

GENERAL PROCEDURES

Robert Burtch
Surveying Engineering Department
Ferris State University

GENERAL PROCEDURES

- 2 devices
 - Sub procedures
 - Function procedures
- Break problem into smaller problems
- Eliminates repetitive code
- Can be reused in other programs

SUB PROCEDURE

- Performs one or more related tasks
- Has its own name
- Written as separate part of program
- Form:

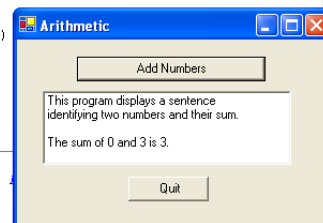
```
Sub ProcedureName()  
    statements  
End Sub
```

- Invoked with statement consisting only of procedure name – called statement

```
ProcedureName()
```

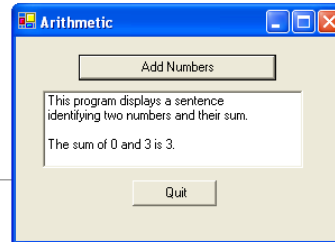
EXAMPLE CODE TO ADD NUMBERS

```
Private Sub btnAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAdd.C:  
    'Display the sum of two numbers  
    Dim num1, num2 As Double  
    lstResult.Items.Clear()  
    lstResult.Items.Add("This program displays a sentence")  
    lstResult.Items.Add("identifying two numbers and their sum.")  
    lstResult.Items.Add("")  
    num2 = 2  
    num2 = 3  
    lstResult.Items.Add("The sum of " & num1 & " and " & num2 & " is " & num1 + num2 & ".")  
End Sub  
  
Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnQuit.C:  
    End  
End Sub
```



CODE USING SUB PROCEDURES

```
Private Sub btnAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAdd.C:  
    'Display the sum of two numbers  
    Dim num1, num2 As Double  
    lstResult.Items.Clear()  
    ExplainPurpose()  
    lstResult.Items.Add("")  
    num1 = 2  
    num2 = 3  
    lstResult.Items.Add("The sum of " & num1 & " and " & num2 & " is " & num1 + num2 & ".")  
End Sub  
  
Sub ExplainPurpose()  
    'Explain the task performed by the program  
    lstResult.Items.Add("This program displays a sentence")  
    lstResult.Items.Add("identifying two numbers and their sum.")  
End Sub
```



SUB PROCEDURES

- Can also transmit data to Sub procedure
 - Called passing
- Example: statement Sum (2,3)
 - In Sub Sum declaration statement, include ByVal num1 As Double, ByVal num2 As Double
 - 2 is num1 and 3 is num2

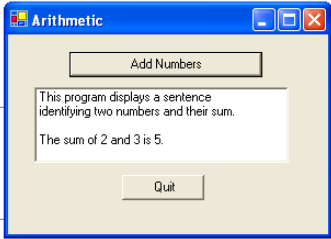


SUB PROCEDURE

```
Private Sub btnAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAdd.Click
    'Display the sum of two numbers
    Dim num1, num2 As Double
    lstResult.Items.Clear()
    ExplainPurpose()
    lstResult.Items.Add("")
    Sum(2, 3)
End Sub

Sub Sum(ByVal num1 As Double, ByVal num2 As Double)
    'display numbers and their sum
    lstResult.Items.Add("The sum of " & num1 & " and " & num2 & " is " & num1 + num2 & ".")
End Sub

Sub ExplainPurpose()
    'Explain the task performed by the program
    lstResult.Items.Add("This program displays a sentence")
    lstResult.Items.Add("identifying two numbers and their sum.")
End Sub
```



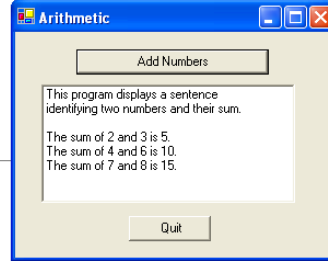
SUB PROCEDURE

- Makes program easy to read, modify, and debug
- Event procedure describes what a program does
 - Sub procedure fills in the details
- Can be called several times during execution

