VBA: Tutorial Week 3

- Non-linear, non-sequential programming using branches and loop
- Nesting: branches and loops
- Branching, looping and the `InlineShapes` collection
- Option Explicit
- Using the VBA debugger

Official resource for MS-Office products: https://support.office.com

Microsoft Introduction/Overview Of VBA

- https://docs.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office

VBA tutorial notes by James Tam

First Tutorial (Monday or Tuesday)

Activities In Tutorial

- TA demos:
  - Used for more complex features (typically multiple steps are required).
  - The tutorial instructor will show on the projector/instructor computer each step for running the feature in Excel.
  - Unless otherwise specified the tutorial material will take the form of a TA demonstrating the use of features in Excel.
  - Slides titled “Lecture Review” are covered for the second time and dealing with less complex material.
    - For this reason they will only be covered briefly in tutorial.
- Student exercises:
  - Used instead of TA demos for simpler features.
  - You will have already been given a summary of how to invoke the feature and the purpose of the exercise is to give you a chance to try it out and get help if needed.

VBA tutorial notes by James Tam

Student Exercise #1

- (You may need to review the lecture notes on collections before trying this exercise)
- Using the starting document called "`1sorting_table_starting`" write a VBA program that will sort the first table (ascending order, assumes there is a header) if the document contains more than one table.
- If the cut-off for the number of tables hasn't been met then the program should display a popup message box with a brief description of why sorting didn't occur as well as the number of tables in the document.
- (JT's comment: test your program by seeing what happens if it contains: no tables, 1 table, 2 or more tables).
- Document containing the solution: `1sorting_table_solution`

VBA tutorial notes by James Tam

Branching: Alternate Courses Of Execution

- What you will know from lecture:
  - Branching allows for alternative courses of execution.
  - Each alternative executes one or more VBA instructions.
  - Branching can be implemented in different ways depending upon the programming language what you will have learned is variations of the `IF` structure.

VBA tutorial notes by James Tam

IF-THEN ; IF-THEN, ELSE

- A Boolean expression (results in either a true or false value) will determine the instruction or instructions that will execute.
- IF-THEN executes an instruction or instructions (body of the IF) when the Boolean expression evaluates to true.
  - E.g.

    ```
    If (true) Then
    ```

    - The above examples always executes the body because the Boolean expression is always true (the expression is a constant value that is true).
- IF-THEN, ELSE reacts for both the true and the false cases.
  - E.g.

    ```
    If (num >= 0) Then
    Else
    ```

    - Executes the IF-body when num is positive and the Else-body when num is not positive.

VBA tutorial notes by James Tam

Branches: Depending Upon The # Of Images

- **Name of the document containing example**: 1ifElse
- Features: checks if the number of images ('InlineShapes' in VBA) is above the defined cut-off value).
  - ifThenExample: The first program reacts (popup appears) if the cut-off has been met.
  - ifThenElseExample: The second program reacts one way if the cut-off has been met (popup appears) and another way (different popup appears)
  - After the branch has been completed both programs will then execute any remaining instructions after the branching structure (after the 'End If')
  - Also:
    - shows an example of defining a named constant (specifies the cut-off value),
    - shows the use of the line continuation character (string argument for the MsgBox in the body of the ELSE-branch in the second program).

VBA tutorial notes by James Tam

Branches: Depending Upon The # Of Images IF-Then Version

```
' First program (IF-THEN)
Sub ifThenExample()
    Const CUT_OFF As Long = 2
    Dim numShapes As Long
    numShapes = ActiveDocument.InlineShapes.Count
    If (numShapes > CUT_OFF) Then
        MsgBox (">" & CUT_OFF & " pics in active Word doc")
    End If
    MsgBox ("Branching structure over: End program")
End Sub
```

