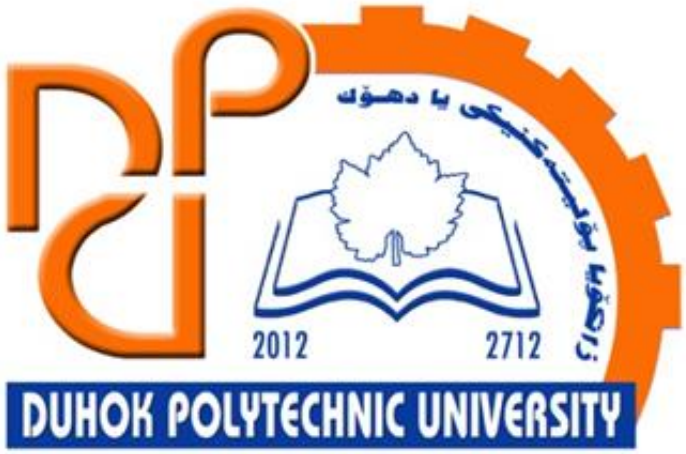


Zakho Technical Institute

Department of Information Technology

Principle of Programming

6. Flow control (while and do...while loop)

Lecturer:

Sipan M. Hameed

1. Table of Contents

6. while and do...while 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 learn to use while loops in C#.

6.1 C# while loop

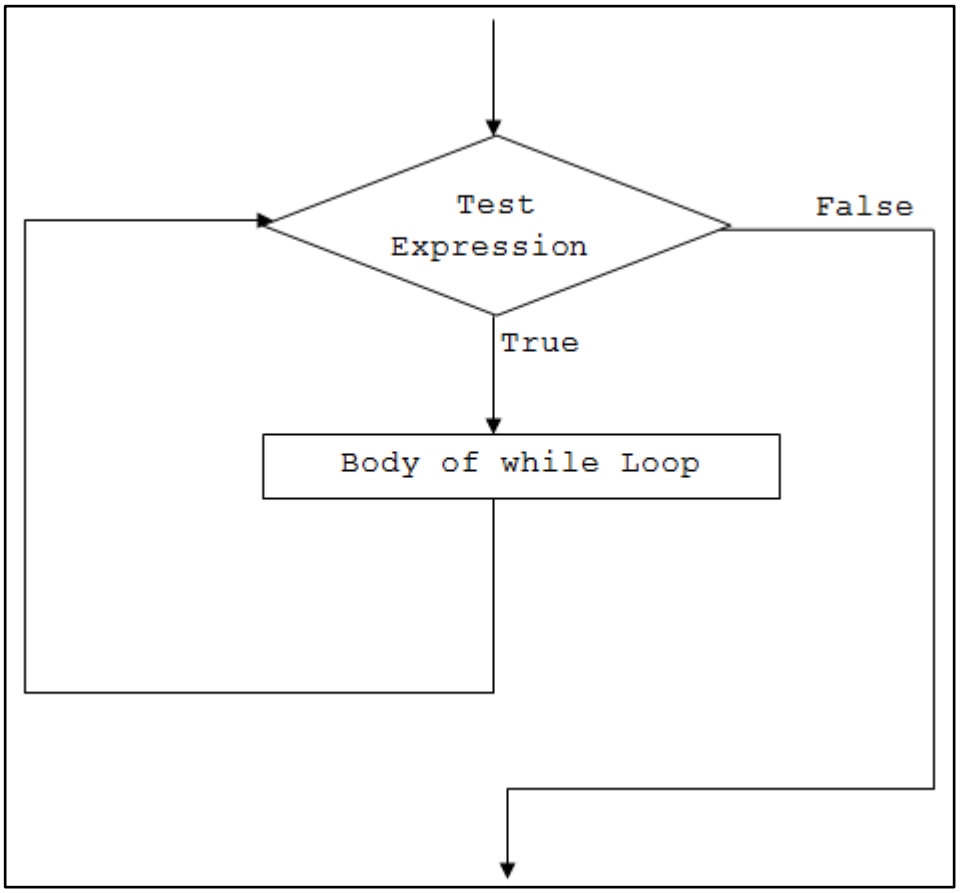
The **while** keyword is used to create while loop in C#. The syntax for while loop is:

```
while (test-expression)
{
      // body of while
}
```

6.1.1 How while loop works?

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

6.1.2 while loop Flowchart



Working of C# while loop

6.1.3 Example 1: while Loop

```
using System;

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

Initially the value of *i* is 1.

When the program reaches the while loop statement,

- the test expression `i <=5` is evaluated. Since *i* is 1 and `1 <= 5` 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 <=5`) is evaluated again. This time too, the expression returns `true` (2 <= 5), 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 6. At this point, the test-expression will evaluate to `false` and hence the loop terminates.

6.1.4 Example 2: while loop to compute sum of first 5 natural numbers

