Visual Basic for Applications

Programming

Damiano SOMENZI

School of Economics and Management

Advanced Computer Skills

damiano.somenzi@unibz.it

Week 4

Outline

1. For ... Next
    - Examples
    - Exercises

For ... Next Statement

For...Next Statement

Repeats a group of statements a **specified number of times**. We look at the following syntax as reference

For counter = *start* **To** *end* [**Step** *step*]

[statements]

Next [counter]

```
For h (counter) = 1 (start) To 10 (end) Step 1 (step)
    s = s + h (statement)
Next h (counter)
```

For...Next Statement

For ... Next

1. **counter** (required): a *numeric variable* used as a loop counter
   - .. an **Integer** or **Long** variable holds the count of processing ..
   - .. the counter must be declared before the **For Next** statement ..
   - .. the execution does not have to modify the counter value ..
   - .. the execution can get the counter value ..
2. **start** (required): the initial value of counter
   - .. the value from which the processing starts ..
3. **end** (required): the final value of counter
   - .. the value to which the processing ends ..
4. **step** (optional): the amount counter that is changed each time through the loop. If not specified, **step defaults to one**.

   The value of the step argument determines loop processing as follows:
   - when step is positive or 0 loop executes if *counter* $<=$ *end*
   - when step is negative loop executes if *counter* $>=$ *end*

For...Next Statement

For ... Next

statements: one or more statements between **For** and **Next** that are executed the specified number of times

- .. after all statements in the loop have executed, step is added to counter
- .. either the statements in the loop execute again because *counter* $<=$ *end* (when step is positive) or *counter* $>=$ *end* (when step is negative) ..
- .. or the loop is exited and execution continues with the statement following the **Next** ..

Outline

1. For ... Next
    - Examples
    - Exercises

For...Next Statement

Set of Examples

one

```
Function squared() As String

    ’ the function produces: " 1 4 9 16 25"
    ’ that is a finite sequence determined computing
    ’ i^2 for i in {1, 2, 3, 4, 5}

    Dim s As String
    Dim i As Integer
    s = ""
    For i = 1 To 5
        s = s & " " & i ^ 2
    Next i
    squared = s
End Function
```

For...Next Statement

Set of Examples

two

```
Function sum(ByVal n As Integer) As Long

    ’ the function for n > = 0 computes the sum of
    ’ the even integers between 0 and n
    ’ if n = 10, the function should sum 0 + 2 + 4 + 6 + 8 + 10

    Dim s As Long
    Dim i As Integer
    s = 0
    For i = 0 To n Step 2
        s = s + i
    Next i
    sum = s
End Function
```

For...Next Statement

Set of Examples

three

```
Function sum(ByVal n As Integer) As Long

    ’ the function for n < = 0 computes the sum of
    ’ the integers between n and 0
    ’ if n = -5, then function should sum 0 - 1 - 2 - 3 - 4 - 5

    Dim s As Long
    Dim i As Integer
    s = 0
    For i = 0 To n Step -1
        s = s + i
    Next i
    sum = s
End Function
```

For...Next Statement

Set of Examples

four

```
Function sum(ByVal p As Integer) As Long

    ’ the function gets an integer p > 0 and computes
    ’ the sum (1)^1 + (1 + 2)^2 + (1 + 2 + 3)^3 + ..
    ’ .. + (1 + 2 + .. + p)^p

    Dim s As Long
    Dim b As Long
    Dim j As Integer
    Dim i As Integer
    For j = 1 To p Step 1
        b = 0
        For i = 1 To j Step 1
            b = b + i
        Next i
        s = s + b ^ j
    Next j
    sum = s
End Function
```

Outline

1. For ... Next
   - Examples
   - Exercises

String Functions

Function	**Description**	**Syntax**
Len	Returns the number of characters in a string	Len(*string*)
Mid	Returns a specified number of characters from a string	Mid(*string*, *start [, length]*)
Left	Returns a specified number of characters from the left side of a string	Left(*string*, *length*)
Right	Returns a specified number of characters from the right side of a string	Right(*string*, *length*)
UCase	Returns the specified string, converted to uppercase	UCase(*string*)
LCase	Returns a string that has been converted to lowercase	LCase(*string*)
Trim	Returns a copy of a specified string without leading and trailing spaces	Trim(*string*)

Exercise

one

Biologists frequently want to know how many times a base (A, C, G, T) occurs in a DNA sequence. A very simple function gets a DNA sequence and a base, and then it provides the occurrences of the base.

