Debugging

Larry Caretto
Mechanical Engineering 209

Computer Programming for Mechanical Engineers

February 16, 2017

California State University Northridge

Outline

- Review choice statements
- Finding and correcting program errors
- Debugging toolbar in VBA editor
- Basic ideas about debugging
  - Breakpoints to halt operations
  - Step-by-step execution
  - Displaying values
  - Use of Watch and Immediate windows

Review If – Else If

- If any condition is true, the statements following the If or ElseIf are executed
- Once any statements are executed transfer to the first statement after the End If
- Statements for only the first true condition are executed
- The Else block is optional
  - If no conditions are true those statements are executed

```
If <condition1> The
    <Statements don
Else If <condition2>
    <Statements don
Else If <condition3>
    <Statements don
<May be other cond
Else
    <Statements don
End If
<Execute here after
```

May have zero or many ElseIf's

In–Class Exercise Solution

```
Function test(x As Double) As String
    If x < 0 Then
        test = "Negative"
    ElseIf x > 0 Then
        test = "Positive"
    Else
        test = "Zero"
    End If
End Function
```


Function Test Example

	A	B
1	-2.9357	Negative
2	-8.0183	Negative
3	-5.9738	Negative
4	0.0000	Zero
5	5.3204	Positive
6	2.1422	Positive
7	-0.0305	Negative
8	-2.9766	Negative
9	3.2078	Positive
10	-3.8545	Negative
11	0.0000	Zero
12	2.0952	Positive
13	4.4570	Positive
14	-4.4392	Negative
15	2.6949	Positive
16	3.6464	Positive
17	-2.2486	Negative
18	-8.8598	Negative
19	-1.4102	Negative
20	6.0504	Positive

	A		B
1	-2.935719195149	=test(A1)	
2	-8.018345742789	=test(A2)	
3	-5.973760742595	=test(A3)	
4	0	=test(A4)	
5	5.320383892017	=test(A5)	
6	2.142150827632	=test(A6)	
7	-0.030534303348	=test(A7)	
8	-2.976633861745	=test(A8)	
9	3.207849071960	=test(A9)	
10	-3.854533153698	=test(A10)	
11	0	=test(A11)	
12	2.095165449840	=test(A12)	
13	4.457003957576	=test(A13)	
14	-4.439226566896	=test(A14)	
15	2.694920775196	=test(A15)	
16	3.646380899032	=test(A16)	
17	-2.248621763140	=test(A17)	
18	-8.859751029899	=test(A18)	
19	-1.410194041907	=test(A19)	
20	6.050393944427	=test(A20)	

Normal view on left

Equation view on right

Immediate Program Errors

- Errors in a newly-written line are colored red immediately after the return key is pressed

```
Option Explicit
                    'Missing function type
Function test(x As double) as

Function isOdd(N As Long) As Boolean
                    'Missing 0 (N Mod 2 = 0)
    isOdd = N Mod 2 =

Function test(score As Long) As String
    if score < 60 test = "Fail"
                    'Missing Then
```

Hard-to-find Errors

- Long statements with errors

```
unevenSimpson = ((y(1) + y(0)) _
              * (x(1) - x(0)) / 2 _
              + getSimpson(x, y, 1, NX) _
              + getSimpson(x, y, 0, NX - 1) _
              + (y(NX) + y(NX - 1)) * _
              * (x(NX) - x(NX - 1)) / 2) / 2
```

- Type apostrophe before underscores (_) marking continuation
- If statement turns black to left of apostrophe, error is NOT there

California State University Northridge 7

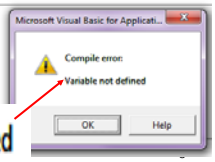
Compilation Error Messages

- Compilation converts code from your VBA into computer commands
- Errors in this stage have explanation in text boxes with error source highlighted

```
Option Explicit
Sub isOdd(N As Long)

    isOdd = R

End Sub
```



