

# VISUAL BASIC FOR APPLICATIONS

Damiano SOMENZI

SCHOOL OF ECONOMICS AND MANAGEMENT

Advanced Computer Skills

`damiano.somenzi@unibz.it`



Exercises

## ① DATE AND TIME

## ② GENERAL EXERCISES

## DATE DATA TYPE

- 1 Variables can be declared as **Date** data type, for example  

```
Dim dateOne As Date
```
- 2 Date variables are stored as floating-point numbers that represent
  - dates ranging from **1 January 100** to **31 December 9999**
  - times from **0:00:00** to **23:59:59**
- 3 Any recognizable date values can be assigned to Date variables
  - Date literals must be enclosed within number signs (#), for example, #January 1, 2007# or #1 Jan 07#
  - The date value could be a string expression representing a date from January 1, 100 through December 31, 9999

## DATE DATA TYPE

- Date variables display dates according to the date format recognized by your computer
- Date variables display times according to the time format (either 12-hour or 24-hour) recognized by your computer
- When other numeric types are converted to Date, values to the left of the decimal represent date information while values to the right of the decimal represent time, for example Midnight is 0 and midday is 0.5

## DATEVALUE

DateValue(date) returns a date; date argument is normally a string expression, however, date can also be any expression that can represent a date, a time, or both a date and time

```
myDate = DateValue(InputBox("Enter a date"))
```

# DATE AND TIME

## EXAMPLES

### EXAMPLES

Value	Equivalent to	Internal rep.
#December 25, 2008#	#12/25/2008#	39807
#Dec 25, 08 12:00#	#12/25/2008 12:00:00 PM#	39807.5
#31/12/2008#	#12/31/2008#	39813
#25 December 2008 20.33#	#12/25/2008 8:33:00 PM#	39807.85625
"10/1/2008"	#1/10/2008#	39823
#10:48#	#10:48:00 AM#	0.45
#22:12#	#10:12:00 PM#	0.925
39807	#12/25/2008 12:00:00 PM#	39807
0.5	#12:00:00 PM#	0.5

# DATE AND TIME

DATE, TIME, NOW

## DATE FUNCTIONS

Name	Description	Syntax
Date	Returns a <b>Date</b> containing the current system date	Date or Date()
Time	Returns a <b>Date</b> indicating the current system time	Time or Time()
Now	Returns a <b>Date</b> specifying the current date and time according your computer's system date and time	Now or Now()

# EXAMPLE

## EXAMPLE ONE

```
Function MoneyValue(t As String) As Date
    Select Case t
        Case "Deposit"
            MoneyValue = Date + 3
        Case "Transfer"
            MoneyValue = Date + 7
        Case "Check"
            MoneyValue = Date + 5
        Case Else
            MoneyValue = Date
    End Select
End Function
```

	A	B	C	D
1				
2		Date	22/01/2010 9.24	
3		<b>Transaction</b>	<b>Value</b>	
4		Transfer	01/29/10	
5		Check	01/27/10	
6		Withdraw	01/22/10	
7				

### DATEADD

Returns a Date containing a date to which a specified time interval has been added

```
DateAdd(interval, number, date)
```

- *interval* (required): a string expression that is the interval of time you want to add { "yyyy" (year), "m" (month), "d" (day), "h" (hour), "n" (minute), "s" (second) }
- *number* (required): a numeric expression that is the number of intervals you want to add. It can be positive (to get dates in the future) or negative (to get dates in the past)
- *date* (required): a Date or a date representation to which the interval is added

# DATE AND TIME

## DATEADD

### EXAMPLES

Interval	Number	Date	Returned
"d"	50	#December 25, 2008#	#2/13/2009#
"d"	-6	#12/31/2008#	#12/25/2008#
"m"	1	#1/31/2009#	#28/2/2009#
"yyyy"	3	Date	#12/5/2010#
"h"	12	Date	#12/5/2008 12:00:00PM#
"n"	1000	#25/12/2008#	#12/5/2008 4:40:00PM#

## DATEDIFF

Returns an integer specifying the number of time intervals between two specified dates:

```
DateDiff(interval, date1, date2)
```

- *interval* (required): a string expression that is the interval of time you use to calculate the difference between *date1* and *date2*  
{ "yyyy" (year), "m" (month), "d" (day), "h" (hour), "n" (minute), "s" (second) }
- *date1*, *date2* (required): two dates (Date) you want to use in the calculation

