Subprograms

CE 311 K - Introduction to Computer Methods

Daene C. McKinney

Introduction

- Modularity
- Sub Procedures
- Arguments
  - Pass By Value
  - Pass By Reference

Modularity

- Decompose problems into components
- Model how components interact with each other
- **Subprograms** are used for this decomposition
  - Independent sections of code that performs specific tasks
  - Written and tested separately
  - Easier to maintain and modify
  - Pre-defined libraries
  - programmer defined libraries

www.tamiya.com

Sub & Functions Procedures

- Sub Procedures
  - Used to perform tasks
  - Can return values in their arguments
  - Used to receive or process input, display output or set properties
- Function Procedures
  - Return a value in the function name
  - Used for calculations

Sub Procedures

- A "Sub" procedure
  - is part of a Program
  - performs a task (or 2)
  - has a name
  - is invoked with a "Call" statement
- A "Sub"
  - has its own code

Program

```
Program
...
Call Name( )
...
End program
```

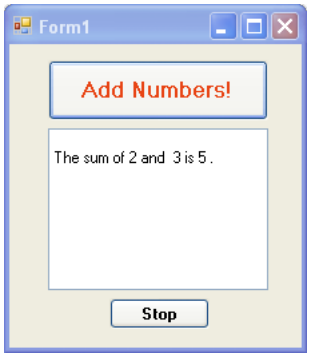
Sub procedure

```
Sub Name( )
    statement(s)
End Sub
```

Example – w/o Sub

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal
    Dim num1, num2 As Double
    num1 = 2
    num2 = 3
    ListBox1.Items.Add("The sum of " & num1 & " and " & _
        " " & num2 & " is " & num1 + num2 & " .")
End Sub
```

This version does NOT use a "sub"

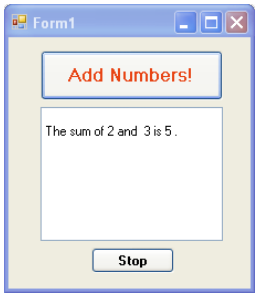


Example – w/Sub

```
Private Sub Button1_Click(ByVal sender As System.Object, ByV
    Dim num1, num2 As Double
    num1 = 2
    num2 = 3
    DisplaySum(num1, num2)
End Sub

Sub DisplaySum(ByVal num1 As Double, ByVal num2 As Double)
    ListBox1.Items.Add("The sum of " & num1 & " and " & _
        " " & num2 & " is " & num1 + num2 & " .")
End Sub
```

This version does use a Sub



Example – w/Sub

Program "calls" Sub procedure

```
DisplaySum(num1, num2)
```

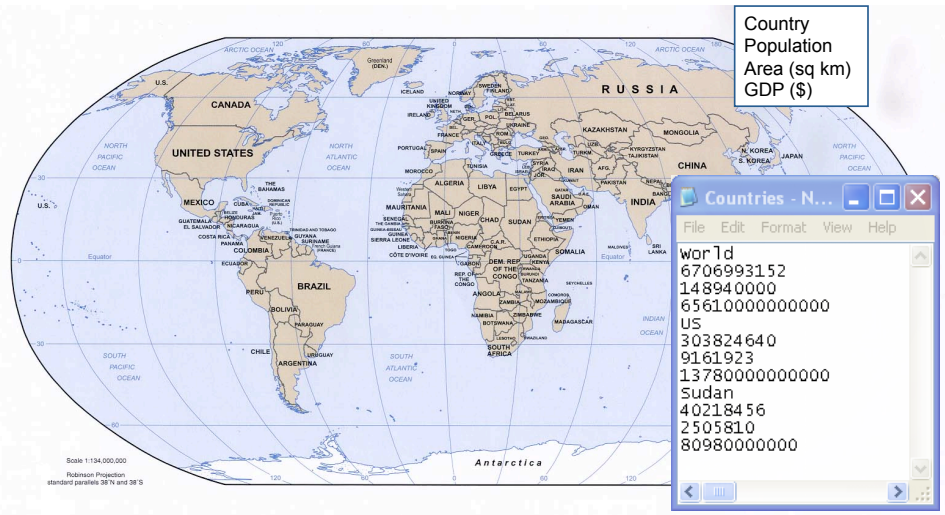
Sub procedure

