Week10

- Student exercises: loops and branches
- The value of “Option Explicit”
- Using the VBA debugger
- The ‘dir’ function
- Text processing and other useful functions/methods

Exercise 1

- Write a program that will prompt the user for the two numbers
- It will determine which number is lower and that lower will number will be the start of a sequence.
- The other number will be the end of the sequence increasing the value by 1 each iteration of the lop
  - E.g. the user first enters 15 and then enters 3. The program will then iterate through the sequence 3, 4, 5…15
  - E.g. the user first enters 0 and then enters 5. The program will then iterate through the sequence 0, 1, 2…5

Author: James Tam

Exercise 2

- Modify this program so that it doesn't increase the count by just one. The program will prompt the user by the amount that 'count' should be increased.
  - E.g. the user first enters 15 and then enters 3. As before the first number in the sequence will be 3 and the last number will be 15. The previous version of the program would increase count by 1 and iterate the following sequence: 1, 2, 3…14, 15
    - With this new version using the previous two inputs the user enters 3 for the 'count', then the program will iterate through the sequence 3, 6, 9, 12, 15
    - With this new version using the previous two inputs the user enters 5 for the count, then the program will iterate through the sequence 3, 8, 13. (Since 13 + 5 = 18 which exceeds the maximum bound in the sequence the last number that the program will count through is 13).

Author: James Tam

Exercise 3

- Write a new program that will prompt the user for the current year and error check the input.
  - The program prompts the user for the current year in the form of a numeric value e.g. 2017.
  - If the year is not one that is within the 21st century (2000 – 2099) then the program will display an appropriate and helpful error message (e.g. "Year must be in the range of 2000-2099") and set the year to the default year of 2035).
  - The program will then display the current value for the year (either the value entered by the user if the year that was entered was valid or the default value if the year was not valid).

Author: James Tam

Exercise 4

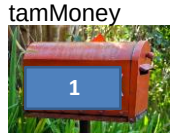
- Modify the previous program (asks the user for the year).
  - It will still prompt the user for a year and display an error message if the year is not within the range of 2000 – 2099.
  - Instead of setting invalid values to the default value the program will instead **repeatedly ask** (using a do-while loop) for the year until a valid value has been entered.
    - E.g.
      - The user enters 1999 for the year. An error message will be displayed and the program again prompt the user for the year.
      - The user then enters 2135 for the year. Again an error message will be displayed and the program re-prompts the user for the year.
      - This time the user enters 2000 for the year. Since this value is within the valid range the program will stop prompting the user for the year and display the year entered.

Author: James Tam

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.

tamMoney

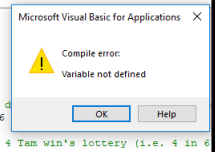



```
Option Explicit
Sub lotteryProgram2()
    Dim tamMoney As Long
    Dim dieRoll As Long

    ' Tam makes $1!
    tamMoney = 1

    ' Generates a random d
    dieRoll = CInt(Int((6

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



Author: James Tam

Example: Option Explicit Used

VBA will automatically catch the error and point out the location

- **Example: 1A_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
```

Author: James Tam

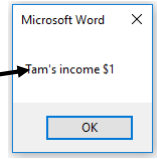
Example: Option Explicit Not Used

- **Example: 1B_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!

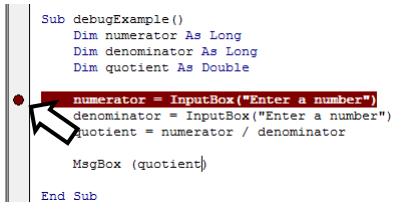
Tam didn't get the "big bucks" ☹
Errors like this can be hard to catch/fix in all but smallest programs



Author: James Tam

The VBA Debugger

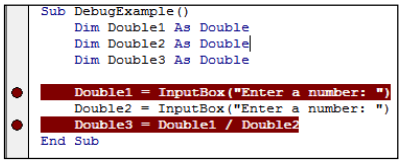
- Debuggers can be used to help find errors in your program
- Setting up breakpoints
  - Points in the program that will 'pause' until you proceed to the next step
  - Useful in different situations
    - The program 'crashes' but you don't know where it is occurring
      - Pause before the crash
 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