VBA tutorial notes by James Tam

Branches: Depending Upon The # Of Images IF-Then, Else Version

```
' Second program (IF-THEN, ELSE)
Sub ifThenElseExample()
    Const CUT_OFF As Long = 2
    Dim numShapes As Long
    numShapes = ActiveDocument.InlineShapes.Count
    If (numShapes > CUT_OFF) Then
        MsgBox (">" & CUT_OFF & " pics in active Word doc")
    Else
        MsgBox ("# pics didn't meet the cutoff of " & CUT_OFF & _
         " pics " & " required ")
    End If
    MsgBox ("Branching structure over: End program")
End Sub
```

VBA tutorial notes by James Tam

Logic And Branching

- Recall how the logical functions, AND() OR(), can be combined with an IF-function in Excel.
  - E.g. (hires U of C graduates with a GPA of 3.3 or higher)
    `IF(AND(A1>3.3,B1="UC", "Hire", "")`
- With programming languages the structure is slightly different, Boolean expressions are chained or connected with AND OR logical operators.
  - **Format**:
    - `If ((Boolean expression 1) <Logical operator> (Boolean expression 2)…) then`
  - **Example**:
    - `If ((age < 0) Or (age > 114)) Then`
  - `(More than 2 Boolean expressions can be evaluated by the use of additional logical operators):`
    - `If ((age < 0) Or (age > 114) Or (hair = "mullet")) Then`

VBA tutorial notes by James Tam

Logic & Branching

- **Name of the document containing example**: `2branchingLogic`
- Features: Error checks user input using branching (both branches produce similar results)
  - First branch example: Employs logical OR (checks if it's true age is outside the valid range).
    - Shows an error message if it's true that the age is outside the allowable range.
    - Shows a confirmation message if it's false that the age is outside the allowable range (i.e. the age is okay)
  - Second branch example: Employs logical AND (checks if it's true age is inside the valid range)
    - Shows a confirmation message if it's true that the age is inside the allowable range
    - Shows an error message if it's false that the age is inside the allowable range (i.e. the age is not okay)
  - Both branches produce the same popup given the same user input (student: verify this for practice).

VBA tutorial notes by James Tam

Logic & Branching: Error Checking Age

- VBA instructions needed for both branches

```
Dim age As Long
age = InputBox("Age (0 - 114)?")
```


Error Checking A Value: IF With OR

```
If ((age < 0) Or (age > 114)) Then
    MsgBox ("OR: Age is outside allowable range of 0 - 114")
Else
    MsgBox ("OR: Age of " & age & " is OK")
End If
```

Error Checking A Value: IF With AND

```
If ((age >= 0) And (age <= 114)) Then
    MsgBox ("AND: Age of " & age & " is OK")
Else
    MsgBox ("AND: Age is outside allowable 0 - 114")
End If
```


Checking Multiple Conditions

- There's two general cases:
  - Zero or one of the conditions is true (no more than one so having one true case excludes the possibility of any other cases being true).
    - Example: getting a letter grade for a class during a particular semester, specifying the current city, town that you reside in.
    - VBA structure to use: IF-ELSEIF
  - Zero, one, two up to all of the cases can be true.
    - Example: for each class taken checking if a perfect score was awarded (letter grade 'A'), checking if a person has ever lived in each of the cities, towns in a particular country (Have you ever lived in Calgary? Have you ever lived in Edmonton? Etc.)
    - VBA structure to use: Multiple and independent IFs

IF-ELSEIF, Multiple IFs

- **Name of the document containing example**: `3multipleIfVSIFElseIF`
- Features:
  - Part I: Check for birth city
    - Prompts the user for the city that person was born in an reacts in different ways based on that information. One approach uses an `IF-ELSEIF` structure, the other employs multiple and separate `IF` structures.
    - Note: the use of the multiple IFs is not an appropriate solution in cases such as this but is included for learning purposes (to show how difficult it can be to check for the 'none of the above' case).
  - Part II: Check education level and if the person is a senior citizen
    - These two checks are independent, education level and the senior check are unrelated.
    - (Alternatively phrased): regardless of what the user enters for the years of education, the program will always check if the person is or is not a senior.

