

# VISUAL BASIC FOR APPLICATIONS

Damiano SOMENZI

SCHOOL OF ECONOMICS AND MANAGEMENT

Advanced Computer Skills

`damiano.somenzi@unibz.it`



Week 6

## ① LOGICAL OPERATORS

## ② STRING FUNCTIONS

## OPERATORS

Type	Name	Symbol
Arithmetic Operators	Addition	+
	Subtraction	-
	Multiplication	*
	Division	/
	Exponentiation	^
String Operator	Concatenation	&
	Comparison	Like
<b>Logical Operators</b>	<b>AND</b>	<b>AND</b>
	<b>OR</b>	<b>OR</b>
	<b>NOT</b>	<b>NOT</b>
Comparison relations	Equal	=
	Less than	<
	Greater than	>
	Less than or equal to	<=
	greater than or equal to	>=
	Not equal to	<>

## AND

**And** operator is used to perform a *logical* conjunction on two expressions

- The syntax is *result = expression1 And expression2*
- *result* (required) should be a boolean variable, *expression1* and *expression2* (required) are any expression
- If both expressions evaluate to **True**, *result* is **True**. If either expression evaluates to **False**, *result* is **False**

# AND OPERATOR

## EXAMPLE

### EXAMPLE

1 result = ((x ≥ 10) AND (x ≤ 20))

x	x ≥ 10	AND	x ≤ 20	Result
15	TRUE	And	TRUE	TRUE
5	FALSE	And	TRUE	FALSE
25	TRUE	And	FALSE	FALSE

2 result = ((font = "blue") AND ( bgcolor = "yellow"))

font	bgcolor	font = "blue"	AND	bgcolor = "yellow"	Result
blue	blank	TRUE	And	FALSE	FALSE
red	blank	FALSE	And	FALSE	FALSE
blue	yellow	TRUE	And	TRUE	TRUE

# AND OPERATOR

## EXAMPLE

### EXAMPLE ONE

```
Sub ValidNumbers(n1 As Long, n2 As Long)
    Dim result As Boolean
    Dim rOne As Boolean
    Dim rTwo As Boolean
    rOne = ((n1 >= 1000) And (n1 <= 2000))
    rTwo = ((n2 >= 1000) And (n2 <= 2000))
    result = rOne And rTwo
    Debug.Print n1 & ", " & n2 & " are valid numbers? " & result
End Sub

Sub Main()
    Call ValidNumbers(900, 1200)
    Call ValidNumbers(1422, 1988)
    Call ValidNumbers(1789, 2012)
End Sub
```

## OR

**Or** operator is used to perform a logical disjunction on two expressions

- The syntax is *result = expression1 Or expression2*
- *result* (required) should be a numeric variable, *expression1* and *expression2* (required) are any expression
- If either or both expressions evaluate to **True**, result is **True**

# OR OPERATOR

## EXAMPLE

### EXAMPLE

① result = ((s = "YES") Or (s = "yes"))

s	s = "YES"	OR	s = "yes"	Result
"YES"	TRUE	Or	FALSE	TRUE
"yes"	FALSE	Or	TRUE	TRUE
"Yes"	FALSE	Or	FALSE	FALSE

② result = ((x <> 0) Or (y <> 1))

x	y	x <> 0	OR	y <> 1	Result
2	1	TRUE	Or	FALSE	TRUE
0	0	FALSE	Or	TRUE	TRUE
0	1	FALSE	Or	FALSE	FALSE

# OR OPERATOR

## EXAMPLE

### EXAMPLE ONE

```
Sub ValidString(s As String)
    Dim result As Boolean
    result = ((s = "YES") Or (s = "yes"))
    Debug.Print s & " is a valid string? " & result
End Sub

Sub Main()
    Call ValidString("YES")
    Call ValidString("Yes")
    Call ValidString("yes")
    Call ValidString("yeh")
End Sub
```

# OR OPERATOR

## EXAMPLE

### EXAMPLE TWO

```
Sub ValidCode(code As String)
    Dim result As Boolean
    result = ((code Like "CR_?????") Or (code Like "CA_*"))
    Debug.Print code & " is a valid code? " & result
End Sub

Sub Main()
    Call ValidCode("CC_12428")
    Call ValidCode("CR_11002")
    Call ValidCode("CR_121102")
    Call ValidCode("CA_121102")
End Sub
```

## EXERCISE 1

Write a procedure **Main** which performs the following tasks:

- 1 It holds two integers  $n_1$  and  $n_2$  and it asks the user for one more integer  $n$
- 2 Calls a procedure named **CheckNumbers** and it passes to it  $n$ ,  $n_1$  and  $n_2$  as arguments

**CheckNumbers** displays using *Message Box* function the following sentences:

- “ $n$  is not equal to  $n_1$  is: ” followed by **true** or **false**
- “ $n$  minus  $n_1$  is greater or equal to  $n_2$  is: ” followed by **true** or **false**
- “ $n$  is larger than  $n_1$  or  $n$  is larger than  $n_2$  is: ” followed by **true** or **false**

# VBA LOGICAL OPERATORS

## SOLUTION

```
Sub CheckNumbers(n As Integer, n1 As Integer, n2 As Integer)
    Dim result As Boolean
    result = n <> n1
    MsgBox (n & " is not equal to " & n1 & ": " & result)
    result = (n - n1) >= n2
    MsgBox (n & " minus " & n1 & " is larger or equal to " & _
        & n2 & ": " & result)
    result = (n > n1) Or (n > n2)
    MsgBox (n & " is larger than " & n1 & " or " & n & _
        " is larger than " & n2 & ": " & result)
End Sub

Sub Main()
    Dim i As Integer
    Dim i1 As Integer
    Dim i2 As Integer
    i = Val(InputBox("Enter an integer"))
    i1 = 22
    i2 = 11
    Call CheckNumbers(i, i1, i2)
End Sub
```

