CSE 1530

Introduction to Computer Use II: Programming

Winter 2005 (Section M)

Topic B: Variables, Data Types and Expressions

Wednesday, January 25 2006

Bill Kapralos

Overview (1):

- **Before We Begin**
  - Some administrative details
  - Some questions to consider
- **Variable declarations Revisited**
  - Local vs. global variables
  - The "Option Explicit" statement
- **Built-in visual Basic Constants**
  - A look at some common Constant modules

Before We Begin

Administrative Details (1):

- **Lab Exercise 2-3**
  - Exercises 2-3 has been graded and will be made available for you to pick up after today's lecture
- **TA Advising Hours**
  - TA will be in the Glade Lab on the following days
    - Tuesdays 5-8pm and Fridays 2:30-4:30pm
- **My Office Hours**
  - I will hold my office hours in the Glade Lab today from 2:30 – 3:30pm

Some Questions to Consider (1):

- Generally, why is it a bad idea to let Visual Basic convert to the appropriate data type for you ?
- What happens when we add two strings e.g., "10" + "1" ?
- What is an objects "Top" property ?
- What is an object's "Left" property ?

Variable Declarations Revisited

Variable Declarations (1):

- **Local vs. Global Variables**
  - Until now, we probably didn't place too much emphasis on when to declare variables global and when to declare them local
  - Typically you can declare all your variables global
    - This is actually bad programming practice!
    - May not seem like a big deal in the simple programs we have encountered so far → this will become an issue for larger, more complex programs!

Variable Declarations (2):

- **Local vs. Global Variables (cont.)**
  - Unnecessary global variables add to the complexity of the program and multiply the possibility of errors!
    - Since a global variable is accessible in any subprogram its value can be changed within any subprogram → if the variable should have been declared local then its value may be inadvertently changed in some other part of the program

Variable Declarations (3):

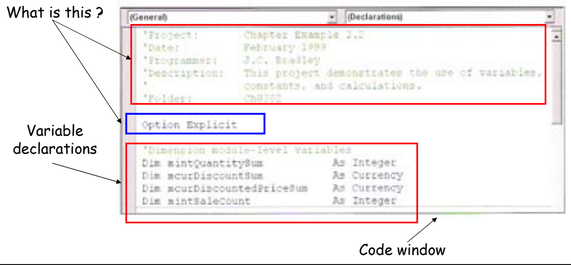
- **Local vs. Global Variables (cont.)**
  - You should ensure that the scope of all your declared variables is appropriate
    - If the variable is needed for one method or function only, then it should not be declared global but rather, locally within the corresponding method or function where it is required
    - Global variables should be used only when the variable is to be used by multiple functions → intended to be shared between components of a program (e.g., forms a connection between the components)

Variable Declarations (4):

- **Local vs. Global Variables (cont.)**
  - Deciding on the "correct" variable scope declaration is of course something that you will pick up with experience!
    - Once again →practice, practice and more practice!

Variable Declarations (5):

- **Declaring Global (Module Level) Variables**
  - These declarations must appear in the code window, in the "General Declarations" section



```
'Project:        Chapter Example 3.2
'Date:           February 1999
'Programmer:     J.C. Bradley
'Description:    This project demonstrates the use of variables,
'                constants, and calculations.
'Folder:         Ch0302

Option Explicit

'Dimension module-level variables
Dim mintQuantitySum               As Integer
Dim mcurDiscountSum               As Currency
Dim mcurDiscountedPriceSum        As Currency
Dim mintSaleCount                 As Integer
```

Variable Declarations (6):

- **The "Option Explicit" Statement**
  - Visual Basic does not require you to declare a variable before using it
    - When the variable is first used in the code, it will automatically be assigned the type Variant
  - For example → curHours = CDbl(Text1.Text)
    - Variable "curHours" has not been declared but this is not an error → when first encountered, it will be automatically assigned of type Variant
    - This is of course bad programming practice since good programming practice requires all variables to be declared!

Variable Declarations (7):

- **The "Option Explicit" Statement (cont.)**
  - Recall the Variant data type
    - You do not need to specify a type for a variable you declare → if no type is specified, Visual Basic will automatically assign it the type Variant and therefore allow its type to be changed as the program executes
    - Variants are slow and consume memory
  - Basically, the "Option Explicit" statement, when present, forces you to declare all your variables and avoid "on the fly" variable declarations!

Variable Declarations (8):

- **The "Option Explicit" Statement (cont.)**
  - An example will best illustrate the importance of the Option Explicit statement
    - Consider the following code segment which does not generate any Visual Basic errors but is incorrect → contains three errors!

```
Private Sub Command1_Click()
  curHours = CDbl(txtHours.Text)
  curPayRate = CDbl(txtPayRate.Text)
  curPay = Hours * PayRate
  curTotalPay curTotlPay + curPay
End Sub
```

Variable Declarations (9):

- **The "Option Explicit" Statement (cont.)**
  - The three errors:
    - curPay and curTotalPay will both be equal to zero given the different variable name spelling!
    - curHours/Hours, curPayRate/payRate, curTotalPay/curTotlPay

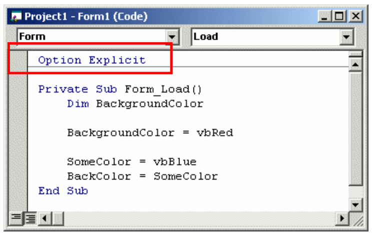
```
Private Sub Command1_Click()
  curHours = CDbl(txtHours.Text)
  curPayRate = CDbl(txtPayRate.Text)
  curPay = Hours * PayRate
  curTotalPay curTotlPay + curPay
End Sub
```

Variable Declarations (10):

- **The "Option Explicit" Statement (cont.)**
  - The errors in the previous code segment can be avoided by including the "Option Explicit" statement in your code window
    - Basically, it would not allow the declaration of the variables payRate, curTotlPay and Hours which are allowed and initialized to a value of zero when the option Explicit statement is missing!

