INTRODUCTION TO VBA PROGRAMMING

LESSON5

dario.bonino@polito.it

e-Lite

Agenda

- Strings
- Iterative constructs
  - For – Next
  - Do – Loop
  - Do While –Loop
  - Do – Loop While
  - Do Until – Loop
  - Do – Loop Until

Strings

Strings

- Variables that hold text
- Can contain up to 2^{31} characters
- Declared as
  - `Dim myString As String`
- Operators
  - & - concatenation
  - `myString = “hello”, myString2=“world”`
  - `myString & myString2 → “helloworld”;`

String Functions

- `Len(myString)`
  - Counts the number of characters in myString
- `Left(myString,8)`
  - Returns the first 8 characters of myString
- `Right(myString,5)`
  - Returns the last 5 characters of myString
- `Mid(myString, 4,3)`
  - Returns 3 characters of myString starting from the 4th character from the left (included)

Examples

- myString = “Hello Everybody”
- X = Len(myString) → X = 15
- A = Left(myString,5) → A = “Hello”
- B = Right(myString,6) → B = “rybody”
- C = Mid(myString,4,5) → C = “lo Ev”

String Functions (cont.d)

- `myString` = "The lazy dog sleeps in the garden"
- `Instr(1,myString,"sleeps")`
  - Returns the position of the first occurrence of the "sleeps" string in myString starting to search from position 1
  - E.g. 14
- `myString = UCase(myString)` → myString = "THE LAZY DOG…"
- `myString = LCase(myString)` → myString = "the lazy dog…"

String Functions (cont.d)

- `Val(myString)`
  - Converts myString to a number
  - If `myString` starts with a number and then includes characters, only the first number is converted
  - If `myString` starts with characters `Val(myString)` returns `0`

ASCII codes

- Characters are represented by numbers
- Codes from 0 to 128 are equal for every machine
- Standard ASCII set
  - (0-128)

	Section
255 128	Extended ASCII Code characters
127 123	Other characters
122 97	a-z (lowercase letters)
96 91	Other characters
90 65	A-Z (uppercase letters)
64 58	Other characters
57 48	0-9 (digits)
47 33	Other characters
32	Character SPACE
31 0	Other characters

ASCII-related functions

- `Asc(string)`
  - Returns the ASCII code of the first character of `string`
  - E.g.
    - Asc("Hello") = 72
    - Asc("hello")=104
- `Chr(number)`
  - Returns the character corresponding to a given ASCII code
  - E.g.
    - Chr(72) = "H"
    - Chr(104) = "h"

Comparison

- `If( myString = “Hello”) Then ...`
  - Comparison performed by comparing ASCII codes from left to right
  - Same characters → compare the next couple
  - Different characters →their relation establishes the relation between the compared strings
  - Same characters, same lenght → strings are equal
  - Different lenght, the shorter is a prefix for the longer → the longer is “greater”

Example

- “aid” < ”basket”
- “help” > “hello”
- “help” = “help”
- “help” < “help me”

String Exercises

Exercise 1

- Write a program that given a string scrambles it by applying the following algorithm:
  1. Split the string in two substrings
     1. The first substring lenght shall be randomly selected
     2. The second substring is given by difference between the original string and the first substring
  2. Compose the final string by concatenating the first half of the first string with the second half of the second string. Then concatenate the resulting string with the second half of the first string and with the first half of the second string.
- Suggestion:
  - Rnd() returns a random number between 0 and 1
  - Rnd() must be preceded by a call to Randomize (at the beginning of the program)

Iterative Constructs

For-Next

Why Loops?

- Suppose we have to compute the factorial of a user-entered number N
  - We do not know N → cannot write the computation a priori
  - We know how to compute 4!
    - 4! = 1*2*3*4
- We can compute N! If we are able to perform
  - N! = 1*2*3*4*….*N-1*N
- We need to repeat the base operation
  - (N)! = (N-1)! * N
  - Given that 0! = 1
    - 1! = 0!*1=1*1=1
    - 2! = 1!*2=1*2 = 2
    - 3! = 2!*3=2*3=6

Why Loops?

