

For Next Loop

A **For Next loop** is used to repeatedly execute a sequence of code or a block of code until a given condition is satisfied. A For loop is useful in such a case when we know how many times a block of code has to be executed. In VB.NET, the For loop is also known as For Next Loop.

Syntax

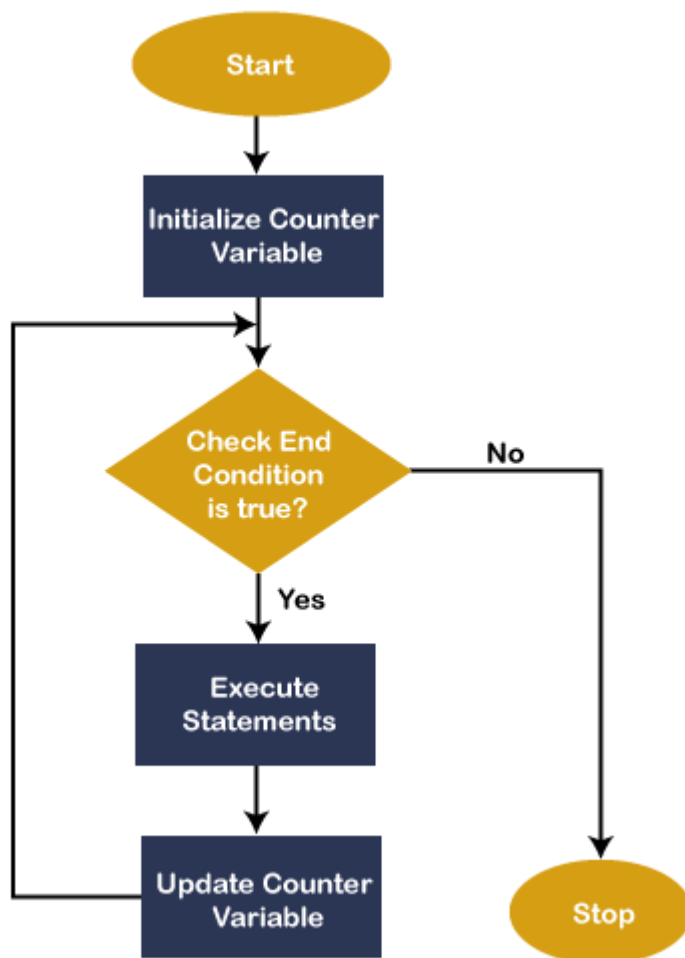
1. For variable_name As [DataType] = start To end [Step step]
2. [Statements to be executed]
3. Next

Let's understand the For Next loop in detail.

- **For:** It is the keyword that is present at the beginning of the definition.
- **variable_name:** It is a variable name, which is required in the For loop Statement. The value of the variable determines when to exit from the **For-Next loop**, and the value should only be a numeric.
- **[Data Type]:** It represents the Data Type of the **variable_name**.
- **start To end:** The **start** and **end** are the two important parameters representing the initial and final values of the **variable_name**. These parameters are helpful while the execution begins, the initial value of the variable is set by the start. Before the completion of each repetition, the variable's current value is compared with the end value. And if the value of the variable is less than the end value, the execution continues until the variable's current value is greater than the end value. And if the value is exceeded, the loop is terminated.
- **Step:** A step parameter is used to determine by which the **counter** value of a variable is increased or decreased after each iteration in a program. If the counter value is not specified; It uses 1 as the default value.
- **Statements:** A statement can be a single statement or group of statements that execute during the completion of each iteration in a loop.
- **Next:** In VB.NET a **Next** is a keyword that represents the end of the **For loop's**

Flowchart of For Next loop

The following flowchart represents the functioning of the For Next loop in the **VB.NET programming language**.



