

A SunCam online continuing education course

Programming MS Excel in Visual Basic (VBA) Part 2-Branching & Looping, Message Boxes & Alerts

by

Kwabena Ofosu, Ph.D., P.E., PTOE



Abstract

This course is the second of a four-part series on computer programming in *Excel Visual Basic for Applications* (VBA), tailored to practicing engineers. In this course the topics, conditional statements, message boxes and alerts, and looping structures are presented. Several examples relevant to engineering are used to illustrate and demonstrate the concepts and methods learned in this class. Two mini-projects are used to demonstrate the programming concepts and methods in situations encountered by practicing engineers.

Computer Programming in Visual Basic (VBA) – Part 1 is not required as a pre-requisite to this course. It would however be helpful to understand the basic principles of computer programming as well as the fundamentals of the Excel VBA language as presented in Part 1 of this series.



TABLE OF CONTENTS

Abstract	2
List of Tables	4
List of Figures	5
1. CONDITIONAL STATEMENTS	6
1.1 Definition	6
1.2 If-Then-Else Statement	6
1.3 Logical Operators	7
1.4 Composite Conditional Expressions	7
1.5 Nested Conditional Statements	9
1.6 Select-Case Statement	
2. MESSAGE BOXES AND ALERTS	15
2.1 Message Box	
2.2 Simple Message Box	16
2.3 Message Box with Options	
2.4 Manipulating Style Value	
2.5 Third VBA Project	
3. LOOPING	
3.1 For-Next Loop	
3.2 Nested Loops	64
3.3 Do-Loop	65
3.4 Exiting a Loop	67
3.5 Fourth VBA Project	67
4. CONCLUSION	
REFERENCES	



List of Tables

Table 1	l: Logical	operators	. 8
---------	------------	-----------	-----



List of Figures

Figure 1: Algorithm for Third VBA Project	42
Figure 2: Plan for Fourth VBA Project	. 72



1. CONDITIONAL STATEMENTS

1.1 Definition

A conditional statement is a feature of a programming language that executes different instructions (lines of code) based on whether some **condition** is met. Conditional statements enable the programmer to control the way an application interacts with the user. Conditional statements are often referred to as **Branching** in some texts, as they provide a means for a program to branch off in some direction or the other as some condition(s) is checked for and met, and the program then proceeds in the relevant direction(s).

1.2 If-Then-Else Statement

The most common conditional statement is the *If*-*Then*-*Else* statement. If the specified condition is met, a block of code will be executed, else a different block of code will be executed. In VBA the syntax is as follows

If condition Then Run this code Else Run that code End If

An alternate format allows the *Else* condition code to be placed on the same line as *Else* word, as follows:

If condition Then Run this code Else: Run that code End If

For more than two conditions, the *ElseIf* condition(s) is added as follows



```
If condition 1 Then

Run code 1

ElseIf condition 2 Then

Run code 2

ElseIf condition 3 Then

Run code 3

:

:

ElseIf condition (n-1) Then

Run code (n-1)

Else

Run code n

End If
```

As with the fundamental two-condition set up, the *Else* statement may be on the same line as the relevant code but separated from the code by a colon. In all of the above cases the *Else* statement is optional.

1.3 Logical Operators

The *condition* in the condition statement is a **logical expression** where a **logical operator** (also called a **Boolean operator**) is applied to compare, evaluate, or check that the inputs (called **operands**) meet the specified condition and give a result of "TRUE", based upon which the relevant block of code will execute.

Examples of logical operators supported in VBA are shown in Table 1.

1.4 Composite Conditional Expressions

Conditional expressions may be combined using the "And" and/ or "Or" operators to form a composite conditional expression.



Table 1: Logical operators

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If true, then the condition is TRUE, otherwise it is FALSE.	If $X = = Y$ Then.
<>	Checks if the values of two operands are not equal. If the values are not equal, then the condition is TRUE, otherwise it is FALSE.	ElseIf p <> q Then.
>	Checks if the value of the left operand is greater than the value of the right operand. If true, then the condition is TRUE, otherwise it is FALSE.	If $m > n$ Then.
<	Checks if the value of the left operand is less than the value of the right operand. If true, then the condition is TRUE, otherwise it is FALSE.	ElseIf x < y Then
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand. If true, then the condition is TRUE, otherwise it is FALSE.	If $a \ge b$ Then
<=	Checks if the value of the left operand is less than or equal to the value of the right operand. If true, then the condition is TRUE, otherwise it is FALSE.	ElseIf q <= r Then
IsNumeric	Checks if an operand is a numeric value. If yes, then the condition is TRUE, otherwise it is FALSE.	If IsNumeric(TextBox2.value) Then
IsNull	Checks if an operand is a null value (empty). If yes, then the condition is TRUE, otherwise it is FALSE.	If IsNull(TextBox2.value) Then



For example, consider a bank account that has been overdrawn. If another charge comes in and the bank pays it, the account goes further into the negative and is charged an overdraft penalty for that transaction. However if a deposit comes in that partially clears the deficit, even though the account is still in the negative, the account is not charged an overdraft fee for that transaction. Therefore, using the negative sign for a charge transaction and positive sign for a deposit, the overdraft penalty fee is applied as follows:

```
If balance < 0 And transaction < 0 Then
newbalance = balance + transaction - fee
ElseIf balance < 0 And transaction > 0 Then
newbalance = balance + transaction
End If
```

1.5 Nested Conditional Statements

A **nested conditional statement** is a conditional statement placed within another conditional statement. The bank account example can be rewritten using nested conditions as follows:



Else

End If

In each case, the *If-ElseIf-Else-End* syntax for each conditional statement must be complete on its own regardless of whether it is nested or not. For instance, in the above example, if the *End If* of the nested *If* statement was omitted, the syntax would be incorrect and a compiler error would occur. A common format to keep track of this, as demonstrated in the above example, is by typing the code such that the *If-ElseIf-Else-End If* for a specific *If* statement are aligned vertically



and that of any nested statements are offset laterally from the main statement in which they are nested. This is called **indenting** the code.

The choice, relevance, or advantage of nesting versus composite conditions must be determined by the programmer based on the specific objectives and requirements of the application.

1.6 Select-Case Statement

This is an alternate method to the If –*Then-Else*. It is advantageous to use when there are too many conditions and the *If*-*Then-Else* statement becomes cumbersome and difficult to follow and keep track of. The syntax is as follows:

```
Select Case variablebeingchecked
Case variablevalue1
Run code 1
Case variablevalue2
Run code 2
:
:
Case Else
Run code
End Select
```

The Select-Case format may also have combined logical expressions, and may involve nesting.



Example 1:

Review the code for a grade calculator for an Engineering professor. A score of 95 and above is +A, from 90 to 91.999 is -A, from 85 to 89.999 is +B, and so on, anything less than 60 is an F. The professor enters the grade in a cell on the spreadsheet and presses the button which fires the code that checks the score and assigns the grade and a comment, and displays them back on the spreadsheet.

X	, ≝) + (≝ + -				-
F	ile Home Inse	rt Page Layout	t Formulas D	ata Review	View D
Pas	Cut Copy ~ te Format Painter Clipboard	Calibri B <i>I</i> <u>U</u> - Fo	• 12 • A A A 		** 📑 Wi
	A1	(f _x	SCORE		
1	А	В		С	D
1	SCORE				
2	94				
3		A-	Ex	cellent	
4					
5					
6					
7			CALCU	ATE GRADE	-
8					
9					_
10					
11					



The code using *If-Then-Else* is as follows;





Example 2:

Alternately the *Select-Case* conditional statement may be used. In this example a simplified grading system is used. 90 and above is A, 80 to 89.99 is B, 70 to 79.99 is C, 60 to 69.99 is D, and anything else is F.

	C2 •	f _x		
	А	В	С	D
1	SCORE	GRADE		
2	89	В		
3				
4				
5		CALCULATE		
6				
7				
8				
9				



The VBA code for the click button is as follows:

```
(General)
  Option Explicit
  Private Sub CommandButton1 Click()
  Dim mark As Double
  Dim grade As String
  mark = Cells(2, 1).Value
                           Note that the spreadsheet object name was not prefixed
                           to the sheet-cell identifier. This is permitted if the
                           application involves only one spreadsheet. The same
                           holds for form controls and the form name prefix.
  Select Case mark
  Case 0 To 59.999
       grade = "F"
       Cells(2, 2) = grade
  Case 60 To 69.999
       grade = "D"
       Cells(2, 2) = grade
  Case 70 To 79.999 🔶

    Composite logic operator

       grade = "C"
       Cells(2, 2) = grade
  Case 80 To 89.999
       grade = "B"
       Cells(2, 2) = grade
  Case 90 To 100
       grade = "A"
       Cells(2, 2) = grade
  Case Else
                                        "Case Else" may have
                                        been omitted as there is
                                        no code for it, and it is
  End Select
                                        not relevant
```



2. MESSAGE BOXES AND ALERTS

2.1 Message Box

A message box acts like a dialog box where a user can interact with the computer. Message boxes are used to alert or prompt the user. Message boxes are able to perform actions in response to what the user selects. Message boxes can be incorporated into conditional statements.

The general structure of the code for a message box is as follows:

MsgBox (Prompt, Style Value, Title)

The *Prompt* is the statement that will appear across the main body of the message box. The *Style Value* relates to the buttons that will appear on the message box such "OK", "Yes", "No", or "Cancel", as well as any symbols, such as a question mark, exclamation point (for a warning message box for instance), informational sign etc. The *Title* is what appears in the title bar across the top of the message box.





2.2 Simple Message Box

A simple message box will be created in this section. The message box will be activated by clicking a push button on a spreadsheet.

From the **Developer** tab, select **Design Mode.** Applications can be built directly on a spreadsheet once in Design Mode.



Click on Insert.

Select Command Button under ActiveX Controls.

Home Insert Page Layout Formulas Data Review View Developer Image: Code Image:	C	9 •	(~ ~) =								
A B Code A B C A B C Command Button (ActiveX Control) Insert a command button control. Command Button (ActiveX Control) Insert a command button control.	9	Home	Insert	Page Layout	Forr	nulas	Data	Revie	w V	iew	Developer
Code Form Controls XML A1 fx fx Image: Control to the second	Visual Basic	Macros	Record Ma Use Relati	acro ve References urity	Insert	Design Mode	Pro View Rur	perties w Code n Dialog	Source	Map Expa	Properties Insion Pack esh Data
A1 f A B C Aa B C Command Button (ActiveX Control) Insert a command button control. S C B C C B C C B C C B C C Command button Control. Control Control. Control Control. Control Control. Control Control. Control Control.	4		Code	6	Form	Control	s				XML
A B C 1 A 2 3 4 5 6 7 8 9		A1	• (• f _x	-						
1 ActiveX Controls 2 Image: Control of the second	1	A	В	С		ab 🛔 ab		F	G	Н	1
2 3 4 5 6 7 8 9	1				Activ	eX Contr	rols				
3 Command Button (ActiveX Control) 4 Insert a command button control. 5 1 6 1 7 1 8 1 9 1	2						abl				
4 Command button (active control) 5 Insert a command button control. 6 7 8 9	3				1 Com	A .e	AP L	ctine V Co	ntroll)	
5 Insert a command button control. 6	4				Com		ICCOIL (A	CUVEN CO	inu oi)		
6	5				Ins	ert a con	imand b	utton cor	itrol.		
7	6										
8 9	7										
9	8										
	9										
10	10										



Holding down the mouse, trace the shape and size of the command button on the spreadsheet.