# STRING FUNCTIONS

## SOME STRING FUNCTIONS

### STRING FUNCTIONS

Function	Description	Syntax
<b>Len</b>	Returns the number of characters in a string	LEN( <i>string</i> )
<b>Mid</b>	Returns a specified number of characters from a string	MID( <i>string</i> , <i>start</i> [, <i>length</i> ])
<b>Left</b>	Returns a specified number of characters <b>from the left side of a string</b>	LEFT( <i>string</i> , <i>length</i> )
<b>Right</b>	Returns a specified number of characters <b>from the right side of a string</b>	RIGHT( <i>string</i> , <i>length</i> )
<b>UCase</b>	Returns the specified string, converted to uppercase	UCASE( <i>string</i> )
<b>LCase</b>	Returns a string that has been converted to lowercase	LCASE( <i>string</i> )
<b>Trim</b>	Returns a copy of a specified string without leading and trailing spaces	TRIM( <i>string</i> )

# STRING FUNCTION

## EXAMPLE

### EXAMPLE ONE

```
Sub SequenceOfCharLen(s As String)
    Dim length As Integer
    length = Len(s)
    MsgBox ("The length of '" & s & "' is " & length)
    length = Len(Trim(s))
    MsgBox ("The length (without leading and trailing) of '" & _
        & s & "' is " & length)
End Sub

Sub Main()
    Dim str As String
    str = " VISUAL BASIC FOR APPLICATIONS "
    Call SequenceOfCharLen(str)
End Sub
```

## EXERCISE 1

Write a procedure **Main** which calls the procedure named **ManipulateStrings** and it passes to it a string as argument. **ManipulateStrings** displays in the *Immediate Window*:

- The input string converted to uppercase
- The input string converted to lowercase
- The first 5 characters of the input string
- The last 10 characters of the input string

## SOLUTION

```
Sub ManipulateStrings(s As String)
    Debug.Print s & " -> " & UCase(s)
    Debug.Print s & " -> " & LCase(s)
    Debug.Print s & " -> " & Left(s, 5)
    Debug.Print s & " -> " & Right(s, 10)
End Sub

Sub Main()
    Dim str As String
    str = InputBox("Enter the sentence")
    Call ManipulateStrings(str)
End Sub
```

## EXERCISE 2

Write a procedure **Main** which asks the users for a password and calls a procedure **ValidatePassword** to validate it.

**ValidatePassword** checks if the password length is at least 8 characters, if the first character is a capital letter and if the ending character is "!". It warns the user if the password is valid or not

## SOLUTION

```
Sub ValidatePassword(p As String)
    MsgBox ("Valid Password: " & (p Like "[A-Z]???????*!"))
End Sub

Sub Main()
    Dim pwd As String
    pwd = InputBox("Write your password")
    Call ValidatePassword(pwd)
End Sub
```

## EXERCISE 3

Write a procedure **Main** which calls a procedure named **Initials** and it passes to it two arguments: first and last name of a person. **Initials** displays in a message box the initials of that person

## SOLUTION

```
Sub Initials(FirstName As String, LastName As String)
    Dim i As String
    i = Left(FirstName, 1) & Left(LastName, 1)
    MsgBox (FirstName & " " & LastName & " = " & UCase(i))
End Sub
```

```
Sub Main()
    Dim fn As String
    Dim ln As String
    fn = InputBox("Enter the first name")
    ln = InputBox("Enter the last name")
    Call Initials(fn, ln)
End Sub
```

## EXERCISE 4

Write a procedure **Main** which calls a procedure named **SubstringOfString** and it passes to it the following arguments: a string **str** (text), an integer **s** (starting character), and an integer **e** (ending character). **SubstringOfString** displays in a message box the part of the string from the  $s^{\text{th}}$  character up to the  $e^{\text{th}}$  character

## SOLUTION

```
Sub SubstringOfString(str As String, s As Integer, e As Integer)
    Dim length As Integer
    length = e - s + 1
    MsgBox( "The part of '" & str & "' from the " & s & "(th) up to the " & _
           & e & "(th) is " & "'" & Mid(str, s, length) & "'")
End Sub

Sub Main()
    Dim s As String
    Dim x As Integer
    Dim y As Integer
    s = InputBox("Enter the sentence")
    x = Val(InputBox("Enter the start position"))
    y = Val(InputBox("Enter the end position"))
    Call SubstringOfString(s, x, y)
End Sub
```

## 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

## EXAMPLES

Start	String1	String2	Returned
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

# STRING FUNCTIONS

## EXAMPLE ONE

```
Sub StringSearch(start As Integer, string1 As String, _
                string2 As String)
    Dim p As Integer
    p = InStr(start, string1, string2)
    Debug.Print "The position of the first occurrence of '" & string2 & _
        "' within '" & string1 & "' is " & p
End Sub

Sub Main()
    Call StringSearch(1, "Free University of Bozen", "Bozen")
    Call StringSearch(1, "Bozen-Bolzano", "Bo")
    Call StringSearch(5, "Bozen-Bolzano", "Bo")
    Call StringSearch(5, "Free University of Bozen", "Free")
    Call StringSearch(1, "Advanced Computer Skills", "Computer")
    Call StringSearch(1, "Advanced Computer Skills", "computer")
End Sub
```