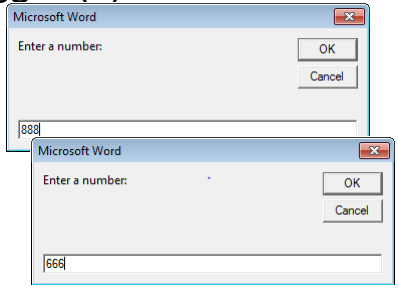


Author: James Tam

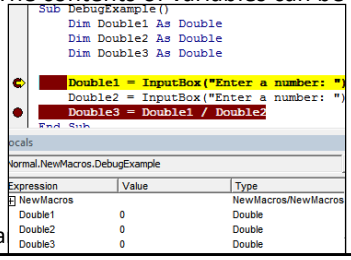
The VBA Debugger (2)

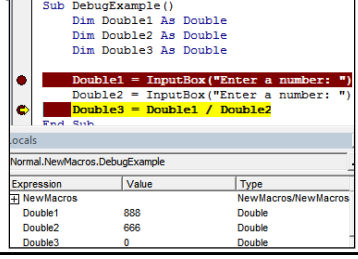
- Multiple breakpoints





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

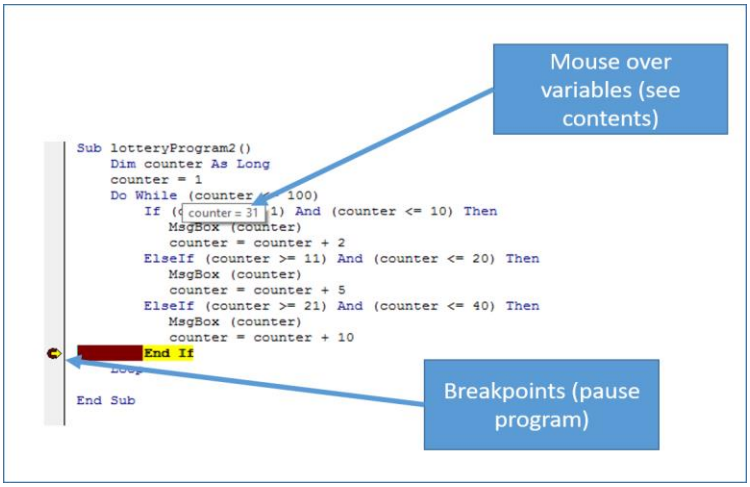




Author: Ja

The VBA Debugger (3)

- Combining breakpoints and viewing variables.

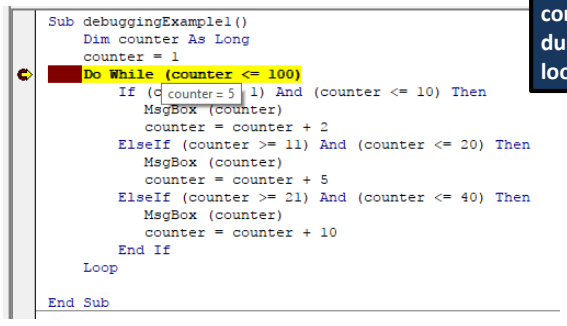


Author: James Tam

An Example To Run With The Debugger

- **Example: 2debuggerExample.docm**
  - 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



Author: James Tam

An Example To Run With The Debugger (2)

- **Example: 2debuggerExample.docm**
  - 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 breakpoints 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'**

Author:

The `Dir` Function

- Allows a VBA program to access the documents in a folder (a 'dir' or directory = folder)
- The function will access documents by their name and the name of the document can then be used to open the document
- Combined with a loop this function can successively access each document in a folder one at a time

Author: James Tam

Dir: Document

- **Example: 3dir.docm**
  - There are four versions of the program in the Word document
  - All require data files in the location C:\temp
  - That is: make sure the 'C' drive contains a 'temp' folder which in turn contains at least one Word document