0) 🖬 🤊 -	(~ ~) ₹						
C	Home	Insert	Page Layout	Form	nulas	Data Rev	iew V	iew
Vis Ba	ual Macros sic	Record M Use Relat Macro Se Code	acro ive References curity	Insert	Design Mode Con	Properties View Code Cale Run Dialog	Source	Exp Re
	A1	, (a fx					
	A	В	с	D	E	F	G	ł
1								
2								
3								
4								
5								
6								
7								
8								
9								

Release the mouse.

0		* (°4 ÷) ;	Ŧ								
C	Home	Insert	Page Lay	out	For	mulas	Data	Revie	w V	lew	Develope
Vis Ba	ual Macros	Code	Macro ative Referenc Security	ces	Insert *	Design Mode	Prope View TRun I rols	erties Code Dialog	Source	Re	ip Properti pansion Pa fresh Data XML
Co	ommandBu	utton1 👻	. (f _x	=EMB	ED("Forr	ms.Com	mandE	Button.1	1","")	
	А	В	С		D	E	F		G	ł	4
1											
2					0		0		ρ		
3										_	
4					- 0	Com	mandBu	tton1	6		
5										_	
6					0						
7							_			_	
8											
9										_	
10											
11											



Right click on the new button and select **Properties**.

Со	CommandButton1 🔹 🕼 =EMBED("Forms.CommandButton.1","")										
	А	В	С	D	E	F	G	Н	1	J	K
1											
2								0			
3						-		[
4				0	Comman	dButton1		6			
5							*	Cu <u>t</u>			
6							B	<u>С</u> ору			
7								<u>P</u> aste			
8							-	Properties			
9								View Code		_	
11								CommandBi	utton Object	•	
12								Grouping		•	
13								Order			
14							A.			-	
15							2	Format Conf	trol		
16											
17											
18											

Change the **Caption** property to "Simple Message Box"

<i>f_x</i> =E	MBED("Forms.Comman	dButton.1	","")							
D	F Formula Bar	G	н		-		К	1	М	N
		-						-		
	0		ρ	Properties						
				CommandB	utton	1 Com	mandButton		•	L
	Simple Message Box		6	Alphabetic	C		1		_	
				Alphabeuc	Categ	orizea			1	
			L	(Name)		Comma	andButton1			
0	0		0	Accelerator						
				AutoLoad		False				
				Autosize		Faise				
				BackColor		1 6-1	20000000F&			
				Caption		Simple	Message Boy	dne 1		
				Enabled True						
				Font Calibri						
				ForeColor 8H800000128						
				Height		61.5				
				Left		144.75	5			
				Locked		True				
				MouseIcon		(None))			
				MousePointe	er 🛛	0 - fm!	MousePointer	Default		
				Picture		(None))			
				PicturePositio	on	7 - fmF	PicturePositio	nAboveCente	r 🛛	
				Placement		2				
				PrintObject		True				
				Shadow	o Cliele	raise Truo				
				Top	ICIICK	11UE				
				Visible		True				
				Width		192				
				WordWrap		False				
						_				
				_						



Right click on the command button again, this time select View Code.

C D E F G H I J K ○ Simple Message Box ∅ □	(•	<i>f</i> _∞ =EME	BED("Forms	s.Comman	dButton.1	","")			
Simple Message Box Cut Copy Paste Properties Yiew Code CommandButton Object > Grouping > Order > Somat Control	С	D	E	F	G	Н	- I	J	K
Grouping → Order → Eormat Control		0	Simple Me	essage Box		Cut Copy Paste Properties View Code CommandBu	tton <u>O</u> bject		
						<u>G</u> rouping O <u>r</u> der <u>F</u> ormat Cont	rol	> 	

The code window opens

A	B	C D	E	F	G	н	1	1	K	L	м	N	0	р	0	
		0		0	ľ											
			<i>c</i> -													
		Ŷ	Co	mmandButt	ant b											
		0		0	ò											
	P Microsoft \	risual Basic - Book1 (design	1 - ISheet1 (C	ode)]											
	FIRE DIA D	die Minur Innent	Eares	at Dahua	Pup Teels	Add Inc	Window	Links				Tune a	nuertion for	help		
	co Die D	au Tiess hitser	*Mun	er Dennå	Dou: Toolo	1000-110	TUIDOW	Deh				types		map		
	🛛 🖾 = 🖌	X 45 65 85	"7	Ge 🕨 11	- 🗹 🕉	🖀 😽 🖄	😺 Ln	4, Col 1								
	Project - VBAP	roject	×	Command	Button1					Click					•	
	🔲 🖬 🗀			Const.	Puplin											
	B & VBAPN	oject (Book1)	-	open	on onprior										-	
	(II) She	et1 (Sheet1)		Prive	te Sub Ce	emandBut	con1_C1	Leic()								
	- III) She	et2 (Sheet2)														
	She She	et3 (Sheet3) Workhook	1	End :	Pub											
	B BE VRAPH	niect (FUNCRES XLA	н т.													
	1 ×															
	Properties - Co	mmandButton1	x													
	CommandBut	ton CommandButton														
	Alphabetic C	stegorized														
	(Name)	CommandButton 1														
	Accelerator		m.													
	AutoLoad	False														
	Autosize BackColor	AHRODODODES														
	BackStyle	1 - fmBackStyleOp														
	Caption	CommandButton1														
	Enabled	True														
	ForeColor	AH30000012A														_
	Height	60.75														
	Left	190.5	E													
	Locked	True	- 11													
	MousePointer	0 - fmMousePointe														
	Picture	(None)														
	PicturePosition	7 - fmPicturePositio	•													
	Placement	2														_
	Shadow	False														
	TakeFocusOnC	lick True														
	Тор	29.25														
	Visible	True	1												-	
	[1.05.75	•												· ·	
					-		_			_		_		_	_	



In the click procedure for CommandButton1 type the following code:





Return the spreadsheet. Click on **Design Mode** to deactivate it.

0	100	· (°* -) =							
	Home	Insert	Page Layout	Form	nulas	Data	Review	View	Deve
Visi Ba	ual Macros sic	Record M Use Relat Macro Se Code	acro ive References curity	Insert	Design Mode Con	Prop View Run I trols	code Dialog	Durce	ap Prop spansio efresh E XM
	A1	、 (fx fx		Design	Mode			
	A	В	С	D	Turn	on or off	design mod	e.	н
1					Pr	ess F1 for	more help.	-	
2	1								
3									
4				c	imple	Μοςτοπο	Box		
5					impre	Message	DOX		
6									
7									
8									

The command button is now "live". Click on the **Simple Message Box** button

	A	В	С	D	E	F	G	Н	I	J	К	L	M	N	0	Р	Q
1																	
2																	
3																	
4					Simple	Accare Boy											
5					Simple N	icosage box											
6																	
7																	
8																	
9																	
10																	
11																	
12																	
13																	
14																	
15																	
16																	
17																	_
18														Microsoft E	xcel		×
19																	
20														Welcome	to VBA Prog	ramming	
21														(All the second s	-		
22															100		
23																OK	
24															-		
25																	
20																	
21																	



2.3 Message Box with Options

Click on **Design Mode** to activate it. Select the Message Box. Right click on it, select Copy

	А	В	С	D	E	F	G	Н	I.	J	1
1											
2				,				0			
3											
4				,	Simple Me	ssage Box					
5										1	
6				,		,	ň	Cu <u>t</u>			
7						1	-13	<u>С</u> ору		ļ	
8								<u>P</u> aste			
9								Properties		L	
10							2	<u>V</u> iew Code			
11							- 1	CommandButto	n Obiect 🕨	1	_
12								Casualan			
13								Grouping		L	
14								Order	•		
15							3	Eormat Control.			
16											
17											

Click on a cell near the Message Box, Right click, select Paste





Drag the new command button to reposition and align it with the previous one. Right click on the new command button. Select Properties.

	А	В	С	D	E	F	G	Н	
1									
2									
3									
4					Simple Me	scare Boy			
5					Simple Me	soage box			
6									
7									
8									
9									
10			c	>	0	>	6	>	
11									
12			c		Simple Me	ssage Box	6	>	
13									
14			c	,		,		>	
15									
16									

Change the Caption property to "Message Box With Options

	А	В	С	D	E	F	G	Н	1	J	К	L	М	N
1														
2									Properties				6	
3									Topercies					
4									CommandBu	tton2 Cor	mmandButton		-	
5					Simple Me	essage Box			Alphabetic C	ategorized	1			
6									(Name)	Comm	andButton2			
7									Accelerator					
1								-	AutoLoad	False				
8									AutoSize	False				
9									BackColor	8	18000000F&			
10			c)	(0		p	BackStyle	1 - fm	BackStyleOpa	que		
11								-	Caption	Messa	age Box With C	Options		
11					lassaga Ray	With Oatin			Enabled	True				
12					lessage box	with Optio	ins	ľ	Font	Calibr	i			
13									ForeColor	8	180000012&			
14			c	,		0		b	Height	61.5				
4.5									Left	143.2	15			
15									Locked	True				
16									MouseIcon	(None	2)			
17									MousePointer	0 - fm	MousePointer	Default		



Double click on the Message Box with Options and type in the following code in its click procedure:

CommandButton2 Click
Option Explicit
<pre>Private Sub CommandButton1_Click()</pre>
MsgBox ("Welcome to VBA Programming")
End Sub
Private Sub CommandButton2_Click()
Dim message As String
<pre>message = MsgBox("Answer this question before you login: Are you having fun ?", vbYesNoCancel + vbQuestion, "Login")</pre>
If message = vbYes Then Cells(18, 1).Value = "YES THIS IS GREAT !" ElseIf message = vbNo Then Cells(18, 1).Value = "NO THIS VBA STUFF IS NOT WORKING FOR ME !" ActiveWorkbook.Close Else 'msgbox("You did not answer the question. Excel will now close") Prompt ActiveWorkbook.Save ActiveWorkbook.Close End If Style Value
End Sub
Code to save and close the Excel file if user selects to Cancel <i>If</i> Statement in conjunction with what user
clicks on the message box. If user clicks on Yes on the message box, or No, the corresponding

line of code will execute



Return to the spreadsheet. Click on **Design Mode** to deactivate it. Click on the **Message Box with Options**



Answer "Yes"





2.4 Manipulating Style Value

Copy and paste a third command button. Change the Caption property to "Message Box with Yes No Only" Reposition and align as needed. In the code window, in the click procedure for this button, type the following code:

End Sub	
Private Sub CommandButton3_Click()	Style Value
Dim message As Integer	4
<pre>message = MsgBox("Click Yes to Continue, No</pre>	to Stop", vbYesNo, "Login")
If message = 6 Then	Alternate
Range("D25").Value = "You may proceed"	conditional
ActiveWorkbook.Activate	statement setup
FigeIf message = 7 Then	
Liberi mebbage / inen	tor message
ActiveWorkbook.Save	tor message
ActiveWorkbook.Save ActiveWorkbook.Close	for message box
ActiveWorkbook.Save ActiveWorkbook.Close End If	for message box
ActiveWorkbook.Save ActiveWorkbook.Close End If End Sub	for message box



Test the button.