SUB PROCEDURE

```
Private Sub btnAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAdd.C
    'Display the sum of two numbers
    Dim num1, num2 As Double
    lstResult.Items.Clear()
    ExplainPurpose()
    lstResult.Items.Add("")
    Sum(2, 3)
    Sum(4, 6)
    Sum(7, 8)
End Sub

Sub Sum(ByVal num1 As Double, ByVal num2 As Double)
    'display numbers and their sum
    lstResult.Items.Add("The sum of " & num1 & " and " & num2 & " is " & num1 + num2 & ".")
End Sub
```



SUB PROCEDURE

- Variables num1 and num2 in Sub procedure called parameters
 - Names of parameters not important
 - Temporary place holders for numbers passed to Sub procedure
 - Essentials: type, quantity, order

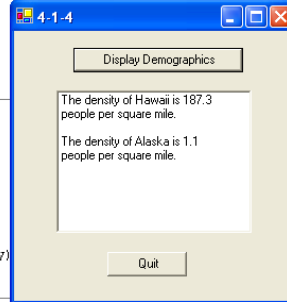


SUB PROCEDURE

- Example of passing a string to Sub procedure

```
Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnD:
    'calculate the population densities of states
    lstDensity.Items.Clear()
    CalculateDensity("Hawaii", 1212000, 6471)
    lstDensity.Items.Add("")
    CalculateDensity("Alaska", 627000, 591000)
End Sub

Sub CalculateDensity(ByVal state As String, _
                    ByVal pop As Double, ByVal area As Double)
    Dim rawDensity, density As Double
    'The density (number of people per square mile)
    'will be displayed rounded one decimal place
    rawDensity = pop / area
    density = Math.Round(rawDensity, 1) 'Round to one decimal place
    lstDensity.Items.Add("The density of " & state & " is " & density)
    lstDensity.Items.Add("people per square mile.")
End Sub
```



VARIABLES AND EXPRESSIONS AS ARGUMENTS

- Items inside parenthesis of call statement referred to as arguments
- Parameter defined for Sub procedure corresponds to argument passed in call statement
- Arguments can be literals, variables or expressions

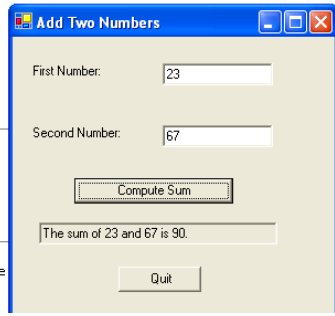
VARIABLES AND EXPRESSIONS AS ARGUMENTS

- Variables x and y

```
Private Sub btnCompute_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnC
    'This program requests two numbers and
    'displays the two numbers and their sum.
    Dim x, y As Double
    x = Cdbl(txtFirstNum.Text)
    y = Cdbl(txtSecondNum.Text)
    sum(x, y)
End Sub

Sub Sum(ByVal num1 As Double, ByVal num2 As Double)
    'Display numbers and their sum
    txtResult.Text = "The sum of " & num1 & " and " & num2 &
        " is " & (num1 + num2) & "."
End Sub

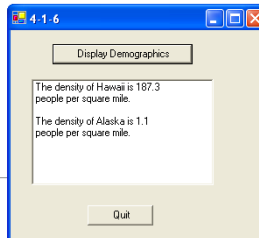
Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e
    End
End Sub
```



VARIABLES AND EXPRESSIONS AS ARGUMENTS

```
Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnD:
    'Calculate the population densities of states
    Dim state As String, pop, area As Double
    Dim s As String, p, a As Double
    Dim sr As IO.StreamReader = IO.File.OpenText("DEMOGRAPHICS.TXT")
    lstDensity.Items.Clear()
    state = sr.ReadLine
    pop = Cdbl(sr.ReadLine)
    area = Cdbl(sr.ReadLine)
    CalculateDensity(state, pop, area)
    lstDensity.Items.Add("")
    s = sr.ReadLine
    p = Cdbl(sr.ReadLine)
    a = Cdbl(sr.ReadLine)
    sr.Close()
    CalculateDensity(s, p, a)
End Sub

Sub CalculateDensity(ByVal state As String, _
    ByVal pop As Double, ByVal area As Double)
    Dim rawDensity, density As Double
    'The density (number of people per square mile)
    'will be displayed rounded to one decimal place
    rawDensity = pop / area
    density = Math.Round(rawDensity, 1) 'Round to one decimal place
    lstDensity.Items.Add("The density of " & state & " is " & density)
    lstDensity.Items.Add("people per square mile.")
End Sub
```