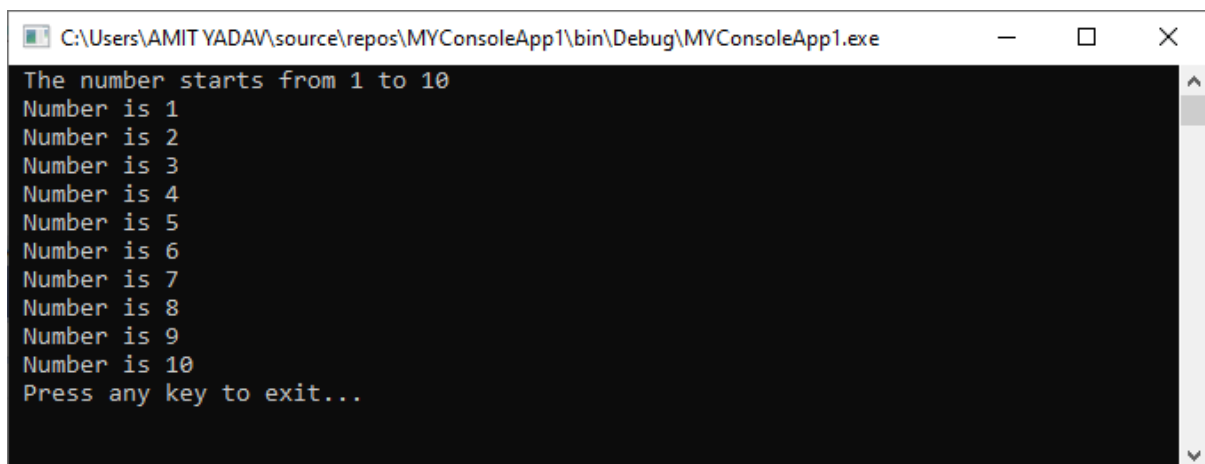
In the above flow chart, the first step is to initialize the variable name with the start value. And then, the value of the variable will be compared to the **end expression** or value. If the condition is true, the control enters the loop body and executes the statements. After that, the value of a variable will be **automatically incremented** by the compiler. Upon **completion** of each iteration, the current value of a variable will be **again compared** to the end expression. If the condition is not true, the controlled **exit** from the loop.

Example 1. Write a simple program to print the number from 1 to 10 using the For Next loop.

Program Name : Number.vb

```
1. Imports System
2. Module Number
3.     Sub Main()
4.         ' It is a simple print statement, and 'vbCrLf' is used to jump in the next line.
5.         Console.WriteLine(" The number starts from 1 to 10 " & vbCrLf)
6.         ' declare and initialize variable i
7.         For i As Integer = 1 To 10 Step 1
8.             ' if the condition is true, the following statement will be executed
9.             Console.WriteLine(" Number is {0} ", i)
10.            ' after completion of each iteration, next will update the variable counter
11.        Next
12.        Console.WriteLine(" Press any key to exit... ")
13.        Console.ReadKey()
14.    End Sub
15. End Module
```

Output:



```
C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe
The number starts from 1 to 10
Number is 1
Number is 2
Number is 3
Number is 4
Number is 5
Number is 6
Number is 7
Number is 8
Number is 9
Number is 10
Press any key to exit...
```

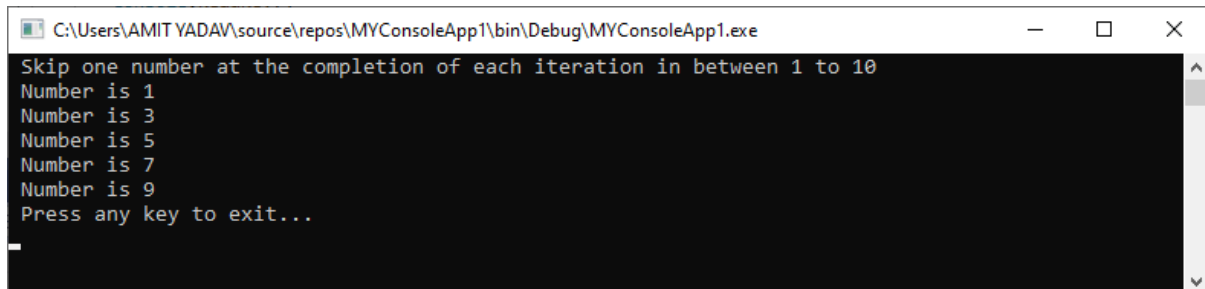
In the above example, we have initialized an integer variable **i** with an initial value 1. The For loop will continuously execute its body until the value of **i** is smaller or equal to 10. After each iteration, the value of **i** is automatically increased with '**Step 1**'. If the value of **i** reached 10, the loop would be **terminated** and control transfer to the **Main()** function.

