

Engineering Analysis & Problem Solving ENG1100

Creating Macros

Optional Text: Chapra, S.C. Power Programming With VBA/EXCEL. Prentice Hall, 2003.

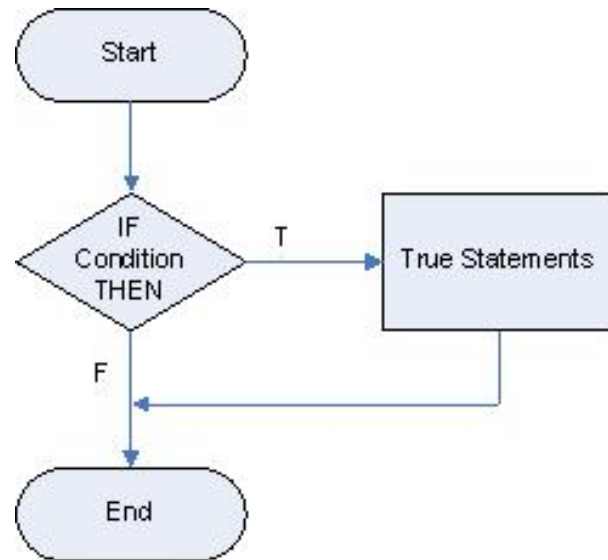
Session Objectives

- Function Review
- Create Spreadsheet Macros

If/Then Statements in VBA

- Do not need to include the **else** if it's not needed
- VBA syntax for an **If/Then** statement

If condition Then
True statements
End If



If/Then/Elseif Statements in VBA

- Can also have nested *If* statements
- VBA syntax for an *If/Then/Elseif* statement

If condition 1 Then

 True statements 1

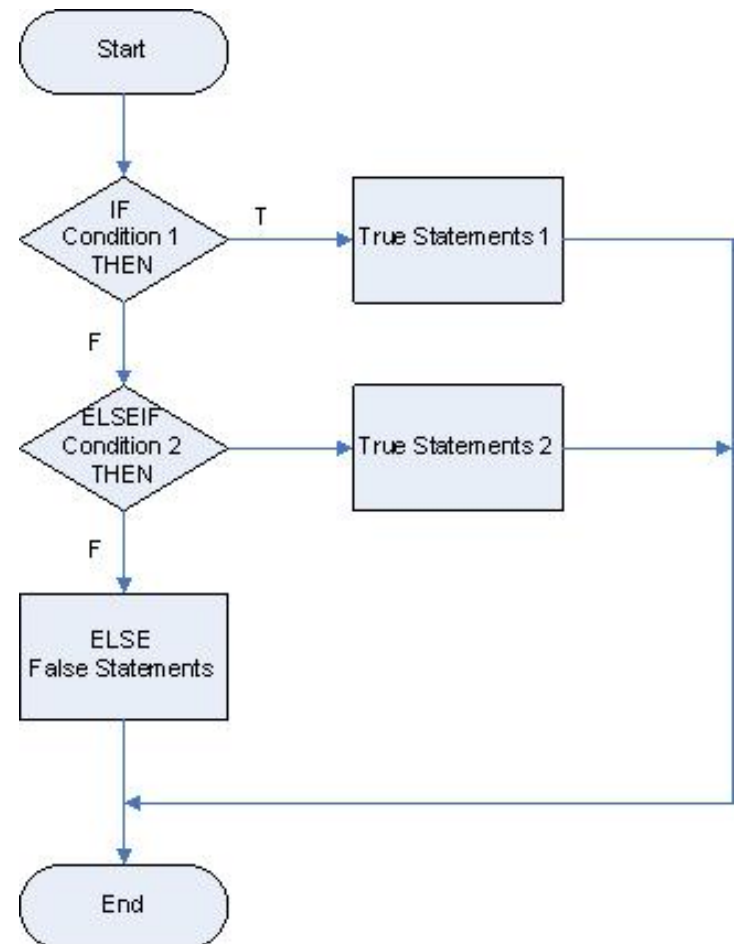
Elseif condition 2 Then

 True statements 2

Else

 False statements

End If



Recall – Parachute Problem

```
Function Vel(t, g, m, cd, vo, tc, cdp)
'Function to compute velocity of falling parachutist
'Name, Team #, Section #
't = time, s
'continue to define other arguments, including units

If t < tc Then
    'compute velocity for free fall
    Vel = vo*Exp(-cd/m*t)+g*m/cd*(1-Exp(-cd/m*t))
Else
    'compute velocity when parachute is first deployed
    vop = vo*Exp(-cd/m*tc)+g*m/cd*(1-Exp(-cd/m*tc))
    'compute velocity after parachute is deployed
    Vel = vop*Exp(-cdp/m*(t-tc))+g*m/cdp*(1-Exp(-cdp/m*(t-tc)))

End If
End Function
```