Input Data File saved in project bin folder

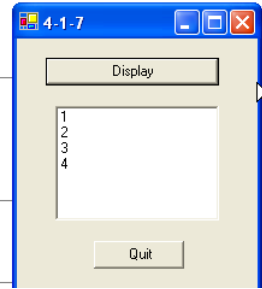
Hawaii
1212000
6471
Alaska
627000
591000

EXAMPLE OF SUB PROCEDURE CALLING ANOTHER SUB PROCEDURE

```
Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDisplay.Click
    'Demonstrate Sub procedure calling other Sub procedures
    FirstPart ()
    lstOutput.Items.Add(4)
End Sub

Sub FirstPart ()
    lstOutput.Items.Add(1)
    SecondPart ()
    lstOutput.Items.Add(3)
End Sub

Sub SecondPart ()
    lstOutput.Items.Add(2)
End Sub
```



SUB PROCEDURES

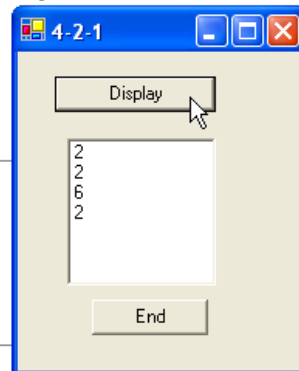
- Rule of thumb – Sub procedures should perform only 1 task or several closely related tasks
 - Keep relatively small
- First line in Sub procedure often a comment statement describing task performed by Sub procedure
- Once Sub procedure defined, VB.NET automatically reminds user of parameters when typing in call statement
 - As soon as left parenthesis is typed, Parameter Info banner appears
 - Figures names and types of parameters

PASSING BY VALUE

- Parameters appearing in Sub procedures preceded by word ByVal
- Variable retains original value after Sub procedure terminated regardless of what is done inside Sub procedure
- 2 memory locations involved
 - When Sub procedure called, temporary 2nd memory set aside for Sub procedure use
 - After leaving Sub procedure, temporary memory location released and value in it lost

PASSING BY VALUE

```
Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal  
    'Illustrate that a change in value of parameter does not alter  
    'value of the argument  
    Dim amt As Double = 2  
    lstResults.Items.Add(amt)  
    Triple(amt)  
    lstResults.Items.Add(amt)  
End Sub  
  
Sub Triple(ByVal num As Double)  
    'Triple a number  
    lstResults.Items.Add(num)  
    num = 3 * num  
    lstResults.Items.Add(num)  
End Sub  
  
Private Sub btnEnd_Click(ByVal sender As System.Object, ByVal e As  
    End  
End Sub
```



PASSING BY REFERENCE

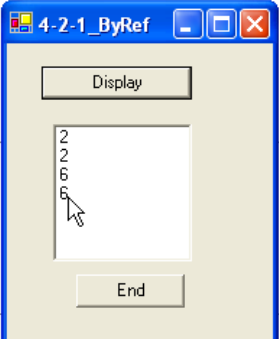
- Parameter preceded by word ByRef
- Current value passes back and forth to and from Sub procedure
- Different names may be used for argument and corresponding parameter
- Only one memory location involved

PASSING BY REFERENCE

```
Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal e As EventArgs)
    'Illustrate that a change in value of parameter does not alter the
    'value of the argument
    Dim amt As Double = 2
    lstResults.Items.Add(amt)
    Triple(amt)
    lstResults.Items.Add(amt)
End Sub

Sub Triple(ByRef num As Double)
    'Triple a number
    lstResults.Items.Add(num)
    num = 3 * num
    lstResults.Items.Add(num)
End Sub

Private Sub btnEnd_Click(ByVal sender As System.Object, ByVal e As EventArgs)
    End
End Sub
```



