

Class: MSc

Subject : Application of IT- Basics and Advance Excel

Chapter: Unit 3 Chapter 1

Chapter Name: Conditional Statements

# Conditional Statements

- *Often while designing a code we are bound to verify functionalities based on certain conditions and make decisions according to the output of the conditional statement.*
- *Conditional Statements are used in programming languages to perform a set of actions depending on the condition specified by the programmer that evaluates to true or false.*
- *These are mainly used to decide the execution flow. If the condition evaluates to true, execute a certain set of actions and if the condition evaluates to false then perform another set of actions.*

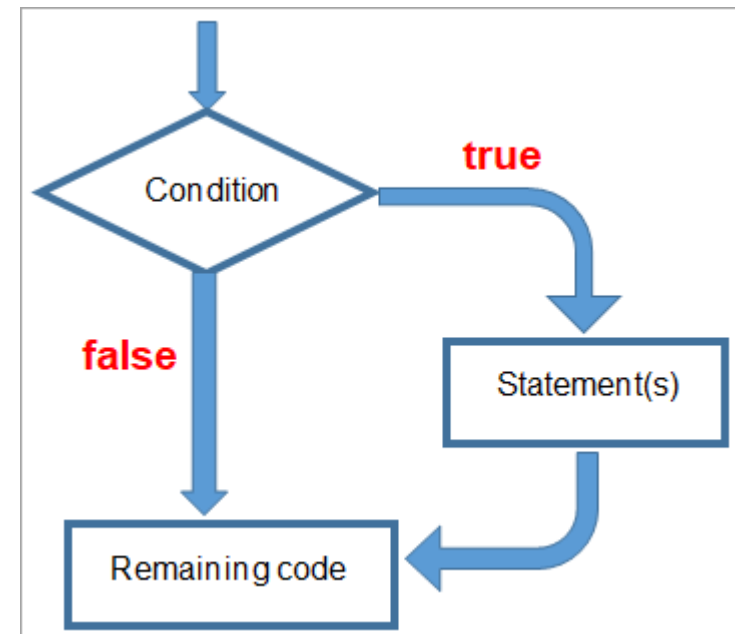
# ***Types of Conditional Statements***

Sl.No	Conditional Statement	Description
1	If...Then	Set of statements are executed only if the condition is true.
2	If.. Then...Else	Set of statements under If block are executed If the condition is true otherwise statements under else block will be executed.
3	If..Elseif	Each Else block if again have a conditional statement based on which the statements will be executed.
4	Nested Ifs	Placing an If statement inside another if statement.
5	Select Case	Each case statement will have a variable value, based on the selection value mentioned in the select case statement, appropriate case will be executed.

# If Statements

- If statements execute a set of actions depending on the condition. If the condition evaluates to true then the code mentioned in the If block will be executed.
- Syntax:
- Condition: This is the required field. Based on the Boolean result of this condition the action will be performed. If the result is true then the statements in the If block will be executed.
- If the condition is Null then it is treated as False.
- Statements: This set of actions will be performed if the condition is true.

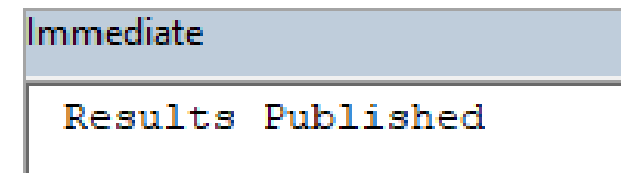
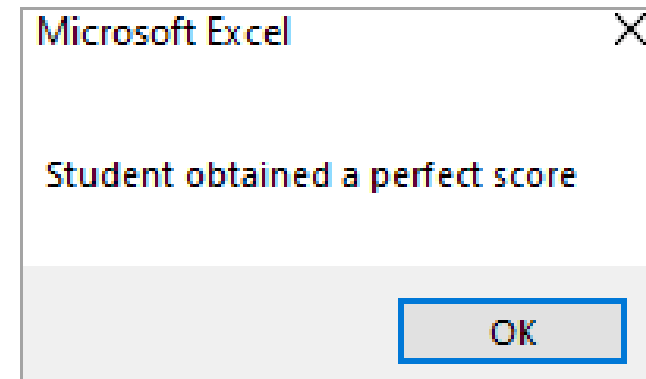
```
If condition Then  
[statements]  
End If
```



