Visual Basic in Access

Microsoft® Access 2007

Instructor: Don Bremer

Contents

Visual Basic in Access

Programming can be an enormously complex and difficult activity. Or it can be quite straightforward. In either case, the basic programming concepts remain the same.

Strictly speaking, the language that is included with Access is not Visual Basic—it is a subset of the full, stand-alone Visual Basic language (which Microsoft sells separately). In Access version 2.0, the subset is called "Access Basic". In version 7.0, it is slightly enlarged subset called "Visual Basic for Applications" (VBA).

History

- Visual Basic is actually a "3GL" or third generation language
    - First generation – binary
    - Second generation – assembler code
    - Visual Basic is not compiled – it is interpreted "on the fly"
- Bill Gates originally saw VB being used in any part of windows, which it is:
    - Visual Basic
    - Windows Script Host
    - VBA

Getting to the Visual Basic Editor

In this exercise, we will simply get the computer to put "Hello World"

- Start Access
- Start a "New Blank Database"
- Save this in My documents as Class Example – XXXX where XXXX is today's date
- Go to the Form objects
- Create a new form using the design view. You should see something like Figure 1.
- Using the toolbox (shown in Figure 2), click on the button tool
- Create a button on the form

At this point, a wizard usually starts up asking what we'd like to do. Since this project is unique, we cannot pick a specific category. Because of this, just click "Finish"

Since the command button is still selected, this is what is shown on the properties tab. This is shown in Figure 3.

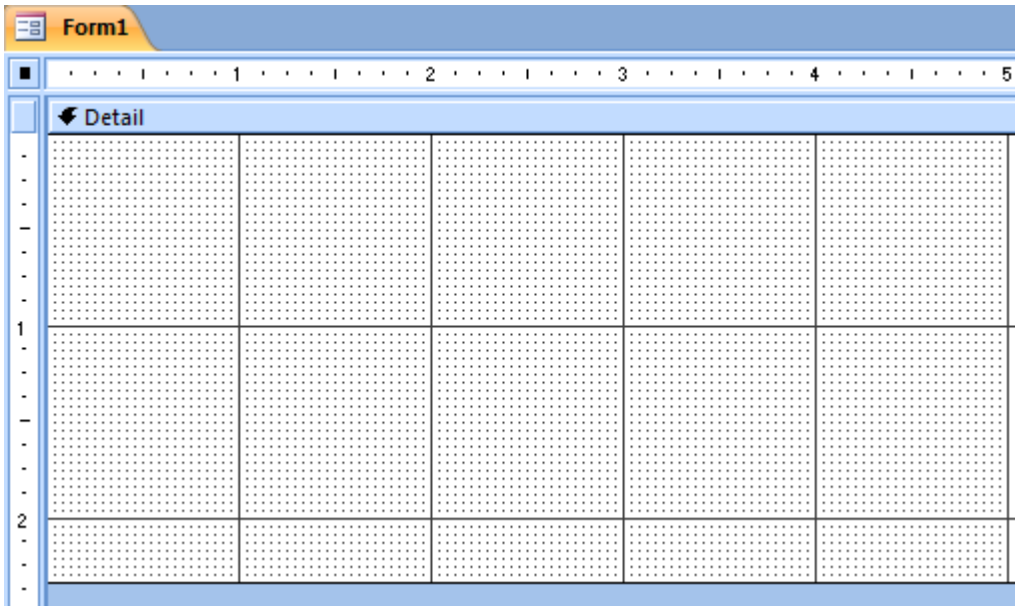
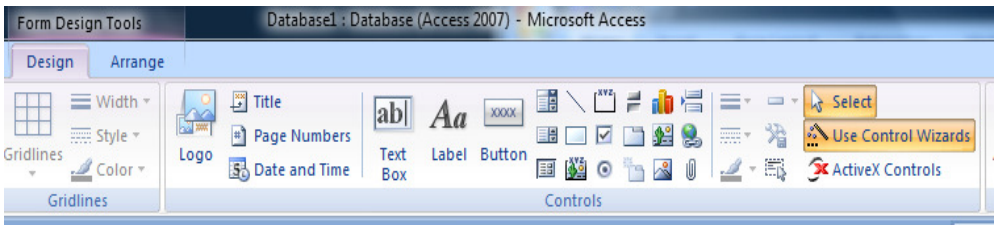


Figure 1: A simple blank form!

Figure 2: The toolbox



Now, look where it says “On Click…” – There is an [Event Procedure]. This is code that is used to get Access to do things.

