Control Statements in VB.NET

Kavita K. Bharti
Assistant Professor
Department of Computer
Durga Mahavidyalaya, Raipur

In **VB.NET**, the **control statements** are the statements that controls the execution of the program on the basis of the specified condition. It is useful for determining whether a condition is true or not. If the condition is true, a single or block of statement is executed. In the control statement, we will use **if- Then, if Then Else, if Then ElseIf** and the **Select case** statement.

We can define more than one condition to be evaluated by the program with statements. If the defined condition is true, the statement or block executes according to the condition, and if the condition is false, another statement is executed.

The following figure shows a common format of the decision control statements to validate and execute a statement:

- If-Then Statement
- If-Then Else Statement
- If-Then ElseIf Statement
- Select Case Statement

If-Then Statement

The **If-Then** Statement is a control statement that defines one or more conditions, and if the particular condition is satisfied, it executes a piece of information or statements.

Syntax:

```
If condition Then
[Statement or block of Statement]
End If
```

In **If-Then** Statement, the **condition** can be a Boolean, logical, or relational condition, and the statement can be single or group of statements that will be executed when the condition is true.

Write a simple program to print greater number between two numbers VB.NET.

```
Module if_statement2
   Sub Main()
      ?Definition of variables
      Dim no1, no2 As Integer
      Console.WriteLine("Enter any two number:")
      no1 = Console.ReadLine() ?read no1 from user
      no2 = Console.ReadLine() ?read no2 from user
      If no1 > no2 Then
         Console.WriteLine("First number is greater than second number")
      End If
      If no1 < no2 Then
         Console.WriteLine("Second number is greater than First number")
      End If
      Console.WriteLine("press any key to exit...")
      Console.ReadKey()
End Sub
End Module
```

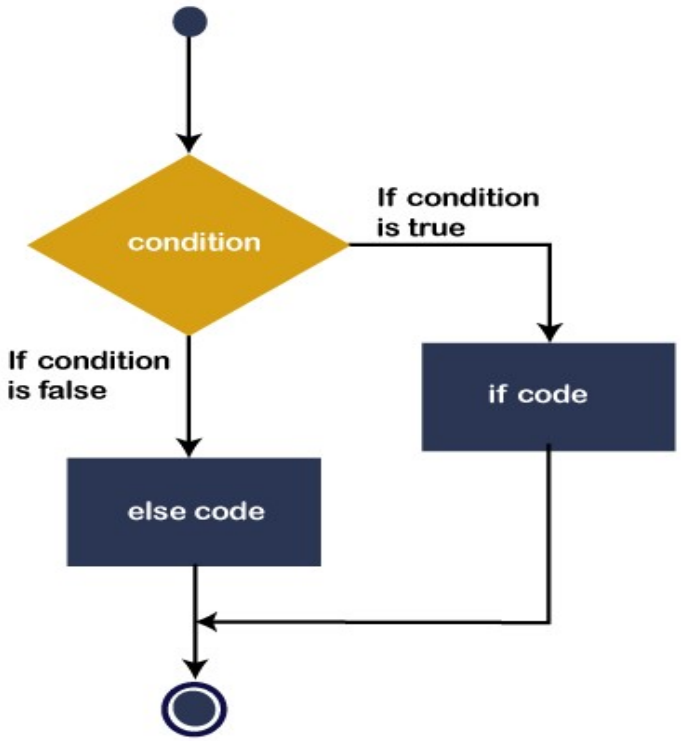
If-Then-Else Statement

The **If-Then** Statement can execute single or multiple statements when the condition is true, but when the expression evaluates to **false**, it does nothing. So, here comes the **If-Then-Else** Statement. The IF-Then-Else Statement is telling what **If** condition to do when if the statement is false, it executes the Else statement. Following is the If-Then-Else statement syntax in VB.NET as follows:

Syntax:

```
If (Boolean_expression) Then
'This statement will execute if the Boolean condition is true
Else
'Optional statement will execute if the Boolean condition is false
End If
```

Flow chart



Write a program to check whether the number is even or odd.

```
Module If_Else_statement
Sub Main()
    Dim num As Integer
    Console.WriteLine("Enter the Number")
    num = Console.ReadLine() 'read data from console
      If (num Mod 2 = 0) Then ' if condition is true, print the if statement
        Console.WriteLine("It is an even number")
      Else 'otherwise, Else statement is executed.
        Console.WriteLine("It is an odd number")
    End If
   Console.WriteLine("press any key to exit...")
    Console.ReadKey()
End Sub
End Module
```