D	E	F	G	Н	1	J	K	L
	Simple Me	ssage Box						
Me	Message Box With Options							
Mess	ag <mark>e</mark> Box W	ith Yes No C	Only		Login Click Yes to) Continue, M	No to Stop	-23
						Yes	No	



The button is working successfully.

	А	В	С	D	E	F	G	Н
1								
2								
3								
4					Simple Me	accade Roy		
5					on pre me			
6								
7								
8								
9								
10								
11					_			
12				Me	ssage Box	With Optic	ons	
13								
14								
15								
16								
17				· (· · · · · · · · · · · · · · · · · ·				
18								
19				Mess	sage Box W	/ith Yes No	Only	
20								
21				·				
22								
23								
24								
25				You may p	roceed	•		— W
26								st
27								l fr



2.5 Third VBA Project

Problem Statement:

Develop an application a city traffic engineer can use to determine whether it is justified to install speed humps on a residential street. There have been complaints of speeding and cut-through traffic. The applicable laws, rules, and industry guidelines are as follows:

- a) The street must be owned and maintained by the City
- b) The road classification must be local access street.
- c) The number of lanes cannot be more than 2.
- d) The posted speed limit must be 40 mph or less.
- e) The average daily traffic must be more than 500 vehicles per day
- f) The 85th percentile speed must at least 12 mph over the posted speed limit
- g) At least 65% of all traffic is cut-through traffic.

For the speed humps to be approved, a) through e) must apply in addition to a combination of e) and f) or e) and g).

The engineer will be using this tool on a laptop in the field. To facilitate the data entry in the field, the engineer prefers to be able to select the applicable criteria from drop down menus rather than typing in all the information. The engineer wants to have a click button which will cause a text box to indicate that the speed humps are or approved or not approved. A pop up will then inform the engineer that all commands have been executed to completion.

Solution:

Based on the criteria and the requirements for this tool, the form will have 7 combo boxes (drop downs), and a text box to display the result. Add two more text boxes so the engineer can enter his/ her name and date. Give the form a caption such as "Neighborhood Traffic Analyst" and add a logo onto the form.

The framework proposed is as follows:





The inputs in the evaluation for speed humps will come predominantly from drop down menus (combo boxes) as requested by the engineer. The form has been split into four sections using the **Frame** control. All buttons pulled onto a Frame can be moved around as one block by clicking and dragging on the Frame. The frames provide clarity by enhancing organization of the controls and being visually appealing. The Caption property of a frame is used to change the wording on the top of the frame.

On clicking the Run button the calculations will execute and a result indicating that speed humps are approved or otherwise will appear in the Result text box in the Analysis frame. One frame holds the company logo. This is an image control dragged onto and sized to fill the frame. By selecting the blank image control and going to its property **Picture**, the programmer can navigate to a picture on the computer and select it to be assigned to the image control.





Combo Boxes

The combo boxes are currently empty. To populate them, code has to be written to load a drop down list of items. This can be done in a number of ways. One common approach is to add the code to the event of opening or activating the main form. The syntax is

UserFormname.ComboBoxname.AddItem "the item"

This code will be repeated for each item in the drop down list. The process will then be repeated for each combo box on the form as follows:



Maintenance Responsibility:

Double click on the form to open the code window. The default event for the form is **Click**, change it to **Activate** in the event menu, or in the code window, simply type over the "Click" to replace it with "Activate".

U	serForm 🗸	Click
	Option Explicit	Activate
	Private Sub CommandButton2_Click()	BeforeDragOver BeforeDropOrPaste
	'Application.Visible = True	DblClick Deactivate Error
	ThisWorkbook.Save	Initialize KeyDown
	'Unload UserForm1	KeyPress KeyUp
	Application.Quit	
	Private Sub UserForm_Click()	
	End Sub	



Now add the list items for Maintenance Responsibility as follows:

UserForm	✓ Activate
Option Explicit	
<pre>Private Sub CommandButton2_Click()</pre>	
'Application.Visible = True	
ThisWorkbook.Save	
'Unload UserForm1	
Application.Quit	
End Sub	
Private Sub TextBox2_Change()	
End Sub	
Private Sub UserForm_Activate()	
'when the form activates (opens) fill the list items	s in the combo boxes
<pre>'maintenance reponsibility UserForm1.ComboBox1.AddItem "City" UserForm1.ComboBox1.AddItem "County" UserForm1.ComboBox1.AddItem "State" UserForm1.ComboBox1.AddItem "Federal Agency"</pre>	
End Sub	
Drivate Sub HeerForm Click()	

Private Sub UserForm Click()



Save your work, as a Macros-Enabled Excel Workbook.

Test the drop down.

Run the form and select the Maintenance Responsibility drop down.

NEIGHBORHOOD TRAFFIC CALMING	
Roadway Data Maintenance Responsibility: Image: City Road Classification: City County State Federal Agency Number of Lanes: Posted Speed Limit (mph):	Traffic Data Average Daily Traffic: 85th Percentile Speed (mph):
	Analysis RUN Result EXIT



Select a drop down item to ensure it populates the box without any error messages. This has been a success.

NEIGHBORHOOD TRAFFIC CALMING	
Roadway Data Maintenance Responsibility: City Road Classification: Number of Lanes: Posted Speed Limit (mph):	Traffic Data Average Daily Traffic: 85th Percentile Speed (mph): Cut Through Traffic (%):
	Analysis RUN Result



The process will be repeated for all the other combo boxes. Append the code to the Maintenance Responsibility code in the UserForm Activate event.

Road Classification:

UserForm Activate		
	Private Sub UserForm_Activate()	
	'when the form activates (opens) fill the list items in the combo boxes	
	'maintenance reponsibility	
	UserForm1.ComboBox1.AddItem "City"	
	UserForm1.ComboBox1.AddItem "County"	
	UserForm1.ComboBox1.AddItem "State"	
	UserForm1.ComboBox1.AddItem "Federal Agency"	
	'Road Classification	
	UserForm1.ComboBox2.AddItem "Principal Arterial"	
	UserForm1.ComboBox2.AddItem "Minor Arterial"	
	UserForm1.ComboBox2.AddItem "Other Arterial"	
	UserForm1.ComboBox2.AddItem "Major Collector"	
	UserForm1.ComboBox2.AddItem "Minor Collector"	
	UserForm1.ComboBox2.AddItem "Local"	


Number of Lanes:

UserForm Activate • Private Sub UserForm Activate() 'when the form activates (opens) fill the list items in the combo boxes 'maintenance reponsibility UserForm1.ComboBox1.AddItem "City" UserForm1.ComboBox1.AddItem "County" UserForm1.ComboBox1.AddItem "State" UserForm1.ComboBox1.AddItem "Federal Agency" 'Road Classification UserForm1.ComboBox2.AddItem "Principal Arterial" UserForm1.ComboBox2.AddItem "Minor Arterial" UserForm1.ComboBox2.AddItem "Other Arterial" UserForm1.ComboBox2.AddItem "Major Collector" UserForm1.ComboBox2.AddItem "Minor Collector" UserForm1.ComboBox2.AddItem "Local" 'Number of lanes UserForm1.ComboBox3.AddItem "2" UserForm1.ComboBox3.AddItem "3" UserForm1.ComboBox3.AddItem "4" UserForm1.ComboBox3.AddItem "6" UserForm1.ComboBox3.AddItem "8"



Posted Speed Limit:

l	IserForm	•	Activate
	UserForm1.ComboBox2.AddItem	"Major Collector"	
	UserForm1.ComboBox2.AddItem	"Minor Collector"	
	UserForm1.ComboBox2.AddItem	"Local"	
	'Number of lanes		
	HerForm1 ComboBox3 AddItem	121	
	UserForm1 ComboBox3 AddItem	"3"	
	UserForm1 ComboBox3 AddItem	5 "4"	
	UserForm1 ComboBox3 AddItem	"6"	
	UserForm1 ComboBox3 AddItem	"8"	
	USEITUTAT.COMDODOX3.Additem	0	
	'posted speed limit		
	UserForm1.ComboBox4.AddItem	"15"	
	UserForm1.ComboBox4.AddItem	"20"	
	UserForm1.ComboBox4.AddItem	"25"	
	UserForm1.ComboBox4.AddItem	"30"	
	UserForm1.ComboBox4.AddItem	"35"	
	UserForm1.ComboBox4.AddItem	"40"	
	UserForm1.ComboBox4.AddItem	"45"	
	UserForm1.ComboBox4.AddItem	"50"	
	UserForm1.ComboBox4.AddItem	"55"	
	UserForm1.ComboBox4.AddItem	"60"	
	UserForm1.ComboBox4.AddItem	"65"	
	UserForm1.ComboBox4.AddItem	"70"	
	UserForm1.ComboBox4.AddItem	"70"	



85th Percentile Speed:

This speed is calculated from the traffic data collected by the traffic monitoring device that provides the average daily traffic. If the speed of all the vehicles recorded were arranged in ascending order, one value will have 85% of all the data below it. This is the 85th percentile speed. In the United States, generally, the 85th percentile speed is used as a baseline to set the speed limit on a particular roadway.

UserForm	
UserForm1.ComboBox3.AddItem	"6"
UserForm1.ComboBox3.AddItem	"8"
'posted speed limit	
UserForm1.ComboBox4.AddItem	"15"
UserForm1.ComboBox4.AddItem	"20"
UserForm1.ComboBox4.AddItem	"25"
UserForm1.ComboBox4.AddItem	"30"
UserForm1.ComboBox4.AddItem	"35"
UserForm1.ComboBox4.AddItem	"40"
UserForm1.ComboBox4.AddItem	"45"
UserForm1.ComboBox4.AddItem	"50"
UserForm1.ComboBox4.AddItem	"55"
UserForm1.ComboBox4.AddItem	"60"
UserForm1.ComboBox4.AddItem	"65"
UserForm1.ComboBox4.AddItem	"70"
'85th percentile speed	
UserForm1.ComboBox6.AddItem	"15"
UserForm1.ComboBox6.AddItem	"20"
UserForm1.ComboBox6.AddItem	"25"
UserForm1.ComboBox6.AddItem	"30"
UserForm1.ComboBox6.AddItem	"35"
UserForm1.ComboBox6.AddItem	"40"
UserForm1.ComboBox6.AddItem	"45"
UserForm1.ComboBox6.AddItem	"50"
UserForm1.ComboBox6.AddItem	"55"
UserForm1.ComboBox6.AddItem	"60"
UserForm1.ComboBox6.AddItem	"65"



Cut-Through Traffic:

This is the volume of traffic trips that do not originate on the street and do not have a location on the street as the destination for the traffic trip. These are vehicles that are using this street as a by-pass route to get to wherever they are going. In this model, cut-throughs will be expressed as a percentage of the average daily traffic volume.

```
'cut through traffic
UserForm1.ComboBox7.AddItem "0"
UserForm1.ComboBox7.AddItem "5"
UserForm1.ComboBox7.AddItem "10"
UserForm1.ComboBox7.AddItem "15"
UserForm1.ComboBox7.AddItem "20"
UserForm1.ComboBox7.AddItem "25"
UserForm1.ComboBox7.AddItem "30"
UserForm1.ComboBox7.AddItem "35"
UserForm1.ComboBox7.AddItem "40"
UserForm1.ComboBox7.AddItem "45"
UserForm1.ComboBox7.AddItem "50"
UserForm1.ComboBox7.AddItem "55"
UserForm1.ComboBox7.AddItem "60"
UserForm1.ComboBox7.AddItem "65"
UserForm1.ComboBox7.AddItem "70"
UserForm1.ComboBox7.AddItem "75"
UserForm1.ComboBox7.AddItem "80"
UserForm1.ComboBox7.AddItem "85"
UserForm1.ComboBox7.AddItem "90"
UserForm1.ComboBox7.AddItem "95"
UserForm1.ComboBox7.AddItem "100"
```



All combo boxes have the property **Locked** set to **False**. This means that the combo boxes are not locked and that users can type in entries in addition to selecting from the drop down list if they so choose.

