



Creating If/Then/Else Routines

You can use `If/Then/Else` routines to give logic to your macros. The process of the macro proceeds in different directions depending on the results of an `If` command. Just like the `IF` function in Excel, the `If/Then/Else` command relies on a logical statement with a true scenario and a false scenario.

We saw one `If/Then/Else` command in Chapter 6, where we created a macro that asked the user which print area he wanted to print (see the following box). Here's a plain-language description of how the macro works. The macro analyzes the user response by determining that, if the user enters 1, then print `PrintArea1`, and the macro then ends. The macro logic continues, however, with an `ElseIf` command that allows for the user to enter something other than a 1. If the user enters something other than 1, the macro doesn't end and instead proceeds to the next step, which determines that, if the user enters 2, then print `PrintArea2`, and then end the macro. Finally, if the user enters a wrong answer, that is, something other than 1 or 2, the macro contains a provision to return to the original question and give the user another chance. (Note that the user also has the right to cancel the macro operation at any time by clicking a `Cancel` button.)

SPECIALPRINT MACRO FROM CHAPTER 6

```

Sub SpecialPrint()
Question:
Report = InputBox("Enter 1 to print Report 1; Enter 2 to print Report 2")
If Report = 1 Then
    GoTo Print1
ElseIf Report = 2 Then
    GoTo Print2
Else
    GoTo Question
End If
Print1:
    Application.Goto Reference:="PrintArea1"
    ExecuteExcel4Macro "PRINT(1,,1,,,,,,1,,TRUE,,FALSE)"
End
Print2:
    Application.Goto Reference:="PrintArea2"
    ExecuteExcel4Macro "PRINT(1,,1,,,,,,1,,TRUE,,FALSE)"
End
End sub

```

148

MEMO

The Else statement in the If/Then/Else routine is not a required command in this If/Then structure. You can create If/Then routines that do not include an alternate direction provided by the Else statement.

The success of any If/Then/Else routine comes from anticipating all the possible responses and providing commands to deal with each possible condition.

Understanding the If/Then/Else Routine

When you enter an If statement in your macro, the If precedes a logical statement. It is up to Visual Basic to determine if this statement is true or false. This statement that follows If is called a *conditional expression*. The condition of this statement can be either true or false. If it is a true statement, the macro proceeds to do what it is told to do in the Then statement.

If you only want an action to occur when the statement is true, then your macro is finished when you have an `If` and a `Then` statement.

Create a Simple If/Then Macro

For example, say you have a worksheet that contains numbers that represent annual sales figures. If the sales figure exceeds 100,000, then you want the macro to calculate a bonus by placing a figure in the cell to the right that equates to the original figure times 2 percent.

Before you jump into programming this little macro, think through the process in baby steps. Here are all the things that this macro needs to do:

- Examine the number in the current cell and determine if it is larger than 100,000.
- If the number is larger than 100,000, move the cellpointer one cell to the right.
- Enter a calculation in the new cell that multiplies the number in the original cell by 2 percent.

Some of this macro can be recorded so that you can harvest the code. You can record yourself moving the cellpointer one cell to the right, and you can record the creation of the formula. The only thing you can't record is the `If/Then` statement.

Recording the cellpointer movement and the formula yields this macro code:

```
ActiveCell.Offset(0, 1).Range("A1").Select
ActiveCell.FormulaR1C1 = "=RC[-1]*0.02"
ActiveCell.Offset(1, 0).Range("A1").Select
```

The first line of code reflects the movement of the cellpointer from the original cell to the cell to the right—`Offset(0, 1)` shows movement of 0 rows and 1 column forward (or right).

The second line of the code provides the formula: No row movement, but one column back, times 0.02.

MEMO

Because we want to use this macro on several different cells, be sure to turn on the Relative References feature before turning on the macro recorder.

MEMO

Cell references in macros are always portrayed as Row first, Column second. Thus an offset of (1,-1) refers to one row forward, or down, and one column back, or left.

The third line of the code moves the cellpointer down one cell, a circumstance that naturally occurs when you press ENTER. `Offset (1, 0)` refers to an advance of 1 row and 0 columns. If instead you want the cellpointer to move to the next cell in the column of existing numbers (in anticipation of applying this macro to the next number), then you should change the offset to (1,-1).