Variable not defined

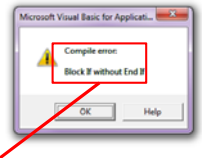
California State University Northridge

Another Compilation Error

```
Sub isOdd(N As Long)

    If N Mod 2 = 0 Then
        MsgBox "N is even"
    Else
        MsgBox "N is Odd"

End Sub
```



Compile error:

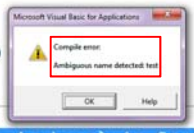
Block If without End If

California State University Northridge 9

Still Another Compilation Error

```
Sub test(N As Long)
    Call isOdd(3)
End Sub

Function test(score As Long) As String
    If score < 60 Then test = "Fail"
End Function
```



Cannot have two procedures with same name

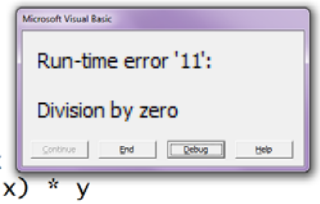
Compile error:

Ambiguous name detected: test

California State University Northridge 10

Execution Error

```
Sub faje()
    x = 8
    y = 0
    Z = 3
    k = 9
    w = Z * x
    k = (w + x) * y
    r = 1 / k
    s = p + q
End Sub
```



Click Debug to see error location

California State University Northridge 11

Execution Error

```
Sub faje()
    x = 8
    y = 0
    Z = 3
    k = 9
    w = Z * x
    k = (w + x) * y
    r = 1 / k
    s = p + q
End Sub
```

Statement causing error is highlighted in yellow

Division-by-zero error is caused by k = 0; can hover mouse over k to confirm

California State University Northridge 12

Execution Error Detection

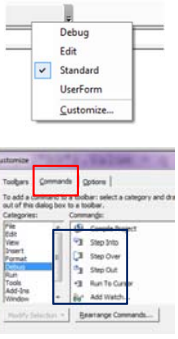
```
w = Z * x
k = (w + x) *
r = 1 / k
s = p + q
nd Sub
```

k = 0

Hovering mouse over k after execution is halted shows value of k confirming k = 0

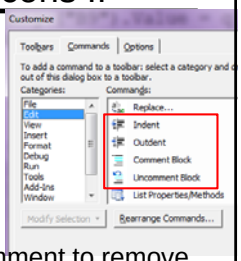
VBE Toolbar Debug Icons

- Right click toolbar in VBE and select Customize
- Select Commands tab in resulting dialog and select Debug from Categories:
- Right-click on desired Commands and drag icons to toolbar
  - Add icons in blue box plus Quick Watch (one at a time)



VBE Toolbar Icons II

- Select Edit from Catagories:
- Right-click on desired Commands (in red box) and drag icons to toolbar
  - Click close to exit the Customize dialog
- Use comment and uncomment to remove statements from executable code, while leaving them in place for later use



Debugging Tools

- Allow user to interact with code as it is running and observe variable values
- Can stop at certain points (breakpoints), go step-by-step, enter or skip subprograms, and examine values of variables to locate incorrect values
- First step is usually setting breakpoints halting program execution at points where you want to examine variables

Setting Breakpoints

- Mouse has I-Beam shape in VBE code: I
- Moving mouse near left margin changes shape to right facing arrow that can select entire line
- Further moving mouse gives left facing arrows; click this arrow to set breakpoint
  - Line has reddish shade with dot in left margin

```
    q = CalcDynamicPressure(rho, V)
    q = CalcDynamicPressure(rho, V)
```

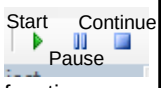
Above: Normal; **Below:** with breakpoint set

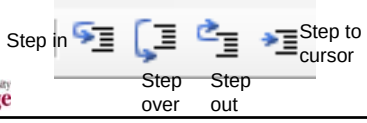
Using Breakpoints

- When code execution comes to a line with a breakpoint, execution stops before the line is executed
- User can then examine values of variables in code to see if there are any errors present
- If errors are noted, correct them and continue execution or restart
- If no errors found, continue execution (F5 or start icon) to another breakpoint