Properties - ComboBo	х7	>	<								1		0					7)				•						
ComboBox7 ComboBo	x	•] [1																-	: :	:						
Alphabetic Categorize	ed				L																	1	:	:		:	: :			•
ColumnHeads	False	*	111	:Ŀ	· · ·	· · ·	· ·		•••	•••	· ·		• •	 	·	 	÷		· ·	· ·	• •	• • •	÷	:	Ŀ÷	•	••••	•		•
ColumnWidths							::	: :				: :	:		:		:		: :	. :	: :		: :	:	: :	:		:		
ControlSource				: :			::	: :				: :	:		:		:		: :	. :	: :		: :	:	: :	:		:		
ControlTipText				::			::	::	: :		: :	: :	:		:		:		: :	. :	::	: :	: :	:	: :	:		:	•	:
DragBehavior	0 - fmDragBehaviorDisablec			11			::	::	: :		: :	: :	:	: :	:	: :	:	: :	::	. :	::	: :	: :	:	: :	:	: :	:		
DropButtonStyle	1 - fmDropButtonStyleArrow			::			::	::	: :	: :	: :	: :	:	: :	:	: :	:	: :	: :	: :	::	::	: :	:	: :	:	::	:		
Enabled	True			11	: : :		11	1	1	: :	: :	: :	1	: :	1	: :	1		11	: :	11	1	: :	1	: :	1	: :	1		:
EnterFieldBehavior	0 - fmEnterFieldBehaviorSe																													
Font	Tahoma																													
ForeColor	&H8000008&																													
Height	18																													
HelpContextID	0																													
HideSelection	True																													
IMEMode	0 - fmIMEModeNoControl																													
Left	108																													
ListRows	8																													
ListStyle	0 - fmListStylePlain																													
Listwidth	, Opi			-																										
Locked	False				-								_																	
MatchEntry	1 - fmMatchEntryComplete				÷.																									
MatchRequired	False																													
MaxLength	0																													
MouseIcon	(None)																													
MousePointer	0 - fmMousePointerDefault	_																												
RowSource		=																												
SelectionMargin	True																													
ShowDropButtonWhen	2 - fmShowDropButtonWhe																													
SpecialEffect	2 - fmSpecialEffectSunken																													
Style	0 - fmStyleDropDownComb																													
T_LT_J	2																													



At this stage all input data has been provided on the form for the calculation to proceed. Based on the narrative of the problem, the following algorithm models the evaluation process.



Figure 1: Algorithm for Third VBA Project











Figure 1 (continued): Algorithm for Third VBA Project



The evaluation process will execute upon clicking the Run button. Therefore the Run button's click event shall contain all variables needed for the evaluation to proceed. Declare variables for each input. Note that as code is typed the VBA Library provides tips and pointers to guide the programmer.

CommandButton1 Click() 'this is the Run button 'variables for maintenance responsibility, and roadway calssification Dim strMaintResp, strRoadClass As String 'variables for number of lans, posted speed limit Dim intNumLanes, intPostSpeed as Integer integer iRibbonControl RibbonControl RibbonControl RibbonCutrol Ribbo		
Private Sub CommandButton1_Click() 'this is the Run button 'variables for maintenance responsibility, and roadway calssification Dim strMaintResp, strRoadClass As String 'variables for number of lans, posted speed limit Dim intNumLanes, intPostSpeed as Integer	CommandButton1	Click
<pre>'variables for number of lans, posted speed limit Dim intNumLanes, intPostSpeed as Integer Interior IPictureDisp IRibbonControl IRibbonEtensibility IRibbonUl IRibbonUl IRitdServer</pre>	Private Sub CommandButton1_Click() 'this is the Run button 'variables for maintenance responsibility, and roadway calssification Dim strMaintResp, strRoadClass As String	
Read Code	<pre>'variables for number of lans, posted speed limit Dim intNumLanes, intPostSpeed as Integer Intege</pre>	
	End Sub	



Note that the programmer may choose any name of their choice for variables provided they meet the variable naming rules and conventions discussed in Chapter 3. The final list of variable declarations used in this application is as follows:

CommandButton1

End Sub

```
Private Sub CommandButton1_Click()
'this is the Run button
'variables for maintenance responsibility, and roadway calssification
Dim strMaintResp, strRoadClass As String
'variables for number of lans, posted speed limit
Dim intNumLanes, intPostSpeed As Integer
'variabel to hold traffic volume. As this may be very large, use
'a Long type rather than Integer type
Dim lngAvDailyTraffic As Long
'Variables for 85th percentile and cut through percentage
'use double type as these may involve decimal values
Dim dbl85Percent, dblCutThu As Double
```



The next step is the assignment of values to the variables from the form. For example, *ComboBox1* contains the information that will be stored in the maintenance responsibility variable; *TextBox1* contains the average daily traffic, and so on. The variable assignments are follows:

```
Private Sub CommandButton1_Click()
'this is the Run button
'variables for maintenance responsibility, and roadway calssification
Dim strMaintResp, strRoadClass As String
'variables for number of lans, posted speed limit
Dim intNumLanes, intPostSpeed As Integer
'variabel to hold traffic volume. As this may be very large, use
'a Long type rather than Integer type
Dim lngAvDailyTraffic As Long
'Variables for 85th percentile and cut through percentage
'use double type as these may involve decimal values
Dim dbl85Percent, dblCutThu As Double
'variable assignments
strMaintResp = UserForm1.ComboBox1.Value
strRoadClass = UserForm1.ComboBox2.Value
intNumLanes = UserForm1.ComboBox3.Value
intPostSpeed = UserForm1.ComboBox4.Value
lngAvDailyTraffic = UserForm1.TextBox4.Value
db185Percent = UserForm1.ComboBox6.Value
dblCutThu = UserForm1.ComboBox7.Value
End Sub
```



The speed humps will not be approved if the street is not under City maintenance responsibility. It will not be approved if the posted speed limit exceeds 40 mph etc. These requirements or conditions must be implemented using conditional statements with the appropriate logical expressions. Note that the string variables must have their values wrapped in double quotation marks whereas the numerical variables do not.

```
If strMaintResp <> "City" Or strRoadClass <> "Local" Or __
intNumLanes > 2 Or intPostSpeed > 40 Then
UserForm1.TextBox2.Value = "SPEED HUMPS NOT APPROVED"
Else
If lngAvDailyTraffic > 500 And (dbl85Percent - intPostSpeed) > 12 Then
UserForm1.TextBox2.Value = "SPEED HUMPS APPROVED"
ElseIf lngAvDailyTraffic > 500 And dblCutThu > 65 Then
UserForm1.TextBox2.Value = "SPEED HUMPS APPROVED"
End If
End If
```

Note the use of a nested *If* statement in addition to the composite logical statements. Think of other ways this can be accomplished.

Note the use of a line continuation at the end of the first line. This is used if the line of code becomes excessively long. A line continuation is created in the following way Hit the spacebar. Type an underscore. Hit Enter.

It may happen that a user may forget to enter information in a box where it is required. This may cause the program to produce a run-time error. To prevent this, code must be added that will alert



the user that there is missing data in the process and will terminate the procedure, enabling the user to go back, review, and make the necessary corrections.

Add an *If* statement that checks if is there are any controls (or variables) with null values, and if so alert the user with a message box and prematurely exit the procedure. This block of code shall be placed anywhere before the variable assignments.

```
Variables for 85th percentile and cut through percentage
'use double type as these may involve decimal values
Dim db185Percent, db1CutThu As Double
'check there are no missing data
If UserForm1.ComboBox1.Value = "" Or UserForm1.ComboBox2.Value = "" Or UserForm1.ComboBox3.Value = "" Or UserForm1.ComboBox4.Value = "" Or UserForm1.ComboBox1.Value = "" Or UserForm1.ComboBox7.Value = "" Or UserForm1.TextBox1.Value = "" Then
MsgBox "Please fill out all input requirements.", vb0KOnly + vbExclamation, "Missing Data"
'exit the procedure and start all over
Exit Sub
End If
'variable assignments
strMaintResp = UserForm1.ComboBox1.Value
extPloadClass = UserForm1.ComboBox1.Value
```



Add a message box at the very end of the procedure alerting the user that the program has run to completion.



The program may now be debugged and compiled before testing.





A compile error is detected and highlighted. The traffic volume variable is being assigned the value in *Textbox4* which does not exist. This is a typo and needs to be changed to *TextBox1*.



The error is corrected and the application re-saved.



Testing may now proceed. Click the Run button.

Roadway Data		Traffic Data		_
Maintenance Responsibility:	•	Average Daily Traffic:		
Road Classification:	-	85th Percentile Speed (mph):	
Number of Lanes:	•	Cut Through Traffic (%):	•	
Posted Speed Limit (mph):			_	
	Please fill out a	all input requirements.		
		ОК		
				_



Fill out the form in its entirety. Click the Run button to conduct the evaluation.

- Roadway Data			- Traffic Data		
Maintenance Responsibility:	City	•	Average Daily Traffic:	1200	_
Road Classification:	Local	-	85th Percentile Speed (mph):	45	•
Number of Lanes:	2	•	Cut Through Traffic (%):	71	-
Posted Speed Limit (mph):	30	-			
	30 35 40 45 50 55 60 65		Analysis RUN Result		
				EXI	T ²



The test is a success. All requirements and features requested by the engineer have been accomplished.

Roadway Data		Traffic Data		T
Maintenance Responsibility:	City 💌	Average Daily Traffic:	1200	
Road Classification:	Local 🗨	85th Percentile Speed (mph):	45 💌	
Number of Lanes:	2 🗸	Cut Through Traffic (%):	71 💌	
Posted Speed Limit (mph):	30 💌			
	COMPLETE			
¢.	ALL COMM	IANDS EXECUTED	_	
		RUN		
		OK		
		SPEED HUMPS APPROVE	ED	
				1
				2



The Exit Button:

The Exit button shall close out the form and close out of Excel entirely when clicked on. In design time select the Exit button.

Click on **View code**.

The syntax to close an open Excel workbook is as follows;





As seen, this tool opens upon activating it through the Excel program. It also opens with the Excel program in the back ground. The client wants this tool to open automatically and as a stand-alone, with no other program open in the background. To write the code for this, open code window for this Excel workbook as follows:

Double click on ThisWorkbook in the Project Window.





In the default **Open** procedure enter the following code



Therefore on clicking to open this Excel file, once it opens, the Excel background will become invisible, and then the form will open.

Save and close out of Excel



Navigate to the folder in which this file resides. Double click on the file to open it.





The form opens with the Excel spreadsheet background deactivated.