- We need to repeat a given operation many times
  - Factorial example
  - Ask many information to the user
  - ...other repetitive tasks
- Constructs for repeating operations → **Iterative Constructs**
- Informally → Loops

For - Next

- The For-Next loop is a “counting loop”
  - Automatically counts the number of times its operations are executed
  - Shall be used when the total number of operations to execute is a priori known
- Structure

```
For index = start_value To end_value
  body
Next index
```

For - Next

- Print all the numbers between 0 and 10

```
Dim i As Integer
For i = 0 To 10
    MsgBox(i)
Next i
```

For – Next

- Reverse loop can be generated using the `Step` clause

```
Dim i As Integer
For i = 100 To 0 Step -1
  MsgBox(i)
Next i
```

Factorial

```
Dim number As Integer
Dim factorial As Double
Dim i As Integer
'ask the number to the user
number = InputBox("Insert a positive number")
'init the factorial to 1
factorial = 1
If (number >= 0) Then
    For i = 1 To number
        factorial = factorial * i
    Next i
    MsgBox (number & "! = " & factorial)
Else
    MsgBox ("Invalid Number...")
End If
```

Exit conditions

- For-next exits when the *index* has reached the *end_value*
- In some case we may want to exit before the *end_value*
  - *E.g. we want to compute the average of a given amount of numbers by entering them one by one and we want to stop the loop at some point.*
- `Exit For`
  - Allows for interrupting a For – Next loop
  - Not really clean (if we need to use Exit for, then the For-Next loop might not be suited to our problem)

Example

```
For i = 1 To 10
  ‘ask the current number
  number = InputBox(“Number?”)
  If(number=“stop”) Then
    Exit For
  Else
    average = (average * (i-1) +
    Val(number))/i
    MsgBox (“Average = ” & average)
  End If
Next i
```

Nested Loop

- We have to fill up a multiplication table
  - We need to compute all the products between row and column indexes

	1	**2**	**3**	**4**
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

Solution

```
Dim i As Integer
Dim j As Integer
Dim table As String
For i = 1 To 4
  Table = Table & “|”
  For j = 1 To 4
      Table = Table & (i*j) & “|”
  Next j
  Table = Table & VBNewLine
Next i
```

Exercise 1

- Write a program that, given a number N, computes the sum of all the numbers between 1 and N
  - E.g N=4, `sum = 4+3+2+1`
  - Solve it using a For – Next cycle
  - Think what is the optimum solution
    - remember the Gauss formula

Exercise 2

- Write a program that, given a number N, computes the value of the Fibonacci series at the Nth iteration
  - N = 1, Fibonacci = **1**
  - N = 2, Fibonacci = 1**1**
  - N = 3, Fibonacci = 11**2**
  - N = 4, Fibonacci = 112**3**
  - N = 5, Fibonacci = 1123**5**
  - N = 6, Fibonacci = 11235**8**
  - N = 7, Fibonacci = 112358**13**

Exercise 3

- Write a program that asks a text to the user and inverts the case of all the text characters
  - Text = “This Is ouR TexT”
  - newText = “tHIS iS Our tEXt”

Exercise 4

- Write a program that given a word provides all the possible rotation of the word
  - Word = “Hello”
  - Rotations = “Hello”, “elloH”, “lloHe”, “loHel”,...

Exercise 5-6-7

- Write a program that, given a text, counts all the uppercase and lowercase characters
- Write a program that asks the user a word and reverses it. The program shall continue to work unless the user enters the word “stop”
- Write a program to check if a word is palindrome, i.e. if it is equal when read from left to right and from right to left

Do While - Loop

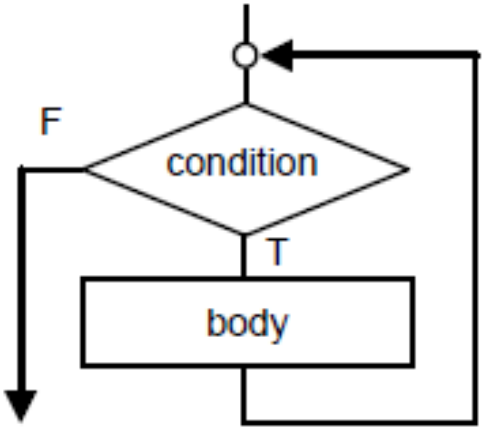
- Does some action `while` a `condition` holds
- Structure:

```
Do While condition
  action
Loop
```

- When we need to use it?
  - Whenever the number of cycles to be executed is not a priori known

Do-While Loop

- WARNING!
  - If *`condition`* is not true the loop is never executed!!