Loops

- Used for repetitive calculations
- For Loops
 - Continue loop for known number of times
 - Repetitively execute statements

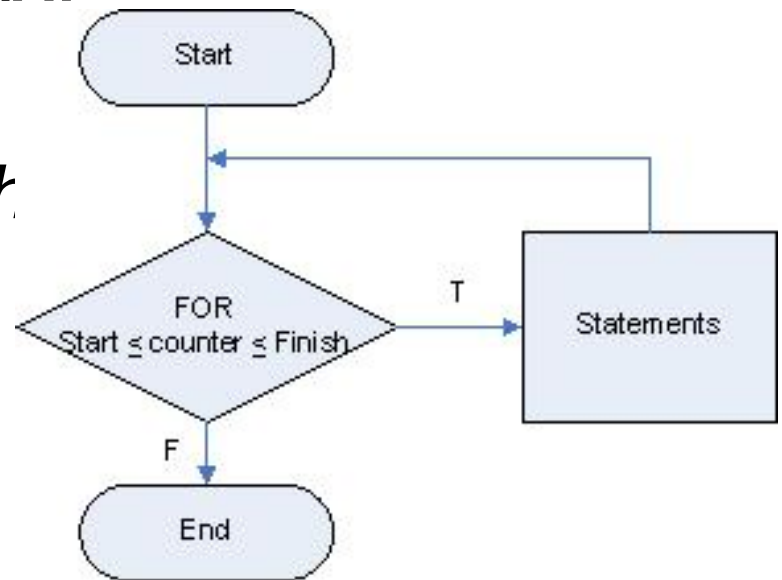
For/Next Loop

- If we know that the loop should run “n” times, there is a special loop to deal with these cases.
- For/Next loop VBA syntax:

For counter = start To finish.

Statements

Next counter



Review: For/Next Loop Options

- Alternative way of exiting the loop:

For counter = start To finish

Statements

If condition Then Exit For

Statements

Next counter

- The loop will terminate when it encounters an Exit For statement.
- Any number of Exit For statements may be placed in the loop.
- Exit For statements may be placed anywhere.

Recall Arrow Function

- Create a VBA function to calculate the maximum height of the arrow

```
Function arrow(t, yo, ay)
    'This is a function to calculate height and velocity of an arrow

    'Step 1. calculate time at maximum height

    tmax = t / 2

    'Step 2. calculate initial velocity using velocity equation (vy=v

    vo = -ay * tmax

    'Step 3. calculate maximum height of the arrow
    arrow = yo + vo * tmax + 0.5 * ay * tmax ^ 2

End Function
```

Functions Review

- We can now write our own functions in Excel
- Functions are good for...
 - Doing repetitive calculations
 - Returning a single value (answer) to the cell
- What if we want to do more?...

Writing Macros

- Ability to do many, many things including:
 - Formatting cells
 - Performing calculations
 - Grabbing data from cells
 - Putting data into cells
 - Display pop-up messages, dialog and input boxes
 - And more...
- Let's start with a simple example.

Factorial Program

- Open a new workbook and enter the following information:

Factorial Program	
Names	
Section #	
Team #	
Date	
Value	5
Factorial	

**Must be in
Cell B8**



Simple Factorial Program

- Open the VBE and a new module and type the following code:

```
Option Explicit
Sub Factorial()

'Names, Section #, Team #
'Date
'Variable Declaration
Dim n As Single, c As Single, initial As Single
Dim i as Integer
'input data
Sheets("Sheet1").Select
Range("B8").Select
n = ActiveCell.Value

Initial =1
'calculation
For i = 1 To n
    'compute factorial
    c = initial * i
    initial = c
Next i
'output results
Range("B9").Select
ActiveCell.Value = c

End Sub
```


Running The Program

- Go back to Excel
 - Go to: Tools – Macro – Macros
 - Select the macro you want (Factorial)
 - Click the run button
- Or..
 - Go to: View – Toolbars – Forms
 - In the forms toolbar select the “button” button (rectangle)
 - Size the button on your spreadsheet
 - In the Assign Macro dialogue box, select Factorial
 - Rename the button to something descriptive like “Run Factorial”

Simple Factorial Program

- Results...