Author: James Tam

Dir: First Example

- Version1: Try running with and without a document called "document.docx" in the temp folder

```
Sub version1()
    Dim result As String
    result = Dir("C:\temp\document.docx")
    If (result = "") Then
        MsgBox ("document.docx NOT found in C:\temp")
    Else
        MsgBox ("document.docx found in C:\temp")
    End If
    MsgBox ("Result of running dir function=" & result)
End Sub
```

Author: James Tam

Dir: Second Example

- Version2: Allows the user to **specify the location**

```
Sub version2()
    Dim result As String
    Dim fullName As String
    fullName = InputBox("Specify path and document e.g.
      C:\temp\document.docx")
    result = Dir(fullName)
    If (result = "") Then
        MsgBox ("document NOT found in " & fullName)
    Else
        MsgBox ("document found in " & fullName)
    End If
    MsgBox ("Result of running dir function=" & result)
End Sub
```

Author: James Tam

Dir: Third Example

- Version3: Augments previous version. It can distinguish between an **invalid location** and a **valid location that does not contain the specified file**

```
Sub version3()
    Dim result As String
    Dim fullName As String
    fullName = InputBox("Specify path and document e.g.
      C:\temp\document.docx")
    If (fullName = "") Then
        MsgBox ("You entered an empty path & filename")
```

Author: James Tam

Dir: Third Example (Location Valid, Check If Document At That Location)

```
        ' Location valid, check for document
        Else
            result = Dir(fullName)
            If (result = "") Then
                MsgBox ("document NOT found in " & fullName)
            Else
                MsgBox ("document found in C " & fullName))
            End If
            MsgBox ("Result of running dir function=" & result)
        End If
    End Sub
```

Author: James Tam

Dir: Four Example

- Version 4: Augments Version 3 so that it steps through all the documents in the folder and displays their name in popup MsgBox

```
Sub version4()
    Dim result As String
    Dim fullName As String

    fullName = InputBox("Specify path to a folder name e.g.
      C:\temp\")
    If (fullName = "") Then
        MsgBox ("You entered an empty path & filename")
```

Author: James Tam

Dir: Four Example (Get And Successively Display Each Document Name Residing In The Specified Folder)

```
    Else
        result = Dir(fullName)
        Do While (result <> "")
            MsgBox ("File name: " & result)
            result = Dir
        Loop
    End If
End Sub
```

Author: James Tam

Text Processing

- Finding/replacing text (and other things such as formatting effects or even formatting styles)
- Tallying occurrences (e.g. typographical mistakes, number of times a search word appears in a document)
- Four versions of the program are included in the document 'textProcessing.docm':
  - countingPartialV1 (counts partial matches of word)
  - countingExactV2 (counts only exact matches of word)
  - countTyposOneDocumentV3 (typos in 1 document)
  - countTyposAllDocumentV4 (typos in each doc in a folder)

Author: James Tam

Counting Occurrences: **Partial Matches** Allowed

- Example: searching for '**the**' will show '**the**re', '**the**ir' and '**the**y're' as matches

```
Sub countingPartialV1()
    Dim count As Long
    With ActiveDocument.Content.Find
        Do While .Execute(FindText:="the", Forward:=True, _
          Format:=True, MatchWholeWord:=False) = True
           count = count + 1
        Loop
    End With
    MsgBox ("Number of instances of 'the' includes partial
matches " & count)
End Sub
```

Author: James Tam

Counting Occurrences: **Only Exact Matches**

- Example: searching for '**the**' with the text (there their' and they're) yields no matches

```
Sub countingExactV2()
    Dim count As Long
    With ActiveDocument.Content.Find
        Do While .Execute(FindText:="the", Forward:=True, _
          Format:=True, MatchWholeWord:=True) = True
           count = count + 1
        Loop
    End With
    MsgBox ("Number of instances of 'the' including only exact
      matches " & count)
End Sub
```