VBE Program Execution

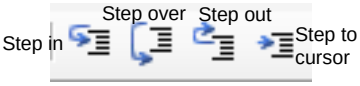
- Start (or continue), pause and stop program execution
  - Start only works for subs, not for functions
- Step in (F8), step out (Shift+F8), step over(Control+Shift+F8), and step to cursor (Control+F8) shown here and discussed on next chart





California State University Northridge 19

Step-by-step Execution



- Execute one statement at a time
  - Step in executes and debugs all statements including those in other procedures
  - Step over executes, but does not debug statements in other procedures
  - Step out completes execution of a called procedure without debugging
  - Step to cursor starts execution then halts at line with cursor

California State University Northridge 20

Examining Variables

- We have seen debug controls
  - breakpoints to stop and restart execution
  - step-by-step single-statement execution
- How do we examine values of variables?
  - Can hover mouse over variables to see their values
  - Can place names of variables in debug windows: “Watch” window, “Locals” window, and “Immediate” window

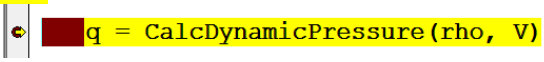
California State University Northridge 21

Halted Execution and Hover

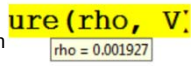
Can drag arrow to change order of execu-tion

When execution halts, the next line to be executed is shaded yellow and has a yellow arrow pointed to it

- Example shows halt at a breakpoint



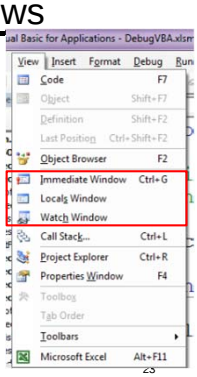
- Hovering mouse over variable shows its value in shaded box
  - Mouse position not shown



California State University Northridge

Setting Windows

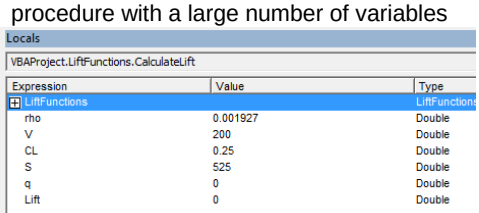
- Use View menu of VB Editor to select various windows
- Select one or more of the three debug windows, Immediate or Watch
- Using more than one window reduces width available for window



California State University Northridge

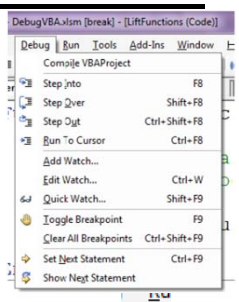
Locals Window

- Locals window shows current values of all variables in a procedure
  - Easiest to use, but can get cluttered for procedure with a large number of variables



Watch Window

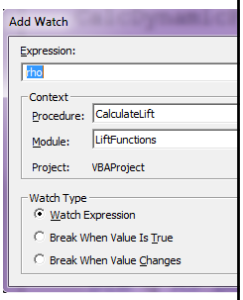
- User selects variables to be watched
  - Select Add Watch from Debug menu or click "Add Watch" icon
  - Usually select variable for which you want to add watch before selecting Add Watch



Toolbar Icon for Add Watch 

"Add Watch" Window

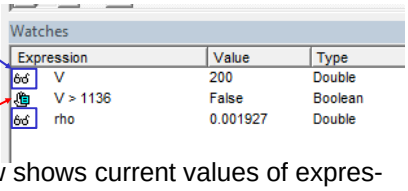
- May select single variable or expression
- Variable or expression selected before getting watch window shows in window
- Can use bottom block to control debug execution of program



Watch Window Example

Watch Expression

Beak when expression is true



- Window shows current values of expressions (which may be just one variable)
- Note that "V > 1136" will cause the program to halt when it becomes true
  - Can check variables that caused this value

Immediate Window

- Can get values of variables (or expressions) and set values of variables

Asks for value of V to be printed with result of V = 200

Asks for value of rho*V^2/2 to be printed with result of 38.54

Sets value of V in program to 250

Remaining statements repeat value requests above with new results for V = 250