VBA tutorial notes by James Tam

Multiple Conditions: Checking City Of Birth

- Solution using multiple `IF`s

```
Dim birthCity As String
birthCity = InputBox("City of birth")
If (birthCity = "Calgary") Then
    MsgBox ("You are 'Part of the Energy'")
End If
If (birthCity = "Edmonton") Then
    MsgBox ("From the City of Champions")
End If
If (birthCity = "Dubai") Then
    MsgBox ("Definitely Dubai!")
End If
If (birthCity = "Fargo") Then
    MsgBox ("You're always warm")
End If
```

VBA tutorial notes by James Tam

Multiple Conditions: Checking City Of Birth (2)

```
' When none of the cases in any of the previous IFs apply
' requires a very awkward solution
If ((birthCity <> "Calgary") And _
    (birthCity <> "Edmonton") And _
    (birthCity <> "Dubai") And _
    (birthCity <> "Fargo")) Then
      MsgBox ("Multiple-IFs: Miscellaneous place")
End If
```


Multiple Conditions: Checking City Of Birth (3)

- Solution using `IF-ELSEIF` (better approach)

```
If (birthCity = "Calgary") Then
    MsgBox ("You are 'Part of the Energy'")
ElseIf (birthCity = "Edmonton") Then
    MsgBox ("From the City of Champions")
ElseIf (birthCity = "Dubai") Then
    MsgBox ("Definitely Dubai!")
ElseIf (birthCity = "Fargo") Then
    MsgBox ("You're always warm")
Else
    MsgBox ("IF-ELSEIFs: Miscellaneous place")
End If
```

Multiple Conditions: Education Level, Senior Citizen

```
' Check for grade level and if senior have nothing to do
' with each other, both checks must always occur so using multiple
'  Ifs is appropriate.
Dim gradeLevel As Long
Dim age As Long
gradeLevel = InputBox("What is your highest grade level: ")
age = InputBox("What is your age: ")
If (gradeLevel >= 13) Then
    MsgBox ("College person!")
End If
If (age >= 65) Then
    MsgBox ("Senior citizen")
End If
```

VBA tutorial notes by James Tam

Multiple Conditions: Education Level, Senior Citizen (Solution With Bug)

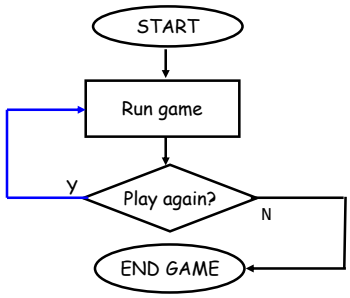
- Student exercise: find the bug in the following program.

```
Dim gradeLevel As Long
Dim age As Long
gradeLevel = InputBox("What is your highest grade level: ")
age = InputBox("What is your age: ")
If (gradeLevel >= 13) Then
    MsgBox ("College person!")
ElseIf (age >= 65) Then
    MsgBox ("Senior citizen")
End If
```

VBA tutorial notes by James Tam

Looping/Repetition

- Used when a part (or the entire) program needs to repeat as long a condition has been met.




```
Do while
(Condition)
    Instruction(s)
Loop
```

- The condition is a Boolean expression.

Example: Counting Program (Up/Increases)

- **Name of the document containing example**: 4loopV1Up
- Features:
  - The program will iterate (count) through the sequence of numbers 1 – 11 in increments of 2 (1, 3, 5, 7, 9, 11)

```
Sub countingLoopV1Up()
    Dim i As Long
    i = 1
    Do While (i <= 11)
        MsgBox ("i=" & i)
        i = i + 2
    Loop
End Sub
```

  - Student exercise: what if the Boolean expression was changed to `(i <= 10)`?

Example: Counting Program (Down/Decreases)

- **Name of the document containing example**: `5countingloopV2Down`
- Features:
  - The program will iterate (count) through the sequence of numbers 10 – 1 in decrements of 3 (10, 7, 4, 1)

```
Sub nineLoopV2()
   Dim i As Long
   i = 10
   Do While (i >= 1)
       MsgBox ("i=" & i)
       i = i - 3
   Loop
End Sub
```

  - Student exercise: what if the Boolean expression was changed to `(i >= 0)`?