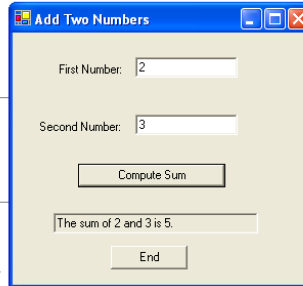
PASSING BY REFERENCE

```
Private Sub btnCompute_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCompute.Click
    'This program requests two numbers and
    'displays the ntwo numbers and their sum.
    Dim x, y As Double
    GetNumbers(x, y)
    Sum(x, y)
End Sub

Sub GetNumbers(ByRef x As Double, ByRef y As Double)
    'Record the two numbers in the text boxes
    x = Cdbl(txtFirstNum.Text)
    y = Cdbl(txtSecondNum.Text)
End Sub

Sub Sum(ByVal num1 As Double, ByVal num2 As Double)
    'Display numbers and their sum
    txtResult.Text = "The sum of " & num1 & " and " & num2 &
        " is " & (num1 + num2) & "."
End Sub

Private Sub btnEnd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEnd.Click
    End
End Sub
```



PASSING BY REFERENCE

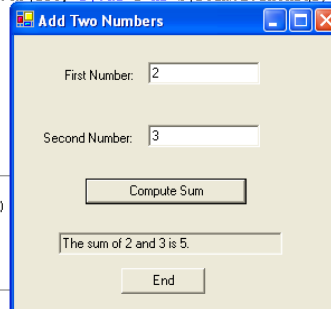
- Procedure in input-process-output style

```
Private Sub btnCompute_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Ha
    'This program requests two numbers and
    'displays the ntwo numbers and their sum.
    Dim x As Double      'First number
    Dim y As Double      'Second number
    Dim t As Double      'Total
    GetNumbers(x, y)
    CalculateSum(x, y, t)
    DisplayResult(x, y, t)
End Sub

Sub GetNumbers(ByRef x As Double, ByRef y As Double)
    'Record the two numbers in the text boxes
    x = Cdbl(txtFirstNum.Text)
    y = Cdbl(txtSecondNum.Text)
End Sub

Sub CalculateSum(ByVal num1 As Double, ByRef num2 As Double, _
    ByRef total As Double)
    'Add the values of num1 and num2
    total = num1 + num2
End Sub

Sub DisplayResult(ByVal num1 As Double, ByVal num2 As Double, _
    ByVal total As Double)
    txtResult.Text = "The sum of " & num1 & " and " & num2 &
        " is " & total & "."
End Sub
```



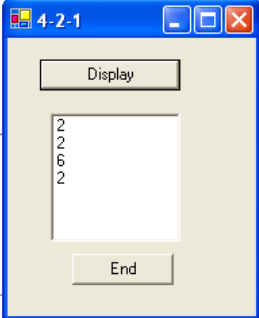
PASSING BY VALUE

- Can use double parenthesis around variable to pass by value
 - Can override even ByRef preceding variable

```
Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    'Illustrate that a change in value of parameter does not alter the
    'value of the argument
    Dim amt As Double = 2
    lstResults.Items.Add(amt)
    Triple(amt)
    lstResults.Items.Add(amt)
End Sub

Sub Triple(ByRef num As Double)
    'Triple a number
    lstResults.Items.Add(num)
    num = 3 * num
    lstResults.Items.Add(num)
End Sub

Private Sub btnEnd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    End
End Sub
```



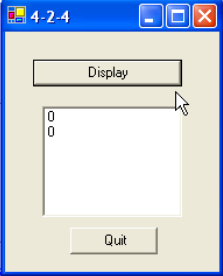
LOCAL VARIABLES

- Variable declared in even or Sub procedure with Dim state is said to be local to procedure

```
Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    'Demonstrate that variables declared in a Sub procedure
    'do not retain their values in subsequent calls
    Three()
    Three()
End Sub

Sub Three()
    'Display the value of num and assign it the value 3
    Dim num As Double
    lstResults.Items.Add(num)
    num = 3
End Sub

Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    End
End Sub
```