If the DNA sequence is empty or the base is mis-typed then the function should return -1

DNA sequence	base	value
`ttaaggaccccatgccctcgaataggcttgagcttgccaattaacgcg`	`t`	**11**
`ttaaggaccccatgccctcgaataggcttgagcttgccaattaacgcg`	`c`	**14**
`ttaaggaccccatgccctcgaataggcttgagcttgccaattaacgcg`	`s`	**-1**

Exercises

one

```
Function dnaAnalisys(ByVal dna As String, ByVal base As String) As Integer

    ’ the function gets a sequence and a base
    ’ the function determines the occurrences of the base
    ’ when the sequence is empty or the base is mis-typed
    ’ the function returns -1

    Dim i As Integer
    Dim s As String
    Dim b As String
    Dim f As Integer
    b = UCase(base)
    If (b = "A" Or b = "T" Or b = "C" Or b = "G") And (dna <> "") Then
        f = 0
        For i = 1 To Len(dna)
            s = Mid(dna, i, 1)
            If UCase(s) = b Then
                f = f + 1
            End If
        Next i
        dnaAnalisys = f
    Else
        dnaAnalisys = -1
    End If
End Function
```

Exercise

two

We need a function “mirror” that gets a sequence of character (the empty string is not admitted), and returns the same sequence, but in the reverse order

(i.e. `University -> ytisrevinU`)

Exercises

two

```
Function mirror(ByVal text As String) As String

    Rem the function gets a text and returns the same text,
    Rem but in the reverse order

    Dim i As Integer
    Dim m As String
    m = ""
    For i = 1 To Len(text) Step 1
        m = Mid(text, i, 1) & m
    Next i
    mirror = m
End Function
```

Exercise

three

An example of texts analysis could be to determine for each paragraph the number of valid characters of the longest sentence.

A very simple function gets a paragraph (the empty string is not admitted) as input and determines the number of characters of the longest sentence.

For this very simple computation only the character . (dot) is a sentence separator, moreover space is not counted

```
Today is Wednesday 2nd November 2011.  (31)
Advanced Compute Skills exercise class starts at 11:00.  (47)
English lecture will start at 14:00.  (30)
```

For this paragraph the function should return 47

Exercises

three

```
Function textAnalysis(ByVal paragraph As String) As Integer

    ’ the function gets a paragraph aiming to determine
    ’ the number of characters of the longest sentence
    ’ the paragraph is not the empty string

    Dim s As String
    Dim t As Integer
    Dim lsentence As Integer
    Dim nchar As Integer
    lsentence = 0
    nchar = 0
    For t = 1 To Len(paragraph)
        s = Mid(paragraph, t, 1)
        If s = "." Then
            If nchar > lsentence Then
                lsentence = nchar
            End If
            nchar = 0
        ElseIf s <> " " Then
            nchar = nchar + 1
        End If
    Next t
    textAnalysis = lsentence
End Function
```

Function

four

We need a function to encrypt a text, however a function to decrypt the encrypted text should be provided.
This function should implement a simple encryption algorithm.
For example could be helpful the following:

- Given a character and an integer number (key), the character is modified shifting its ASCII coding by the key applying the formula char(mod(ASCII(character) + key), 255)
- The corresponding VBA expression to encrypt a character is

  ```
  Chr((Asc(code) + key) Mod 255)
  ```

- Whereas the VBA expression to decrypt the character is

  ```
  Chr((255 + (Asc(Mid(text, i, 1)) - (key Mod 255))) Mod 255)
  ```

- The encrypted characters make up in the reverse order the encrypted text

Function

four

encryption

```
Function encryption(ByVal text As String, ByVal key As Long) As String

    Rem the function gets a text and an integer key
    Rem each character is encrypted shifting it by the key
    Rem the encrypted text is made up concatenating the resulting
    Rem encrypted characters in the reverse order

    Dim e As String
    Dim c As String
    Dim i As Integer
    e = ""
    For i = 1 To Len(text) Step 1
        c = Chr((Asc(Mid(text, i, 1)) + key) Mod 255)
        e = c & e
    Next i
    encryption = e
End Function
```

Function

Exercise

decryption

```
Function decryption(ByVal text As String, ByVal key As Long) As String

    Rem the function gets an encrypted text
    Rem and the integer key used to encrypt it,
    Rem then the function decrypts the text

    Dim e As String
    Dim c As String
    Dim i As Integer
    e = ""
    For i = 1 To Len(text) Step 1
        c = Chr((255 + (Asc(Mid(text, i, 1)) - (key Mod 255))) Mod 255)
        e = c & e
    Next i
    decryption = e
End Function
```

Exercises

five

In computer science numbers are typically managed in binary system.

A simple but useful function gets a binary number (representing a positive integer) and converts it in decimal system
(i.e. `11001111 -> 207`)

We observe that a binary number is a sequence of characters (**0**/**1**), however a decimal number is a “literal” integer number

Exercises

five

```
Function binaryTodecimal(ByVal n As String) As Long

    Rem the function gets a number n in binary system
    Rem n represents a positive integer
    Rem the function returns n in decimal system
    Rem if n is the empty string the function returns -1

    Dim i As Integer
    Dim c As String
    Dim d As Long
    d = 0
    If Len(n) = 0 Then
        binaryTodecimal = -1
    Else
        For i = 1 To Len(n) Step 1
            c = Val(Mid(n, i, 1))
            d = d + c * (2 ^ (Len(n) - i))
        Next i
        binaryTodecimal = d
    End If
End Function
```