Further, we can change the **Step** in For Next loop. Write the following program to skip the number is 2.

Number.vb

```
1. Imports System
2. Module Number
3.     Sub Main()
4.         ' declaration of variable i
5.         Dim i As Integer
6.         Console.WriteLine(" Skip one number at the completion of each iteration in between 1 to 1
0 " & vbCrLf)
7.         ' initialize i to 1 and declare Step to 2 for skipping a number
8.         For i = 1 To 10 Step 2
9.             ' if condition is true, it skips one number
10.            Console.WriteLine(" Number is {0} ", i)
11.            ' after completion of each iteration, next will update the variable counter to step 2
12.        Next
13.        Console.WriteLine(" Press any key to exit... ")
14.        Console.ReadKey()
15.    End Sub
16. End Module
```

Output:



```
C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe
Skip one number at the completion of each iteration in between 1 to 10
Number is 1
Number is 3
Number is 5
Number is 7
Number is 9
Press any key to exit...
```

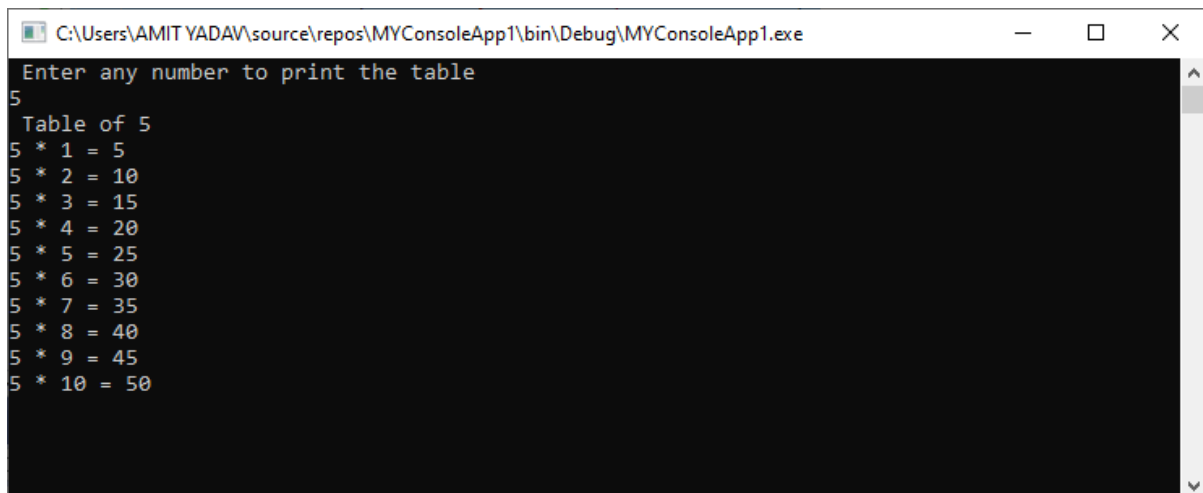
As we can see in the above output, the value of the variable **i** is **initialized with 1**, and the **value of i is skipped by 'Step 2' in the loop for each iteration** to print the skipped number from 1 to 10.

Example 2: Write a simple program to print a table in the VB.NET.

Table.vb

```
1. Imports System
2. Module Table
3.     Sub Main()
4.         'declaration of i and num variable
5.         Dim i, num As Integer
6.         Console.WriteLine(" Enter any number to print the table")
7.         num = Console.ReadLine() ' accept a number from the user
8.         Console.WriteLine(" Table of " & num)
9.         'define for loop condition, it automatically initialize step to 1
10.        For i = 1 To 10
11.            Console.WriteLine(num & " * " & i & " = " & i * num)
12.        Next
13.        Console.ReadKey()
14.    End Sub
15. End Module
```

Output:



```
C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe
Enter any number to print the table
5
Table of 5
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
```

