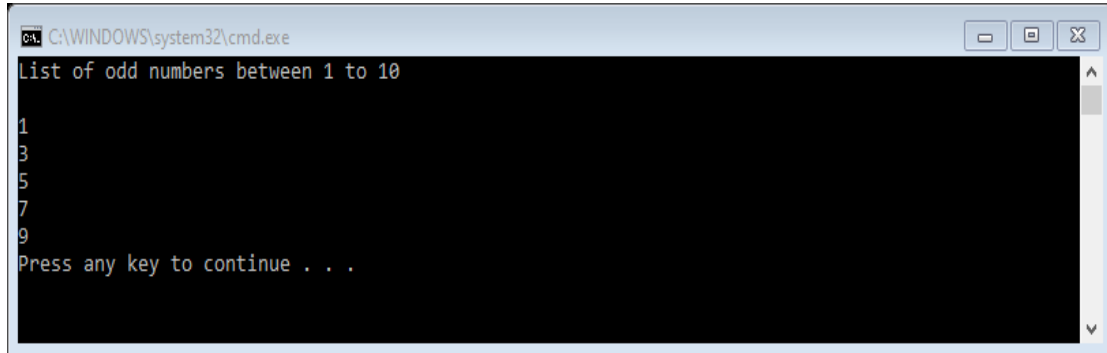
```
namespace LoopClass5
{
    class Program
    {
        static void Main(string[] args)
        {
            int x=1;

            Console.WriteLine("List of odd numbers between 1 to 10 \n");
            while (x<=10)
            {
                if ((x % 2) ==0)
                {
                    x++;
                    continue;
                }
                else
                {
                    Console.WriteLine(x);
                    x++;
                }
            }
        }
    }
}
```

```

    }
}
}
}
}
}

```



```

C:\WINDOWS\system32\cmd.exe
List of odd numbers between 1 to 10
1
3
5
7
9
Press any key to continue . . .

```

Example-2

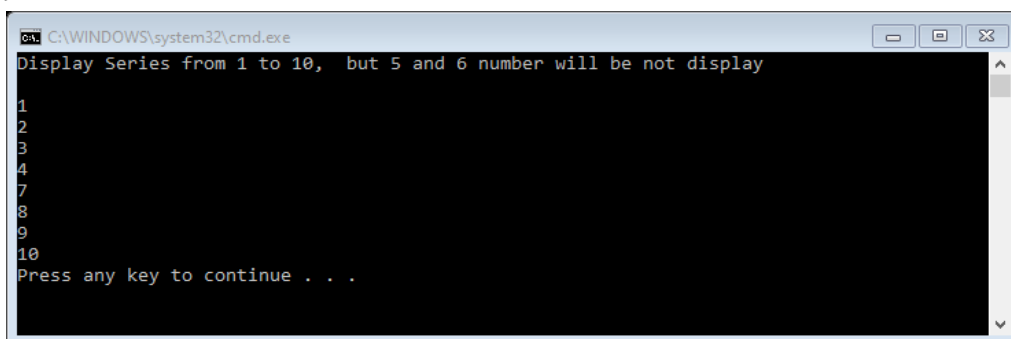
```

using System;

namespace LoopClass6
{
    class Program
    {
        static void Main(string[] args)
        {
            int x=1;

            Console.WriteLine("Display Series from 1 to 10, but 5 and 6 number will be not display \n");
            while (x<=10)
            {
                if (x == 5 || x == 6)
                {
                    x++;
                    continue;
                }
                Console.WriteLine(x);
                x++;
            }
        }
    }
}

```



```

C:\WINDOWS\system32\cmd.exe
Display Series from 1 to 10, but 5 and 6 number will be not display
1
2
3
4
7
8
9
10
Press any key to continue . . .

```

4. foreach loop

The foreach loop in C# iterates items in a collection, like an array or a list. It proves useful for traversing through each element in the collection and displaying them. The foreach loop is an easier and more readable alternative to for loop. This loop we discuss in chapter Array.

Exercise

Theory Questions

1. Write down the syntax of Nested **while** and Nested **do while** loops.
2. Describe the **continue** keywords according to loop.
3. Which type of post-test loop and pre-test loop?
4. What is loop and how many types of loop in C# language?
5. Where we use to foreach loop?

Practical Questions

1. Write a program to print table of a number entered by user using **while** loop and **do while** loop.
2. Write a program to print following output using nested **for** loop

```
1
12
123
1234
12345
```

3. Write down the code to display series from 0 to 5 and their sum numbers, using loop of your choice.

Count	Total
0	0
1	1
2	3
3	6
4	10
5	15

4. Write down the code to display following output

```
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25
```

5. Write a program series of prime number from 1 to 25.

Objective MCQ's:

1. Each repetition of a looping statement is called a(n) _____
 - a) Recurrence
 - b) Iteration
 - c) Duplication
 - d) Re-execution

2. Which of the following is the correct syntax for a for loop statement.
- a) `for(i=0; i <=10; ++)`
`Console.WriteLine("display from a for statement ");`
 - b) `for(i=0, i <=10, ++)`
`Console.WriteLine("display from a for statement ");`
 - c) `for(i=0; i <=10);`
`Console.WriteLine("display from a for statement ");`
 - d) `for`
{
`Console.WriteLine("display from a for statement ");`
`}while (i=0; i<10; i++);`
3. When the keywords inside a loop the control automatically passes/move to the beginning of loop.
- a) break
 - b) Exit
 - c) continue
 - d) not in all
4. Which of the following is the correct syntax for while statement?
- a. `while (i <10 , i++)`
{ `Console.WriteLine(" i ");`
}
 - b. `while (i <10)`
{ `Console.WriteLine (" i ");`
`i++;`
}
 - c. `wile (i <10 , i++)`
`Console.WriteLine(" i ");`
`i++;`
 - d. `while (i <10 , i++);`
{ `Console.WriteLine(" i ");`
}
5. These keywords are used in the loops, if depend on requirement.
- a) continue
 - b) break;
 - c) a and b, both
 - d) case