VBA tutorial notes by James Tam

Student Exercise

- Write a program that will, using a loop, display all the multiples of 5 in the range from 5 – 15,625.
- The program will display each multiple of 5 within this range in a `MsgBox` one-at-a-time.
- **Document containing the solution**: `2multiples_of_five_solution`

VBA tutorial notes by James Tam

Nesting: Branches And Loop

- Branches and loops can be nested within each other

Scenario 1

```
If (Boolean) then
    If (Boolean) then
        ...
    End if
End if
```

Scenario 2

```
Do while (Boolean)
   If (Boolean) then
       ...
   End if
Loop
```

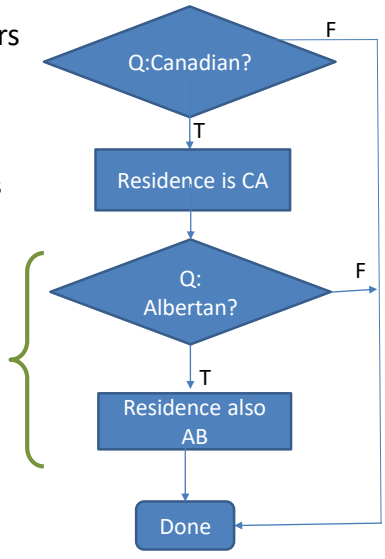
Scenario 3

```
If (Boolean) then
    Do while (Boolean)
        ...
    Loop
End if
```


Recognizing When **Nesting** Is Needed

- **Scenario 1**: Only if a question answers true then check if another question answers true or false.
  - Example: If the user entered Canada as country of residence then ask if the user's province of residence is Alberta.
  - Type of nesting: a IF-branch nested inside of an IF-branch

```
If (Boolean) then
    If (Boolean)
        ...
    End If
End If
```



(Key Part: IF) Nested Inside An IF

- Nesting: a structure (e.g. IF) is nested inside of another structure (e.g. IF) when the second structure is part of the body of the first structure.
- **Word document containing the example**: 6nesting_branch_within_branch

```
'Some parts excluded for brevity.
country = InputBox("Current country of residence: ")
If (country = "Canada") Then
    message = message & "Great country, "
    province = InputBox("Current province of residence: ")
    If (province = "AB") Then
        message = message & " Greatest place on earth ^-*"
    End If 'Checking province
End If 'Checking country
```

- Recall: the check for the Boolean expression for the second IF does not occur unless the first Boolean expression is true. (Don't bother checking if province is AB if country isn't Canada).

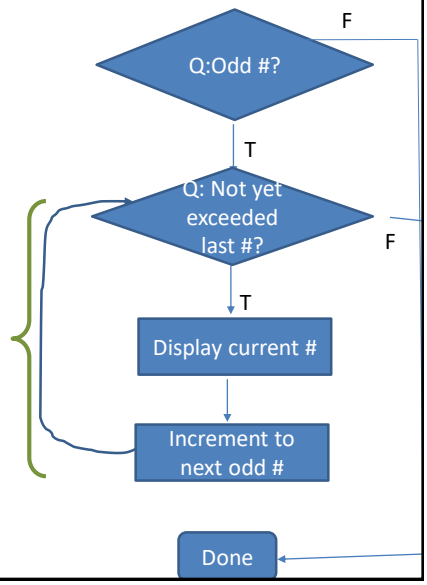
VBA tutorial notes by James Tam

Second Tutorial (Wednesday or Thursday)

Recognizing When **Nesting** Is Needed

- **Scenario 2**: If a question answers true then check if a process should be repeated.
  - Example: If the user entered an odd number then count through a sequence 1 to this number and display each odd number in this sequence.
  - Type of nesting: a `Do-While` loop nested inside of an `IF`-branch

```
If (Boolean) then
    Do While (Boolean)
        ...
    Loop
End If
```



