Visual Basic for Applications

Programming

Damiano SOMENZI

School of Economics and Management

Advanced Computer Skills

damiano.somenzi@unibz.it

Week 3

Outline

If..Then..Else..End If Statement

If..Then..Else..End If

Conditionally executes a group of statements, depending on the value of an expression. We take the following syntax as reference

If condition **Then** [statements] [**Else** elsestatements]

or the block form syntax

If condition **Then** [statements]
[**ElseIf** condition-*n* **Then** [elseifstatements]] ...
[**Else** [elsestatements]]
End If

If..Then..ElseIf..Else..EndIf

Example

If..Then..ElseIf..Else..EndIf

```
Function weather(t As Double) As String

    Rem the function gets as input
    Rem a temperature (Celsius degree)
    Rem and provides the corresponding
    Rem climate conditions (hot, warm, fine, cold)

    If t > 30 (condition-1) Then
        weather = "HOT" (then statement)
    ElseIf t > 20 (condition-2) Then
        weather = "WARM" (else if statement)
    ElseIf t > 10 (condition-3) Then
        weather = "FINE" (else if statement)
    Else
        weather = "COLD" (else statement)
    End If

End Function
```

If..Then..Else..End If Statement

If..Then..Else..End If

- condition (required): expression that evaluates to **True** or **False**
- statements (optional in block form while required in single-line form that has no Else clause): one or more statements executed if condition is **True**
- condition–n (optional), same as condition
- elseifstatements (optional): one or more statements executed if the associated condition–n is **True**
- elsestatements (optional): one or more statements executed if no previous condition or condition-n expression is **True**

Case One - example

If `Len(Text) > Max` **Then** `MsgBox ("The text is too long")`

Text	Max	Condition	Result
”Advanced Computer Skills”	20	True	The text is too long
”Computer Science”	20	False	–

If `Income >= 20000` **Then** `Tax = Income * 0.35`

Income (€)	Upper Bound	Condition	Tax (€)
32450	20000	True	11357.5
14600	20000	False	0

If..Then..Else..End If

Case Two

Case Two - Example

```
If Len(Text) < Max Then
    MsgBox ((Max - Len(Text)) & "more character(s) ...  ")
Else
    MsgBox ("The text is too long")
End If
```

Text	Max	Condition	Result
”Advanced Computer Skills”	20	False	The text is too long
”Computer Science”	20	True	4 more character(s) ...

If..Then..Else..End If

Case Two

Case Two - Example

```
If Income >= 20000 Then
    Tax = Income * 0.35
Else
    Tax = Income * 0.20
End If
```

Income (€)	Upper Bound	Condition	Tax (€)
32450	20000	True	11357.5
14600	20000	False	2920

If..Then..ElseIf..Then..Else..End If

Case Three

Case Three - Example

```
If Len(Text) > Max Then
    MsgBox ("The text is too long")
ElseIf Len(Text) < Min Then
    MsgBox ("The text is too short")
Else
    MsgBox ((Max - Len(Text)) & " more character(s) ...  ")
End If
```

Text	Min	Max	C_1	C_2	Result
”Advanced Computer Skills”	10	20	True	–	The text is too long
”Computer Science”	10	20	False	False	4 more character(s)..
”Excel VBA”	10	20	False	True	The text is too short

If..Then..ElseIf..Then..Else

Case Three

Case Three - Example

```
If Income >= 20000 Then
    Tax = Income * 0.35
ElseIf Income <= 10000 Then
    Tax = Income * 0.08
Else
    Tax = Income * 0.20
End If
```

Income (€)	U_Bound	L_Bound	Cond1	Cond2	Tax (€)
32450	20000	10000	True	-	11357.5
9750	20000	10000	False	True	780
14600	20000	10000	False	False	2920

If..Then..Else..End If

Case Four

Case Four - Example

```
If proportion(k, n) < = 0.66 Then
    classA = classA + 1
Else
    classB = classB + 1
End If
```

k	n	Condition	classA	classB
155	1321	TRUE	1	0
980	1211	FALSE	1	1
799	991	FALSE	1	2

If..Then..ElseIf..Else..End If

Example

If..Then..ElseIf..Else..End If

```
Function belong(r As Double) As Integer

    ' the function gets as input a real number
    ' and returns the code:
    ' -1 if r belongs to (-infty, -2)
    '  0 if r belongs to [-2, 2]
    '  1 if r belongs to (2, +infty)

    If r > 2 Then
       belong = 1
    ElseIf r < -2 Then
        belong = -1
    Else
        belong = 0
    End If
End Function
```