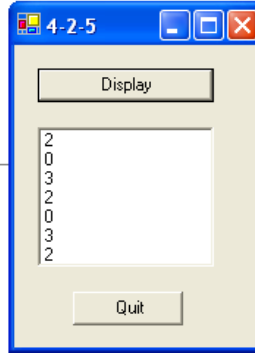


LOCAL VARIABLES

- Same name can be used in different Sub procedures but if they are local, VB.NET treats them differently

```
Private Sub btnDisplay_Click(ByVal sender As System.Object,
    'Demonstrate the local nature of variables
    Dim x As Double = 2
    lstResults.Items.Clear()
    lstResults.Items.Add(x)
    Trivial()
    lstResults.Items.Add(x)
    Trivial()
    lstResults.Items.Add(x)
End Sub

Sub Trivial()
    'Do something trivial
    Dim x As Double
    lstResults.Items.Add(x)
    x = 3
    lstResults.Items.Add(x)
End Sub
```



CLASS-LEVEL VARIABLES

- Variable visible to every procedure in form's code without being passed
- Dim statement for class-level variable placed anywhere between statements Public Class Form1 and End Class, provided Dim is not in a procedure
 - Generally placed right after Windows Form Designer generated code
- When class-level variable has its value changed by procedure, value persists even after procedure finished executing
- Variable said to have class-level scope

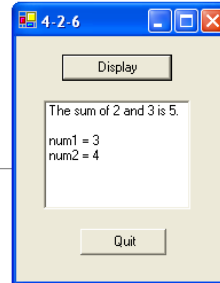
CLASS-LEVEL VARIABLES

Windows Form Designer generated code

```
Dim num1, num2 As Double 'Class-level variable

Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    'Display the sum of two numbers
    num1 = 2
    num2 = 3
    lstResults.Items.Clear()
    AddAndIncrement()
    lstResults.Items.Add("")
    lstResults.Items.Add("num1 = " & num1)
    lstResults.Items.Add("num2 = " & num2)
End Sub

Sub AddAndIncrement()
    'Display numbers and their sum
    lstResults.Items.Add("The sum of " & num1 & " and " & _
        num2 & " is " & (num1 + num2) & ".")
    num1 += 1 'Add 1 to the value of num1
    num2 += 1 'Add 1 to the value of num2
End Sub
```

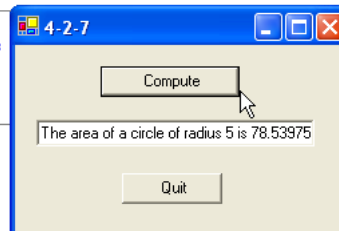


CLASS-LEVEL VARIABLES

- Can assign values to class-level variables as soon as created

```
Private Sub btnCompute_Click(ByVal sender As System.Object, ByVal e As Sys
    'Display the area of a circle of radius 5
    txtArea.Text = "The area of a circle of radius 5 is " & (pi * 5 * 5)
End Sub

Private Sub btnQuit_Click(ByVal sender As
End
End Sub
End Class
```



FUNCTION PROCEDURES

- VB.NET has number of built-in functions
- User can also define functions
 - Called Function procedures or user-defined functions
 - Designed similar to Sub procedures
 - Used same way as built-in functions used
 - Have single output of any data type

FUNCTION PROCEDURES

- Format of function procedure:

```
Function FunctionName (ByVal var1 As Type1, _  
                        ByVal var2 As Type2, ...) As dataType  
    statement(s)  
    Return expression  
End Function
```

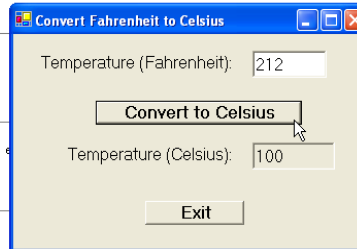
- Variable in top line called parameter
- Variable passed to function normally passed by value, but could be by reference