```
using System;

namespace Loop
{
      class WhileLoop
      {
            public static void Main(string[] args)
            {
                  int i=1, sum=0;

                  while (i<=5)
                  {
                        sum += i;
                        i++;
                  }
                  Console.WriteLine("Sum = {0}", sum);
            }
      }
}
```

When we run the program, the output will be:

```
Sum = 15
```

This program computes the sum of first 5 natural numbers.

- Initially the value of *sum* is initialized to 0.
- On each iteration, the value of *sum* is updated to `sum+i` and the value of *i* is incremented by 1.
- When the value of *i* reaches 6, the test expression `i<=5` will return false and the loop terminates.

Let's see what happens in the given program on each iteration.

Initially, *i* = 1, *sum* = 0

While 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.

6.2 C# do...while loop

The **do** and **while** keyword is used to create a do...while loop. It is similar to a while loop, however there is a major difference between them.

In while loop, the condition is checked before the body is executed. It is the exact opposite in do...while loop, i.e. condition is checked after the body is executed.

This is why, the body of do...while loop will execute at least once irrespective to the test-expression.

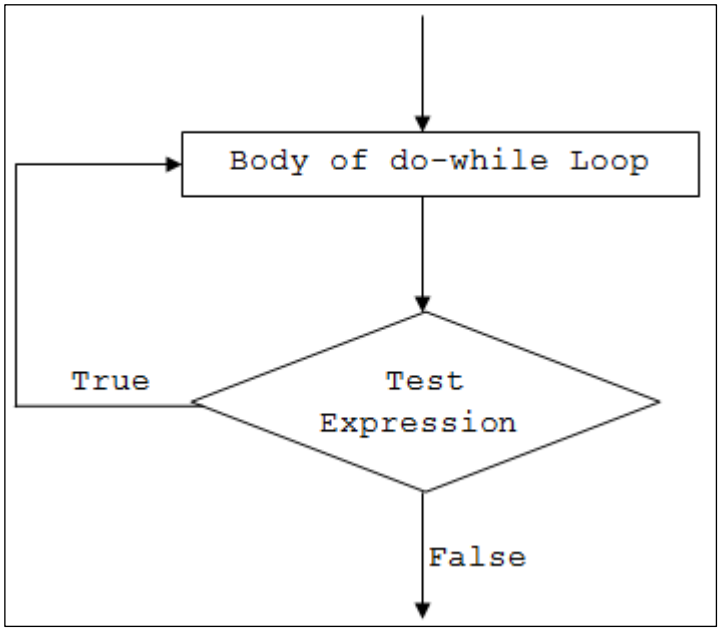
The syntax for do...while loop is:

```
do
{
        // body of do while loop
} while (test-expression);
```

6.2.1 How do...while loop works?

1. The body of do...while loop is executed at first.
2. Then the `test-expression` is evaluated.
3. If the `test-expression` is `true`, the body of loop is executed.
4. When the `test-expression` is `false`, do...while loop terminates.

6.2.2 do...while loop Flowchart



Working of C# do...while loop

6.2.3 Example 3: do...while loop

```
using System;

namespace Loop
{
      class DoWhileLoop
      {
            public static void Main(string[] args)
            {
                  int i = 1, n = 5, product;

                  do
                  {
                        product = n * i;
                        Console.WriteLine("{0} * {1} = {2}", n, i, product);
                        i++;
                  } while (i <= 10);
            }
      }
}
```

When we run the program, the output will be:

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
```

```
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

As we can see, the above program prints the multiplication table of a number (5).

- Initially, the value of *i* is 1. The program, then enters the body of do..while loop without checking any condition (as opposed to while loop).
- Inside the body, *product* is calculated and printed on the screen. The value of *i* is then incremented to 2.
- After the execution of the loop's body, the test expression `i <= 10` is evaluated. In total, the do...while loop will run for 10 times.
- Finally, when the value of *i* is 11, the test-expression evaluates to `false` and hence terminates the loop.

6.3 Infinite while and do...while loop

If the test expression in the while and do...while loop never evaluates to `false`, the body of loop will run forever. Such loops are called infinite loop.

6.3.1 Infinite while loop