Comput					
€ C v intro_prog	gramming 🕨 submittal 🕨 tute	orial_files 🔹 🗸	Search tutorial_files	٩	
Organize 👻 Open	▼ Print Burn Ne	ew folder	8	• 🔟 🔞	
Recycle I 🔆 Favorites	Name	Date modified 12/26/2013 8:27	Type 7 AM Microsoft Excel M	Size 22 KB	
Downloads	FourthVBAProject SecondVBAProject	12/26/2013 8:23 12/26/2013 8:24	AM Microsoft Excel M AM Microsoft Excel M	1,266 KB 20 KB	
Adaba	ThirdVBAProject	12/26/2013 11:5	i2 Microsoft Excel M	28 KB	
Reader Libraries	VBAExamples	12/26/2013 8:49 12/26/2013 11:5	AM Microsoft Excel M i4 Microsoft Excel M	85 KB 1 KB	
H.T.E Videos		NEIGHBORHOOD TRAFFIC CALMIN	NG		X
Computer		Roadway Data Maintenance Responsibility:	Traffic Average I S5th Perc	Daily Traffic:	
Cel 20: Favorites graded_work intro_programi lane_closure		Number of Lanes:	Cut Throu	ugh Traffic (%):	•
tlook 2 Malabar My V-drive			Analys	sis RUN	1
hicrosoft Excel	ct Macro-Enabled Worksheet Aut	200		Result	
🙆 🌯 🦓					
kio Creator sanfilippo Home		L			
imeClock					EXIT



If it ever becomes necessary to reactivate the Excel background for example to update some codes, perform the following: Save and close out of Excel. Open the folder containing the file. Hold down the **Shift** key Double click on the file to open it The full Excel view is opened.



The test is a success. This project has been completed to the satisfaction of the client.



3. LOOPING

Looping is a procedure in a programming language that performs repetitive (iterative) tasks. The loop is a sequence of instructions that is executed repeatedly while or until some condition is met or satisfied. Looping is fundamental to all programming languages. Most programming languages have numerous looping constructs. VBA has two types loops based on whether the number of iterations is known beforehand or otherwise.

3.1 For-Next Loop

In VBA this type of loop is used when it is pre-known how many times the loop will be repeated. The syntax is

For loopvariable 1 To n

Code that is to be repeated n times

Next loopvariable

The **loop variable** (or **loop counter variable**) is an integer or long type. It will start at 1, and the code will run. The *Next* will increment the loop variable to 2 and send it back to the *For* line where it will be checked against the n value. If it is less than or equal to n the next iteration of the code proceeds, if it is greater than n the loop has run to completion and the code will not be repeated further. The cursor then moves to the line after the *Next*.

The loop variable call on the *Next* line is optional in VBA, and is by and large omitted or made a comment. If the increment needed is to be in steps of 2 or more the code is as follows

For loopvariable 1 To n Step m

Code that is to be repeated up to n/m times

Next

In this case the loop variable will jump from 1 to 1 + m, and so on, up to the value of n.



Loops can also be set up to run "backwards", for example

For loopvariable n To 1 Step -1

Code that is to be repeated n times but in reverse order

Next

Example:

In this example the main body of the loop is such that the value of the loop variable is displayed in a cell on the spreadsheet in the column 1 (or A). The cell row designator is the loop variable i, therefore each time i increments, the display location on the spreadsheet will jump to the next row below, hence the results filling down the spreadsheet. Change the number of iterations to 10,000 and see what happens.

	E14	1	• ()	f_{x}											
	A	В	С	D	Formula Bar	G	Н	L	J	K	L	М	N	C)
1	1														
2	2					2	Microsoft Vi	sual Basic -	vba tutorial.	xlsm [desiar	1 - [Sheet7 (Code)]			
3	3					1	h Eile Ed	it View	Incart Form	nat Dehu	Dun T	Cools Add	Inc Wind	Ll	da
4	4				CommandButton1	: 0-			insen r <u>o</u> n		y <u>r</u> un <u>i</u>	Loois Add.	-ins <u>vv</u> inu	оw <u>п</u>	eib
5	5					: 2	S 🔚 - 🎽	1% D		Ci 🕨	🗹	S 🖀 🖁	i 🛠 🕜	Ln 8, 0	Col 5
6	6					Proj	ect - VBAPro	oject	X	Comma	ndButton1				
7	7								Ŧ	Opt	ion Expl	licit			
8	8						🕺 Interne	t_Assistant	(HTML.XLA						
9	9					. E	SOLVER	(SOLVER.XL	AM)	Pri	vate Sub	o Command	Button1	Clic	c ()
10	10					Ē.	🐉 sumif (S	SUMIF.XLAM)			1.0000000			
11							VBAProj	ect (FUNCR	torial vism	Din	1 1 AS II	iteger			
12							Modu	le1	ICON UNIXISIII	For	i = 1 1	To 10			
13							Bhee	t1 (Sheet1)			Cells (i	i, 1).Val	lue = i		
14							Bhee	t2 (Sheet2)		Nex	t				
15							Bhee	t3 (Sheet3)							
16							I Shee	t5 (Sheet5)		Enc	Sub				
17							Bhee	t6 (Sheet6)			C. Strategy				
18							- Di Shee	t7 (Sheet7)							
19							*************************************	Vorkbook							
20															
21															
22															



3.2 Nested Loops

A nested loop is a loop inside of another loop.

Example:

Fill cells A18 through E35 with a value which is calculated by addition of the cell's row number and its column number.

	А	В	С	D	E	F	G	Н	1	J		K	L	М	Ν	0	Р
13																	
14							/ 🥭 N	Aicrosoft Vis	ual Basic -	vba tutorial v	lsm -	[Sheet]	7 (Code)]				
15					Comman	dButton2	i naka	Ella E-lla	View	Incert Econ		Dahum	Dun .	Teels Add	Inc. Mind	uu Hala	
16							: 😂	<u>File E</u> dit	<u>v</u> iew	insert F <u>o</u> rm	lat	Debug	Kun	Tools <u>A</u> dd	-ins <u>w</u> ind	ow <u>H</u> eip	
17								- 🚽	XD	12 # 9	6	▶ 00	i 🗉 🕍	8 🖀 🖁	; 🔆 🛛	Ln 18, Col 1	17
18	19	20	21	22	23		Proje	ct - VBAPro	ject	×	C	ommar	ndButton2				
19	20	21	22	23	24			-8		÷	h	Ont	ion Exp	licit			
20	21	22	23	24	25			🕺 Internet	Assistan	t (HTML.XLA		opt.	ION EAD	11010			
21	22	23	24	25	26		÷	SOLVER	_ (SOLVER.X	LAM)		Priv	vate Su	b Command	dButton1	Click()	
22	23	24	25	26	27		÷	sumif (S	UMIF.XLAI	1)							
23	24	25	26	27	28		±…£	S VBAProje	ect (FUNCI ect (vha t	RES.XLAM) utorial vism		Dim	1 AS I	nteger			
24	25	26	27	28	29			Modul	e1			For	i = 1	To 10			
25	26	27	28	29	30			Bheet	1 (Sheet1)				Cells(i, 1).Val	lue = i		
26	27	28	29	30	31			Sheet	2 (Sheet2)			Next	t				
27	28	29	30	31	32			Sheet	3 (Sheet3) 4 (Sheet4)								
28	29	30	31	32	33			Bheet	5 (Sheet5)			End	Sub				
29	30	31	32	33	34				6 (Sheet6)								
30	31	32	33	34	35			Bheet	7 (Sheet7)			Priv	vate Su	b Command	dButton2_	Click()	
31	32	33	34	35	36			······································	orkbook			Dim	i i b	s Integer	~		
32	33	34	35	36	37							Dim	1, J A	o incege	-		
33	34	35	36	37	38							For	i = 18	To 35		_	
34	35	36	37	38	39					Loop to	6 J		For j	= 1 To 5			
35	36	37	38	39	40					fill row			Nevt i	115(1,])	.value =	• 1 +]	
36										1111 10 1		Next	t i		Nest	ed Lo	on to
37															fill		
38												E a d	Curb			olullill	8
39												End	อนอ				
40																	
41																	
42																	



3.3 Do-Loop

If the number of iterations needed is not known *a piori*, then the *For-Next* loop cannot be used. In that case a *Do-Loop* structure will have to be used. In the *Do-Loop*, the programmer sets conditions upon which the looping will terminate but will not know how much iteration will actually run until termination occurs.

The syntax is as follows:

Do while termination condition

Code to be repeated

Loop

The termination condition is generally some logical expression. After each iteration, the termination condition will be checked. If it has been met the loop will stop and the cursor moves to the line after the "Loop" line, otherwise it will move to the next iteration of the code in the main body of the loop.

The termination condition must be chosen carefully and studied closely otherwise the program may fall into an infinite loop. An infinite loop is a loop that lacks a functioning exit routine . As a result the loop repeats continuously until the operating system senses the issue and terminates the program, or until some event, for instance having the program terminate automatically after a certain duration or number of iterations, occurs. Infinite loops are capable of easily crashing personal computers.

There are other styles of the Do-Loop, namely

- Do.....Loop While
- Do until.....Loop
- Do.....Loop until

Each style requires the termination statement be set up in a certain way that is consistent with the logic of that style.



Example:

C	ommandButton1	<u>`</u>	A1		• (>	f _x	x		
	Option Explicit		А	В	C	D		F	F
	Private Sub CommandButton1 Click()	1	x	Y	X+Y	-	-	_	
		2	1	2	3				
	No	3	2	4	6		Cor	nmandButt	on1
	Dim counter, sum As Integer	4	3	6	9				
		5	4	8	12				
		6	5	10	15				
		7	6	12	18				
		8	7	14	21				
		9	8	16	24				
	Cells(1, 1) = "X"	10	9	18	27				
	Cells(1, 2) = "Y" Cells(1, 3) = "X+Y"	11	10	20	30				
		12							
	Do While counter < 10	13							
	counter = counter + 1	••							
	Cells(counter + 1, 1) = counter $($ Cells(counter + 1, 2) = counter $($ 2								
	<pre>sum = Cells(counter + 1, 1) + Cells(counter Cells(counter + 1, 3) = sum Loop</pre>	er +	1, 2)						
	End Sub 'In this example, the program will display the 'The value of Y is X2 and the values are displ 'Finally, it shows the values of X+Y in column	e va Lay 13,	lues of) in column i.e. fro	X in cel n 2, i.e om cells	ls(1,1) t . from ce s(3,1) to	o cel 11s(2 cells	lls(2,1) s(3,	11,1). to cells 11)	9(2,11).

In this simplified example the loop variable is *counter*. Note that unlike the *For-Next* loop where the program automatically increments the loop variable, in the *Do-Loop*, the loop variable must be incremented "manually" by the programmer. However this makes the incrementing more flexible than in the *For-Next* loop. In this example the termination condition is based on the value of the counter variable, however that is not always the case. The termination condition may be based on any of the variables in the main body of the loop and based on specific requirements of the application. Do-Loops may be nested with each other and they may be nested within For-Next loops and vice versa.



