CSE 1530

Introduction to Computer Use II: Programming

Winter 2005 (Section M)

Topic B: Variables, Data Types and Expressions

Monday, January 23 2006

Bill Kapralos

Overview (1):

- **Before We Begin**
  - Some administrative details
  - Some questions to consider
- **Data Types and Visual Basic**
  - Data type conversion
    - String to data type conversion functions
    - Other data type to String conversion functions
- **Positioning Objects on a Form**
  - Example

Overview (2):

- **"Live Demos" (Time Permitting)**

Before We Begin

Administrative Details (1):

- **Lab Exercise 3-4**
  - This week, you should be working on Ex. 3-4 from your textbook
  - Follow instructions given on the course website
  - Due Monday, January 30 2005 before noon
    - Place in the assignment drop-box located on the 1st floor of the CSE building just by the elevator and CSE undergraduate offices
  - Wednesday's office hours will be held in the Glade lab

Some Questions to Consider (1):

- How do we declare a constant ?
- How do we declare a variable ?
- What is a variable's scope ?
- Is Visual Basic case-sensitive with respect to variable declarations ?
- What is a function ?
- What is an argument ?

Data Types & VB

Data Type Conversion (1):

- **Built in VB Conversion Functions**
  - Visual Basic functions to convert between data types
  - As an aside → what is a function ?
    - A convenient way to encapsulate some computation that can be then used many times over without worrying about its implementation
    - Allows us to ignore *how* a job is done
    - All we need to know is *what* is done (outcome)
    - Imagine having to compute some computation many times → you can replicate the code many times or you can write the code once within a function and simply call the function

Data Type Conversion (2):

- **Built in VB Conversion Functions (cont.)**
  - In general these conversion functions take one or more arguments and produce a single result (called the function return value) of a particular type
    - Argument → when you call and use the function, you may have to supply it zero or more values – these values are known as arguments
    - Function return value → the value of a particular type returned by the function - the value can be used by the caller of the function where appropriate

Data Type Conversion (3):

- **Built in VB Conversion Functions (cont.)**
  - Arguments may be a single variable or a single value as an argument provided it is of the required type
  - Argument may also be an expression that, when evaluated, will result in a single value
  - Examples → assume a function named "myFunction" that takes one Integer argument
    - myFunction(Command1.Width)
    - myFunction(Command1.Width / 2)
    - myFunction((Command1.Width / 2) + 50)
    - myFunction(795 - Command1.Width * 10)

Data Type Conversion (4):

- **Built in VB Conversion Functions (cont.)**
  - Many times it is common to convert from a string to some other value
    - Usually, user input will be in the form of a string (e.g., entering data in a textbox) and we therefore must convert to the appropriate type
    - Visual Basic functions that convert a string to any other data type are widely used
    - Lets take a look at these functions…

Data Type Conversion (5):

- **Converting Strings to Other Types**

Function Name	Return Type	Argument
CBool	Boolean	Any valid string of numeric expression
CCur	Currency	String with range of Currency values
CDbl	Double	String with range of Double values
CInt	Integer	String with range of Integer values
CLng	Long Integer	String with range of Long Integer values
CSng	Single	String with range of Single values
CDate	Date	Any value that can be interpreted as a date

Data Type Conversion (6):

- **Converting Strings to Other Types (cont.)**
  - The conversion functions take one argument (the string value) and return a single value
  - The argument can be either a previously defined String variable or entering the String directly
  - Examples
    - String to Double → CDbl(TextBox1.Text)
    - String to Double → CDbl("100.11")
    - String to Integer → CInt(TextBox1.Text)
    - String to Integer → CInt("100")

Data Type Conversion (7):

- **Converting Strings to Other Types (cont.)**
  - Of course, in order to be of any use, we must make use of the return type!
  - We can use the return type anywhere that particular type is used
  - Basically, treat the "function(argument)" as a value and the type of the value is the function return type
    - Command1.Top = CInt(TextBox1.Text)
    - Command1.Top = 100 + CInt(TextBox1.Text) / 2
  - When part of an expression, the function is evaluated and its return value will replace the function call

Data Type Conversion (8):

- **Converting Other Types to Strings**
  - Functions are also available to convert from any other data type to a string
    - Generally, when Visual Basic performs this conversion for us without explicitly calling a function, the conversion is less ambiguous than going the other way
    - We should still explicitly call the appropriate conversion functions → failure to use the conversion functions promotes a lack of awareness of data types and leads to bad habits that may eventually lead to errors!

Data Type Conversion (9):

- **Converting Other Types to Strings**
  - The CStr function
    - Takes argument of any type and returns a String representation of the argument
    - Used as any other conversion function!
    - Textbox1.text = CStr(100.0)
    - Textbox1.text = CStr(1000)
    - Textbox1.text = CStr(True)
    - What happens when the argument is also a Sting → Textbox1.text = CStr("100.0") ???