	A	B	C	D
1	Factorial Program			
2				
3	Names			
4	Section #			
5	Team #			
6	Date			
7				
8	Final Value	5		
9	Factorial	120		
10				
11				



What is going on??

```
Option Explicit
Sub Factorial()

'Names, Section #, Team #
'Date
'Variable Declaration
Dim n As Single, c As Single, initial As Single
Dim i as Integer
'input data
Sheets("Sheet1").Select
Range("B8").Select
n = ActiveCell.Value

Initial =1
'calculation
For i = 1 To n
    'compute factorial
    c = initial * i
    initial = c
Next i
'output results
Range("B9").Select
ActiveCell.Value = c

End Sub
```

Forces you to define the type of all variables (good programming practice).

Declaring the macro program to be a subroutine program, "Factorial". The parentheses are for the routine's arguments. In this case, there are no arguments.

Defining the variable types

Declaring Variables

- Why declare variables?
 - Macros will run faster and use less memory
 - You will avoid problems with misspelled variable names
- VBA syntax for declaring variables:
`Dim varname As type, varname As type, ...`
- *Good practice to:*
 - *Place all your Dim statements at the beginning of the program.*
 - *Group them by type.*

```
Option Explicit  
Sub Example()
```

```
'Variable declaration
```

```
Dim i As Integer, n As Integer
```

```
Dim t As Long
```

```
Dim x As Single, v As Single
```

```
Dim z As Double
```

```
.  
. .  
. .
```

Numeric Variable Types

- **Integers** (whole numbers used for counting)
 - **Integer** type (range from -32,768 to 32,757 or 2 bytes)
 - **Long** type (range from -2,147,483,648 to 2,147,483,647 or 4 bytes)
- **Real or floating-point** (decimal numbers used for measuring)
 - **Single** type (about 7 significant digits of precision or 4 bytes)
 - Good for most engineering and scientific calculations
 - **Double** type (about 15 significant digits of precision or 8 bytes)

Declaring your Data Type

- Your code will work if you declare the variables like this, but not as you may expect...

```
Dim x, v As Single
```

Variant Data Type

- Your code will work if you declare the variables like this, but not as you may expect...
`Dim x, v As Single`
- VBA will make the following assignments:
 - v is a single type
 - x is a variant type
 - The variant type allows the macro to choose which data type it should be.
- Warning: this will slow down your macro.

Declaring your Data Type

- If you want both x and v to be single instead of:

~~Dim x, v As Single~~

Use:

Dim x As Single, v As Single

What is going on??

```
Option Explicit  
Sub Factorial()
```

```
'Names, Section #, Team #
```

```
'Date
```

```
'Variable Declaration
```

```
Dim n As Single, c As Single, initial as Single
```

```
Dim i as Integer
```

```
'input data
```

```
Sheets("Sheet1").Select
```

```
Range("B8").Select
```

```
n = ActiveCell.Value
```

```
Initial =1
```

```
'calculation
```

```
For i = 1 To n
```

```
    'compute factorial
```

```
    c = initial * i
```

```
    initial = c
```

```
Next i
```

```
'output results
```

```
Range("B9").Select
```

```
ActiveCell.Value = c
```

```
End Sub
```

Grabbing data from your spreadsheet and assigning the values to variables

Outputting value of variable "c" to cell B9.

These are Object-Oriented Programming (OOP) statements

More Useful OOP Statements

- Clear a section of the worksheet
 - `Range("t3:z40").ClearContents`
- Move the current active cell
 - `ActiveCell.Offset(0,1).Select`
 - Moves the active cell 0 rows, 1 column to the right
 - `ActiveCell.Offset(1,0).Select`
 - Moves the active cell 1 row down, 0 columns