Creating Macros

Last Time

- For Loops
Registration for ENG1101 and ENG1102 Cohorts

- Open Friday 3/2/07 Preview Only
- Open Tuesday 3/6/07 8AM-3:30PM athletes (Dillman 112)
- Open Wednesday 3/7/07 10PM everyone
- Close Friday 3/9/07 Noon
- Select Cohort based on current Engineering AND current Math courses
- See Tina in Dillman 112 with any problems

- Preview at website  db.geneng.mtu.edu/cohorts/

- Register on normal BanWEB  www.banweb.mtu.edu

- Look at other required courses – select a cohort that allows room for those courses
- Cohorts can not be changed once regular registrations starts
Registration for ENG1101 and ENG1102 Cohorts

- Preview at website  db.geneng.mtu.edu/cohorts/
  - Use this site to select your favorite cohorts
  - (Please Note: not all listed cohorts are open, check the seating capacity)
  - Open ENG1101 cohorts = 2,3,7,13,16,18 21,30
  - Open ENG1102 cohorts = A,B,C,D,E,F,J,K,L,M,N

- Register on normal BanWEB  www.banweb.mtu.edu
Registration for ENG1101 and ENG1102 Cohorts

● Once you Previewed

● On March 7, 10pm Go to BanWEB

  www.banweb.mtu.edu

  – Sign in
  – Select Fall07 for the term
  – Select Engineering Fundamentals
  – Course Number “0000”

  – Then Click on the CRN which matches your choice of cohort
More on cohort registration

- You will still need to register for additional classes during normal registration.
- Do not drop or change sections of cohort classes as the entire group of classes will be dropped.
- Plan which other classes you want before you select a cohort.
- Have several alternatives ready.
- Cohort registration will help you and is based on your current math class. Pick the correct one!
- If you sign up for a cohort with a class you don’t need (example: you have already had physics) see Tina to drop that course.
As a Team

Create a flowchart and write a new function using a `for/next loop` that will determine the amount of money in your account after 10 years if you invest $4000 initially and deposit $1000 at the end of each year for the each of the following cases:

4.5% Annual compounded annually
5.0% Annual compounded annually
5.5% Annual compounded annually
## Check your Work

Problem: Determine the amount after 10 years with an initial deposit of 4000 and an additional 1000 at the end of each year for the annual interest rates of 4.5%, 5.0%, and 5.5%

<table>
<thead>
<tr>
<th>Principle</th>
<th>p</th>
<th>4000</th>
<th>4000</th>
<th>4000</th>
</tr>
</thead>
<tbody>
<tr>
<td>deposit</td>
<td>d</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>rate</td>
<td>i</td>
<td>0.045</td>
<td>0.05</td>
<td>0.055</td>
</tr>
<tr>
<td>years</td>
<td>yr</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>amount after 10yrs</td>
<td>18500</td>
<td>19093</td>
<td>19708</td>
<td></td>
</tr>
</tbody>
</table>
Function interest(P, D, I, yr)
'P=principle
'D=yearly deposit
Dim j As Integer
For j = 1 To yr
    P = P * (1 + I) + D
Next j
interest = P
End Function
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.
As a Team

- Insert a break into your financial calculator when you reach your goal of $15,000.
Check your work

For Loop with break when target is reached

Problem: Determine the amount after 10 years with an initial deposit of 4000 and an additional 1000 at the end of each year for the annual interest rates of 4.5%, 5.0%, and 5.5%

<table>
<thead>
<tr>
<th>Principle</th>
<th>p</th>
<th>4000</th>
<th>4000</th>
<th>4000</th>
</tr>
</thead>
<tbody>
<tr>
<td>deposit</td>
<td>d</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>comp/year</td>
<td>n</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>rate</td>
<td>i</td>
<td>0.045</td>
<td>0.05</td>
<td>0.055</td>
</tr>
<tr>
<td>years</td>
<td>yr</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>target</td>
<td></td>
<td>15000</td>
<td>15000</td>
<td>15000</td>
</tr>
<tr>
<td>amount after 10yrs</td>
<td>15068.41607</td>
<td>15458.9</td>
<td>15860.3</td>
<td></td>
</tr>
</tbody>
</table>

When is your target reached?
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:

<table>
<thead>
<tr>
<th>Factorial Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names</td>
</tr>
<tr>
<td>Section #</td>
</tr>
<tr>
<td>Team #</td>
</tr>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Factorial</td>
</tr>
</tbody>
</table>

Must be in Cell B8

Simple Factorial Program

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

```vba
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...

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Factorial Program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Names</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Section #</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Team #</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Final Value</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Factorial</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is going on??

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:
  
  ```vba
  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.

```vba
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
```

```vba
. . .
```
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...
  
  ```vba
  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
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
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
As a Team

- Convert your interest function to a macro
- Have the program return the amount and the time when the target has been exceeded
Today

- Macros
  - Allow the return of more than one variable
Individual Homework
DUE: Week 9, Session A

- Convert your fibonacci series function to a macro that reads “N” from your spreadsheet and returns the first N terms of the Fibonacci series
Factorial Function

- What does this do to a loop function?

```
For i = 1 To n
    Factorial = Factorial * i
Next i
```

```
i = 1
Do
    If i > n Then Exit Do
    Factorial = Factorial * i
    i = i + 1
Loop
```

Do/If Exit Loop

For/Next Loop