# If Statements

## Example:

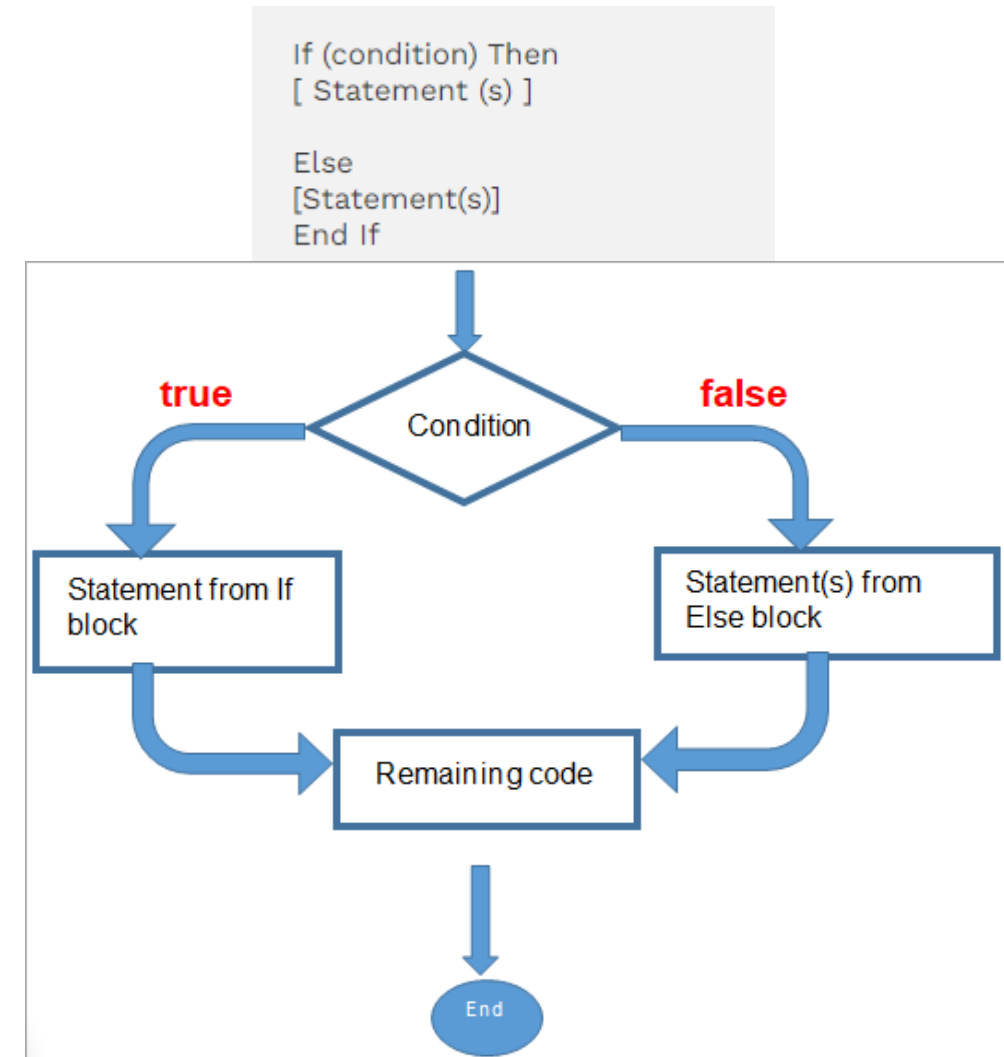
```
Option Explicit
Sub ifExample()
Dim Obtained_Marks, Total_Marks As Integer
Obtained_Marks = 100
Total_Marks = 100
If (Obtained_Marks = Total_Marks) Then
MsgBox "Student obtained a perfect score"
End If
Debug.Print "Results Published"
End Sub
```



*The output from the above code will be a msgbox as shown below and whether the condition is true or false "Result Published" will be printed in the immediate window.*

# IF... Then... Else Statements

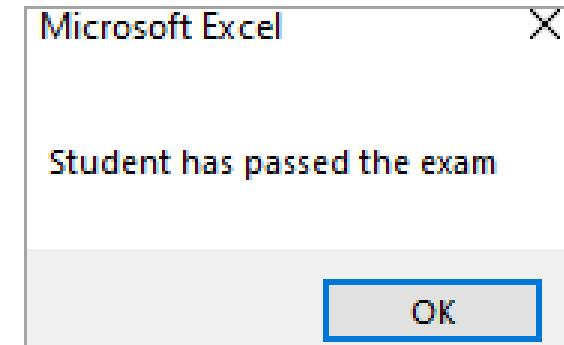
- *If the condition returns a boolean true, then the set of actions defined under the if block will be executed but if the conditional expression returns a boolean false then the statements under the else block will be executed.*
- *Syntax:*
- *Once the code reaches the conditional statement, it evaluates the value of the expression. The If-block is executed if the condition is true and the Else block is executed if the condition is false. It is not possible to execute both the If and Else blocks in a single run.*