- Click on the [Event Procedure]
- Click on the ellipse (…) on the side

This is what comes up in the Visual Basic Editor:

```
Private Sub Command0_Click()
On Error GoTo Err_Command0_Click

    Screen.PreviousControl.SetFocus
    DoCmd.FindNext

Exit_Command0_Click:
```

Property Sheet
Selection type: Command Button
Command0
Format | Data | Event | Other | All

Name	Command0
Caption	Command0
Picture Caption Arrangement	No Picture Caption
Visible	Yes
Cursor On Hover	Default
Picture	(none)
Picture Type	Embedded
Width	1.7917"
Height	0.5"
Top	0.625"
Left	1.7917"
Back Style	Normal
Transparent	No
Font Name	Calibri
Font Size	11
Alignment	Center
Font Weight	Normal
Font Underline	No
Font Italic	No
Fore Color	System Button Text
Hyperlink Address	
Hyperlink SubAddress	
Gridline Style Top	Transparent
Gridline Style Bottom	Transparent
Gridline Style Left	Transparent
Gridline Style Right	Transparent
Gridline Color	#000000
Gridline Width Top	1 pt
Gridline Width Bottom	1 pt
Gridline Width Left	1 pt
Gridline Width Right	1 pt
Top Padding	0.0208"
Bottom Padding	0.0208"
Left Padding	0.0208"
Right Padding	0.0208"
Horizontal Anchor	Left
Vertical Anchor	Top
Display When	Always
Reading Order	Context
Enabled	Yes
On Click	
On Got Focus	
On Lost Focus	
On Dbl Click	
On Mouse Down	
On Mouse Up	
On Mouse Move	
On Key Down	
On Key Up	
On Key Press	
On Enter	
On Exit	
Default	No
Cancel	No
ControlTip Text	
Tab Index	0
Tab Stop	Yes
Status Bar Text	

Figure 3 – A lot of what you can do with a command button!

```
    Exit Sub

Err_Command0_Click:
    MsgBox Err.Description
    Resume Exit_Command0_Click

End Sub
```

Wow! That's a lot of stuff that was written for us. But, that will come later. Now, erase everything between Private Sub … and End Sub.

- Type in:
  `MsgBox("Hello World")`
- Select the form again
- Select the form view ()
- Our form is active
- Click on the button…

Great! We've just written our first VB program.

MsgBox()

This is a command that will put text on the screen if some action has occurred. This is a perfect for a message saying that a command cannot take place because some information hasn't been given.

The entire syntax of the command is:
MsgBox(Prompt, Button Style, Title, HelpFile, Context)

This shows we can change a lot of how we see it. Change the command to what is below and try it again. NOTE: No ()!

```
MsgBox "Hello World", vbYesNo, "VB is Cool"
```

Now, click on the button again.

Basic programming constructs

Statements

Statements are special keywords in a programming language that do something when executed. This is like our MsgBox example.

Variables and assignment

Statements are special keywords in a programming A variable is space in memory to which you assign a name. When you use the variable name in expressions, the programming language replaces the variable name with the contents of the space in memory at that particular instant.

Now, let's change our program. Type this in as our program:

```
Dim HelloWorld As String

HelloWorld = "Hello "
HelloWorld = HelloWorld & "World"

MsgBox HelloWorld, vbYesNo, "VB is Cool"
```

In the first statement, a variable HelloWorld is created and the string "Hello" is assigned to it. In the next statement, another word in concatenated to it ("World") and then that variable is used in the MsgBox command.

In computer programming, a function is a small program that takes one or more **arguments** (or **parameters**) as input, does some processing, and returns a value as output. A predefined (or built-in) function is a function that is provided as part of the programming environment.

For example, `cos(x)` is a predefined function in many computer languages—it takes some number `x` as an argument, does some processing to find its cosine, and returns the answer. Note that since this function is predefined, you do not have to know anything about the algorithm used to find the cosine, you just have to know the following:

1. what to supply as inputs (e.g., a valid numeric expression representing an angle in radians),
2. what to expect as output (e.g., a real number between -1.0 and 1.0).

Remark statements

When creating large programs, it is considered good programming practice to include adequate internal documentation—that is, to include comments explaining what the program is doing.

Let's change around our comments to make it more user friendly. Add the line:

```
‘ VB is cool – Let’s tell the world!
```

The character on front is the single tick mark. Notice that the line turned green? This means that the line will not be executed – it is a remark.

Returning something

Now, what is Yes or No returning? Add these lines:

```
    Dim HelloWorld As String
    Dim Answer As Variant

    HelloWorld = "Hello "
    HelloWorld = HelloWorld & "World"

    Answer = MsgBox(HelloWorld, vbYesNo, "VB is Cool")
    MsgBox (Answer)
```

The numbers 6 and 7 refer to the constants vbYes and vbNo, respectively.