```
while (true)
{
     // body of while loop
}
```

6.3.2 Infinite do...while loop

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

The infinite loop is useful when we need a loop to run as long as our program runs.

For example, if your program is an animation, you will need to constantly run it until it is stopped. In such cases, an infinite loop is necessary to keep running the animation repeatedly.

6.4 Flow control-Examples

6.4.1 Ex-1

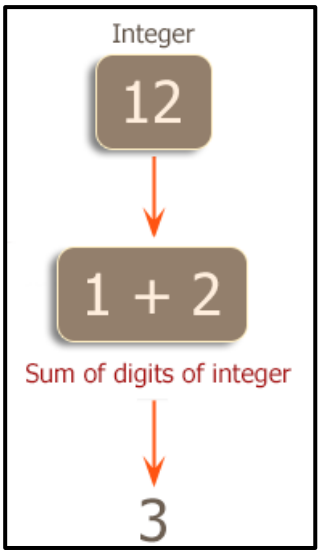
Write a C# program to print the odd numbers from 1 to 99. Prints one number per line.

Sample Output:

```
using System;
public class Ex65
{
    public static void Main()
      {
          Console.WriteLine("Prints one number per line.");
          for (int n = 1; n < (99 + 1); n++)
            {
                if (n % 2 != 0)
                {
                    Console.WriteLine(n.ToString());
                }
            }

    }
}
```

6.4.2 Compute the sum of the digits of an integer



C# Sharp Code:

```
using System;
public class Ex67
{
    public static void Main()
    {
        Console.Write("Input  a number(integer): ");
        int n = Convert.ToInt32(Console.ReadLine());
        int sum = 0;
        while (n != 0)
        {
            sum += n % 10;
            n /= 10;
        }
        Console.WriteLine("Sum of the digits : " + sum);
    }
}
```

Sample Output:

```
Input  a number(integer): 12
Sum of the digits: 3
```

6.4.3 Ex-3

Write a C# program to print the following shape using nested for loop

```
333333
333333
333333
333333
333333
333333
```

Code:

```
using System;
public class Ex68
{
    public static void Main()
    {
        for (int i = 1; i <= 6; i++)
        {
            for (int j = 1; j <= i ; j++)
            {
                Console.Write("3");
            }
            Console.WriteLine();
        }
    }
}
```

6.4.4 Ex-4

Write a C# program to print the following shape using nested for loop:

```
3
33
333
3333
33333
333333
```

Code:

```
using System;
public class Ex69
{
    public static void Main()
    {
        for (int i = 1; i <= 6; i++)
        {
            for (int j = 1; j <= i ; j++)
            {
                Console.Write("3");
            }
            Console.WriteLine();
        }
    }
}
```

6.4.5 Ex-5

Write a C# program to print the following shape using nested for loop:

```
88888888
8888888
888888
88888
8888
888
88
8
```

Code:

```
using System;
public class Ex611
{
    public static void Main()
    {
        for (int i = 1; i <= 9; i++)
        {
            for (int j = i; j < 9; j++)
            {
                Console.Write("8");
            }
            Console.WriteLine();
        }
    }
}
```

6.4.6 Ex-6

Write a C# program to print the following shape using nested for loop:

```
        5
       55
      555
     5555
    55555
   555555
  5555555
 55555555
555555555
```

Code:

```
using System;
public class Ex622
{
    public static void Main()
    {
        for (int i = 1; i <= 9; i++)
        {
            for (int j = i; j < 9; j++)
            {
                Console.Write(" ");
            }
            for (int j = 1; j <= i; j++)
            {
                Console.Write("5");
            }
            Console.WriteLine();
        }
    }
}
```

6.4.7 Ex-7

Write a C# program to print the following shape using nested for loop:

```
        1
       22
      333
     4444
    55555
   666666
  7777777
 88888888
999999999
```

Code:

```
using System;
public class Ex622
{
    public static void Main()
    {
        for (int i = 1; i <= 9; i++)
        {
            for (int j = i; j < 9; j++)
            {
                Console.Write(" ");
            }
            for (int j = 1; j <= i; j++)
            {
                Console.Write(i);
            }
            Console.WriteLine();
        }
    }
}
```

6.4.8 Ex-8

Write a C# program to print the following shape using nested for loop:

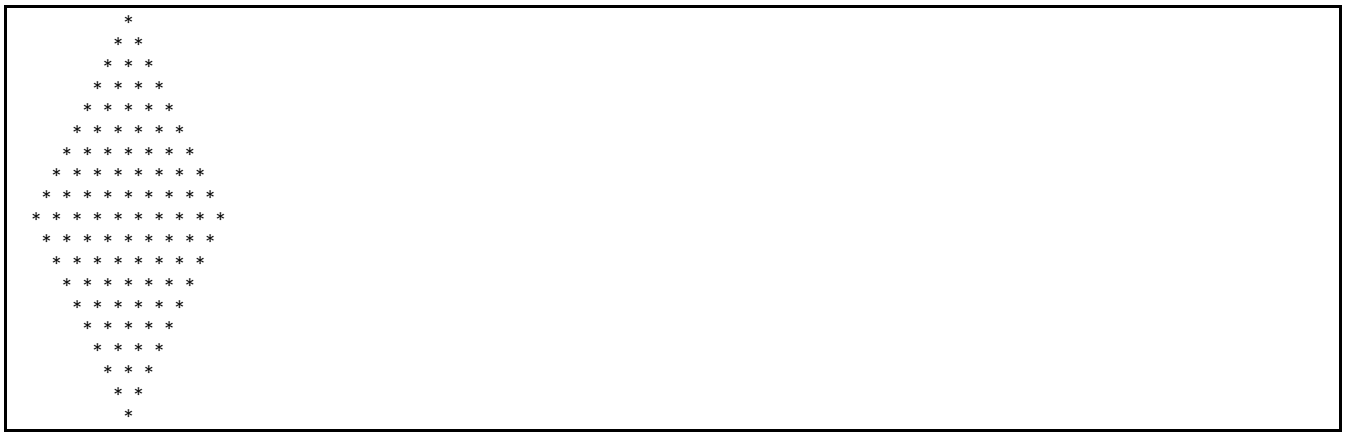
```
                   *
                  **
                 ***
                ****
               *****
              ******
             *******
            ********
           *********
          **********
         ***********
        ************
       *************
      **************
  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