```
Sub DisplaySum(ByVal num1 As Double, ByVal num2 As Double)
```

Sub receives the variable "value"

Sub declares the variables in title line

Example – w/ Sub & File I/O

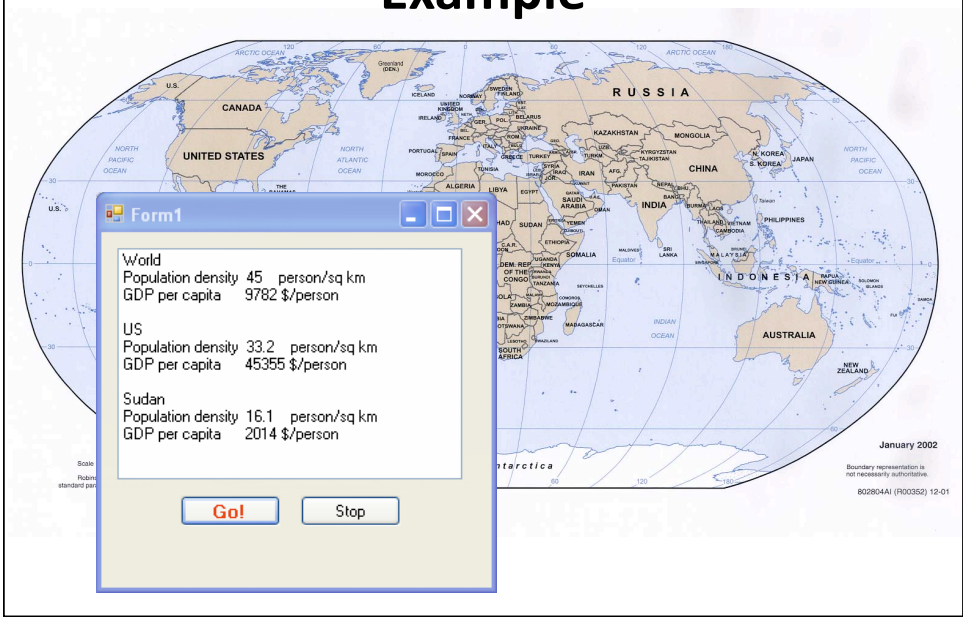


Example

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As Syste
    Dim country As String, pop, area, gdp As Double
    Dim sr As IO.StreamReader = IO.File.OpenText("C:\temp\Countries.txt")
    country = sr.ReadLine
    pop = CDbl(sr.ReadLine)
    area = CDbl(sr.ReadLine)
    gdp = CDbl(sr.ReadLine)
    Calculate(country, pop, area, gdp)
    country = sr.ReadLine
    pop = CDbl(sr.ReadLine)
    area = CDbl(sr.ReadLine)
    gdp = CDbl(sr.ReadLine)
    Calculate(country, pop, area, gdp)
```

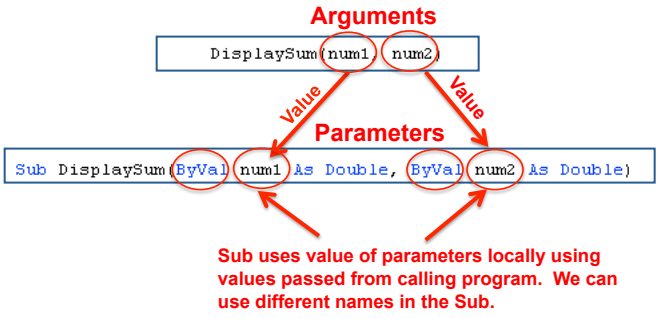
```
Sub Calculate(ByVal country As String, _
              ByVal pop As Double, ByVal area As Double, _
              ByVal gdp As Double)
    Dim pop_density, gdp_per_capita As Double
    pop_density = Math.Round(pop / area, 1)
    gdp_per_capita = Math.Round(gdp / pop, 0)
    ListBox1.Items.Add(country)
    ListBox1.Items.Add("Population density  " & _
                       pop_density & "    person/sq km")
    ListBox1.Items.Add("GDP per capita      " & _
                       gdp_per_capita & " $/person")
    ListBox1.Items.Add(" ")
End Sub
```

