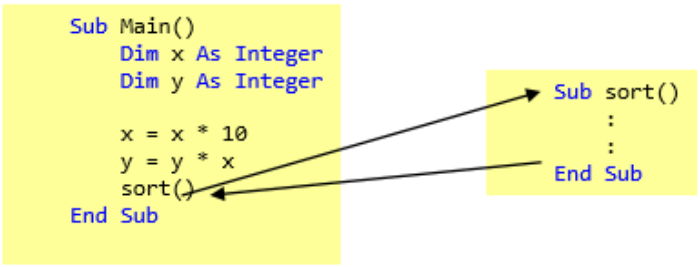
knowledgeplus
PROMOTING IT

VB Note Lesson 7

Subroutines & Procedures

Initially, a program was written as one monolithic block of code. The program started at the first line of the program and continued to the end.

Program languages have now been developed to be structured. A problem can be divided into a number of smaller subroutines (also called procedures). From within one subroutine, another subroutine can be called and executed:



This gives the advantage of reusing common subroutines. It also makes it easier to program, as each procedure can be tested before moving onto the next (breaking a large problem into smaller ones).

With so many subroutines, the computer needs to know which one to execute first. In VB.Net, the `main()` subroutines is always executed first. When writing programs with lots of subroutines, it is good practice to create the subroutine before it is uses and the last subroutine is the very last in the module

```
Module Module1
    Dim num1 As Integer
    Dim num2 As Integer
    Dim answer As Integer

    Sub input_sub()
        Console.Clear()
        Console.WriteLine("Enter number 1")
        num1 = Console.ReadLine
        Console.WriteLine("Enter number 2")
        num2 = Console.ReadLine
    End Sub

    Sub Calculation()
        answer = num1 * num2
    End Sub

    Sub output_sub()
        Console.Write("the product of " & num1 & " and " & num2 & " is ")
        Console.WriteLine(answer)
        Console.ReadLine()
    End Sub

    Sub Main()
        input_sub()
        Calculation()
        output_sub()
    End Sub
End Module
```

Variable Scope

As described on page 7, a variable holds data while the program is running. The scope of a variable defines where it can be seen. They are classifies as either global or local

Global Variable

A global variable is declared in a module and is accessible from any procedure or function within that module.

Local Variables

A local variable is declared in a procedure or function and is only accessible within that procedure of function.

```
Module Module1
    Dim num1 As Integer
    Dim num2 As Integer
    Dim answer As Integer

    Sub input_sub()
        Console.Clear()
        Console.WriteLine("Enter number 1")
        num1 = Console.ReadLine
        Console.WriteLine("Enter number 2")
        num2 = Console.ReadLine
    End Sub

    Sub Calculation()
        answer = num1 * num2
    End Sub

    Sub output_sub()
        Console.Write("the product of " & num1 & " and " & num2 & "
is ")
        Console.WriteLine(answer)
    End Sub

    Sub Main()
        Dim answer As Char

        Do
            input_sub()
            Calculation()
            output_sub()

            Console.WriteLine("another go? Y/N")
            answer = Console.ReadLine()
        Loop Until UCase(answer) = "N"

    End Sub
        Console.WriteLine("Enter number 2")
        num2 = Console.ReadLine
    End Sub

    Sub Calculation()
        answer = num1 * num2
    End Sub

    Sub output_sub()
```

Global variables, declared before any subroutines and are available throughout the

Local variable declared within a subroutine and is only available within this subroutine.

Scope of a Variable

The scope of a variable defines which subroutines can see and use a variable.

The sample code on the previous page defines the variables num1 and num2 as global; their scope is in subroutines:

- Input_sub
- Calculations
- Output_sub

However, the variable answer twice; once as a global variable and again as a local variable within main().

The scope of the integer version is available in:

- Input_sub
- Calculations

And the car version is available within

- Output_sub

If a global and a local variable share the same name, the local variable takes precedence.

Parameters

As mentioned above, local variables only have a lifespan of the procedure. Sometimes it is useful to pass a value from one procedure to another. This is done by using parameters (or arguments)

A parameter can be passed from one procedure to another by value or by reference.