Loops

Loops are used to get the computer to do some commands until some condition is met. You can do statements for a certain amount of time (a “for” loop) or until some condition is met (a “while” loop).
Let's make the computer say “Hello World” five times. And have it count all five time to show us. Adjust the program accordingly:

```
Dim HelloWorld As String
Dim Answer As Variant

For i = 1 To 5

    HelloWorld = "Hello "
    HelloWorld = HelloWorld & "World –" & Str(i)

    Answer = MsgBox(HelloWorld, vbYesNo, "VB is Cool")
    'MsgBox (Answer)

Next
```

Notice the second message box is commented out. This is common – to comment out code that isn't being used right now. After all, you may need it again!

If statements

If statements allow the computer to do different things depending on conditions. For an example, put in the following code:

```
Dim Question As String
Dim Answer As Variant

Question = "Do you like VB?"

Answer = MsgBox(Question, vbYesNo, "VB is Cool")

If Answer = vbYes Then
    MsgBox ("Cool")
Else
    MsgBox ("Bummer")
End If
```

If you like VB, it will say “Cool”. Otherwise, it will say “Bummer”. Not only can we say things, but different pieces of code can be executed.

Subroutines

The true test of any programmer is if you can use your code in other places or other applications. The easiest way to do this is to make small sections of code that do specific tasks and set that off on its own. In VB, this is called a subroutine.
Let’s take our example and split it up.

```
Private Sub Command0_Click()

    Dim Question As String
    Dim Answer As Variant

    Question = "Do you like VB?"

    Answer = MsgBox(Question, vbYesNo, "VB is Cool")

    TellUser (Answer)

End Sub

Private Sub TellUser(Answer)

    If Answer = vbYes Then
        MsgBox ("Cool")
    Else
        MsgBox ("Bummer")
    End If
```

```
End Sub
```

Works about the same, right? Now, let's add a new button on the form and ask the user if they like quiche. The code for the button will look something like:

```
Private Sub Command1_Click()

    Dim Question As String
    Dim Answer As Variant

    Question = "Do you like Quiche?"

    Answer = MsgBox(Question, vbYesNo, "Quiche is Cool")

    TellUser (Answer)

End Sub
```

We have now reused our code and deserve a donut!

Class Project: Make a calculator

The next project is to create a calculator. This calculator will add, subtract, multiply, and divide. It will show the answer in a message box. We will use a subroutine to show all answers!

- Create a form like the one in Figure 4
- To create an hot key button, put an & in front of the letter
- The best way to name objects is using the Lesenski method:
    - Cmd for command
    - Frm for form
    - Tbl for table
    - Qry for query
    - Txt for textbox
    - Lbl for label

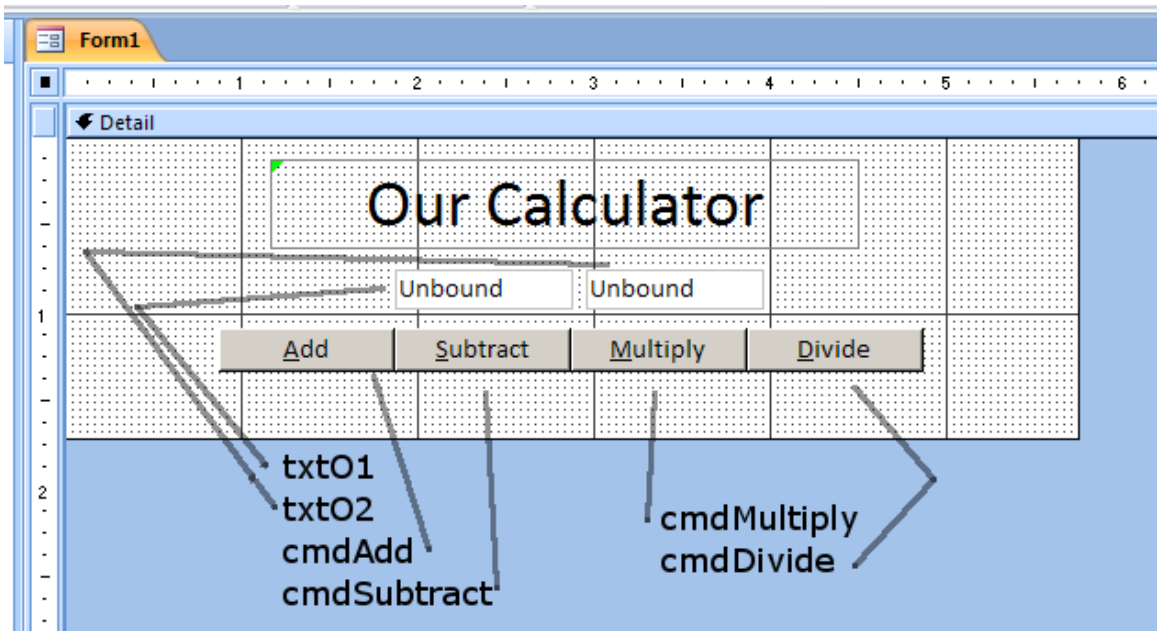


Figure 4: The form

ShowAnswer function

The code for the ShowAnswer function:

```
Private Sub ShowAnswer(numAnswer)

    Dim strAnswer As String

    strAnswer = "The answer to your math problem is: " & Str(numAnswer)

    MsgBox (strAnswer)

End Sub
```