3.4 Exiting a Loop

In some cases it may be necessary to abruptly or prematurely exit a loop based on the progress of the program. The syntax is

Exit Do Or Exit For

The program will exit the loop at the location the *Exit* code is written. The program will exit out of the current (or most recent) loop. Typically the *Exit* command will be associated with some conditional statement nested in the loop such that if that condition is met abort the loop and move to some other area of the code or to some other relevant command. The *Exit* code can also be called to exit out of a conditional statement (*Exit If*). The advantage of that being if for example the conditions) and related code that need to be checked or evaluated, the program will exit out once the first "True" condition is encountered, and not have to continue evaluating through the other hundreds or thousands of other conditions and instructions.

3.5 Fourth VBA Project

Problem Statement:

A Florida bridge engineer maintains a large bridge inventory with the following attributes:

	A	В	С	D	E	F	G	Н
1	Bridge_Number	County_Code	Facility_Carried_by_Structure	Features_Under	Year_Built	Num_of_Lanes	Ave_Daily_Traffic	Percent_Truck_Traffic
2	490805	37	#13 ROAD	#13 Rd./ DRAINAGE DITCH	1997	1	25	5
3	124053	71	CRYSTAL DRIVE	10 MILE CANAL	1986	2	10800	2
4	125745	71	WINKLER AVENUE	10 MILE DRAIN	2000	5	27800	10
5	720253	31	I-295 SB (SR-9A)	103rd ST.	1970	3	51500	8
6	720253	31	I-295 SB (SR-9A)	103rd ST.	1970	3	48000	5
7	720346	31	I-295 NB (SR-9A)	103rd ST.	1970	3	51500	8
8	720346	31	I-295 NB (SR-9A)	103rd ST.	1970	3	48000	5
9	870437	86	1-95	119 ST	1962	10	190500	6
10	870437	86	1-95	119 ST	1962	10	20100	2
11	720132	31	US-ALT-1 (SR-115)	11th ST.	1970	4	49500	4
12	720132	31	US-ALT-1 (SR-115)	11th ST.	1970	4	49500	4
13	790013	127	US-1	11th St. Canal	1931	5	28500	3
14	794096	127	Williamson Blvd	11th Street Canal	1994	٩	14168	2



The inventory currently holds 15,332 records that are updated periodically by inspectors. The engineer periodically performs spreadsheet calculations for update and reporting purposes. The inspectors' data entry requirements are different from what the engineer would prefer. For example, the County Codes are given by the US Census Code number but the engineer would prefer the name of the County. The engineer wants a button which when clicked on will run through the entire data and for each record will conduct the following manipulations and calculations:

- 1. Converts the County Code to the County name
- 2. In a separate column calculates the Age of the bridge
- 3. In a separate column calculates Average Daily Truck Traffic by multiplying the Average Daily Traffic by the Percent Truck Traffic

At the beginning of the program the engineer wants pop up to appear stating the number of records that are about to be manipulated and giving the engineer the options to continue or stop the process. At the end of the process a pop up shall appear notifying the engineer that all commands have been successfully executed.

As the program runs it will not be visually possible for the engineer to follow the calculations on the screen. Set up a digital dashboard which displays the following information:

- 1. Number of records in the inventory.
- 2. Current record at which calculations are taking place.
- 3. Number of records remaining to be processed.



Solution:

Insert a command button onto the spreadsheet (alternatively, create a form and add a command button onto the form)

X	🚽 🌒 • (° - 🖓								bennouna_lal	b5_data_Sol - Microsoft Excel
F	File Home	Insert Page La	rout Formulas	Data Reviev	v View	Developer Traffi	c Acrobat			
Vis Ba	sual Macros Asic Code	ord Macro Relative Reference: cro Security e	Add-Ins Add-Ins Add-Ins	Insert Form Controls	Properties View Code Run Dialog	Source Map Prop Expansio Refresh D XM	oerties 📑 In n Packs 📑 E Data IL	xport Document Panel Modify		
	C1	iii 📝 🗹 🊔 🗉	8 0							
	А	В	С	🎬 Aa 불 ab 🗐		D	E	F	G	Н
1	Bridge_Number	County_Code	acility_Carried_by	ActiveX Controls	s 🗟 불 s_Unde	r	Year_Built	Num_of_Lanes	Ave_Daily_Traffic	Percent_Truck_Traffic
2	490805	37	13 ROAD			ACEDITCH	1997	1	25	5
3	124053	71 (RYSTAL DRIVE	Command But	ton (ActiveX Co	ontrol)	1986	2	10800	2
4	125745									-
5		5 71	VINKLER AVENUE	Insert a comm	nand button co	ontrol.	2000	5	27800	10
<u> </u>	720253	5 71 31	VINKLER AVENUE -295 SB (SR-9A)	Insert a comm 103	nand button co Brd ST.	ontrol.	2000 1970	5 3	27800 51500	10 8
6	720253 720253	71 31 31	VINKLER AVENUE -295 SB (SR-9A) -295 SB (SR-9A)	Insert a comm 103 103	nand button co Brd ST. Brd ST.	ontrol.	2000 1970 1970	5 3 3	27800 51500 48000	10 8 5
6 7	720253 720253 720346	71 31 31 31 31	VINKLER AVENUE -295 SB (SR-9A) -295 SB (SR-9A) -295 NB (SR-9A)	Insert a comm 103 103 103	nand button co Brd ST. Brd ST. Brd ST.	ontrol.	2000 1970 1970 1970	5 3 3 3	27800 51500 48000 51500	10 8 5 8
6 7 8	720253 720253 720346 720346	i 71 31 31 i 31 i 31 i 31	VINKLER AVENUE -295 SB (SR-9A) -295 SB (SR-9A) -295 NB (SR-9A) -295 NB (SR-9A)	Insert a comm 103 103 103 103	nand button co Brd ST. Brd ST. Brd ST. Brd ST.	ontrol.	2000 1970 1970 1970 1970	5 3 3 3 3 3	27800 51500 48000 51500 48000	10 8 5 8 5 5
6 7 8 9	720253 720253 720346 720346 720346 870437	5 71 3 31 5 31 5 31 7 86	VINKLER AVENUE 295 SB (SR-9A) 295 SB (SR-9A) 295 NB (SR-9A) 295 NB (SR-9A) -295 NB (SR-9A)	Insert a comn 103 103 103 103 103 103	nand button co Brd ST. Brd ST. Brd ST. Brd ST. 9 ST	ontrol.	2000 1970 1970 1970 1970 1970	5 3 3 3 3 10	27800 51500 48000 51500 48000 190500	10 8 5 8 5 5 6
6 7 8 9 10	720253 720253 720346 720346 870437 870437	5 71 3 31 5 31 5 31 5 31 7 86 7 86	VINKLER AVENUE -295 SB (SR-9A) -295 SB (SR-9A) -295 NB (SR-9A) -295 NB (SR-9A) -95 -95	Insert a comm 103 103 103 103 103 103 119 119	nand button co Brd ST. Brd ST. Brd ST. Brd ST. 9 ST 9 ST	ontrol.	2000 1970 1970 1970 1970 1962 1962	5 3 3 3 10 10	27800 51500 48000 51500 48000 190500 20100	10 8 5 8 5 6 2
6 7 8 9 10 11	720253 720253 720346 720346 870437 870437 720132	71 31 31 31 31 31 31 31 31 31 31 31 31 31	VINKLER AVENUE -295 SB (SR-9A) -295 SB (SR-9A) -295 NB (SR-9A) -295 NB (SR-9A) -95 -95 JS-ALT-1 (SR-115)	Insert a comm 103 103 103 103 103 103 115 115 115	nand button co Brd ST. Brd ST. Brd ST. Brd ST. 9 ST 9 ST h ST.	ontrol.	2000 1970 1970 1970 1970 1962 1962 1970	5 3 3 3 3 10 10 10 4	27800 51500 48000 51500 48000 190500 20100 49500	10 8 5 8 5 6 2 4
6 7 8 9 10 11 12	720253 720253 720346 720346 870437 870437 870437 720132 720132	5 71 3 31 5 31 5 31 5 31 7 86 7 86 7 86 8 31 7 86 8 31 7 8 8 31 8 31	VINKLER AVENUE -295 SB (SR-9A) -295 SB (SR-9A) -295 NB (SR-9A) -295 NB (SR-9A) -95 -95 JS-ALT-1 (SR-115) JS-ALT-1 (SR-115)	Insert a comm 103 103 103 103 103 103 115 115 115 115 115	nand button co Brd ST. Brd ST. Brd ST. Brd ST. 9 ST 9 ST h ST. h ST.	ontrol.	2000 1970 1970 1970 1970 1962 1962 1970 1970	5 3 3 3 10 10 4 4	27800 51500 48000 51500 48000 190500 20100 49500 49500	10 8 5 8 5 6 2 4 4



Resize the command button as needed

1ap Properties 📑	Import											
xpansion Packs 👹	Export	<u> </u>										
efresh Data	. [Document										
XMI		Modify										
70012		mouny										
E		F	G	Н		J	K	L	М	N	0	P
res_Under		Year_Bu	It Num_of_Lanes	Ave_Daily_Traffic	Percent_Truck_Traffic							
d./ DRAINAGE D	ITCH	1997	1	25	5							
LE CANAL		1986	2	10800	2							
LE DRAIN		2000	5	27800	10							
ST.		1970	3	51500	8			0				
ST.		1970	3	48000	5			Ĭ		-	Ĺ	
ST.		1970	3	51500	8				_			
ST.		1970	3	48000	5			P	Comm	andButton1	P	
Г		1962	10	190500	6							
Г		1962	10	20100	2					-0		
эт.		1970	4	49500	4							
эт.		1970	4	49500	4							
St. Canal		1931	5	28500	3							
Street Canal		1994	3	14168	2							
Street Canal		1999	2	354	2							
LE SWAMP		1957	5	18900	3							
LE SWAMP		1966	6	66000	15							
LE SWAMP		1966	6	66000	15							
I AVE NORTH		1990	6	50500	3							
I AVE NORTH		1990	6	30000	5							
IAVE		1985	3	61000	10							
LAVE		1985	2	500	20							



Double click on the command button to open the code window

Page Layout		ig <u>R</u> un <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indow <u>H</u> elp	Туре
	I 🛛 🔄 - 🔒 I X 🖻 🛍 🐴 🧐 (* i)	1) 💷 🚾 📽 🎘 🕐 🕐 Ln 2, Col 1 💦 💂	
A	Project - VBAProject	CommandButton1 Click	
,	_ <u> </u>	Private Sub CommandButton1_Click()	
f _x :	Solver (SOLVER.XLAM) WBAProject (bennouna lab5 data Solvis	End Sub	
	Microsoft Excel Objects		
0.1	Sheet3 (bridges) Sheet4 (EL Counties)		
_Code Coun 37 FRAN	ThisWorkbook		
71 LEE	⊡ 🔄 Modules		
71 LEE	Module2		
31 DUVA 31 DUVA	E - S VBAProject (FUNCRES.XLAM)		
31 DUV			
31 DUVA			
86 MAR			
31 DUV/			
31 DUVA			
127 VOLU			
127 VOLU			
109 PUTN			



Convert Census Code to Florida County name. The list of Florida counties by census code may be downloaded from the internet. An abridged list is provided.