(Key Part: Do-While) Nested Inside An IF

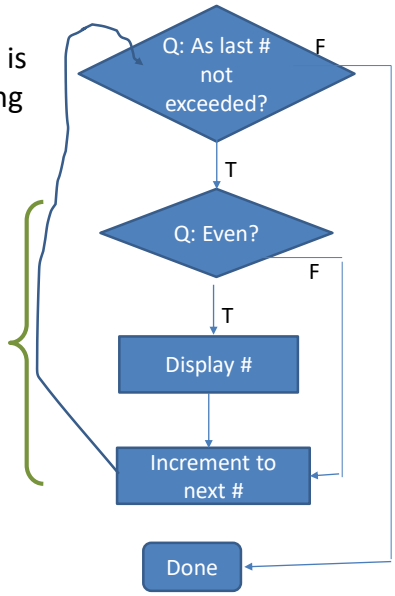
- **Word document containing the example**: 7nesting_loop_in_branch

```
'Variable & constant declaration excluded for brevity
lastOdd = InputBox("Enter last odd number in sequence: ")
remainder = lastOdd Mod 2
If (remainder = 0) Then
    MsgBox (lastOdd & " is even not odd.")
Else
    If (lastOdd <= MAX_ODD) Then
        count = 1
        Do While (count <= lastOdd)
            MsgBox ("Current number = " & count)
            count = count + 2
        Loop
    End If 'End: checks size of last #
End If 'End: checks if # is odd or even
```

Recognizing When **Nesting** Is Needed

- **Scenario 3**: As long some condition is met a question will be asked. As long as some condition is met a popup will be displayed.
  - Example: While the last number in a sequence hasn't been exceeded if the current number is even it will be displayed.
  - Type of nesting: an `IF`-branch nested inside of a `Do-While` loop

```
Do While (Boolean)
    If (Boolean) then
        ...
    End If
Loop
```



(Key Part: `IF` Nested) Inside A `Do-While`

- **Word document containing the example**: `8nesting_branch_in_loop`

```
Const MAX_NUMBER As Long = 20
Dim lastNumber As Long
Dim count As Long
Dim remainder As Long
lastNumber = InputBox("Enter last number in a sequence: ")
If (lastNumber <= MAX_NUMBER) Then
    count = 1
    Do While (count <= lastNumber)
       remainder = count Mod 2
       If (remainder = 0) Then
           MsgBox ("Current  even #: " & count)
       End If
       count = count + 1
    Loop
End If
```

Return To Collections

- Recall with the collections you have seen: `Documents`, `InlineShapes`, `Shapes`, `Tables` you can access a particular element or item in the collection by that item's index.
  - Example (accesses the first `InlineShape`): `ActiveDocument.InlineShapes(1)`
- Also the number of items in the collection can be accessed through the collection's count attribute.
  - Example (the variable `numTables` will contain the current number of tables in the currently active Word document):

```
numTables =  ActiveDocument.Tables.count
```

VBA tutorial notes by James Tam

Return To Collections (2)

- Now that you have learned how to use looping and branching structures, you can:
  - Access each item in a collection (using a loop).
  - Check if the number of items in the collection is the desired amount (using a branch).

VBA tutorial notes by James Tam

Collections: Inline Shapes

- **Example program**: 3_loop_branch_inline_shapes
  - For documents containing 2 - 4 InlineShapes (images) the program will halve the size of odd numbered images.

```
Sub reduceOddInlineShapes()
    Const MIN_SHAPES As Long = 2
    Const MAX_SHAPES As Long = 4
    Dim count As Long
    Dim numShapes As Long
    Dim tempWidth As Long
    numShapes = ActiveDocument.InlineShapes.count
    If ((numShapes < MIN_SHAPES) Or (numShapes > MAX_SHAPES)) Then
        MsgBox ("Number of Inline Shapes not " & MIN_SHAPES & _
          "-" & MAX_SHAPES)
```


Collections: Inline Shapes (2)

```
    Else
        count = 1
        Do While (count <= numShapes)
            If ((count Mod 2) = 0) Then
                tempWidth = ActiveDocument.InlineShapes(count).Width / 2
                ActiveDocument.InlineShapes(count).Width = tempWidth
            End If
            count = count + 1
        Loop
    End If
End Sub
```