Now all you need in order to have this macro make the logical decision is the `If` statement that asks if the original cell contains a number greater than 100000.

```
If ActiveCell.Value > 100000 Then
```

The preceding line of code asks if the value of the active cell is greater than 100,000. If the answer is true, then the next line of the VBA code executes. If the answer is false, nothing happens.

Note one final thing—any time you use an `If/Then` statement, you must conclude the `If/Then` section with an `End If` statement.

Thus the final macro code looks like this:

```
Sub Bonus ()
'
' Macro to calculate bonus
'
    If ActiveCell.Value > 100000 Then
        ActiveCell.Offset (0, 1).Range ("A1").Select
        ActiveCell.FormulaR1C1 = "=RC[-1]*0.02"
        ActiveCell.Offset (1, -1).Range ("A1").Select
    End If
End Sub
```

To execute this macro, place your cellpointer on a cell containing a number that you want to analyze. Click the Macros button to find your macro on the list, click on the macro, and then click Run. If the number you chose is greater than 100,000, your `Then` sequence is activated and the calculation appears in the cell to the right of the original number. The cellpointer returns to the cell beneath the original number (see Figure 10-1).

	A	B	C	D
1	Annual sales			
2	2000	121578	2431.56	
3	2001	105775	2115.5	
4	2002	98255		
5	2003	168540	3370.8	
6	2004	124851	2497.02	
7	2005	95123		
8	2006	99123		
9	2007	100254	2005.08	
10				
11				

Figure 10-1 The worksheet after the macro has executed

Add an Else Operation for a False Answer

So far we've created a macro that analyzes a situation, and if the situation is true, a command is executed. If the situation is not true, the macro ends. But we don't have to stop there. We can tell the macro to perform some other task if the answer to the initial question is false. In the Bonus macro that we already created, the macro does nothing if the sales figure is less than 100,000. Instead, we can make the macro continue to the Bonus column and enter a zero.

You can record yourself performing this task if you like, but if you look at the macro code that already exists, it should be a pretty simple step to add the Else clause to this macro, without even recording. We want our Else clause to have the operation move one cell to the right (we already have that code in place), and then enter zero. This code should do the trick:

```
ActiveCell.FormulaR1C1 = "0"
```

Then you also need another line of code to provide instructions for where the cellpointer should end up—just as in the Then part of the macro:

```
ActiveCell.Offset(1, -1).Range("A1").Select
```

And so the completed macro, with instructions for how to behave if the statement is true and if it is false, looks like this:

```
Sub Bonus()
' Macro to calculate bonus
  If ActiveCell.Value > 100000 Then
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*0.02"
    ActiveCell.Offset(1, -1).Range("A1").Select
  Else
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "0"
    ActiveCell.Offset(1, -1).Range("A1").Select
  End If
End Sub
```

Add an Elself Operation

So we've seen how to create a macro containing an `If/Then/Else` statement, and that handles the situation when there is only one right and one wrong answer. Now we'll take this a level deeper, and add a second `If` statement, known as an `ElseIf` statement, so that if the macro returns a false answer to the first `If` statement, there is another opportunity for a true statement to occur.

This time we'll add a level of the macro that occurs after the first `If` statement executes and produces a false answer. Instead of immediately assuming there is no bonus to compute and placing a zero in the Bonus cell, we'll apply a second criterion—the ability to calculate a bonus if the sales figure exceeds 75000. This time the bonus calculation will be 1 percent instead of 2 percent. So the complete bonus calculation is 2 percent if sales exceed 100,000 and 1 percent if sales are in the 75,000 to 100,000 range.