# ***IF... Then... Else Statements***

## **Example:**

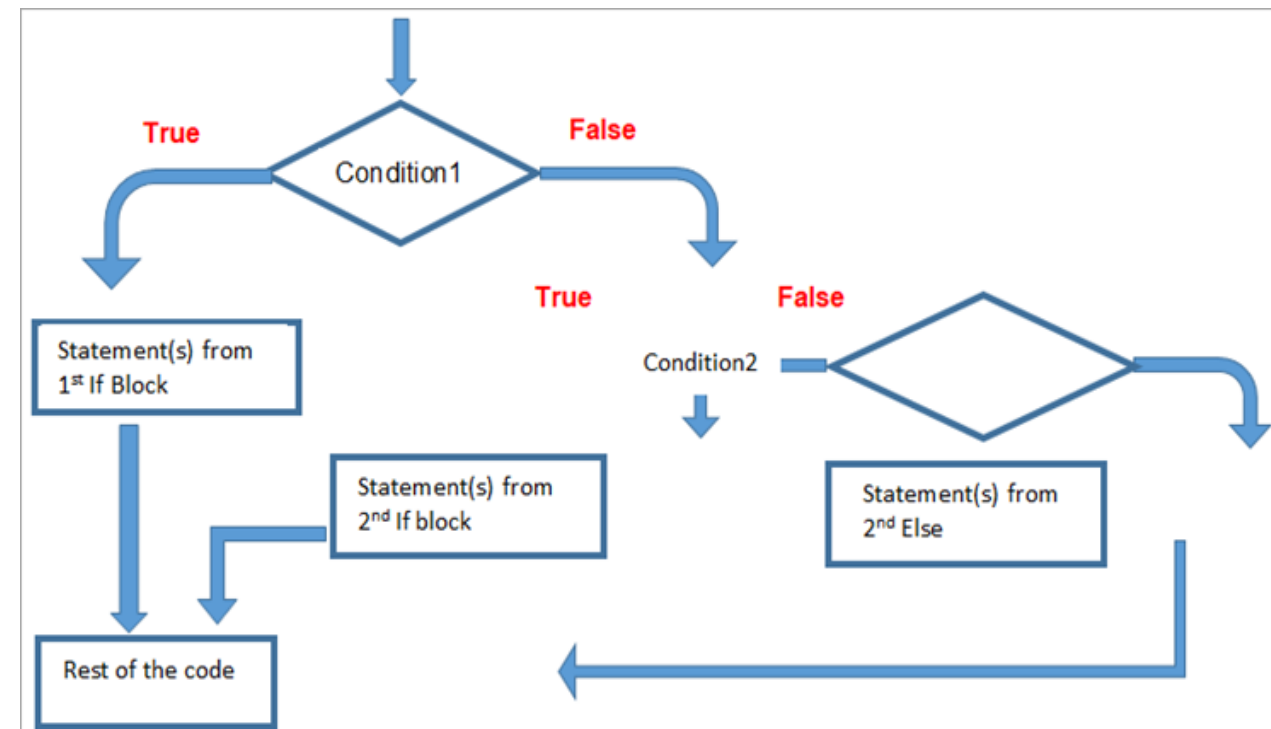
```
Sub ifElseExample()  
Dim Obtained_Marks, Passing_Marks As Integer  
Obtained_Marks = 35  
Passing_Marks = 35  
If (Obtained_Marks >= Passing_Marks) Then  
    MsgBox "Student has passed the exam"  
Else  
    MsgBox "Student did not clear the exam"  
End If  
End Sub
```



# ElseIf Statements

- To test a second condition we can add ElseIf statements to a simple If..Then..Else. An If statement is allowed to be followed by multiple ElseIf statements each consisting of a conditional statement.
- Syntax:
- Once the code reaches the conditional expression, it evaluates either to True or False. If the condition is true then the statements under the 1st IF block will be executed and the control exists in the conditional block, but if the expression returns false then the control will enter the 2nd conditional expressions and repeats the process.

```
If(condition) Then  
    [Statement(s)]  
Elseif (condition)Then  
    [Statement (s)]  
End If  
End If
```