Student Exercise 3

- Write a program that will prompt the user for a positive integer value (1 or greater).
- The program will then double the size of the item # of the `Inline Shape` in the currently active document.
- **Name of the document containing the solution**: `3enlarge_A_pic_solution`

Student Exercise 4

- Modify the solution to the previous exercise so that the program error checks the user's input.
- If the value enter by the user is less than 1 or it exceeds the current number of In line shapes in the document:
  - The program will display an error message specifying the correct range of values that can be entered (it needs to be based on the actual number of shapes in the currently active document).
  - The program will repeat the prompt until a value within the correct range has been entered.
- **Name of the document containing the solution**: `4enlarge_A_pic_solutionV2`

Counting Occurrences Of A Word

- It's an application of the '`Find`' method of the `ActiveDocument` object combined with looping.
- Why count occurrences:
  - Evaluating resumes by matching skills sought vs. skills listed by the applicant.
  - Ranking the relevance of a paper vs. a search topic by the number of times that the topic is mentioned.
    - Word frequency may be one criteria employed when websites rank search results according to relevance

VBA tutorial notes by James Tam

Checking Occurrences

- **Word document containing the macro** (actually it checks if word is or isn't found rather than doing an actual count but a small modification will allow a count to be performed):
- `10determine_if_word_occurs`

```
Sub checkingOccurence()
    Dim occurs As Boolean
    Dim searchWord As String
    searchWord = InputBox("Word to search for")
    occurs = False
    With ActiveDocument.Content.Find
        Do While .Execute(FindText:=searchWord, Forward:=True, _
          MatchWholeWord:=True) = True
            occurs = True 'Word was found change state
        Loop
    End With
```

VBA tutorial notes by James Tam

- True: search for exact word
- False: partial match counted e.g. when looking for 'the' words like 'there' are counted

Checking Occurrences

Search not started, assume Word not in document

- **Word document containing the macro** (actually it checks if word is or isn't found rather than doing an actual count but a small modification will allow a count to be performed):
- 10determine_if_word_occurs

```
Sub checkingOccurence()
    Dim occurs As Boolean
    Dim searchWord As String
    searchWord = InputBox("Word to search for")
    occurs = False
    With ActiveDocument.Content.Find
        Do While .Execute(FindText:=searchWord, Forward:=True, _
          MatchWholeWord:=True) = True
            occurs = True
        Loop
    End With
```

Word to find in document

Body of Do-While entered when a match occurs (in this case can set variable to indicate that it's true that wo was found)

Checking Occurrences

- Once the search is complete display the results of the search

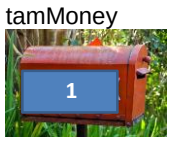
```
    If (occurs = True) Then
        MsgBox ("'" & searchWord & "'" & " was found")
    Else
        MsgBox ("'" & searchWord & "'" & " could not be found")
    End If
End Sub
```

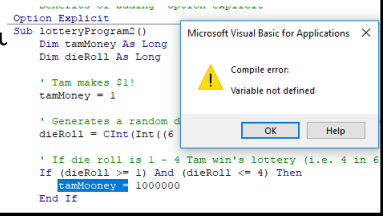
Student Exercise 3

- Modify the previous program. Instead of determining if the search word was or was not found have your program count the number of occurrences.
  - A word should be counted if it's a partial match e.g. when search for 'the' the words 'the', 'their', 'they're' and 'there' should all be counted.
- After the search is complete the number of occurrences should be displayed in a popup
- **Name of the document containing the solution**: `exercise3`
  - Example data used to test the correctness of your solution.
  - Search for 'the', count should be 2
  - Search for 'at', count should be 2
  - Search for 't', count should be 4

Option Explicit Used

- Including 'Option Explicit' requires that variables must be created via 'Dim' variable declaration
  - E.g. `Dim tamMoney As Long`
- After creating/declaring the variable the memory location can be used by assigning a value into that location.
  - E.g. `tamMoney = 1`
- Advantage: helps catch bugs
  - If you type in the wrong variable name if you use Option Explicit then VBA may tell you exactly where the error lies.