You can probably figure out this new piece of code without recording any steps. As a reminder, here's the code that calculates the first bonus:

```
If ActiveCell.Value > 100000 Then
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*0.02"
    ActiveCell.Offset(1, -1).Range("A1").Select
```

Now here's all you have to do add an `ElseIf` layer that asks if the `ActiveCell.Value` exceeds 75,000, and applies a 1 percent bonus:

```
ElseIf ActiveCell.Value > 75000 Then
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*0.01"
    ActiveCell.Offset(1, -1).Range("A1").Select
```

The `ElseIf` format works the same way as the `If` code—you must accompany the `ElseIf` statement with a `Then` statement. This piece of new code can be inserted in the macro. The finished product appears in Figure 10-2. The results appear in Figure 10-3.

```

Chapter 10 annual sales.xlsm - Module1 (Code)
(General) Bonus
Sub Bonus()
' Macro to calculate bonus
  If ActiveCell.Value > 100000 Then
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*0.02"
    ActiveCell.Offset(1, -1).Range("A1").Select
  ElseIf ActiveCell.Value > 75000 Then
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*0.01"
    ActiveCell.Offset(1, -1).Range("A1").Select
  Else
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "0"
    ActiveCell.Offset(1, -1).Range("A1").Select
  End If
End Sub

```

Figure 10-2 The finished If/Then/Else macro

	A	B	C	D
1	Annual sales			
2	2000	121578	2431.56	
3	2001	105775	2115.5	
4	2002	98255	982.55	
5	2003	168540	3370.8	
6	2004	124851	2497.02	
7	2005	95123	951.23	
8	2006	72451	0	
9	2007	100254	2005.08	
10				
11				

Figure 10-3 The Bonus results based on the calculations in the Bonus macro

MEMO

In Chapter 11 you'll learn about creating For/Next loops and you'll be able to apply that skill to macros like this If/Then/Else macro so that you don't have to call the macro on each cell—the macro can run through the entire list of sales figures with one command.

Create a Multilevel If/Then/Else Macro

Back in Chapter 6, you learned how to create a macro using the Case command to offer the user a selection of several different criteria. We created one macro that calculated U.S. corporate income tax. We can perform a similar operation using the If/Then/Else format. Since I'm a tax accountant, I like to revert to tax examples for my macros. This time we'll create a macro called SingleTax that calculates U.S. individual income tax for a single individual, using the If/Then/Else macro style. In this way you'll see how you can nest several layers of If conditions within a single macro.

Here's a chart showing the 2008 U.S. income tax rates for a single individual:

- 10% on income between \$0 and \$8,025
- 15% on the income between \$8,025 and \$32,550; plus \$802.50

- 25% on the income between \$32,550 and \$78,850; *plus* \$4,481.25
- 28% on the income between \$78,850 and \$164,550; *plus* \$16,056.25
- 33% on the income between \$164,550 and \$357,700; *plus* \$40,052.25
- 35% on the income over \$357,700; *plus* \$103,791.75

Our macro must examine an income number, establish what income range the income amount falls in, and then calculate the appropriate income tax and place that amount in a cell.

It's easier to start this calculation at the top, the 35% tax rate, because then you can ask if the income is over a certain amount, whereas if you start at the bottom you have to determine if the income is within a certain range. You can structure the macro either way, but starting at the top results in fewer keystrokes.

We could follow the example set in the previous macro and have the macro examine the contents of a cell. Instead, let's use the techniques learned in Chapter 7, and use an `InputBox`. This way, we'll ask the worksheet user to enter the income to be analyzed, and then perform the tasks on the amount entered in the box.

The first portion of the code sets up the input box:

```
TaxableIncome = InputBox("Enter your taxable income")
If TaxableIncome > 357700 Then
    ActiveCell.Value = 103791.75 + (TaxableIncome - 357700) * 0.35
ElseIf TaxableIncome > 164550 Then
    ActiveCell.Value = 40052.25 + (TaxableIncome - 164550) * 0.33
ElseIf TaxableIncome > 78850 Then
    ActiveCell.Value = 16056.25 + (TaxableIncome - 78850) * 0.28
ElseIf TaxableIncome > 32550 Then
    ActiveCell.Value = 4481.25 + (TaxableIncome - 32550) * 0.25
ElseIf TaxableIncome > 8025 Then
    ActiveCell.Value = 802.50 + (TaxableIncome - 78850) * 0.15
ElseIf TaxableIncome > 0 Then
    ActiveCell.Value = TaxableIncome * 0.10
End If
```

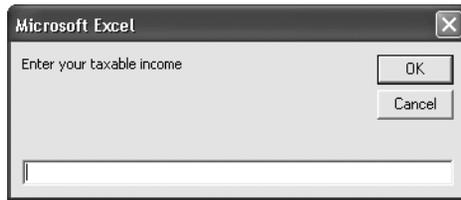


Figure 10-4 SingleTax macro uses an input box to request taxable income amount.

