Lab 8 - Functions

Introduction

Visual Basic programs are comprised of one or more modules or procedures. In VB, these procedures are either "sub-procedures" or "functions". Each procedure is a self-contained block of code that performs a specific task. Often the algorithm to solve a problem is more complex than those we have seen and a programmer would like to break the problem up into subproblems to develop the solution. In attempting to solve a subproblem at one level, we introduce new subproblems at lower levels. This process, called “top-down design”, proceeds from the original problem at the top level to the subproblems at each lower level. One way that programmers implement top-down design in their programs is by defining their own functions. Often a programmer will write one sub-program or function for each subproblem.

Functions

A function is invoked (or “called”) when its name is encountered in the execution of a program. When a function is called, program control passes to the function. The calling program can pass information to the function by passing arguments “by value” or “by reference”. When passing information “by value”, the numerical values of the arguments are evaluated in the calling program and the values are passed to the function. (Note: For info passing information “By Reference”, see the lecture slides or the book.)

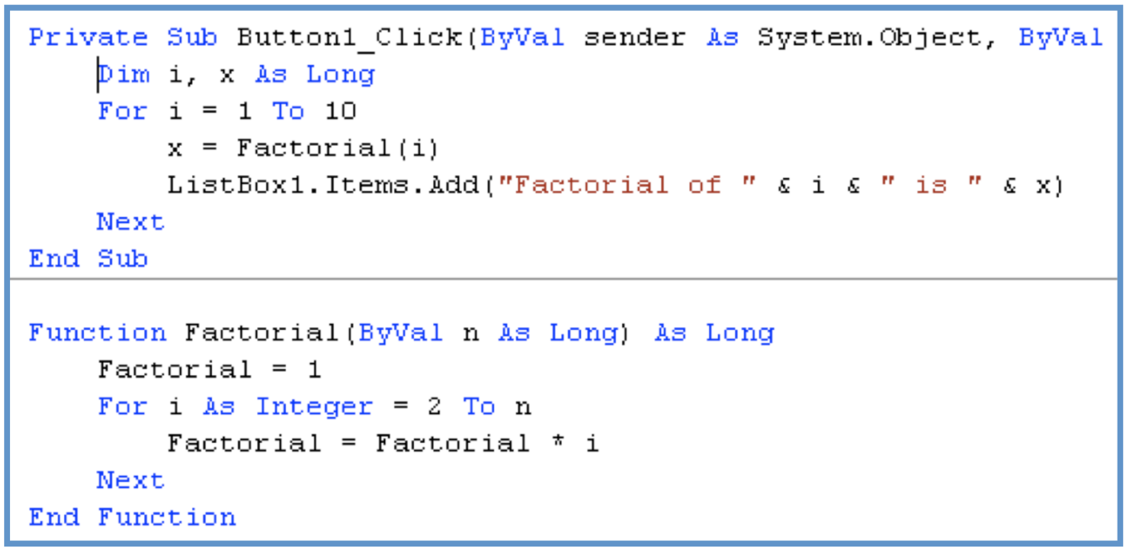
Every function has a header and a body. A function header consists of:

- The function name;
- A list of argument names and types; and
- The data type of the function

A function header is followed by the function body. When the function completes executing the statements in the function body, program control returns to the calling program. The called function may return information (a value) to the calling program.