Florida Counties

County name	FIPS code	County name	FIPS code
Alachua	1	Okeechobee	93
Baker	3	Orange	95
Bay	5	Osceola	97
Bradford	7	Palm Beach	99
Brevard	9	Pasco	101
Broward	11	Pinellas	103
Calhoun	13	Polk	105
Charlotte	15	Putnam	107
Citrus	17	St. Johns	109
Clay	19	St. Lucie	111
Collier	21	Santa Rosa	113
Columbia	23	Sarasota	115
Dade	25	Seminole	117
DeSoto	27	Sumter	119
Dixie	29	Suwannee	121
Duval	31	Taylor	123
Escambia	33	Union	125
Flagler	35	Volusia	127
Franklin	37	Wakulla	129
Gadsden	39	Walton	131
Gilchrist	41	Washington	133
Glades	43		
Gulf	45		
Hamilton	47		
Hardee	49		
Hendry	51		
Hernando	53		
Highlands	55		
Hillshorough	57		

Figure 2: Plan for Fourth VBA Project


Consider the bridge data is stored on a spreadsheet named "bridges". Also, the county code number is in column "B" (column 2). So for the first record the reference to the county code number will be

Sheets("bridges").cells(2, 2).value

For the county code of the next record on the spreadsheet is

Sheets("bridges").cells(3, 2).value

And so on.

The county code for a record can therefore be generalized as follows

Sheets("bridges").cells(*i*, 2).value

where *i* is the row number for that record.

An *If* statement (or Select-Case statement) may be used to reference county code number to county name as follows:

If Sheets("bridges").cells(i, 2).value = 1 Then

Sheets("bridges").cells(i, 2).value = "Alachua"

Elself

End If

As there are 67 counties in Florida, the *If* statement will have 67 conditions to check. Note that in this code, the name of the county will replace the county code number in column 2. If the intent is to post the county name in a separate column, then the column number in the *cells* code must be changed to that column number and the county name will now be posted in that column at row i.



The full If statement will therefore be of the set up





Save and Compile to check for errors.

Resave.

Next declare the variables that will be used for the calculation of age and truck traffic. (preferably at the top of the procedure)

```
CommandButton1
                                                             Click
                                                          •
  Private Sub CommandButton1 Click()
  'this is the click button on the spreadsheet
  'variable for age of bridge
  Dim intAge As Integer
  'year bridge was built
  Dim intYearBuilt
  'variable for trick traffic
  Dim intAveDailyTruckTraffic As Integer
  'Truck volume will calculated from
  Dim intAveDailyTraffic As Long
  'and % trucks
  Dim intPercentTrucks As Integer
  'assign correct county name from census code
  If Sheets("bridges").Cells(i, 2).Value = 1 Then
               Sheets("bridges").Cells(i, 2).Value = "ALACHUA"
  ElseIf Sheets("bridges").Cells(i, 2).Value = 3 Then
               Sheate ("bridges") Calle (i 2) Value = "RAKED"
```



Next, assign the values of the input variables for the calculations. For example, age of bridge is the current year (2014) minus the year built. The year built is in column "E" (column 5).

```
Private Sub CommandButton1 Click()
'this is the click button on the spreadsheet
'variable for age of bridge
Dim intAge As Integer
'year bridge was built
Dim intYearBuilt
'variable for trick traffic
Dim intAveDailyTruckTraffic As Integer
'Truck volume will calculated from
Dim intAveDailyTraffic As Long
'and % trucks
Dim intPercentTrucks As Integer
'assign values from the data
intYearBuilt = Sheets("bridges").Cells(i, 5).Value
intAveDailyTruckTraffic = Sheets("bridges").Cells(i, 7).Value
intPercentTrucks = Sheets("bridges").Cells(i, 8).Value
'assign correct county name from census code
If Sheets("bridges").Cells(i, 2).Value = 1 Then
            Sheets("bridges").Cells(i, 2).Value = "ALACHUA"
Fleaff Sheate ("bridges") Calle (i - 2) Value = 3 Then
```



Save and Compile to check for errors.

Resave.

Next perform the calculations. For example Truck Traffic is the product of Average Daily Traffic and Percent Trucks in the traffic (divide spreadsheet value by 100).

```
'assign values from the data
intYearBuilt = Sheets("bridges").Cells(i, 5).Value
intAveDailyTraffic = Sheets("bridges").Cells(i, 7).Value
intPercentTrucks = Sheets("bridges").Cells(i, 8).Value
'calculate bridge age
intAge = 2014 - intYearBuilt
'calculate truck traffic
intAveDailyTruckTraffic = intAveDailyTraffic * intPercentTrucks / 100
'assign correct county name from census code
If Sheets("bridges").Cells(i, 2).Value = 1 Then
            Sheets("bridges").Cells(i, 2).Value = "ALACHUA"
ElseIf Sheets("bridges").Cells(i, 2).Value = 3 Then
            Sheets("bridges").Cells(i, 2).Value = "BAKER"
ElseIf Sheets("bridges").Cells(i, 2).Value = 5 Then
            Sheets("bridges").Cells(i, 2).Value = "BAY"
ElseIf Sheets("bridges").Cells(i, 2).Value = 7 Then
```



Save the result to the spreadsheet. Save truck volume to column "I", and Age to column "J".

```
'assign values from the data
intYearBuilt = Sheets("bridges").Cells(i, 5).Value
intAveDailyTraffic = Sheets("bridges").Cells(i, 7).Value
intPercentTrucks = Sheets("bridges").Cells(i, 8).Value
'calculate bridge age
intAge = 2014 - intYearBuilt
Sheets("bridges").Cells(i, 10).Value = intAge -
'calculate truck traffic
intAveDailyTruckTraffic = intAveDailyTraffic * intPercentTrucks / 100
Sheets("bridges").Cells(i, 9).Value = intAveDailyTruckTraffic +
'assign correct county name from census code
If Sheets("bridges").Cells(i, 2).Value = 1 Then
            Sheets("bridges").Cells(i, 2).Value = "ALACHUA"
ElseIf Sheets("bridges").Cells(i, 2).Value = 3 Then
            Sheets("bridges").Cells(i, 2).Value = "BAKER"
ElseIf Sheets("bridges").Cells(i, 2).Value = 5 Then
            Sheets("bridges").Cells(i, 2).Value = "BAY"
ElseIf Sheets("bridges").Cells(i, 2).Value = 7 Then
```



Save and Compile to check for errors.

Resave.

The calculations will be performed for each record (each row in the table) therefore the calculation code must be placed within a For-Next loop so as to iterate down the table for each record. The row number *i* must therefore be in the *For* statement.

Note that it is not necessary to put the variable declaration within the *For* loop, in fact it will cause a run-time error. Once the variables are declared once, they can be recycled through each iteration of the For-Next loop.





Note that a *For* statement must always have a corresponding *Next* statement at the bottom otherwise the syntax is incomplete and a run-time error will occur





Save and Compile to check for errors.

Resave.

On the spreadsheet highlight the columns receiving the calculated results to facilitate review.

4	Α	В	С	D	E	F	G	Н	1	J	
1	Bridge_Number	County_Code	Facility_Carried_by_Structure	Features_Under	Year_Built	Num_of_Lanes	Ave_Daily_Traffic	Percent_Truck_Traffic	Ave Daily Truck Traffic	Age of Bridge	
2	490805	37	#13 ROAD	#13 Rd./ DRAINAGE DITCH	1997	1	25	5			
3	124053	71	CRYSTAL DRIVE	10 MILE CANAL	1986	2	10800	2			
4	125745	71	WINKLER AVENUE	10 MILE DRAIN	2000	5	27800	10			
5	720253	31	I-295 SB (SR-9A)	103rd ST.	1970	3	51500	8			
6	720253	31	I-295 SB (SR-9A)	103rd ST.	1970	3	48000	5			
7	720346	31	I-295 NB (SR-9A)	103rd ST.	1970	3	51500	8			
8	720346	31	I-295 NB (SR-9A)	103rd ST.	1970	3	48000	5			
9	870437	86	1-95	119 ST	1962	10	190500	6			
10	870437	86	I-95	119 ST	1962	10	20100	2			
11	720132	31	US-ALT-1 (SR-115)	11th ST.	1970	4	49500	4			
12	720132	31	US-ALT-1 (SR-115)	11th ST.	1970	4	49500	4			
13	790013	127	US-1	11th St. Canal	1931	5	28500	3			
14	794096	127	Williamson Blvd	11th Street Canal	1994	3	14168	2			
15	795503	127	Alta Drive	11th Street Canal	1999	2	354	2			
16	780022	109	US-1 (SR-5)	12 MILE SWAMP	1957	5	18900	3			

Now set up the digital dashboard. The purpose of the digital dashboard is to enable the user "watch" the progress of the program as it moves down the table performing the calculations, as it will be physically impossible to see the calculations in real-time as the computer monitor is not large enough to display the entire spreadsheet of 15,332 records.

The progress of the program will therefore be monitored by viewing the dashboard information.



In the project, the client requested to be able to track the total number of records, the current record being processed, and the number of records remaining to be processed, on the dashboard.

G	H	l I	J	K	L		M	N	(
Ave_Daily_Traffic	Percent_Truck_Traffic	Ave Daily Truck Traffic	Age of Bridge						
25	5								
10800	2								
27800	10								
51500	8								
48000	5								
51500	8					C			
48000	5					Con	nmandb	utton i	
190500	6								
20100	2				-				
49500	4								
49500	4								
28500	3								
14168	2				DIGITAL D	ASHBOA	ARD		
354	2								
18900	3								
66000	15			Tota	a <mark>l Number o</mark>	of Record	ds		
66000	15								
50500	3								
30000	5				Current Re	ecord			
61000	10								
500	30								
61000	10			Nur	nber of Rec	ords Lef	ft		
500	30								
40500	8								
5000	5								



Save and Compile to check for errors.

Resave.

Code must be added to send the relevant values to the dashboard.

The number of records is known from the spreadsheet as 15,332. Report this number to the dashboard. The current record is the current value of the loop variable *i*. Report it to the dashboard at the start (or end) of each iteration. Subtracting the current record from the total number of records will give the number of records left to be processed.





Next, add a message box at the end that alerts the user that the program has run to completion



It has been requested by the engineer that before the process runs a message box pops up giving a caution that a large data set is about to be processed, and give the user the option to continue or stop the process. Therefore before the *For* loop starts a message box with options must appear

```
'total records
Sheets("bridges").Cells(18, 12).Value = 15532
'caution user that data is very large and processing will
'consume siginificant computer resources
Dim xmassage, zmessage As String
xmessage = MsgBox("You are about to process a very large data set which may tie up computer resources"
               & vbNewLine & "Do you want to continue?", vbYesNo + vbExclamation, "CAUTION")
'so if user selects to continue, start the For loop
If xmessage = vbYes Then
For i = 2 To 15332
'dashboard reporting
'current record
Sheets("bridges").Cells(21, 12).Value = i
'dashboard reporting
'number of records left
Sheets("bridges").Cells(24, 12).Value = 15532 - i
'assign values from the data
intYearBuilt = Sheets("bridges").Cells(i, 5).Value
```



Always remember to complete an If statement with the ElseIf/-Else-End If.

```
status ("bridges").Cells(i, 2).Value = 131 Then
Sheets("bridges").Cells(i, 2).Value = "WALTON"
ElseIf Sheets("bridges").Cells(i, 2).Value = 133 Then
Sheets("bridges").Cells(i, 2).Value = "WASHINGTON"
End If
Next 'i
MsgBox "ALL COMMANDS EXECUTED", vboKOnly, "SUCCESSFUL COMPLETION"
ElseIf xmessage = vbNo Then
'this is from the xmessage question you answered after being warned yo are
'about to process a large data set.
zmessage = MsgBox("You have chosen not to process this large data set.", vboKOnly, "Goodbye")
End If
End Sub
```

Save and Compile to check for errors. Resave.