Author: James Tam

Counting Typographical Errors: Single Document

- Counts the number of typos in the current active document
- Also writes text into the current document

```
Sub countTyposOneDocumentV3()
   Dim numTypos As Long
   Dim comment As String

   numTypos = ActiveDocument.SpellingErrors.count
   Selection.Font.Size = 24
   ' vbCR = Newline
   ' Writes string out where the cursor is currently located
   comment = vbCr & "Number typographical errors: " & numTypos
      & " " & vbCr
   Selection.TypeText (comment)
End Sub
```

Author: James Tam

Counting Typographical Errors: All Documents In A Folder

```
Sub countTyposAllDocumentsV4()
   Dim numTypos As Long
   Dim comment As String
   Dim path As String
   Dim currentFile As String

   directoryPath = "C:\temp\203\"
   currentFile = Dir(directoryPath & "*.doc*")
   If (currentFile = "") Then
       MsgBox ("No Word documents in location " &
   directoryPath)
   End If
```

Author: James Tam

Counting Typographical Errors: **All Documents In A Folder** (2)

```
    Do While (currentFile <> "")
        currentFile = directoryPath & currentFile
        Documents.Open (currentFile)
        numTypos = ActiveDocument.SpellingErrors.count
        comment = "# typos in " & currentFile & "=" & numTypos
        MsgBox (comment)
        currentFile = Dir
    Loop
End Sub
```

Author: James Tam

Exercise 5:

- Modify the starting program:
  - Text will be written in the document located at the start or end of the document (user choice)
  - The choice will come in the form of an InputBox ('s' = start of document, 'e' = end of document)
  - If the user enters anything else then the program will repeatedly prompt again until one of these two values have been entered and an error message (in the form of MsgBox) should appear when the user enters a value for the location that is not one of the above two values.
  - Formatting of the text:
    - Font = Arial Black
    - The text written to the document should reside on its own line. (To put it another way: there should be a carriage return before the text and after it).

Author: James Tam

Exercise 5: Sample Run

Needless to say this exercise will take quite a bit longer than previous ones

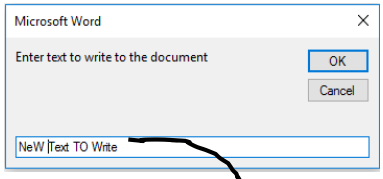


Figure 1: Prompt to user

What the user enters written to top of document

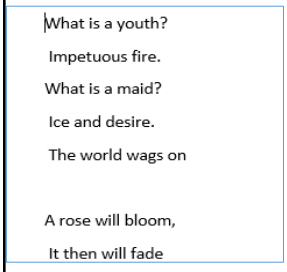


Figure 2: Document before write

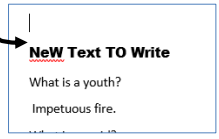


Figure 3: Document after write

Author: James Tam

Exercise 5: Starting Code

- Look for the document: exercise5_starting.docm

```
Sub variableTextWriting()
    Dim textToWrite As String
    textToWrite = "Don't fall asleep - Nightmare on Elm
      Street"
    Selection.TypeText (textToWrite)
End Sub
```

Author: James Tam

Exercise 6

- Modify the starting code so the user can get via an InputBox the filename suffix
  - This suffix specifies the type of document that the program will process
  - e.g. User enters 'xlsx', program only accesses Excel files in the C:\temp\203 folder
  - (Currently the program only opens Word2007 '.docx' documents).

Author: James Tam

Exercise 6: Starting Code

- Look for the document: exercise6_starting.docm

```
Sub displayAllDocumentsOfAnyType()
   Dim numTypos As Long
   Dim comment As String
   Dim path As String
   Dim currentFile As String
   directoryPath = "C:\temp\203\"
   currentFile = Dir(directoryPath & "*.docx")
   If (currentFile = "") Then
       MsgBox ("No Word documents" & " in location " & _
                directoryPath)
   End If
```

Author: James Tam

Exercise 6: Starting Code

```
    Do While currentFile <> ""
         currentFile = directoryPath & currentFile
         MsgBox (currentFile)
         currentFile = Dir
     Loop
End Sub
```

Author: James Tam