FUNCTION PROCEDURES

```
Private Sub btnConvert_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnConvert
    Dim FahrenheitTemp, CelsiusTemp As Double
    FahrenheitTemp = CDb1(txtTempF.Text)
    CelsiusTemp = FtoC(FahrenheitTemp)
    txtTempC.Text = CStr(CelsiusTemp)
    'Note: The above four lines can be replaced with the single line
    'txtTempC.Text = CStr(FtoC(CDb1(txtTempF.Text)))
End Sub

Function FtoC(ByVal t As Double) As Double
    'Convert Fahrenheit temperature to Celsius
    Return (5 / 9) * (t - 32)
End Function

Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnExit
    End
End Sub
End Class
```

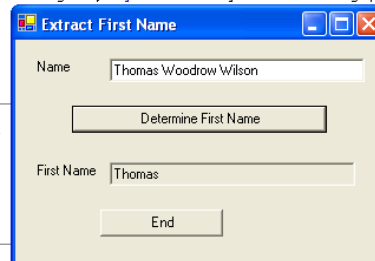


FUNCTION PROCEDURES

```
Private Sub btnDetermine_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDetermine
    'Determine a person's first name
    Dim name As String
    name = txtFullName.Text
    txtFirstName.Text = FirstName(name)
End Sub

Function FirstName(ByVal name As String) As String
    'Extract the first name from a full name
    Dim firstSpace As Integer
    firstSpace = name.IndexOf(" ")
    Return name.Substring(0, firstSpace)
End Function

Private Sub btnEnd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEnd
    End
End Sub
```



USER DEFINED FUNCTION HAVING SEVERAL PARAMETERS

- Input to user-defined function can consist of one or more values

```
Private Sub btnCalculalte_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCalculalte.Click
    'Calculate the length of the hypotenuse of a right triangle
    Dim a, b As Double
    a = Cdbl(txtSideOne.Text)
    b = Cdbl(txtSideTwo.Text)
    txtHyp.Text = CStr(Hypotenuse(a, b))
End Sub

Function Hypotenuse(ByVal a As Double, ByVal b As Double) As Double
    'Calculate the hypotenuse of a right triangle
    'having sides of lengths a and b
    Return Math.Sqrt(a ^ 2 + b ^ 2)
End Function

Private Sub btnQuit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnQuit.Click
    End
End Sub
```

USER DEFINED FUNCTION HAVING SEVERAL PARAMETERS

```
Private Sub btnCompute_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCompute.Click
    'Find the future value of a bank deposit
    Dim p As Double 'principal, the amount deposited
    Dim r As Double 'annual rate of interest
    Dim c As Double 'number of times interest is compounded per year
    Dim n As Double 'number of years
    InputData(p, r, c, n)
    DisplayBalance(p, r, c, n)
End Sub

Sub DisplayBalance(ByVal p As Double, ByVal r As Double, _
    ByVal c As Double, ByVal n As Double)
    'Display the balance in a text box
    Dim balance As Double
    balance = FutureValue(p, r, c, n)
    txtBalance.Text = FormatCurrency(balance)
End Sub

Function FutureValue(ByVal p As Double, ByVal r As Double, _
    ByVal c As Double, ByVal n As Double) As Double
    'find the future value of a bank savings account
    'p principal, the amount deposited
    'r annual rate of interest
    'c number of times interest is compounded per year
    'n number of years
    Dim i As Double 'interest rate per period
    Dim m As Double 'total number of times interest is compounded
    i = r / c
    m = c * n
    Return p * ((i + 1) ^ m)
End Function

Sub InputData(ByRef p As Double, ByRef r As Double, _
    ByRef c As Double, ByRef n As Double)
    'Get the four values from the text boxes
    p = Cdbl(txtAmount.Text)
    r = Cdbl(txtRate.Text)
    c = Cdbl(txtNumComp.Text)
    n = Cdbl(txtNumYrs.Text)
End Sub
```