VBA tutorial notes by James Tam

Example: Option Explicit Used

VBA will automatically catch the error and point out the location

- **Example:** 6A_optionExplicitUsed.docm

```
Option Explicit
Sub lotteryProgram2()
    Dim tamMoney As Long
    Dim dieRoll As Long

    ' Tam makes $1!
    tamMoney = 1

    ' Generates a random dice roll (value returned in range of 1 - 6
    dieRoll = CInt(Int((6 * Rnd()) + 1))

    ' If die roll is 1 - 4 Tam win's lottery (i.e. 4 in 6 or 75% chance to win)
    If (dieRoll >= 1) And (dieRoll <= 4) Then
       tamMooney = 1000000
    End If

    MsgBox ("Tam's income $" & tamMoney)
End Sub
```

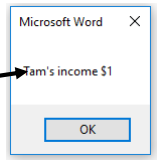

Example: Option Explicit Not Used

- **Example:** 6B_optionExplicitNotUsed.docm

```
Sub lotteryProgram2()
    Dim tamMoney As Long
    Dim dieRoll As Long
    tamMoney = 1
   dieRoll = CInt(Int((6 * Rnd()) + 1))
   If (dieRoll >= 1) And (dieRoll <= 4) Then
       tamMooney = 1000000
    End If
    MsgBox ("Tam's income $" & tamMoney)
End Sub
```

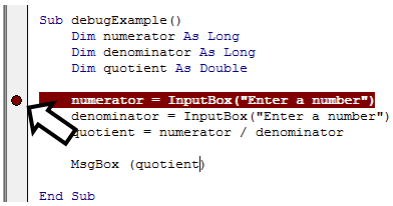
The program erroneously set the wrong variable!

Tam didn't get the "big bucks" ☹
Errors like this can be hard to catch/fix in all but smallest programs



The VBA Debugger

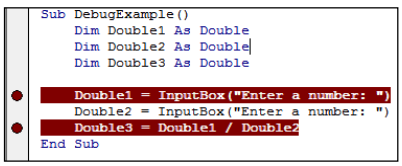
- Debuggers can be used to help find errors in your program
- Setting up breakpoints
  - Points in the program that will 'pause' until you proceed to the next step
  - Useful in different situations
    - The program 'crashes' but you don't know where it is occurring
      - Pause before the crash
    - An incorrect result is produced but where is the calculation wrong
- Set up breakpoints
  - Click in the left margin

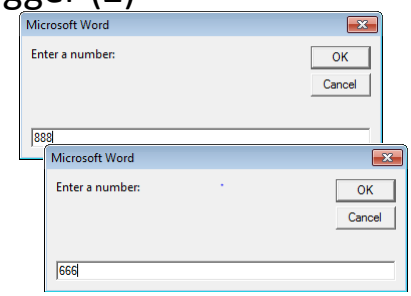


VBA tutorial notes by James Tam

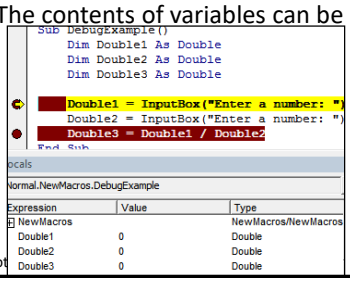
The VBA Debugger (2)

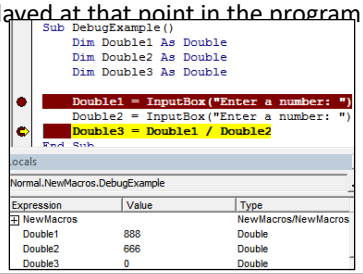
- Multiple breakpoints





- Program pauses when breakpoints are reached
  - The contents of variables can be displayed at that point in the program