VB.NET While End Loop

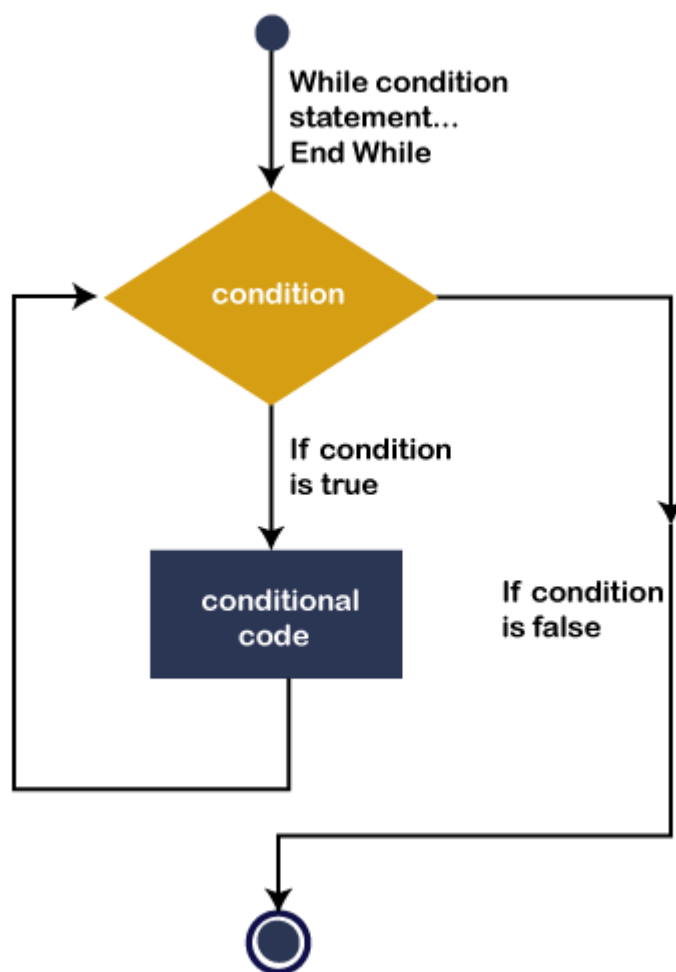
The **While End loop** is used to execute blocks of code or statements in a program, as long as the given **condition** is true. It is useful when the number of executions of a block is not known. It is also known as an **entry-controlled loop** statement, which means it initially checks all loop conditions. If the condition is true, the body of the while loop is executed. This process of repeated execution of the body continues until the condition is not false. And if the condition is false, control is transferred out of the loop.

Syntax:

1. While [condition]
2. [Statement to be executed]
3. End While

Here, **condition** represents any **Boolean condition**, and if the logical condition is true, the **single or block of statements** define inside the body of the while loop is executed.

Flow Diagram of the While End Loop in VB.NET



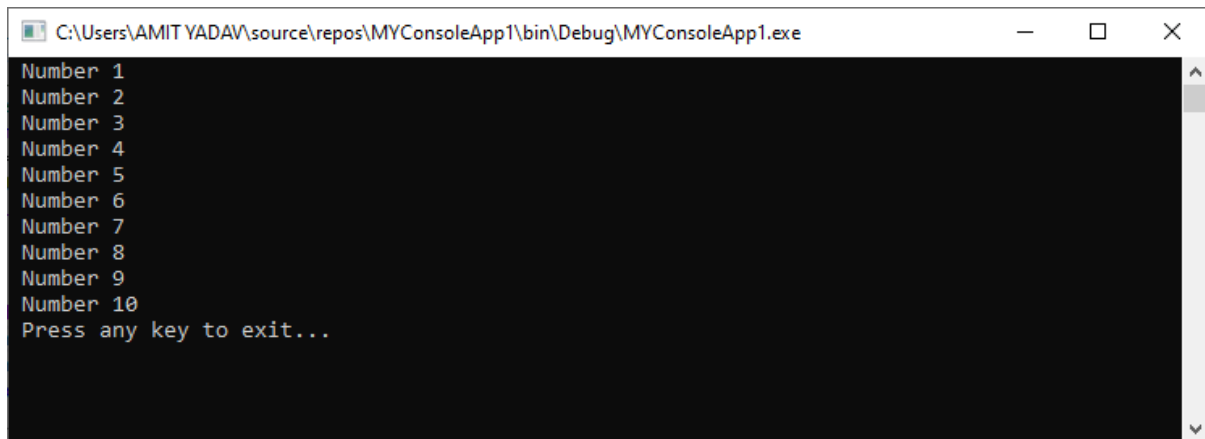
As we know, the **While End loop** is an **entry-controlled** loop used to determine if the condition is true, the statements defined in the body of the loop are executed, and the execution process continues till the **condition** is satisfied. Furthermore, after each iteration, the value of the counter variable is incremented. It again checks whether the defined condition is true; And if the condition is again true, the body of the While loop is executed. And when the condition is not true, the control transferred to the end of the loop.