String Functions

Some String Functions

String Functions

Function	Description	Syntax
Len	Returns the number of characters in a string	LEN(*string*)
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*)

String Functions

InStr

InStr function returns the position of the first occurrence of one string within another

- The used syntax is `InStr`([*start*,] *string1*, *string2*)
- **start** (optional), is a numeric expression that sets the starting position for each search. If omitted, search begins at the first character position (start = 1)
- **string1** (required), is the string expression being searched
- **string2** (required), is the string expression sought
- If string2 is not found INSTR returns 0

String Functions

InStr

Examples

Start	String1	String2	
1	Free University of Bozen	Bozen	20
1	Bozen-Bolzano	Bo	1
5	Bozen-Bolzano	Bo	7
5	Free University of Bozen	Free	0
1	Advanced Computer Skills	Computer	10
1	Advanced Computer Skills	computer	0

Exercise

Biologists frequently need to find patterns in a DNA sequence. A very simple function gets a DNA sequence and a possible pattern and then it should provide one of the following coding:

1	the pattern matches at least once and is a prefix of the sequence as well
2	the pattern matches at least once and is a suffix of the sequence as well
3	the pattern matches at least once, but it is neither a prefix nor a suffix
-1	the pattern does not occur

DNA squence	pattern	code
`ttaaggaccccatgccctcgaataggcttgagcttgccaattaacgcg`	`ccaat`	**3**
`ttaaggaccccatgccctcgaataggcttgagcttgccaattaacgcg`	`acgcg`	**2**

Exercise

```
Function dna(ByVal sequence As String, ByVal pattern As String) As Integer

    ' the function getes a dna sequence and a pattern to find,
    ' it checks if it is not a pattern, in this case it returns -1,
    ' if it is a pattern it checks if it is a prefix, in this case it returns 1,
    ' otherwise it checks if it is a suffix, in this case it returns 2,
    ' in the other cases it returns that the pattern simply occurs, with the code 3

    Dim p As Integer
    p = InStr(1, UCase(sequence), UCase(pattern))
    If p = 0 Then
        dna = -1
    ElseIf UCase(pattern) = UCase(Left(sequence, Len(pattern))) Then
        dna = 1
    ElseIf UCase(pattern) = UCase(Right(sequence, Len(pattern))) Then
        dna = 2
    Else
        dna = 3
    End If
End Function
```

Select Case..End Select Statement

Select Case..End Select

It executes one of several groups of statements, depending on the value of an expression. We take the following syntax as reference

Select Case **testexpression**
[**Case** **expressionlist-n** [statements–n]] ...
[**Case Else** [elsestatements]]
End Select

Select Case..End Select

Example

```
Function set(i As Double) As Integer

    ' the function gets a number and determines the corresponding set
    ' 1 = [10, 20], 2 = [21, 30]
    ' 3 = [31, 40], 4 = [31, 40], 0 = outside every set

    Select Case n (test expression)
    Case 10 To 20 (expression-1)
        set = 1
    Case 21 To 30 (expression-2)
        set = 2
    Case 31 To 40 (expression-3)
        set = 3
    Case Else
         set = 0
    End Select
End Function
```

n	Case	Result
25	21 **To** 30	"n $\in [21, 30]$"
31	31 **To** 40	"n $\in [31, 40]$"
88	**Else**	"n $\notin [10, 40]$"

Select Case..End Select Statement

Select Case..End Select

- **testexpression** (required), any expression
- **expressionlist-n** (required if a Case appears): delimited list of one or more of the forms:
  1. *expression*
  2. *expression* **To** *expression* (`To` keyword specifies a range of values)
  3. **Is** *comparison operator expression* (`Is` keyword with comparison operators to specify a range of values)
- statements–n (optional): one or more statements executed if **testexpression** matches any part of **expressionlist–n**
- If testexpression matches an expressionlist in more than one Case clause, only the statements following the first match are executed
- elsestatements (optional): one or more statements executed if **testexpression** does not match any of the Case clause

Select Case..End Select

Example

Example One

```
Function daysofmonth(m As Integer) As Integer

  ' the function gets a month (1 = January .. 12 = December)
  ' and it provides the number of days for that month,
  ' if the month is wrong then it returns zero

    Dim days As Integer
    Select Case m
    Case 4, 6, 9, 11
        days = 30
    Case 1, 3, 5, 7, 8, 10, 12
        days = 31
    Case 2
        days = 28
    Case Else
        days = 0
    End Select
    daysofmonth = days
End function
```