Code:

```
using System;
public class Ex622
{
    public static void Main()
    {
        for (int i = 1; i <= 19; i++)
        {
            for (int j = i; j < 19; j++)
            {
                Console.Write(" ");
            }
            for (int j = 1; j <= i; j++)
            {
                Console.Write("*");
            }
            Console.WriteLine();
        }
    }
}
```

6.4.9 Ex-9

Write a C# program to print the following shape using nested for loop:



Code:

```
using System;
public class Ex688
{
    public static void Main()
    {
        for (int i = 1; i <= 10; i++)
        {
            for (int j = i; j < 10; j++)
            {
                Console.Write(" ");
            }
            for (int k = 1; k <= i; k++)
            {
                Console.Write(" *");
            }
            Console.WriteLine();
        }

        for (int i = 1; i <= 10; i++)
        {
            for (int k = 1; k <= i; k++)
            {
                Console.Write(" ");
            }
            for (int j = i; j < 10; j++)
            {
                Console.Write(" *");
            }

            Console.WriteLine();
        }

    }
}
```

6.4.10 Ex-10

Write a C# program for the following formula:

$$z = x^1 + x^2 + x^3 + \ldots\ldots\ldots + x^n$$

```
using System;
public class Ex691
{
    public static void Main()
    {
        double z=0, n,p=1,x;
        n = Convert.ToDouble( Console.ReadLine());
        x = Convert.ToDouble(Console.ReadLine());


        for (int i = 1; i <= n; i++)
        {
            p = 1;
            for (int j = 1; j <= i ; j++)
            {
                p = p * x;
            }

            z = z + p;
        }

        Console.WriteLine(z);
    }
}
```

6.4.11 Ex-11

Write a C# program for factorial:

0! = 1

1! = 1

2! = 1*2 = 2

3! =1*2*3 = 6

7! = 1* 2 * 3 * 4* 5* 6* 7 = 5040

Code:

```
using System;
public class Ex692
{
    public static void Main()
    {
        double n,f=1;

        n = Convert.ToDouble(Console.ReadLine());



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

            f = f * i;
        }

        Console.WriteLine(f);
    }
}
```

6.4.12 Ex-12

Write a C# program for the following formula:

$$z = \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ \ldots\ldots\ldots + \frac{x^n}{n!}$$

Code:

```
using System;
public class Ex691
{
    public static void Main()
    {
        double z = 0, n, p = 1, x,f=1;
        n = Convert.ToDouble(Console.ReadLine());
        x = Convert.ToDouble(Console.ReadLine());



        for (int i = 1; i <= n; i++)
        {
            p = 1;
            f = 1;
            for (int j = 1; j <= i; j++)
            {
                p = p * x;
                f = f * i;
            }

            z = z + p/f;
        }

        Console.WriteLine(z);
    }
}
```

6.4.13 Ex-13

Write a C# program for the following formula:

$$z = \frac{x^1}{1!+n} + \frac{x^2}{2!+n} + \frac{x^3}{3!+n} + \ \ldots\ldots\ldots + \frac{x^n}{n!+n}$$

Code:

```
using System;
public class Ex691
{
    public static void Main()
    {
        double z = 0, n, p = 1, x,f=1;
        n = Convert.ToDouble(Console.ReadLine());
        x = Convert.ToDouble(Console.ReadLine());



        for (int i = 1; i <= n; i++)
        {
            p = 1;
            f = 1;
            for (int j = 1; j <= i; j++)
            {
                p = p * x;
                f = f * i;
            }

            z = z + p/(f+n);
        }

        Console.WriteLine(z);
    }
}
```