Test the program.

Deactivate Design mode

Click on the command button.

Test the option where the user chooses not to proceed after the caution message box

В	C	D	E	F	G	Н	1	J	K	L	M	N	
County_Co	de Facility_Carried_by_Structure	Features_Under	Year_Built	Num_of_Lanes	Ave_Daily_Traffic	Percent_Truck_Traffic	Ave Daily Truck Traffic	Age of Bridge					
•	37 #13 ROAD	#13 Rd./ DRAINAGE DITCH	1997	1	25	5							
	71 CRYSTAL DRIVE	10 MILE CANAL	1986	2	10800	2							
	71 WINKLER AVENUE	10 MILE DRAIN	2000	5	27800	10							
	31 I-295 SB (SR-9A)	103rd ST.	1970	3	51500	8							
1	31 I-295 SB (SR-9A)	103rd ST.	1970	3	48000	5							
5	31 I-295 NB (SR-9A)	103rd ST.	1970	3	51500	8							
2	31 I-295 NB (SR-9A)	103rd ST.	1970	3	48000	5				Comment	dian 1		
7	86 1-95	119 ST	1962	10	190500	6				Commanded	atton i		
7	86 1-95	119 ST	1962	10	20100	2					_		
2	31 US-ALT-1 (SR-115)	11th ST.	1970	4	49500	4					_	_	
2	31 US-ALT-1 (SR-115)	11th ST.	1970	4	49500	4							
1 1	27 US-1	11th St. Canal	1931	5	28500	3							
i 1	27 Williamson Blvd	11th Street Canal	1994	3	14168	2				DIGITAL DASH	BOARD		
1 1	27 Alta Drive	11th Street Canal	1999	2	354	2							
1	09 US-1 (SR-5)	12 MILE SWAMP	1957	5	18900	3							
1 1	.09 I-95 (SR-9)	12 MILE SWAMP	1966	6	66000	15			Tota	I Number of Re	ords		
1 1	09 I-95 (SR-9)	12 MILE SWAMP	1966	6	66000	15				15532			
1 1	03 US 19 SR 55	126TH AVE NORTH	1990	CAUTION			83						
7 1	03 US 19 SR 55	126TH AVE NORTH	1990							Current Record			
)	57 I-75 SB (SR-93A)	127TH AVE	1985										
)	57 I-75 SB (SR-93A)	127TH AVE	1985		You are about to pro	cess a very large data set w	hich may tie up						
L	57 I-75 NB (SR-93A)	127TH AVE	1985	-	Do you want to cont	inue?			Nun	nber of Records	Left		
L	57 I-75 NB (SR-93A)	127TH AVE	1985		00 900 Hair to com	and c.							
t .	57 I 275 NB & SB	127TH AVENUE	1966										
5	57 I 275 NB & SB	127TH AVENUE	1966										
5	99 I-95 (SR-9)	12th Avenue South	1975			Yes	No						
1	99 I-95 (SR-9)	12th Avenue South	1975	13	0000	,							
	33 12TH AVENUE	12TH AVENUE/BAYOU TEXAR	1956	4	15000	5							
5	57 SR-618 (CROSSTOWN)	12TH ST TO 26TH ST	2006	3	6341	0							
)	57 SR-618 (CROSSTOWN)	12TH ST TO 26TH ST	2006	3	10000	5							
)	57 SR-618 (CROSSTOWN)	12TH ST TO 26TH ST	2006	3	500	5							
)	57 SR-618 (CROSSTOWN)	12TH ST TO 26TH ST	2006	3	10000	5							
)	57 SR-618 (CROSSTOWN)	12TH ST TO 26TH ST	2006	3	1500	3							
)	57 SR-618 (CROSSTOWN)	12TH ST TO 26TH ST	2006	3	15000	13							



Click on **No**. Success.

25	5				
10800	2				
27800	10				
51500	8				
48000	5				
51500	8				
48000	5				
190500	6		CommandBu	utton1	
20100	2				
49500	4				
49500	4				
28500	3				
14168	2		DIGITAL DASH	BOARD	
354	2				
18900	3				
66000	15		CommandButton1		
66000	15		15532		
50500	3				
Goodbye		×	Current Record		
You have chosen	not to process this large data	a set.	Number of Records	Left	
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,		ОК			
6000	5				
15000	5				
C2.44	~	1		i i i	



Repeat the test.

This time select **Yes** to continue after the caution message.

Dim intPercentTrucks As Integer						
'dashboard reporting 'total records Sheets("bridges") Cells(18, 12) Value = 15532	Microsoft Visual Basic					
Sheebs (Shinges). Series (15, 12). Saine 19992	Run-time error '6': Overflow					
'caution user that data is very large and processing will 'consume siginificant computer resources						
Dim xmassage, zmessage As String						
<pre>xmessage = MsgBox("You are about to process a very large</pre>	Continue End Debug Help					
'so if user selects to continue, start the For loop If xmessage = vbYes Then						
For i = 2 To 15332						
<pre>'dashboard reporting 'current record Sheets("bridges").Cells(21, 12).Value = i 'dashboard reporting</pre>						

An **Overflow Error** occurs.

This is a problem with a variable not being able to hold the data given to it.



Click Debug

The line where the error occurred is highlighted. Study the code carefully.

```
II AMESSAGE - VDIES INEN
   For i = 2 To 15332
   'dashboard reporting
   'current record
   Sheets("bridges").Cells(21, 12).Value = i
   'dashboard reporting
   'number of records left
   Sheets("bridges").Cells(24, 12).Value = 15532 - i
  'assign values from the data
  intYearBuilt = Sheets("bridges").Cells(i, 5).Value
intAveDailyTruckTraffic = Sheets("bridges").Cells(i, 7).Value
  intPercentTrucks = Sheets("bridges").Cells(i, 8).Value
  'calculate bridge age
   intAge = 2014 - intYearBuilt
   Sheets("bridges").Cells(i, 10).Value = intAge
   'calculate truck traffic
   intAveDailyTruckTraffic = intAveDailyTraffic * intPercentTrucks
   Sheets("bridges").Cells(i, 9).Value = intAveDailyTruckTraffic
   'assign correct county name from census code
```

The data from column 7 is actually the Average Daily Traffic, not Average Daily Truck Traffic. Average Daily Truck Traffic will be calculated subsequently. Correct the variable name to intAveDailyTraffic.

Click on Reset



Save and Compile to check for more bugs and errors.

If none, Resave.

Re-Run.

The program proceeds. The dashboard updates rapidly as the program proceeds

Hold down the **Ctrl** key and press **Break**. (In some operating systems it is **Shift + Break**, or **Fn + Break**)

The program pauses and goes in the **Break mode** Select **Debug**

As Integer	
d from)ng	
:ger	Microsoft Visual Basic
12).Value = 15332	Code execution has been interrupted
very large and processing will er resources ring	<u>C</u> ontinue <u>E</u> nd <u>D</u> ebug <u>H</u> elp
bout to process a very large & "Do you want to continue?",	data set which may tie up computer resources" vbYesNo + vbExclamation, "CAUTION")

.nue, start the For loop



Click on the spreadsheet window to view it. Review the calculated columns. Review the digital dashboard

0			J	IV.	L	IVI	IN	
Ave Daily Traffic	Percent Truck Traffic	Ave Daily Truck Traffic	Age of Bridge					
25	5	1	17					
10800	2	216	28					-
27800	10	2780	14					
51500	8	/120	14					
48000	5	2400	44					
51500	0	2400	44	ſ				
48000	0 5	4120	44					
190500	5	2400	44 52		CommandBu	itton1		
190500	0	11430	52					
20100	2	402	52					
49500	4	1960	44				2	
49500	4	1980	44					
28500	3	000	03			0400		
14168	2	283	20		DIGITAL DASH	BOARD		
354	2	1	15					_
18900	3	567	57					
66000	15	9900	48	Tota	I Number of Red	cords		
66000	15	9900	48		15332			
50500	3	1515	24					
30000	5	1500	24		Current Record			
61000	10	6100	29		793			
500	30	150	29					
61000	10	6100	29	Num	ber of Records	Left		
500	30	150	29		14539			
40500	8	3240	48					
5000	5	250	48					
77500	7	5425	39					

The results so far are satisfactory. It can be concluded that the program is working as intended.



Select VBA window.

Click on **Run**/ **Resume** to resume execution of the program from where it was paused.

The program successfully runs to completion.

Save and close the workbook.

All project requirements have been met.

1	0			J	IX	L	IVI	IN			
Num_of_Lanes	Ave_Daily_Traffic	Percent_Truck_Traffic	Ave Daily Truck Traffic	Age of Bridge							
1	25	5	1	17							
2	10800	2	216	28							
5	27800	10	2780	14							
3	51500	8	4120	44							
3	48000	5	2400	44							
3	51500	8	4120	44							
3	48000	5	2400	44							
10	190500	6	11430	52		CommandBu	itton1				
10	20100	2	402	52							
4	49500	4	1980	44	_			-			
4	49500	4	1980	44							
5	28500	3	855	83							
3	14168	2	283	20		DIGITAL DASH	BOARD				
2	354	2	7	15							
5	18900	3	567	57							
6	66000	15	9900	48	Tota	I Number of Red	cords				
6	66000	15	9900	48		15332					
6	50500	3	1515	24			-				
6	300 SUCCESS	FUL COMPLETION	1500	24		Current Record					
3	610		6100	29		15332					
3	50 411 COL		150	29							
3	610	IVIIVIAINDS EXECUTED	6100	29	Nun	nber of Records	Left				
3	50		150	29		0					
6	405		3240	48							
6	50	OK	250	48							
13	775	,	5425	39							
13	6000	5	300	39							
4	15000	5	750	58							
			-	-							



4. CONCLUSION

This course has presented a broad overview of fundamental concepts and principles of computer programming, and presented them in situations encountered by practicing engineers and scientists. All codes were developed using the *Visual Basic for Applications* (VBA) programming language.

In this course the topics, conditional statements, message boxes and alerts, and looping structures were covered in detail. Several examples from engineering were used to illustrate and demonstrate the concepts and methods learned in this class. Two mini-projects were used to demonstrate these programming concepts and methods in situations encountered by practicing engineers.

This course has enabled participants to identify situations where programming is relevant and will be of advantage to the professional. Practitioners are strongly encouraged to look out for situations in their domains of expertise where programming solutions are applicable and will be of benefit to their work and their organization.

Computer programming requires a careful and meticulous approach, and can only be mastered and retained by practice and repetition.

Good Luck and Happy Programming.



REFERENCES

Bradley, J. C., & Millspaugh, A. C. (1999). *Programming in Visual Basic 3.0.* Irwin McGraw-Hill.

- FunctionX Inc. (2013). VBA for Microsoft Office Excel 2007. Retrieved December 21, 2013, from FunctionX Tutorials: www.functionx.com/
- Microsoft. (2013). *Excel 2013 developer reference*. Retrieved October 15, 2013, from Office Dev Center: http://msdn.microsoft.com/en-us/library/office/ee861528.aspx

Images were all drawn/ prepared by K. Ofosu