This macro gets placed between the Sub SingleTax() and End Sub lines, and you've got a complete macro. Test the macro by opening a worksheet and running the SingleTax macro. A dialog box like the one in Figure 10-4 appears, asking you to input your taxable income.

When you enter the amount of income and click OK, the correct income tax calculation appears in the currently active cell.

Create a Nested If/Then Macro

Sometimes one question isn't enough information to get you to the results you need. You can create a macro that asks more than one If question, and produce different layers of actions based on the answers.

For example, let's say we want to confirm that the taxable income used in the previous macro is really 2008 taxable income, so that the correct rates apply. Rather than just providing an input box that asks for the income, we can first ask for the income, and then ask for a confirmation that this is 2008 income. If the answer is Yes, the macro operation continues and the tax is calculated. If the answer is No, the macro execution stops and a message appears telling the user that tax rates for different years are not available.

The first line of the macro remains intact—the InputBox command asks the user for 2008 taxable income.

Next we need a new If statement, asking if this is really 2008 taxable income. The following code results in the message shown in Figure 10-5:

```
x = MsgBox("Is this your 2008 taxable income?", 3)
If x = 6 Then
```

Next you insert all of the macro code from the original SingleTax macro. It is wise to indent this code so that you can keep track of your If statements and the related End If statement. Finally, you need



Figure 10-5 Revised macro asks user to confirm that he entered 2008 taxable income.

MEMO

The message box codes (3 for a Yes/No box, and 6, which stands for the answer Yes) can be found in Chapter 7.



Figure 10-6 This message appears if the user indicates that 2008 income was not entered.

to provide for the possibility that the user did not enter a Yes answer in the message box. We'll take care of this contingency by providing an additional message box advising the user that the tax can't be calculated, as shown in Figure 10-6. If the user clicks OK (the only option), the box disappears and the macro operation ends.

```
Else
MsgBox ("Unable to calculate your income tax")
End If
```

The completed macro (named SingleTax2 to distinguish it from its predecessor) is shown in Figure 10-7.

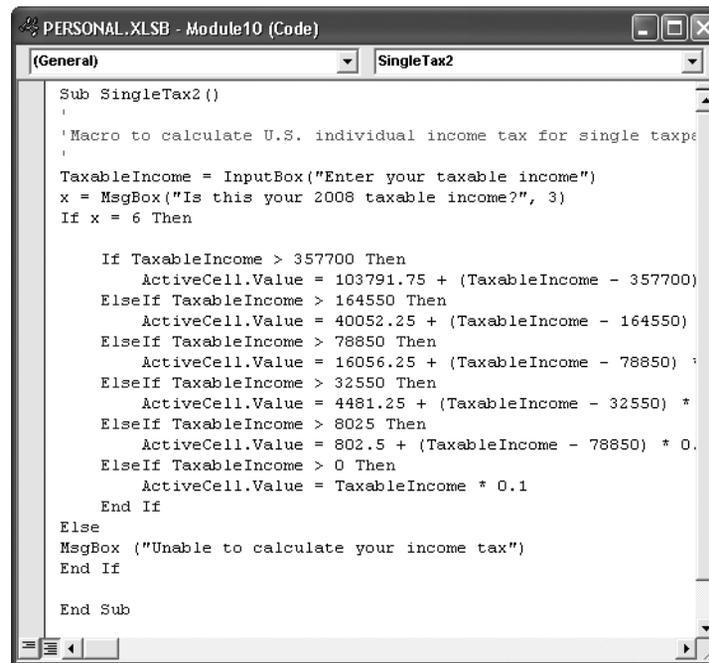


Figure 10-7 Complete code for the SingleTax2 macro