Example: Write a simple program to print the number from 1 to 10 using while End loop in [VB.NET](#).

while_number.vb

```
1. Imports System
2. Module while_number
3.     Sub Main()
4.         'declare x as an integer variable
5.         Dim x As Integer
6.         x = 1
7.         ' Use While End condition
8.         While x <= 10
9.             'If the condition is true, the statement will be executed.
10.            Console.WriteLine(" Number {0}", x)
11.            x = x + 1 ' Statement that change the value of the condition
12.        End While
13.        Console.WriteLine(" Press any key to exit...")
14.        Console.ReadKey()
15.    End Sub
16. End Module
```

Output:

A screenshot of a Windows console window titled "C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe". The console output shows a list of numbers from 1 to 10, each on a new line, followed by the text "Press any key to exit...". The numbers are: Number 1, Number 2, Number 3, Number 4, Number 5, Number 6, Number 7, Number 8, Number 9, and Number 10. The console has a black background and white text.

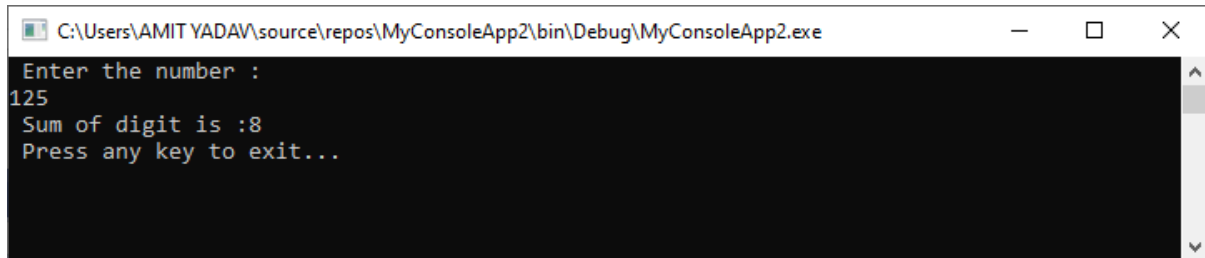
In the above example, while loop executes its body or statement up to the defined state ($i \leq 10$). And when the value of the variable i is 11, the defined condition will be false; the loop will be terminated.

Example 2: Write a program to print the sum of digits of any number using while End loop in VB.NET.

Total_Sum.vb

```
1. Public Class Total_Sum ' Create a Class Total_sum
2.     Shared Sub Main()
3.         'Declare an Integer variable
4.         Dim n, remainder, sum As Integer
5.         sum = 0
6.
7.         Console.WriteLine(" Enter the number :")
8.         n = Console.ReadLine() ' Accept a number from the user
9.
10.        ' Use While loop and write given below condition
11.        While (n > 0)
12.            remainder = n Mod 10
13.            sum += remainder
14.            n = n / 10
15.        End While
16.        Console.WriteLine(" Sum of digit is :{0}", sum)
17.        Console.WriteLine(" Press any key to exit...")
18.        Console.ReadKey()
19.    End Sub
20. End Class
```


Output:



```
C:\Users\AMIT YADAV\source\repos\MyConsoleApp2\bin\Debug\MyConsoleApp2.exe
Enter the number :
125
Sum of digit is :8
Press any key to exit...
```