While-do loop

Do-Loop While

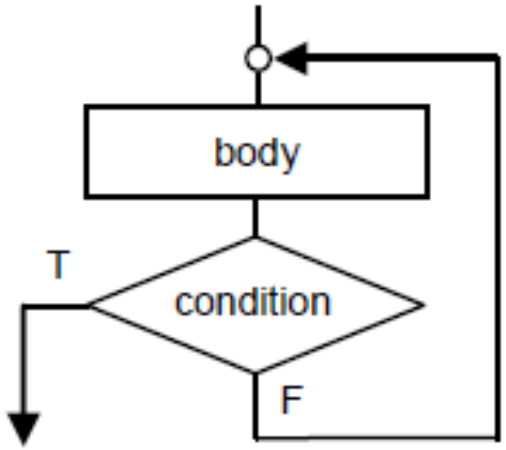
- Similar to the Do-While but executes *action* at least 1 time
- Structure

```
Do
  action
Loop While condition
```

- When we need to use it?
  - Whenever the number of cycles to be executed is not a priori known
  - AND *action* must be executed at least once

Do-Loop While

- WARNING!
  - The action is executed at least one time!



Exercise 1

- Write a program which lets you enter the price of several articles. The user will enter a price of zero to indicate that there are no more articles to be added. Once all the articles have been entered, the program must display the total price of all the articles.

Exercise 2

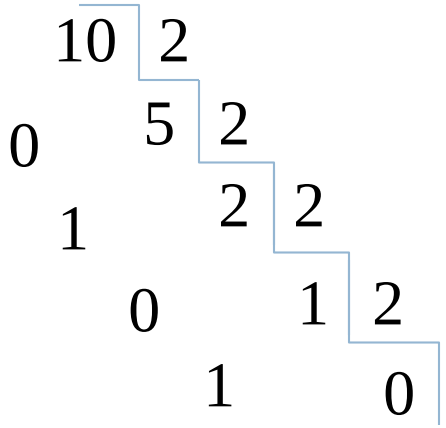
- Write a program that asks the user a word and reverses it. The program shall continue to work unless the user enters the word “stop”

Exercise 3

- write a program which allows the user to enter a number of pollution readings (integers). As each pollution reading is entered the program should test the reading and print one of the following messages:
  - comfortable (for readings less than 30)
  - moderate (for reading less from 30 to 60 inclusive)
  - dangerous (for readings over 60).
- the program should stop when the user enters any negative pollution reading.

Exercise 4

- Write a program that given a decimal number N prints out the binary representation of the number



- $10_{10} \rightarrow 0101_2$

Exercise 5

- Write a number guessing game where the computer picks a number N between 1 and 1000 and the user is required to guess which number the computer has selected.
- The game ends if the user guesses the correct number
- The game ends if the user makes more than N guesses
- At each new guess the program displays
  - the string “higher” if the guess is higher than the selected number
  - the string “lower” if the guess is lower than the selected number

Do Until - Loop

- Does some action `until` a `condition` becomes true
- Structure:

```
Do Until condition
  action
Loop
```

- When we need to use it?
  - Whenever the number of cycles to be executed is not a priori known
  - Need to loop until *`condition`* becomes true
  - Equivalent to
    - **`While Not`**`(condition)`

Do-Loop Until

- Similar to the Do-Until but executes *action* at least 1 time
- Structure

```
Do
  action
Loop Until condition
```

- When we need to use it?
  - Whenever the number of cycles to be executed is not a priori known
  - AND *action* must be executed at least once
  - AND the cycle shall be executed until *condition* becomes true
  - Equivalent to
    - **Loop While Not** (condition)