

First Tutorial

In this tutorial, we will be discussing branches and their different types and ways of using them.

We will start with the simplest form of If statement:

```
Sub SimpleIfMacro()  
  
    Dim Age As Integer  
    Const VOTING_AGE = 18  
  
    Age = InputBox("What is your age?", "Getting Age")  
  
    If (Age >= VOTING_AGE) Then  
        MsgBox ("Eligible to vote")  
    End If  
  
End Sub
```

[Typed]

- They have already seen everything; other than the if statement; in the previous tutorials
 - Defining a variable
 - Defining a constant
 - Getting input from the user
 - Displaying output to the user
- For using an if statement, you have to
 - Start with an “If” and end up with “End If”
 - Similar to “Sub” and “End Sub”
 - Just after the “If” we place our condition between two brackets “()”
 - The condition is similar to what they have used in excel
 - = (equal)
 - > (greater than)
 - < (less than)
 - >= (greater than or equal)
 - <= (less than or equal)
 - <> (not equal)
 - Once you are done with the condition, you add “Then” to the end of the sentence
 - All this means, if the condition defined between the two brackets “()” is right, then execute whatever comes after “Then” till the end of the if statement “End If”

Students to do

- This will be their first macro using branches.
- Ask the students to get an input from the user asking if he/she is attending the course 203.
- They should type “yes” or “no”.

```
Sub StudentsIfMacro()  
  
    Dim Attending As String  
  
    Attending = InputBox("Are you attending the course?", "Checking Attendance")  
    If (Attending = "yes") Then  
        MsgBox ("See you there!")  
    End If  
  
End Sub
```

[Displayed, done by students]

So far, we only covered simple examples of branching.

We can take it one step forward.

```
Sub IfElseMacro()  
  
    Dim Age As Integer  
    Const VOTING_AGE = 18  
  
    Age = InputBox("What is your age?", "Getting Age")  
    If (Age >= VOTING_AGE) Then  
        MsgBox ("Eligible to vote")  
    Else  
        MsgBox ("Not eligible to vote")  
    End If  
  
End Sub
```

[Typed]

- It is similar to the simple if statement but now we have extra situations
- We used “Else”
 - When we used “If” only, it will only execute when the condition is right
 - In the current case, it will always execute whether the condition is right or not
 - When the condition is right, it will run everything before the “Else”
 - When the condition is wrong, it will run everything after the “Else”

Now it is time to discuss the nested if statements.

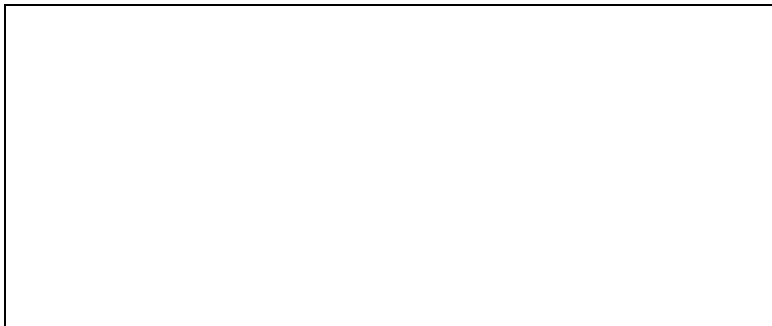
```
Sub NestedIfsMacro()  
  
    Dim Citizenship As String  
    Dim Age As Integer  
    Const VOTING_AGE = 18  
  
    Age = InputBox("What is your age?", "Getting Age")  
    If (Age >= VOTING_AGE) Then  
        Citizenship = InputBox("What is your age?", "Checking Citizenship")  
        If (Citizenship = "Y" Or Citizenship = "y") Then  
            MsgBox ("Eligible to vote")  
        Else  
            MsgBox ("Not eligible to vote")  
        End If  
    Else  
        MsgBox ("Not eligible to vote")  
    End If  
  
End Sub
```

[Displayed]

- It is important to explain to the students that within the “If” code block or “Else” code block, we can type any code we want
- That might also include another if statement

Students to do

Ask the students to get the grade from the user and check whether it is a pass/fail grade.



The last type of if statements they have to know is the one with several different conditions.

```
Sub IfElseIfElseMacro()  
  
    Dim Grade As Integer  
    Const PASS_MARK = 50  
    Const EXCELLENT_MARK = 90  
  
    Grade = InputBox("What is your grade?", "Getting Grade ")  
    If (Grade >= EXCELLENT_MARK) Then  
        MsgBox ("Excellent")  
    ElseIf (Grade >= PASS_MARK) Then  
        MsgBox ("Pass")  
    Else  
        MsgBox ("Fail")  
    End If  
  
End Sub
```

[Typed]

- It is important to explain to the students how the if-elseif-else statement works and how it differs from several single if statements
- The if-elseif-else will
 - Go from top to down
 - Check each condition
 - The first condition to be right is the one to be executed
 - The other conditions will be ignored/skipped
 - In case none of the conditions were right, the else will be executed (“none of the above” case)
 - It is not obligatory to have the “Else” but it can be useful for error checking
 - On that note there is a problem with error checking and the above version of the program, can any students spot the problem?

```
Sub IfElseIfElseMacro()  
  
    Dim Grade As Integer  
    Const FAIL_CUT_OFF  
    Const PASS_MARK = 50  
    Const EXCELLENT_MARK = 90  
  
    Grade = InputBox("What is your grade?", "Getting Grade ")  
    If (Grade >= EXCELLENT_MARK) Then  
        MsgBox ("Excellent")
```

```

ElseIf (Grade >= PASS_MARK) Then
    MsgBox ("Pass")
ElseIf (Grade > FAIL_CUT_OFF) Then
    MsgBox ("Fail")
Else
    MsgBox ("Error: cannot have negative grade")
End If
End Sub

```

Finally, they should learn how to use the condition operators

- They have used it with access and excel

```

Sub ConditionsOperatorsMacro()

    Dim AverageGrade As Integer
    Dim FinalExamGrade As Integer
    Const PASS_MARK = 50

    AverageGrade = InputBox("What is your course average grade?", "Getting Grade ")
    FinalExamGrade = InputBox("What is your final exam grade?", "Getting Grade ")

    If ((AverageGrade >= PASS_MARK) AND (FinalExamGrade >= PASS_MARK)) Then
        MsgBox ("Pass") ' Must pass both final and the overall average of each component
    Else
        MsgBox ("Fail")
    End If

End Sub

```

[Typed]

- Simply just remind them that they have already seen the “AND” and only briefly describe how it works

```
If ((AverageGrade < PASS_MARK) OR (FinalExamGrade < PASS_MARK)) Then
    MsgBox ("Fail")
Else
    MsgBox ("Pass")
End If
```

[Typed]

- Just change the conditions but keep the same output
 - Use "OR"