Learning Objectives

- Why Subroutines?
- How to declare subroutine variables?
- Passing values in and out of subroutines
- What are functions?
- How to call an EXCEL function from VB?
- How to use a VB function in EXCEL?

Subprograms/functions

- VBA allows both subprograms and functions, and they are different
- Functions are similar to subprograms, but are shorter and can compute just one value
- Subroutines are longer and can compute multiple values

Trap rule program (without sub)

n, i should be integers
h, fa, fb, a, b, x, fx, fxSum, sum should be single
n = 30
a = 1
b = 8
fa = (a * 2) * log(a)
fb = (b * 2) * log(b)
h = (b - a) / n
For i = 1 To n - 1
x = a + (h * i)
fx = (x * 2) * log(x)
fxSum = fxSum + fx
Next
sum = h * (fa + (2 * fxSum) + fb) / 2
Cells(1, 1) = sum
End Sub

Note, here we are repeating the formula three times
Also, note LOG is VB is natural log, LN

Trap rule program (with sub)

n, i should be integers
fa, fb, a, b, x, sum, h, x should be single
a = 30
b = 8
h = (b - a) / n
Call Myexpress (a, fa)
Call Myexpress (b, fb)
For i = 1 To n - 1
x = a + h * i
Call Myexpress (x, fx)
fxSum = fxSum + fx
Next
sum = h * (fa + (2 * fxSum) + fb) / 2
Cells(1, 1) = sum
End Sub

Sub Myexpress (x1 as Single, fx1 as Single)
fx1 = (x1 * 2) * Log(x1)
End Sub

Trap rule program (with sub)

- You describe the f(x) only once in the sub
- If you need to change f(x), you have to just change it in the subroutine alone
- Less chance of making errors
- Call the subroutine Myexpress every time you want to evaluate the value of the function at x.

Limitation of the program without a subroutine

- We have to compute f(x) at least 3 times in the program
- If the expression is large, mistakes can happen. Difficult to debug such errors.
- If you want to change the expression, you have to change it in 3 places.
Subprograms – How does it work?

- When you call the subroutine Myexpress from the main program, the control of the program will shift to sub Myexpress
- Once the subprogram is executed, the control will come back to the line after the calling line
- Example,
  Call Myexpress (x as Single, fx as Single)
  We send the value of x into the sub program and get the value of fx, which is the value of the f(x) at x.

Passing by Reference (default, and also our preferred method)

- All the variables declared in the main program are stored in a memory address within the computer.
- Consider two variables in the call: a, fa
- Let a = 100, fa = 500 in the main program

Passing by Value (another option)

- Sometimes you may wish to protect the value of a variable in the main program (not allowing it to change)
- This is accomplished by placing variable names in calling statement using an extra bracket around each variable:
  Call Myfunc((a), (b), c, d, e) – here a and b are passed by value, and c,d, and e are passed by reference.

WE WILL NOT USE THIS OPTION IN THIS CLASS

Beware of the order of variables in the argument list!

- Argument list of variables are the variables we specify within the brackets. These variables are exchanged between the main and calling programs
- In our example, argument list in the main program has: “a” and “fa”
  Call Myexpress (fa, fa)
- In the subprogram variables in the argument list are “x1” and “fx1”
  Sub Myexpress (x1 as Single, fx1 as Single)
- In this example, a is linked to x1, and fa to fx1
- Order is very important
  – If values in the call is reversed as fa and a then sub will treat x1 as fa, and fx1 as a

Passing by Reference

- In Visual basic, by default, the value is passed to the subroutine by reference.
- When you pass the variable by reference, you pass the memory address of the location where the variable is stored.

Advantages of using subroutines

- Subroutines can handle many variables
- Subroutines can modify the input parameters
- Subroutines can output several parameters
- In this class, it is recommended that you use subroutines (not functions)
- Also, in the argument list place the input variables first and then place the output variables. (This is not required by VB, but it is a good programming practice)
Construction site example (subroutine)

<table>
<thead>
<tr>
<th>Construction site example (subroutine)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No of plots</strong></td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

**Example:**

```vba
Sub subroutine
    Dim r1 As Single
    Dim r2 As Single
    Dim r3 As Single
    Dim a1 As Single, a2 As Single, a3 As Single
    Dim total_area As Single
    Dim total_perimeter As Single
    Cells(1, 1) = a1
    Cells(2, 1) = a2
    Cells(3, 1) = a3
    Cells(1, 2) = r1
    Cells(2, 2) = r2
    Cells(3, 2) = r3
    Cells(1, 3) = total_area
    Cells(2, 3) = total_perimeter
End Sub
```

**Note:**

- The **Subroutine** example uses **Dim** to declare variables.
- **Call** statements are used to invoke the **subroutine**.
- **End Sub** marks the end of the subroutine.