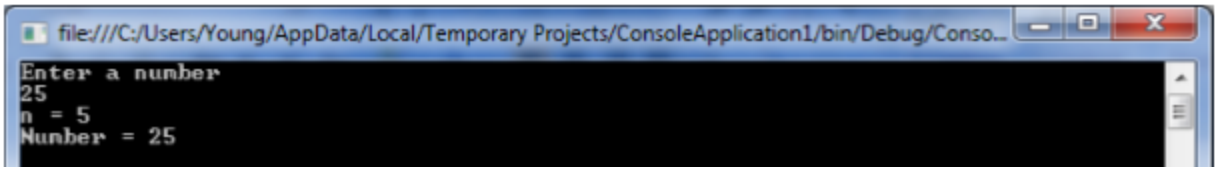
By Value

The word ByVal is short for "By Value". What it means is that you are passing a **copy** of a variable to your Subroutine. You can make changes to the copy and the original will not be altered.

```
Module Module1

    Sub WriteSQRT(ByVal n As Double)
        n = Math.Sqrt(n)
        Console.WriteLine("n = " & n)
    End Sub

    Sub Main()
        Dim number As Double
        Console.WriteLine("Enter a number")
        number = Console.ReadLine
        WriteSQRT(number)
        Console.WriteLine("Number = " & number)
        Console.ReadLine()
    End Sub

End Module
```

This procedure us expecting a double variable, which is known locally as **n**. Any changes to n do not effect the original variable

The variable **number** is passed to the subroutine *WriteSQRT*



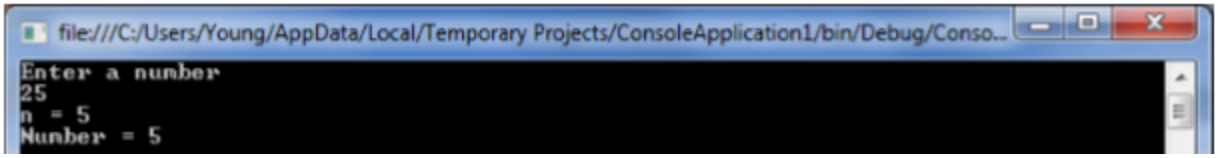
By Reference

ByRef is the alternative. This is short for By Reference. This means that you are not handing over a copy of the original variable but pointing to the original variable. Any change you make to the variable within your subroutine will effect the variable itself.

```
Module Module1

    Sub WriteSQRT(ByRef n As Double)
        n = Math.Sqrt(n)
        Console.WriteLine("n = " & n)
    End Sub

    Sub Main()
        Dim number As Double
        Console.WriteLine("Enter a number")
        number = Console.ReadLine
        WriteSQRT(number)
        Console.WriteLine("Number = " & number)
        Console.ReadLine()
    End Sub

End Module
```

This procedure us expecting a double variable, which is known locally as **n**. Any changes **WILL** effect the original variable

The variable **number** is passed to the subroutine *WriteSQRT*



Programming Projects

Write and test a procedure Swap, which takes two integers as parameters and returns the first value passed to the procedure as the second value returned by the procedure and vice versa.

Write and test a procedure OutputSymbols, which takes two parameters: an integer n and a character symbol. The procedure is to display, on the same line, the symbol n times.

For example, the call OutputSymbols (5, ' # ') should display #####.

Write and test a procedure Sort, which takes two integers as parameters and returns them in ascending order.

For example, if No1 contained 5 and No2 contained 3, then the call Sort(No1, No2) will leave the value 3 in No1 and the value 5 in No2, but the call Sort(No2, No1) will leave the variable contents as they are.

Write a program to let the computer guess a number the user has thought of, within a range specified by you as the programmer.

Challenge Projects

The game 'Last one loses' is played by two players and uses a pile of n counters. Players take turns at removing 1, 2 or 3 counters from the pile. The game continues until there are no counters left and the winner is the one who does not take the last counter. Using procedures, write a program to allow the user to specify n in the range 10 — 50 inclusive and act as one player, playing at random until fewer than 5 counters remain. Try playing against your program, and then playing to win.

Create a procedure GetLotteryNumbers that will supply 6 unique random numbers between 1 and 49. One possible strategy, or algorithm, is:

> *Initialise an array by using a for loop to store the values 1 to 49*
> *Repeatedly select a random element from array until a non-zero value is selected*
> *Display this value*
> *Set that element to zero*
> *Repeat the above three steps until six numbers have been selected.*