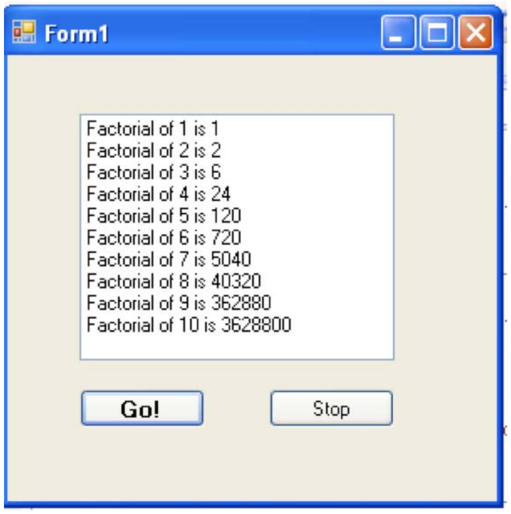
Example

Consider the following example for computing the factorial of integers from 1 to 6. It consists of a main program and a computational function that computes the factorial of an integer passed to it.



```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal
    Dim i, x As Long
    For i = 1 To 10
        x = Factorial(i)
        ListBox1.Items.Add("Factorial of " & i & " is " & x)
    Next
End Sub

Function Factorial(ByVal n As Long) As Long
    Factorial = 1
    For i As Integer = 2 To n
        Factorial = Factorial * i
    Next
End Function
```



```
Factorial of 1 is 1
Factorial of 2 is 2
Factorial of 3 is 6
Factorial of 4 is 24
Factorial of 5 is 120
Factorial of 6 is 720
Factorial of 7 is 5040
Factorial of 8 is 40320
Factorial of 9 is 362880
Factorial of 10 is 3628800
```

Assignment

Green and Ampt Infiltration (Chow, V., D. Maidment and L. Mays, Applied Hydrology, McGraw-Hill, 1988, pages 116 - 117)

Infiltration is the process of water penetrating from the ground surface into soil (usually form rainfall). Depending on the amount of infiltration and the soil properties, water may penetrate a few centimeters to several meters into a soil. Cumulative infiltration is the total amount of water infiltrated during a given time period (say a whole storm period). A commonly used infiltration equation is the Green-Ampt equation for the cumulative infiltration F after a period of time t

$$F = Kt + \psi\Delta\theta \ln\left(1 + \frac{F}{\psi\Delta\theta}\right) \qquad (1)$$

where:

F = cumulative infiltration (cm), (unknown)
K = soil conductivity (cm/hr) = 0.65 cm/hr
t = time of infiltration (hr) = 1 hr
ψ = suction head (cm) = 16.7 cm
$\Delta\theta$ = difference in dry versus wet soil moisture = 0.34
ln = natural logarithm

F, often used in hydraulic drainage design calculations, can be found by solving this equation for F. One numerical method that can be used to solve this equation is “fixed-point iteration”. In fixed-point iteration, we guess the value of F and substitute it into the right hand side of the equation

$$F_1 = Kt + \psi\Delta\theta \ln\left(1 + \frac{F_0}{\psi\Delta\theta}\right) \qquad (2)$$

where F_0 is the guessed value and F_1 is the new value. Then we substitute this new value, F_1, into the right-hand-side of the equations and obtain the next value F_2

$$F_2 = Kt + \psi\Delta\theta \ln\left(1 + \frac{F_1}{\psi\Delta\theta}\right) \qquad (3)$$

We continue this process until the values of F stop changing from one iteration to the next (that is, they converge). So, after i iterations of this process, we have

$$F_{i+1} = Kt + \psi\Delta\theta \ln\left(1 + \frac{F_i}{\psi\Delta\theta}\right) \quad (4)$$

The relative approximate error (in percent) can be written as

$$e_A = \left|\frac{F_{i+1} - F_i}{F_{i+1}}\right| * 100 \quad (5)$$

When the relative approximate error is less than the precision we require (say, e_{min}), we can stop our calculations. That is, when

$$e_A < e_{\min} \qquad (6)$$

stop computing new approximations. Notice that the right hand side of equation 4 can be written as a function

$$\begin{aligned} F_{i+1} &= g(F_i) \\ &= Kt + \psi\Delta\theta \ln\left(1 + \frac{F_i}{\psi\Delta\theta}\right) \quad (7) \end{aligned}$$

1. Write a Visual Basic program that uses Fixed-Point Iteration to solve for the infiltration F into the soil.
2. In your program, use a function that evaluates $g(F_i)$, like in equation 7.
3. Call this function from a loop.
4. The loop should stop when the relative approximate error is less than $e_{\min} = 1x10^{-6}$.
5. The program should print out the final value of F.

Answer

Oh, by the way, the answer is 3.17 cm of water infiltrated after one hour of infiltration.

Turn in:

1. A copy of the Visual Basic code for your program.
2. A copy of the output file for your program.