Typical sub program

**Example:**

```vba
Sub Myexpres(x1 As Single, x2 As Single)
    fx = (x1 ^ 2) * Log(x2)
End Sub
```

**Subroutines are defined as:**

- **Sub** name( as Single, n as integer, etc., y as single, fx, etc.)

**Input variables:**

- x1, x2, etc.

**Output variables:**

- y, fx

**Most of the time, x and n will not be modified.**

**Argument list contains both input and output variables**

**Name of the subroutine “name” is never used.**

Defining Functions

- **Functions are defined as:**

  ```vba
  Function name(x, n as Single, etc.) as Single
  ... only input variables are used in the argument list
  ... name = ...
  End Function
  ```

  **Note:**
  - The outer single declares the variable “name” as single
  - Name of function must be used as variable calculated within function as name is example
  - Function to calculate the f(x) in trapezoidal rule:

  ```vba
  Function Myvalue(x as Single) as Single
  Myvalue = x^2 * Log(x)
  End Function
  ```

Function Example

**Calling the function accomplished by:**

```vba
fa = Myvalue(a)
fh = Myvalue(b)
fx = Myvalue(x)
```

**Note:**

- A function can be used multiple times in the same line

Trap rule program (with function)

```vba
Function Myvalue(x as Single) as Single
    Myvalue = (x^2) * Log(x)
End Function
```

**Example:**

```vba
For i = 1 To n - 1
    x = x + h * i
    fx = Myvalue(i)
    fxSum = fxSum + fx
Next i
sum = h * (fa + (2 * fxSum) + fb) / 2
Cells(i, 1) = sum
End Sub
```
### Construction site example (function)

<table>
<thead>
<tr>
<th>No of sites</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radius of the sites</td>
<td>10, 12, 15</td>
</tr>
<tr>
<td>Total Area</td>
<td>1474</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function Name</th>
<th>metertofeet</th>
</tr>
</thead>
<tbody>
<tr>
<td>x As Single</td>
<td>x * 3.28</td>
</tr>
</tbody>
</table>

```vbnet
Public Function metertofeet(x As Single) As Single
    metertofeet = x * 3.28
End Function
```

### Subroutines Vs Functions

- A subroutine takes a number of input parameters and computes and outputs a number of parameters.
- The output variable are passed within the parenthesis
- Invoke by using a call statement. E.g.,
  ```vbnet
  Call Myexpress(b, fb)
  ```

- A function takes a number of input parameters and computes and returns a single value.
- Name of the function is the output value returned by the function.
- Invoke by using the function name within the equation of interest. E.g.
  ```vbnet
  sum = h * (Myvalue(a) + (2 * fxSum) + Myvalue(b)) / 2
  ```

### How to use a VB function in EXCEL

- You can invoke a VB function from your EXCEL spreadsheet. This can be useful way to evaluate certain formula
- Click on Visual Basic editor icon under developer tab
- Insert module
- Insert procedure
- Select type as function, enter name as: “metertofeet” and then type the following code
- Go to EXCEL and use the function, as you would use any normal EXCEL function

```vbnet
Private Sub CommandButton1_Click()
    Dim nfact As Integer
    nfact = Application.WorksheetFunction.Fact(Cells(1, 1))
    Cells(1, 2) = nfact
End Sub
```

### How to use an EXCEL function in VB

- Visual basic can invoke specialized EXCEL functions with the VB code. For example, EXCEL has function called “fact” to compute factorials. This function can be invoked as shown below

```vbnet
Private Sub CommandButton1_Click()
    Dim nfact As Integer
    nfact = Application.WorksheetFunction.Fact(Cells(1, 1))
    Cells(1, 2) = nfact
End Sub
```