If-Then-ElseIf statement

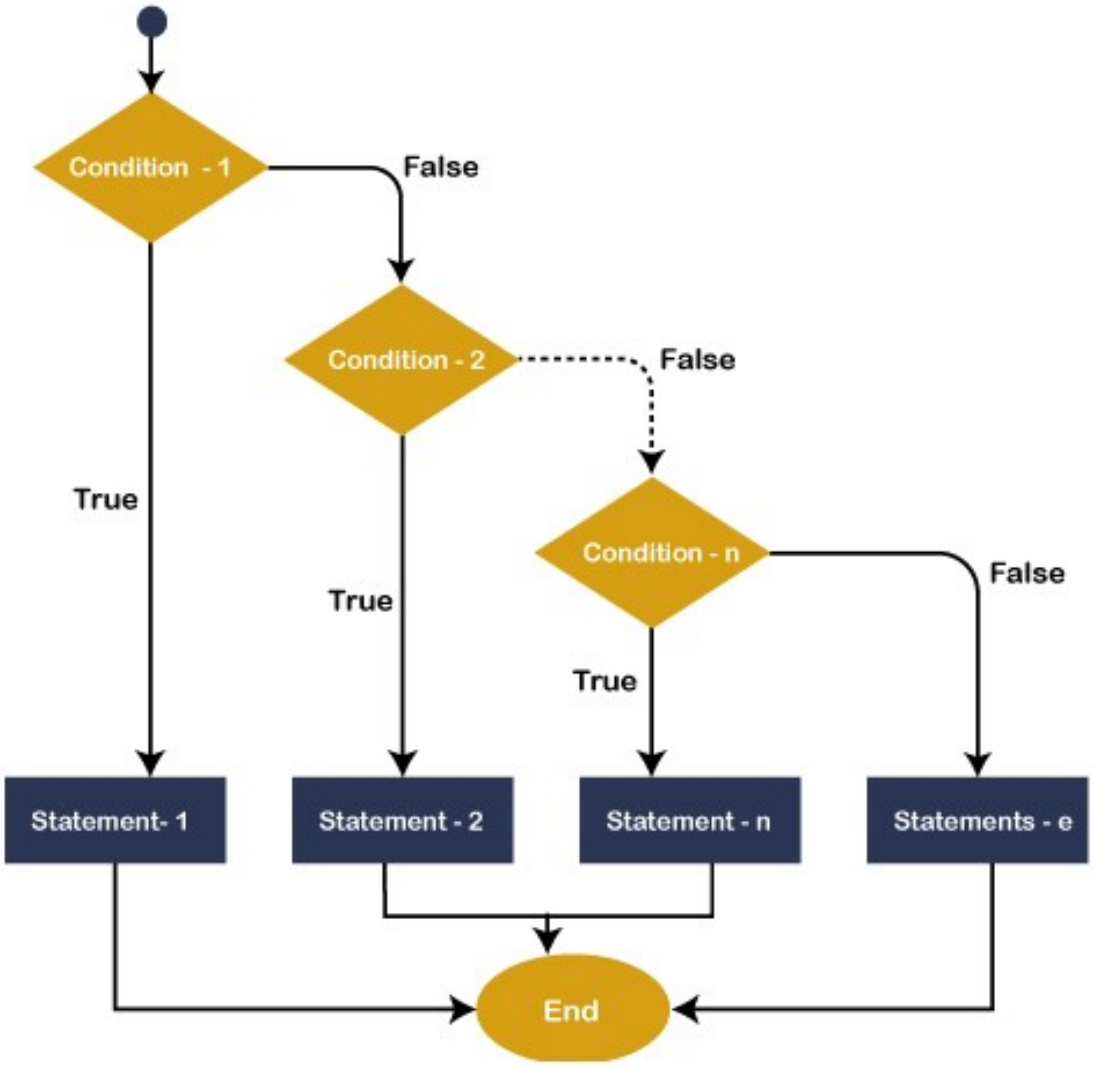
The **If-Then-ElseIf** Statement provides a choice to execute only one condition or statement from multiple statements. Execution starts from the top to bottom, and it checked for each If condition. And if the condition is met, the block of If the statement is executed. And **if none** of the conditions are true, the last **block** is executed. Following is the syntax of If-Then-ElseIf Statement in VB.NET as follows:

Syntax

```
If(condition 1)Then
  ' Executes when condition 1 is true
ElseIf( condition 2)Then
  ' Executes when condition 2 is true
ElseIf( boolean_expression 3)Then
  ' Executes when the condition 3 is true
Else
  ' executes the default statement when none of the above conditions is true.
End If
```

Flowchart

The following diagram represents the functioning of the If-Else-If Statement in the VB.NET programming language.



If this condition is true in the flowchart of the if-else-if statement, the statement is executed within the if block. If the condition is not true, it passes control to the next ElseIf condition to check whether the condition is matched. And if none of the conditions are matched, the else block is executed.