Variable Declarations (11):

- **Two Ways To Set Option Explicit**
  - Include the statement "Option Explicit" in the code window before any variable declarations

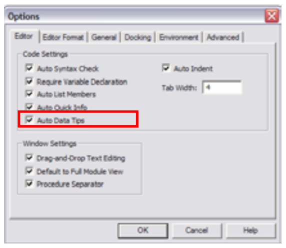


```
Option Explicit

Private Sub Form_Load()
    Dim BackgroundColor

    BackgroundColor = vbRed

    SomeColor = vbBlue
    BackColor = SomeColor
End Sub
```

Code window

Variable Declarations (12):

- **Two Ways To Set Option Explicit (cont.)**
  - Set it as an option in the Visual Basic editor
- From the *Tools menu* select *Options*
- On the editor tab, make sure the *Require Variable Declaration* is selected and click *Ok*

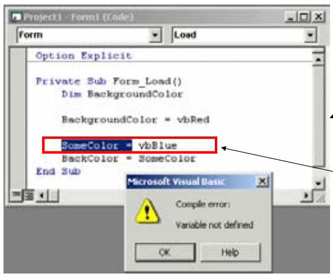


Code Settings
- ✓ Auto Syntax Check
- ✓ Require Variable Declaration
- ✓ Auto List Members
- ✓ Auto Quick Info
- ✓ Auto Data Tips
- ✓ Auto Indent
- Tab Width: 4

Window Settings
- ✓ Drag-and-Drop Text Editing
- ✓ Default to Full Module View
- ✓ Procedure Separator

Variable Declarations (13):

- **Two Ways To Set Option Explicit (cont.)**
  - With the Option Explicit statement present, you must declare all variables!



```
Option Explicit

Private Sub Form_Load()
    Dim BackgroundColor

    BackgroundColor = vbRed

    SomeColor = vbBlue
    BackColor = SomeColor
End Sub
```

```
Microsoft Visual Basic
Compile error:
Variable not defined
```

Variable Initial Values (1):

- **Built in VB Conversion Functions (cont.)**
  - When we declare a variable, what is its initial value ?
    - Does Visual basic assign it a “default” value or is that our responsibility ?
  - Visual basic does provide initial (default) values for all variables you declare
    - These values may not necessarily be the value you need/want however so you may need to provide your own initial variable values!
    - Typically, the numerical types are initialized to zero, Strings to the null string etc. How can you test this for all the data types ??

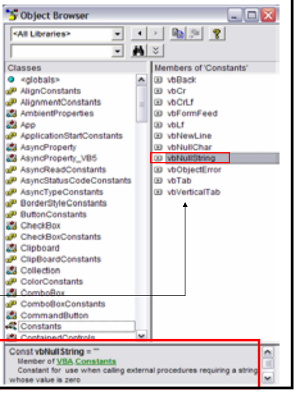
Built-In Constants in Visual Basic

VB Built-in Constants (1):

- **Several Available “Built-in” Constants**
  - Visual Basic contains various commonly used pre-defined constants for you to make use of
    - Good programming practice to make use of such constants → promotes code consistency and no need to define your own constants for constants that are already defined for you!
    - Commonly used constant → vbNullString
    - To view the VB constants → View Menu -> Object Browser
    - Various “groups” (modules) of constants

VB Built-in Constants (2):

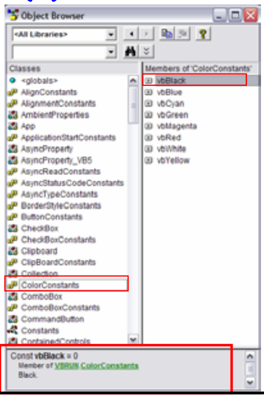
- **Several Available “Built-in” Constants (cont.)**
  - Constants module



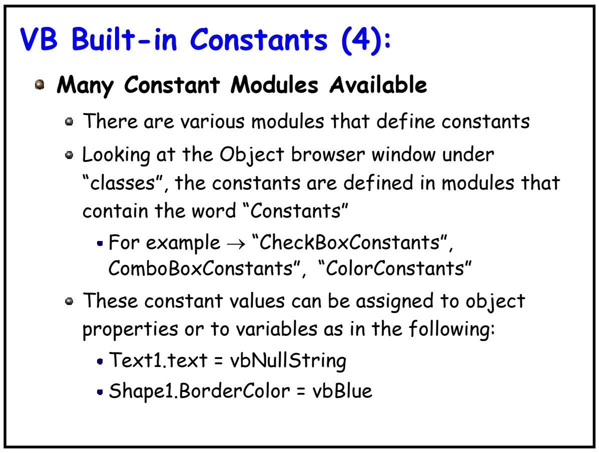
```
Const vbNullString = ""
    Member of VBA.Constants
    Constant for  use when calling external procedures requiring a string
whose value is zero
```

VB Built-in Constants (3):

- **Several Available “Built-in” Constants (cont.)**
  - ColorConstants module
    - Color related constant definitions



```
Const vbBlack = 0
    Member of VBRUN.ColorConstants
    Black.
```



VB Built-in Constants (4):

- **Many Constant Modules Available**
  - There are various modules that define constants
  - Looking at the Object browser window under "classes", the constants are defined in modules that contain the word "Constants"
    - For example → "CheckBoxConstants", ComboBoxConstants", "ColorConstants"
  - These constant values can be assigned to object properties or to variables as in the following:
    - Text1.text = vbNullString
    - Shape1.BorderColor = vbBlue