Data Type Conversion (10):

- **The Dangers of Not Converting Types**
  - Recall that Visual Basic will attempt to perform conversion of data types but the result may not always be what you expect!
  - Take the "+" operation
    - With numeric values (e.g., Integer, Single) the addition of two such values is also a number
    - But, the "+" operator has a different meaning with strings! → concatenation
    - Visual Basic will not attempt to convert two strings that are to be added

Data Type Conversion (11):

- **The Dangers of Not Converting Types (cont.)**
  - Example
    - 100 + 1100 = 1200
    - "100" +"1100" = 1001100 → Two strings will be concatenated
  - What if one arguments of the "+" operator is a string only ?
    - 100 + "1100" → what will happen here ???

Data Type Conversion (12):

- **The Dangers of Not Converting Types (cont.)**
  - Actual example of using the conversion functions → Exercise 3-3 revisited

```
Interest1.Text = CDbl(Interest.Text) * Cdbl(InitialCap.Text)
Capital1.Text = CDbl(InitialCap.Text) + CDbl(Interest1.Text)
```

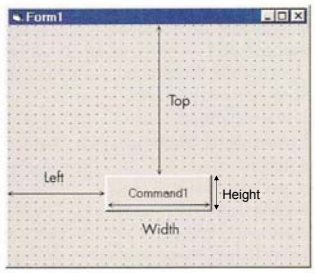
  - Above expressions are relying on Visual Basic to convert to String → in principle, we should have

```
Interest1.Text = CStr( CDbl(Interest.Text)*CDbl(InitialCap.Text)
Capital1.Text = CStr( CDbl(InitialCap.Text) + CDbl(Interest1.Text) )
... (etc)
```

Positioning Objects on a Form

Object Positions on a Form (1):

- **Form "Coordinate System"**
  - Every object we place on a form contains a position relative to the form's coordinate system



Object Positions on a Form (2):

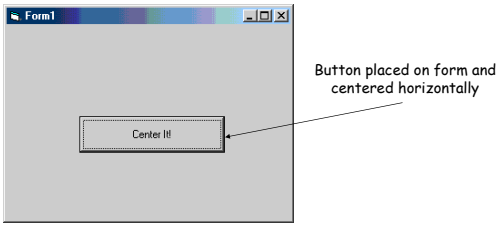
- **Form "Coordinate System" (cont.)**
  - Measurements (integer numbers) are specified in twips → measurement system from the printing industry
    - One twip → 1/20 of a printer's point or 1/1440 of an inch
    - If a control's width property is 1440 twips, then it is basically 1 inch wide → of course, on the screen it may appear smaller of larger depending on your screen's resolution

Object Positions on a Form (3):

- **Form "Coordinate System" (cont.)**
  - Top
    - Integer value representing the position of the top edge of the object relative to top window edge
  - Left
    - Integer value representing the position of the left edge of the object relative to left window edge
  - Object width
    - The width of the object
  - Object height
    - The height of the object

Positioning Objects on a Form (1):

- **Centering a Control on a Form**
  - We will add a button to a form and eventually place it in the center of the form (horizontally)



Positioning Objects on a Form (2):

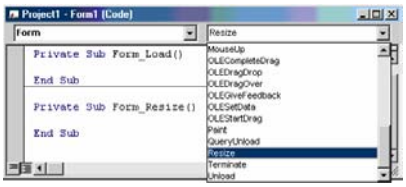
- **Centering a Control on a Form (cont.)**
  - Lets begin by setting the width of our button (which we will call cmdCenter) to half the width of the form
    - cmdCenter.Width = Form1.Width / 2
  - Now lets center the button horizontally on the form
    - cmdCenter.left = Form1.Width / 2
    - Is this sufficient – will this work ? → no! We have to account for the buttons own width as well!
    - cmdCenter.Left = (Form1.Width / 2) – (cmdCenter.Width / 2)

Positioning Objects on a Form (3):

- **Centering a Control on a Form (cont.)**
  - But where do these statements go in our code ?
    - How about placing them in the cmdCenter_Click() method ? → not a good solution since, the button must be pressed to center the button but what if we resize the window ?
    - What about placing the statements in the Form1_Resize() event handler method ? → a much better approach!

Positioning Objects on a Form (4):

- **Centering a Control on a Form (cont.)**
  - Form resize method
    - Automatically called whenever the form itself is resized → makes sense to place the code here since the code will be executed any time the form's size changes!



Positioning Objects on a Form (5):

- **Centering a Control on a Form (cont.)**
  - Placing the code in the Resize() methods still doesn't ensure the correct placement of the button when the form first appears for the first time
    - Can also place the code in the Load() event handler that gets called when the form is first "loaded" (e.g., just before it first appears on the display)
  - But what about centering the object with respect to the height of the form ?
    - Should be the same process?
    - Experiment with this!