Example 1: Write a program to show the uses of If... ElseIf statements.

```
Module if_elseIf
   Sub Main()
      Dim var1 As Integer

      Console.WriteLine(" Input the value of var1: ")
      var1 = Console.ReadLine()
      If var1 = 20 Then
         'if condition is true then print the following statement'
         Console.WriteLine(" Entered value is equal to 20")
      ElseIf var1 < 50 Then
         Console.WriteLine(" Entered value is less than 50")

       ElseIf var1 >= 100 Then
         Console.WriteLine(" Entered value is greater than 100")
      Else
         'if none of the above condition is satisfied, print the following statement
         Console.WriteLine(" Value is not matched with above condition")
      End If
      Console.WriteLine(" You have entered : {0}", var1)
      Console.WriteLine(" press any key to exit...")
      Console.ReadKey()
   End Sub
End Module
```

Select Case Statement

n VB.NET, the Select Case statement is a collection of multiple case statements, which allows executing a single case statement from the list of statements. A selected case statement uses a variable to test for equality against multiple cases or statements in a program. If the variable is matched with any test cases, that statement will be executed. And if the condition is not matched with any cases, it executes the default statement.

Using the select case statement in VB.NET programming, you can replace the uses of multiple If-Then-Else If statement from the program for better readability and easy to use.

Syntax

Following is the syntax of the Select Case statement in VB.NET, as follows:

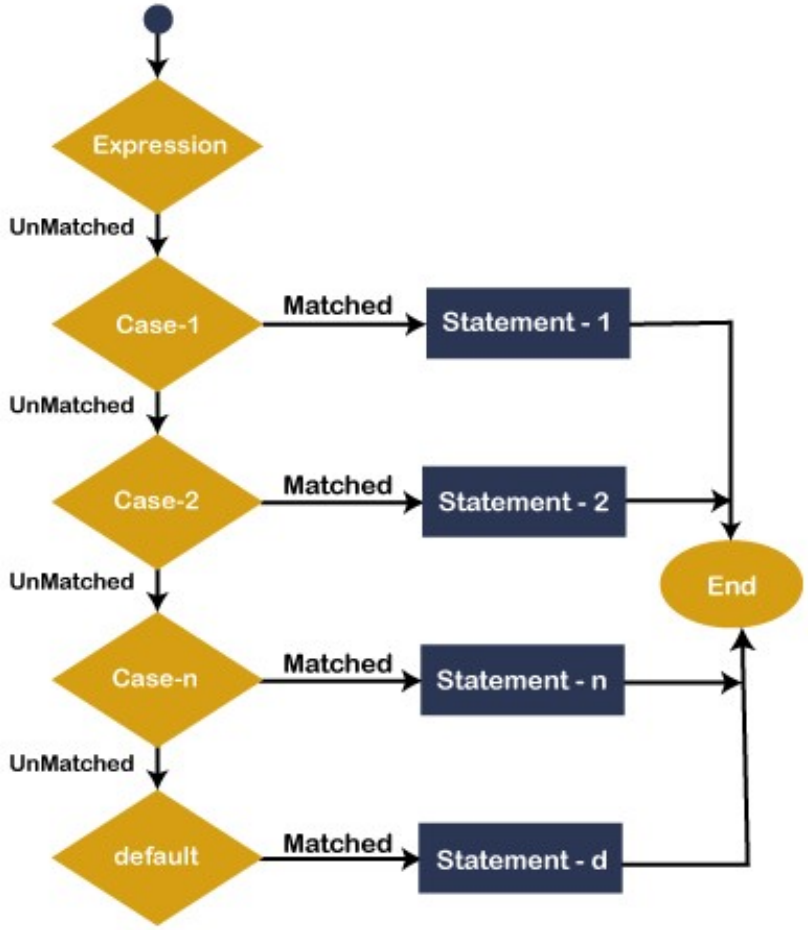
```
Select Case [variable or expression]
Case value1 'defines the item or value that you want to match.
// Define a statement to execute

Case value2 'defines the item or value that you want to match.
// Define a statement to execute

Case Else
// Define the default statement if none of the conditions is true.
End Select
```

Flowchart of Select Case Statement

The following flowchart represents the functioning of the Select case statement in the VB.NET programming language.



In Flowchart, the Select Case statement represents the evaluating of the process start from top to bottom. If the expression or value is matched with the first select case, statement -1 is executed else the control transfer to the next case for checking whether the expression is matching or not. Similarly, it checks all Select case statements for evaluating. If none of the cases are matched, the Else block statement will be executed, and finally, the Select Case Statement will come to an end.

Write a program to display the Days name using the select case statement in VB.NET.

```
Imports System
Module Select_case
    Sub Main()
```

```
        'define a local variable.
        Dim Days As String
        Days = "Thursday"
        Select Case Days
            Case "Monday"
                Console.WriteLine(" Today is Monday")
            Case "Tuesday"
                Console.WriteLine(" Today is Tuesday")
            Case "Wednesday"
                Console.WriteLine("Today is Wednesday")
            Case "Thursday"
                Console.WriteLine("Today is Thursday")
            Case "Friday"
                Console.WriteLine("Today is Friday")
            Case "Saturday"
                Console.WriteLine("Today is Saturday")
            Case "Sunday"
                Console.WriteLine("Today is Sunday")
            Case Else
                Console.WriteLine(" You have typed Something wrong")

        End Select
        Console.WriteLine("You have selected : {0}", Days)
        Console.WriteLine("Press any key to exit...")
        Console.ReadLine()
    End Sub
End Module
```