Do While Loop

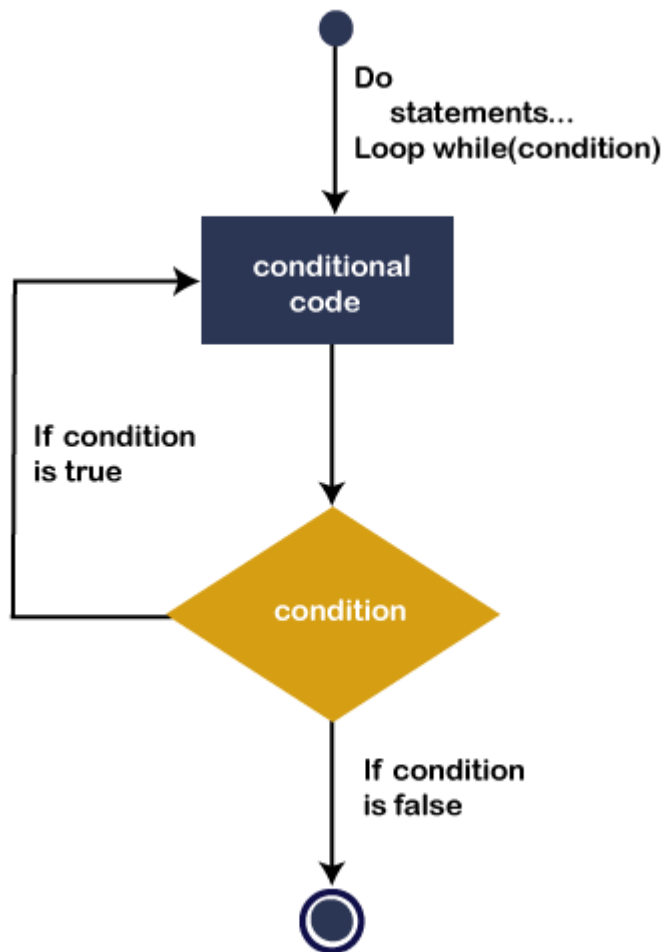
In VB.NET, Do While loop is used to execute blocks of statements in the program, as long as the condition remains true. It is similar to the [While End Loop](#), but there is slight difference between them. The **while** loop **initially checks** the defined condition, if the condition becomes true, the while loop's statement is executed. Whereas in the **Do** loop, is opposite of the while loop, it means that it executes the Do statements, and then it checks the condition.

Syntax:

1. Do
2. [Statements to be executed]
3. Loop While Boolean_expression
4. **// or**
5. Do
6. [Statement to be executed]
7. Loop Until Boolean_expression

In the above syntax, the **Do** keyword followed a block of statements, and **While** keyword checks **Boolean_expression** after the execution of the first Do statement.

Flowchart of Do loop



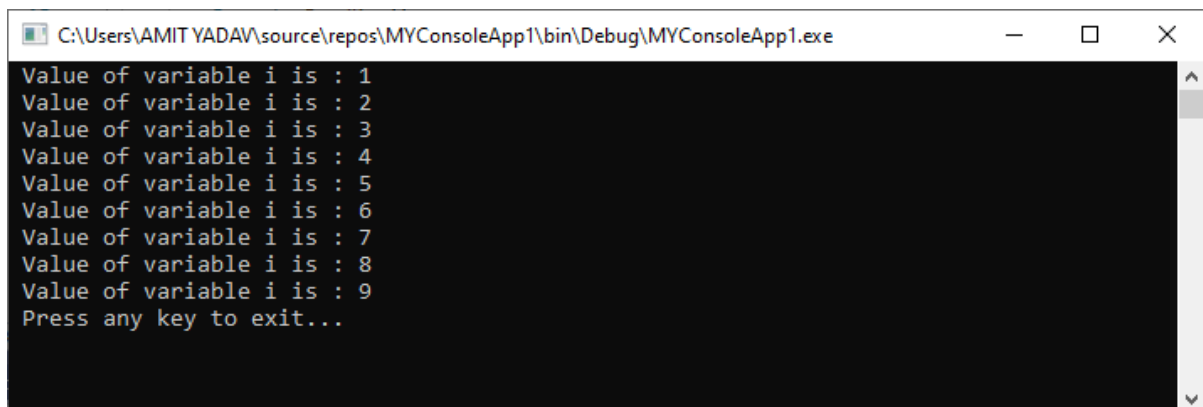
The above flow chart represents the flow of Do While loop. It is used to **control the flow of statements**, such that it executes the statement at least once before checking the While or Until condition. If the condition is true, the next iteration will be executed till the condition become false.

Example 1. Write a simple program to print a number from 1 to 10 using the Do While loop in VB.NET.