Add

The code for the Add function:

```
Private Sub cmdAdd_Click()

     Dim numOperand1
     Dim numOperand2
     Dim numAnswer

     numOperand1 = Me.txtO1.Value
     numOperand2 = Me.txtO2.Value

     numAnswer = Val(numOperand1) + Val(numOperand2)

     ShowAnswer (numAnswer)
```

```
End Sub
```

Subtract

The code for the Subtract function:

```
    Dim numOperand1
    Dim numOperand2
    Dim numAnswer

    numOperand1 = Me.txtO1.Value
    numOperand2 = Me.txtO2.Value

    numAnswer = Val(numOperand1) - Val(numOperand2)

    ShowAnswer (numAnswer)
```

Multiply

The code for the Multiply function:

```
    Dim numOperand1
    Dim numOperand2
    Dim numAnswer

    numOperand1 = Me.txtO1.Value
    numOperand2 = Me.txtO2.Value

    numAnswer = Val(numOperand1) * Val(numOperand2)

    ShowAnswer (numAnswer)
```

Divide

The code for the Divide function:

```
Private Sub cmdDivide_Click()

    Dim numOperand1
    Dim numOperand2
    Dim numAnswer

    numOperand1 = Me.txtO1.Value
    numOperand2 = Me.txtO2.Value

    numAnswer = Val(numOperand1) / Val(numOperand2)

    ShowAnswer (numAnswer)
```

```
End Sub
```

With error checking making the denominator not 0!

Try dividing by zero… We get an error. Let's do some error checking!

```
Private Sub cmdDivide_Click()

    Dim numOperand1
    Dim numOperand2
    Dim numAnswer

    numOperand1 = Me.txtO1.Value
    numOperand2 = Me.txtO2.Value

    If (Val(numOperand2) = 0) Then
        MsgBox ("You cannot change the laws of mathematics!")
        Exit Sub
    End If

    numAnswer = Val(numOperand1) / Val(numOperand2)

    ShowAnswer (numAnswer)

End Sub
```

Objects to know:

Me

This is the form you are dealing with. This is useful in getting the values from a form without knowing the forms name.

DoCmd

You can use the methods of the DoCmd object to run Microsoft Access actions from Visual Basic. An action performs tasks such as closing windows, opening forms, and setting the value of controls. For example, you can use the OpenForm method of the DoCmd object to open a form, or use the Hourglass method to change the mouse pointer to an hourglass icon.

Example

```
Sub ShowNewRecord()
    DoCmd.OpenForm "Employees", acNormal
    DoCmd.GoToRecord , , acNewRec
```

```
End Sub
```

Application

You can use the Application property in Visual Basic to access the active Microsoft Access Application object and its related properties.

Example

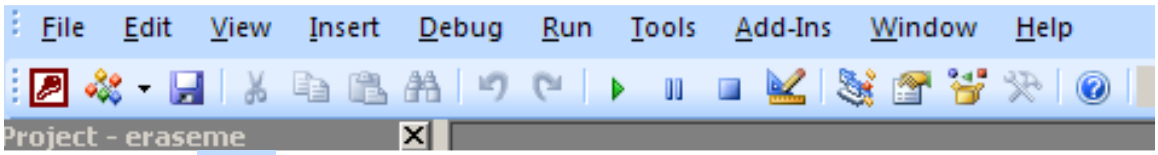
```
Private Sub Command1_Click()
    DisplayApplicationInfo Me
End Sub

Function DisplayApplicationInfo(obj As Object) As Integer
    Dim objApp As Object, intI As Integer, strProps As String
    On Error Resume Next
        ' Form Application property.
        Set objApp = obj.Application
        MsgBox "Application Visible property = " & objApp.Visible
        If objApp.UserControl = True Then
        For intI = 0 To objApp.DBEngine.Properties.Count - 1
            strProps = strProps & objApp.DBEngine.Properties(intI).Name & ", "
        Next intI
        End If
        MsgBox Left(strProps, Len(strProps) - 2) & ".", vbOK, "DBEngine Properties"
End Function
```

Object Browser

You can use the Object Browser to view and navigate among the objects available in Microsoft Access and other applications that support Microsoft Visual Basic, as well as the methods and properties of each object. When you locate a method or property in the Object Browser, you can paste it into the active module.

Toolbar:



This button :