Select Case..End Select

Example

Example Two

```
Function group(name As String) As Integer

' the function gets student's last name and then determines
' depending on the name first letter the group he should belong
' letter A or B -> group 1
' letter C -> group 2
' letter S -> group 3
' other letters -> group 4

    Select Case UCase(Left(name, 1))
    Case "A", "B"
       group = 1
    Case "C"
       group = 2
    Case "S"
       group = 3
    Case Else
       group = 4
    End Select
End Function
```

Exercises

Exercise

Some employees have a 9 to 5 job (8 working hours). It could happen that they work more or less than the 8 working hours.
A function should determine worked extra time or not.
Consider as not worked time (and therefore not paid) if it exceeds 10 minutes, whereas consider as worked extra time (and therefore extra paid) only if it exceeds 30 minutes.
The function should gets as input the clock in time and the clock out time and then providing:

- not worked time (with minus sign)
- worked extra time (with plus sing)
- 0 in the other cases

Date	Clock In	Clock Out	Difference
18/10/2011	9:00	16:45	-15
18/10/2011	9:00	17:15	0
18/10/2011	8:45	17:20	35

Exercise

```
Function working(ByVal clockin As Date, ByVal clockout As Date) As Integer

    ' the function takes two valid times and determines
    ' worked extra time or not applying the defined rule

    Dim t As Integer
    t = DateDiff("n", clockin, clockout) - DateDiff("n", #9:00:00 AM#, #5:00:00 PM#)
    If t < -10 Then
        working = t
    ElseIf t > 30 Then
        working = t
    Else
        working = 0
    End If
End Function
```

Exercise

Winter Session 2010-11

Exercise

Build function **reducedTicket** that takes as input

- the monetary amount of the standard fare,
- the passenger age and
- whether he has required a seat reservation (true or false),

returns the corresponding ticket fare, following the rules: if the passenger is less or equal to 15 years old then the standard fare is reduced by 30%; for all passengers having requested a seat reservation an extra flat charge (without reduction) of 3 € is applied

Exercise

Winter Session 2010-11

```
Function reducedTicket(standard As Single, age As Integer, reservation As Boolean) As Single

    ' .....

    Dim fare As Single
    If age <= 15 Then
        fare = standard * (1 - 0.3)
    Else
        fare = standard
    End If
    If reservation Then
        fare = fare + 3
    End If
    ticket = fare
End Function
```

Exercise

Spring Session 2010-11

Exercise

Build a function **reimbursement** that takes as input a delivery cost (double), whether the book price is a seasonal offer (boolean), the delay in days (integer) and the daily reimbursement percentage perc (double) and returns the corresponding amount of reimbursement (double), following the rules:

- if the book price is a seasonal offer then a flat reimbursement equal to 0.3 is granted
- if the delay is higher than 5 days then an additional reimbursement is granted for each of the extra days (those exceeding the fifth) corresponding to percentage perc of the delivery cost

  (For example: for book in row 10 with perc=5% reimbursement is $0.3 + (18 - 5) \times 10\% \times 9.2$)
- If the book price is not a seasonal offer then a flat reimbursement equal to 0.8 is granted
- if the delay is higher than 3 days then an additional reimbursement is granted for each of the extra days (those exceeding the third), corresponding to percentage perc of the delivery cost

Exercise

Spring Session 2010-11

```
Function reimbursement(cost As Double, seasonal As Boolean, delay As Integer, perc As Double)

  ' .....

  If seasonal Then
    If delay <= 5 Then
      reimbursement = 0.3
    Else
      reimbursement = 0.3 + cost * ((delay - 5) * perc)
    End If
  Else
    If delay <= 3 Then
      reimbursement = 0.8
    Else
      reimbursement = 0.8 + cost * ((delay - 3) * perc)
    End If
  End If
End Function
```

Exercise

Summer Session 2010-11

Exercise

We interviewed 100 subjects on the amount of time they typically spend per week on Internet services (e-mail, chat, social networks, etc.).
The main task is to compute for different classes of ages the average hours (rounded) spent, following the schema:

Class Code	Age	Average (hours)
1	6 - 9	
2	10 - 15	
3	16 - 18	
4	19 - 25	

Build function **classCode** that takes as input the age of the interviewed, as integer, and returns the corresponding class code, as integer, 1, 2, 3 or 4 to which he/she belongs. If his/her age is outside all the four considered ranges (for example, it is 26), then the function returns class code -1

Exercise

Summer Session 2010-11

```
Function classCode(age As Integer) As Integer

  ' .....

    Dim class As Integer
    Select Case age
    Case 6 To 9
        class = 1
    Case 10 To 15
        class = 2
    Case 16 To 18
        class = 3
    Case 19 To 25
        class = 4
    Case Else
        class = -1
    End Select
    classCode = class
End Function
```