Do_loop.vb

```
1. Imports System
2. Module Do_loop
3.     Sub Main()
4.         ' Initializatio and Declaration of variable i
5.         Dim i As Integer = 1
6.         Do
7.             ' Executes the following Statement
8.             Console.WriteLine(" Value of variable I is : {0}", i)
9.             i = i + 1 'Increment the variable i by 1
10.        Loop While i <= 10 ' Define the While Condition
11.
12.        Console.WriteLine(" Press any key to exit...")
13.        Console.ReadKey()
14.    End Sub
15. End Module
```

Now compile and execute the above program by clicking on the Start button, it shows the following output:

A screenshot of a Windows console application window. The title bar shows the file path: C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe. The console output displays the values of variable i from 1 to 9, each on a new line, followed by the prompt "Press any key to exit...". The text is white on a black background.

```
C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe
Value of variable i is : 1
Value of variable i is : 2
Value of variable i is : 3
Value of variable i is : 4
Value of variable i is : 5
Value of variable i is : 6
Value of variable i is : 7
Value of variable i is : 8
Value of variable i is : 9
Press any key to exit...
```

In the above program, the Do While loop executes the body until the given condition becomes **false**. When the condition becomes false the loop will be terminated.

Use of Until in Do Until Loop statement

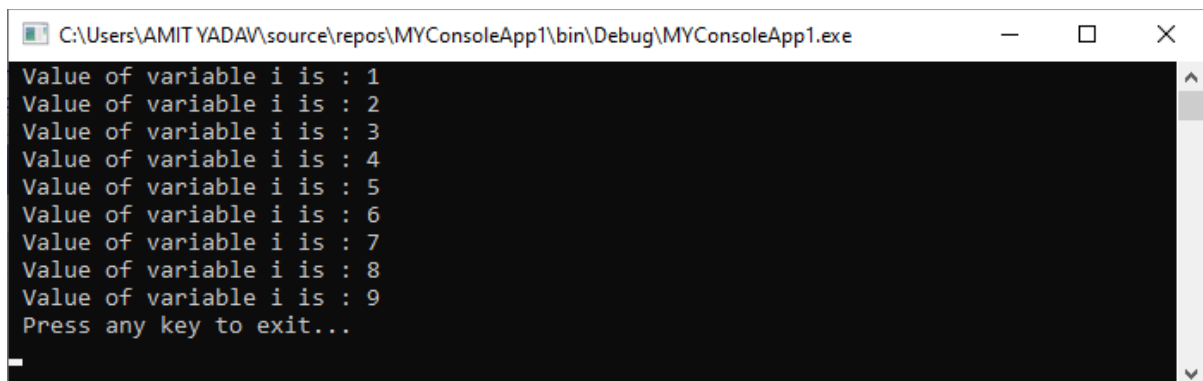
In the VB.NET loop, there is a **Do Until loop** statement, which is similar to the **Do While loop**. The Do Statement executes as long as Until condition becomes true.

Example: Write a program to understand the uses of Do Until Loop in VB.NET.

Do_loop.vb

```
1. Imports System
2. Module Do_loop
3.     Sub Main()
4.         ' Initialization and Declaration of variable i
5.         Dim i As Integer = 1
6.         Do
7.             ' Executes the following Statement
8.             Console.WriteLine(" Value of variable i is : {0}", i)
9.             i = i + 1 'Increment variable i by 1
10.        Loop Until i = 10 ' Define the Until Condition
11.
12.        Console.WriteLine(" Press any key to exit...")
13.        Console.ReadKey()
14.    End Sub
15. End Module
```

Output:



```
C:\Users\AMIT YADAV\source\repos\MYConsoleApp1\bin\Debug\MYConsoleApp1.exe
Value of variable i is : 1
Value of variable i is : 2
Value of variable i is : 3
Value of variable i is : 4
Value of variable i is : 5
Value of variable i is : 6
Value of variable i is : 7
Value of variable i is : 8
Value of variable i is : 9
Press any key to exit...
```

In the above program, a Do Until loop is executed their statement until the given condition **Until (i =10)** is not meet. When the counter value of the **variable i** becomes 10, the defined statement will be false, and the loop will be terminated.

Nested Do While Loop Statement

In VB.NET, when we use one Do While loop inside the body of another Do While loop, it is called Nested Do While loop.

Syntax

1. Do
2. `// Statement of the outer loop`
3. Do
4. `// Statement of outer loop`
5. While (condition -2)
6. `// Statement of outer loop`
7. While (condition -1)