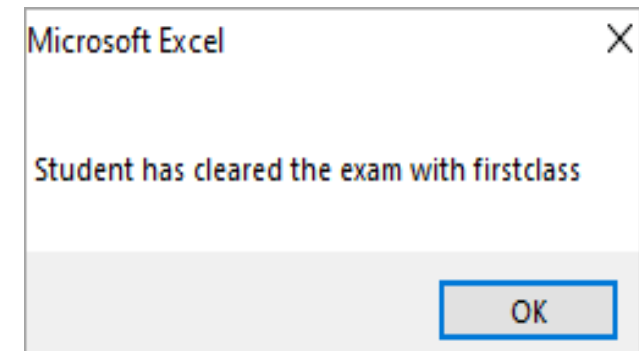




# ***ElseIf Statements***

## **Example:**

```
Sub ifElseifExample()  
Dim Obtained_Marks, Passing_Marks As Integer  
Obtained_Marks = 60  
Passing_Marks = 35  
If (Obtained_Marks < Passing_Marks) Then  
    MsgBox "Student did not clear the exam"  
Elseif (Obtained_Marks >= 60) Then  
    MsgBox "Student has cleared the exam with firstclass"  
Else  
    MsgBox "Student passed with second class"  
End If  
End Sub
```



# Nested IF Statements

- *VBA allows us to place control statements inside another control statement.*
- *Placing an If statement inside another if statement. This procedure of placing one control statement within another is called to be nested.*
- *Control structures in VBA can be nested to as many levels as you wish. By intending the body of each control statement, it will be better readable.*

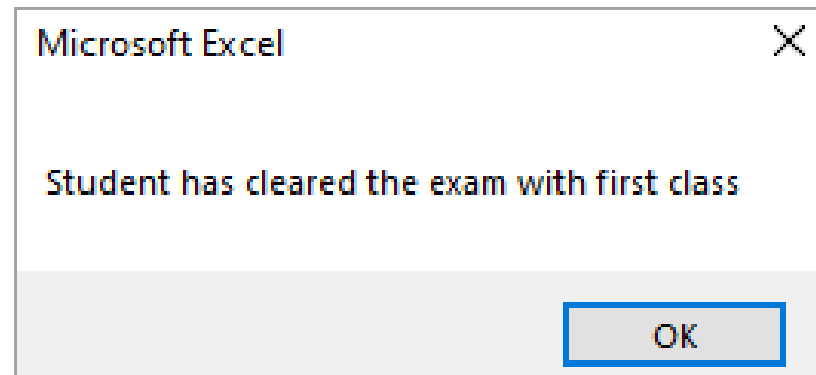
- *Syntax:*

```
If (condition) Then
Statement(s)
If(condition) Then
Statement(s)
Elseif (condition) Then
Statement(s)
Else
    Statement(s)
End If
Else
Statement(s)
End If
```

# Nested IF Statements

## Example:

```
Sub NestedIFExample()  
Dim Obtained_Marks  
Obtained_Marks = 67  
If (Obtained_Marks > 0) Then  
    If (Obtained_Marks = 100) Then  
        MsgBox "Student has got a perfect score"  
    ElseIf (Obtained_Marks >= 60) Then  
        MsgBox "Student has cleared the exam with first class"  
    ElseIf (Obtained_Marks >= 50) Then  
        MsgBox "Student cleared the exam with second class"  
    ElseIf (Obtained_Marks >= 35) Then  
        MsgBox "Student has cleared"  
    Else  
        MsgBox " Student did not clear the exam"  
    End If  
    ElseIf (Obtained_Marks = 0) Then  
        MsgBox "Student scored a zero)"  
    Else  
        MsgBox "student did not attend the exam"  
    End If  
End Sub
```



# Select Case

- From the above nested if statement we have seen how cumbersome it is to deal with multiple if..else statements. If you misplace a single If or Else then it is difficult to debug and hence it is more error-prone. To deal with such a problem we can use Select Case.
- In Select Case, you can enter the block of code to be executed under a particular case statement. Each case statement will have a variable value to identify. Before we begin the execution, we have to specify which case is to be executed by entering the variable value in the Select Case Statement.
- Select Case has a 3 part syntax:
  1. Testexpression: Mandatory field and takes any numeric or string expression as input.
  2. expressionlist-n: List of expressions using which the appropriate case will be selected.
  3. statements-n: Set of actions performed if the test expression matches the case expression list.
  4. elstatements: Set of actions to be executed if the test expression does not match any of the case statements.

```
Select Case testexpression  
[ Case expressionlist-n ]  
    [ statements-n ]  
[ Case Else ]  
    [ elstatements ]  
End Select
```

# Select Case

**Example:** *You can try to run the code by putting different marks*

```
Sub selectExample()  
Dim marks As Integer  
marks = InputBox("Enter Total Marks")  
Select Case marks  
Case 100  
    MsgBox "Perfect score"  
Case 60 To 99  
    MsgBox "First Class"  
Case 50 To 59  
    MsgBox "Second class"  
Case 35 To 49  
    MsgBox "Pass"  
Case 1 To 34  
    MsgBox "Not Cleared"  
Case 0  
    MsgBox "Scored zero"  
Case Else  
    MsgBox "Did not attend the exam"  
End Select  
End Sub
```