Example



Argument Passing By Value

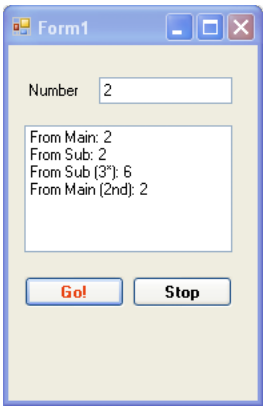
- **Pass By Value**
  - Sends the ***value*** of the argument to the Subprogram
  - Argument is evaluated and its value is passed and used ***locally*** in the Subprogram
  - Subprogram can not change the value of the variable in the calling program



Example – Pass By Value

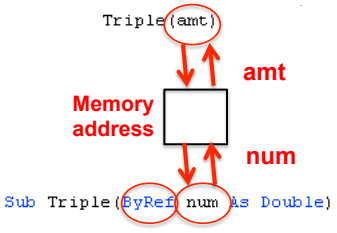
```
Private Sub Button1_Click(ByVal sender As System.Obje
    Dim amt As Double
    amt = CDbl(TextBox1.Text)
    ListBox1.Items.Add("From Main: " & amt)
    Triple(amt)
    ListBox1.Items.Add("From Main (2nd): " & amt)
End Sub

Sub Triple(ByVal num As Double)
    ListBox1.Items.Add("From Sub: " & num)
    num = 3 * num
    ListBox1.Items.Add("From Sub (3*): " & num)
End Sub
```



Argument Passing By Reference

- **Pass By Reference**
  - ***Memory address*** of argument passed
  - Argument value ***can*** be changed in the Subprogram
  - Value in the calling program is also changed

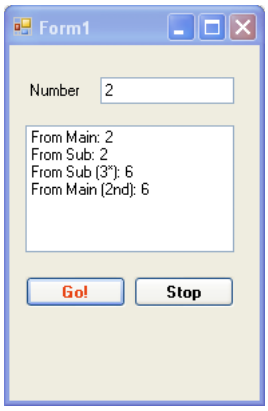


Sub uses value of parameters locally using address passed from calling program. Sub changes value residing at the address of the argument. This also changes it in the calling program.

Example – Pass By Reference

```
Private Sub Button1_Click(ByVal sender As System.Obj
    Dim amt As Double
    amt = CDbl(TextBox1.Text)
    ListBox1.Items.Add("From Main: " & amt)
    Triple(amt)
    ListBox1.Items.Add("From Main (2nd): " & amt)
End Sub

Sub Triple(ByRef num As Double)
    ListBox1.Items.Add("From Sub: " & num)
    num = 3 * num
    ListBox1.Items.Add("From Sub (3*): " & num)
End Sub
```

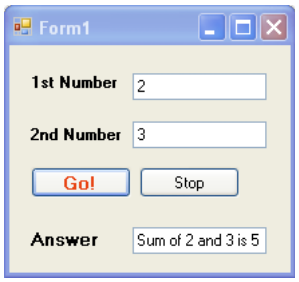


Example – Pass By Reference

```
Private Sub Button1_Click(ByVal sender As System.Object
    Dim x, y As Double
    GetNumbers(x, y)
    DisplaySum(x, y)
End Sub

Sub GetNumbers(ByRef x As Double, ByRef y As Double)
    x = CDbl(TextBox1.Text)
    y = CDbl(TextBox2.Text)
End Sub

Sub DisplaySum(ByVal x As Double, ByVal y As Double)
    Dim sum As Double
    sum = x + y
    TextBox3.Text = "Sum of " & x & " and " & y & _
    " is " & sum
End Sub
```



Summary

- Modularity
- Sub Procedures
- Arguments
  - Pass By Value
  - Pass By Reference