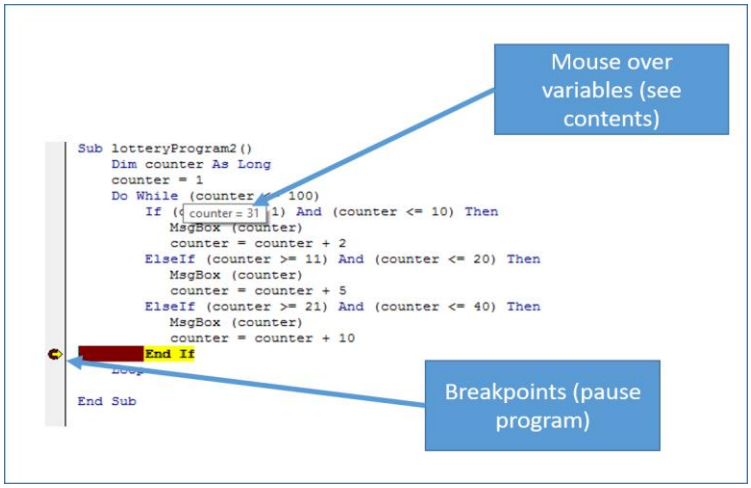




VBA tutorial not

The VBA Debugger (3)

- Combining breakpoints and viewing variables.



An Example To Run With The Debugger

- **Example:** `7debuggerExample.docm`
  - `Sub: DebuggingExample1`

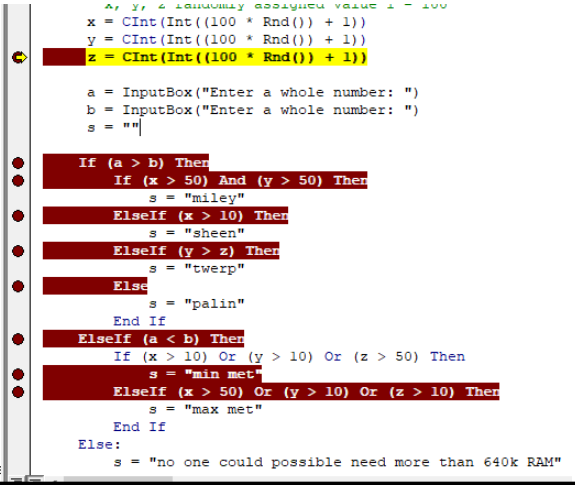
Set up a breakpoint and trace through the program step-by-step while viewing the contents of the loop control during each iteration of the loop

```
Sub debuggingExample1()
    Dim counter As Long
    counter = 1
    Do While (counter <= 100)
        If (counter = 5) And (counter <= 10) Then
            MsgBox (counter)
            counter = counter + 2
        ElseIf (counter >= 11) And (counter <= 20) Then
            MsgBox (counter)
            counter = counter + 5
        ElseIf (counter >= 21) And (counter <= 40) Then
            MsgBox (counter)
            counter = counter + 10
        End If
    Loop

End Sub
```

An Example To Run With The Debugger (2)

- **Example (cont'):** `7debuggerExample.docm`
  - `Sub: DebuggingExample2`



```
    x = CInt(Int((100 * Rnd()) + 1))
    y = CInt(Int((100 * Rnd()) + 1))
    z = CInt(Int((100 * Rnd()) + 1))

    a = InputBox("Enter a whole number: ")
    b = InputBox("Enter a whole number: ")
    s = ""

    If (a > b) Then
        If (x > 50) And (y > 50) Then
            s = "miley"
        ElseIf (x > 10) Then
            s = "sheen"
        ElseIf (y > z) Then
            s = "twerp"
        Else
            s = "palin"
        End If
    ElseIf (a < b) Then
        If (x > 10) Or (y > 10) Or (z > 50) Then
            s = "min met"
        ElseIf (x > 50) Or (y > 10) Or (z > 10) Then
            s = "max met"
        End If
    Else:
        s = "no one could possible need more than 640k RAM"
```

Program randomly assigns value into x, y, z (1 – 100)
Set up multiple breakpoint and mouse over variables at the breakpoints to view their contents.

- **This time x = 71, y = 54, z = 1**
- **Which branches execute**
- **What values will be assigned to the string 's**

VBA tutoria