# DATE AND TIME

## DATEDIFF

### EXAMPLES

Interval	Date1	Date2	Returned
"d"	#12/31/2008#	#3/1/2009#	60
"d"	#1/1/2009#	#12/1/2008#	-31
"m"	#8/15/2008#	#12/25/2008#	4
"yyyy"	Date	#6/30/2010#	1
"h"	#12/1/2008#	#12/25/2008#	576
"n"	#12/1/2008#	#12/25/2008#	34560

## EXERCISE 1

Write the following procedures:

- A function **DaysInterval** that given a past date returns the number of days between that date and now
- A function **FutureDate** that given a number of days returns a date that represents a future date (sum of that number of days to today)
- A procedure **Main** which asks the user for a date and then displays using a *Message Box* the time interval between that date and now. Furthermore it asks for an interval and then it displays using a *Message Box* the current date/time, the days entered by the user and the future date

## SOLUTION

```
Function DaysInterval(d As Date) As Long
    DaysInterval = DateDiff("d", d, Date)
End Function

Function FutureDate(i As Integer) As Date
    FutureDate = DateAdd("d", i, Date)
End Function

Sub Main()
    Dim dt As Date
    Dim it As Integer
    dt = DateValue(InputBox("Enter the date"))
    MsgBox ("Number of days between " & dt & " and " & Date & ": " & _
        & DaysInterval(dt))
    it = Val(InputBox("Enter an interval (days)"))
    MsgBox ("Current date: " & Date & " Interval: " & it & _
        " Future date: " & FutureDate(it))
End Sub
```

# DATE AND TIME

## DATE SERIAL

### DATE SERIAL

Returns a Date for a specified year, month, and day

`DateSerial(year, month, day)`

- *year* (required): an integer between 100 and 9999, inclusive, or a numeric expression
- *month* (required): an integer, any numeric expression
- *day* (required): an integer, any numeric expression

### DAY MONTH YEAR

Function	Description	Syntax
Day	Returns an Integer specifying a whole number between 1 and 31, inclusive, representing the day of the month	Day( <i>date</i> )
Month	Returns an Integer specifying a whole number between 1 and 12, inclusive, representing the month of the year	Month( <i>date</i> )
Year	Returns an Integer containing a whole number representing the year	Year( <i>date</i> )

# DATE AND TIME

DAY MONTH YEAR

## EXAMPLES

<b>d</b>	<b>Day(d)</b>	<b>Month(d)</b>	<b>Year(d)</b>
#12/1/2008#	1	12	2008
40200	22	1	2010
#December 1, 2008#	1	12	2008
"January 1, 2009"	1	1	2009

<b>Year(y)</b>	<b>Month(m)</b>	<b>Day(d)</b>	<b>DateSerial(y,m,d)</b>
2010	2	2	#2/2/2010#
2006	2	Day(Date)	#2/5/2006#
2007	Month(Date)	30	#12/30/2007#

## EXERCISE 2

Write a function **nextDate** which given a date, an interval type and a number computes and returns the next date that is the date within  $n$  days if the interval type is "day", within  $n$  months if the interval type is "month" and within  $n$  years if the interval type is "year".

A procedure **Main** calls **nextDate** and it passes to it the arguments, then it displays the returned result

## SOLUTION

```
Function NextDate(d As Date, s As String, n As Integer) As Date
    Select Case s
        Case "day"
            NextDate = DateAdd("d", n, d)
        Case "month"
            NextDate = DateAdd("m", n, d)
        Case "year"
            NextDate = DateAdd("yyyy", n, d)
        Case Else
            NextDate = Date
    End Select
End Function
Sub Main()
    Dim dt As Date
    Dim it As String
    Dim num As Integer
    dt = DateValue(TextBox("Enter the date"))
    it = TextBox("Enter the interval type")
    num = Val(TextBox("Enter a time for the interval"))
    MsgBox "(" & dt & ", " & it & ", " & num & ") -> " & NextDate(dt, it, num)
End Sub
```

## EXERCISE 3

Write the following procedures:

- A function **TimeInterval** that given a date and an interval type ("day", "month", "year") returns the time interval between that date and now depending on the interval type
- A procedure **Main** which asks the user for an amount of money, an interest rate, a date, and an interval type. **Main** checks whether the entered data are valid (amount and interest rate are greater than zero, date is greater than now) and in this case it calls the function and then it displays the following results:

- 1  $m \times \left(1 + \frac{i}{365}\right)^{TimeInterval(days)}$
- 2  $m \times \left(1 + \frac{i}{12}\right)^{TimeInterval(months)}$
- 3  $m \times (1 + i)^{TimeInterval(years)}$

If the entered data are not valid, the procedure warns the user

## SOLUTION

```
Function TimeInterval(d As Date, t As String) As Integer
    Select Case t
        Case "day"
            TimeInterval = DateDiff("d", Date, d)
        Case "month"
            TimeInterval = DateDiff("m", Date, d)
        Case "year"
            TimeInterval = DateDiff("yyyy", Date, d)
    End Select
End Function
Sub Main()
    Dim m As Double, i As Double, dt As Date, t As String
    Dim fv As Double
    m = Val(InputBox("Write the amount"))
    i = Val(InputBox("Write the interest rate"))
    dt = DateValue(InputBox("Write the date"))
    t = InputBox("Write interval type")
    If (i > 0) And (m > 0) And (dt > Date) Then
        Select Case t
            Case "day"
                fv = m * (1 + i / 365) ^ TimeInterval(dt, "day")
            Case "month"
                fv = m * (1 + i / 12) ^ TimeInterval(dt, "month")
            Case "year"
                fv = m * (1 + i) ^ TimeInterval(dt, "year")
        End Select
        MsgBox ("Future Value (Time Interval: " & t & " ): " & fv)
    Else
        MsgBox ("Entered data are not valid")
    End If
End Sub
```

## EXERCISE 4

- Write a function **nextBirthDay** that takes your birthdate and then returns your **next** birthday. This function should compare the current date with your birthdate. If this year your's birthday is occurred then the function should return the next year's birthday otherwise it should return this year's birthday
- Create an Excel Worksheet and fill a range of cells with the name, surname, and birthdate of some people. Use the function **nextBirthDay** to display in the same worksheet their birthdays

## SOLUTION

```
Function nextBirthDay(d As Date) As Date
    Dim b As Date
    b = DateSerial(Year(Date), Month(d), Day(d))
    If b < Date Then b = DateSerial(Year(Date) + 1, Month(d), Day(d))
    nextBirthDay = b
End Function
```

	A	B	C	D	E
1					
2		<b>Name</b>	<b>BirthDate</b>	<b>NextBirthDay</b>	
3		Paul Miller	12/12/1969	December 12, 2010	
4		Ken Parker	02/01/1982	January 2, 2011	
5					
6					

## EXERCISE 1

Write the following procedures:

- A function **Sum** that given an array of real numbers returns a number that is the sum of the numbers available in the array
- A function **Average** that given an array of real numbers returns a number that is the average of the numbers available in the array
- A procedure **Main** which takes a set of real numbers from the range B1:C10 of the first worksheet, it stores them within an array, and then it displays the sum and the average of these numbers

## SOLUTION

```
Function Sum(num() As Double) As Double
    Dim i As Integer
    Dim t As Double
    t = 0
    For i = LBound(num) To UBound(num)
        t = t + num(i)
    Next i
    Sum = t
End Function
```

```
Function Average(num() As Double) As Double
    Dim i As Integer, n As Integer
    Dim t As Double
    t = 0
    n = 0
    For i = LBound(num) To UBound(num)
        t = t + num(i)
        n = n + 1
    Next i
    Average = t / n
End Function
```

## SOLUTION

```
Sub Main()  
    Dim SetOfNumbers(1 To 20) As Double  
    Dim r As Integer, c As Integer  
    Dim n As Range  
    Dim i As Integer  
    Set n = Worksheets(1).Range("B1:C10")  
    For c = 1 To 2  
        For r = 1 To 10  
            SetOfNumbers(i) = n.Cells(r, c).Value  
        Next r  
    Next c  
    MsgBox ("Sum -> " & Sum(SetOfNumbers) & _  
        Chr(13) & "Average -> " & Average(SetOfNumbers))  
End Sub
```

## EXERCISE 2

Write the following procedures:

- A function **AverageNeg** that given an array of real numbers returns a number that is the average only of the negative numbers available in the array
- A function **AveragePos** that given an array of real numbers returns a number that is the average only of the positive numbers available in the array
- A procedure **Main** which takes a set of real numbers from the range A1:A10 of the first worksheet, it stores them within an array, and then it displays the average of the negative numbers and the average of the positive numbers

## SOLUTION

```
Function AverageNeg(num() As Double) As Double
    Dim i As Integer, n As Integer, sum As Double
    sum = 0
    n = 0
    For i = LBound(num) To UBound(num)
        If num(i) < 0 Then
            sum = sum + num(i)
            n = n + 1
        End If
    Next i
    If n = 0 Then
        AverageNeg = 0
    Else
        AverageNeg = sum / n
    End If
End Function
```

## SOLUTION

```
Function AveragePos(num() As Double) As Double
    Dim i As Integer, n As Integer, sum As Double
    sum = 0
    n = 0
    For i = LBound(num) To UBound(num)
        If num(i) > 0 Then
            sum = sum + num(i)
            n = n + 1
        End If
    Next i
    If n = 0 Then
        AveragePos = 0
    Else
        AveragePos = sum / n
    End If
End Function

Sub Main()
    Dim SetOfNumbers(1 To 10) As Double
    Dim n As Range
    Dim i As Integer
    Set n = Worksheets(1).Range("A1:A10")
    For i = 1 To 10
        SetOfNumbers(i) = n.Cells(i, 1).Value
    Next i
    MsgBox ("Average negative numbers - > " & AverageNeg(SetOfNumbers) & _
        Chr(13) & "Average positive numbers -> " & AveragePos(SetOfNumbers))
End Sub
```

## EXERCISE 3

Write the following procedures:

- A function **Variance** that given an array of real numbers returns the variance of these numbers
- A procedure **Main** that firstly takes the real numbers from the range A1:A10 of the first worksheet, it stores them within an array, and then it displays the variance of these numbers. After that it takes the real numbers from the range D1:D10 of the first worksheet, it stores them within a new array, and then it displays the variance for these numbers

$$Var = \frac{\sum(x-\bar{x})^2}{n}$$

## SOLUTION

```
Function Average(num() As Double) As Double
    Dim i As Integer, n As Integer
    Dim sum As Double
    sum = 0
    n = 0
    For i = LBound(num) To UBound(num)
        sum = sum + num(i)
        n = n + 1
    Next i
    Average = sum / n
End Function
```

```
Function Variance(num() As Double) As Double
    Dim a As Double, i As Integer, sum As Integer, n As Integer
    a = Average(num)
    sum = 0
    n = 0
    For i = LBound(num) To UBound(num)
        sum = sum + (num(i) - a) ^ 2
        n = n + 1
    Next i
    Variance = sum / n
End Function
```

## SOLUTION

```
Sub Main()  
    Dim setNumOne(1 To 10) As Double  
    Dim setNumTwo(1 To 10) As Double  
    Dim n1 As Range  
    Dim n2 As Range  
    Dim i As Integer  
    Set n1 = Worksheets(1).Range("A1:A10")  
    Set n2 = Worksheets(2).Range("D1:D10")  
    For i = 1 To 5  
        setNumOne(i) = n1.Cells(i, 1).Value  
        setNumTwo(i) = n2.Cells(i, 1).Value  
    Next i  
    MsgBox ("Variance (A1:A10) -> " & Variance(setNumOne))  
    MsgBox ("Variance (D1:D10) -> " & Variance(setNumTwo))  
End Sub
```

## EXERCISE 4

Write the following procedures:

- A function **NewStringOne** that given an array of strings returns a string that consists of the concatenation by " - - " (space line line space) of the strings within the array, for example given the strings "free", "university" and "Bozen" the function returns "free - - university - - Bozen"
- A function **NewStringTwo** that similarly to the function **NewStringOne** concatenates the strings of a given array of strings but in the reverse order
- a function **AverageLength** that given an array of strings returns the average length of the strings within the array
- A procedure **Main** which asks the user for 5 strings, it stores them within an array, and then displays the new strings (**NewStringOne** and **NewStringTwo**) and the average length of these strings

## SOLUTION

```
Function NewStringOne(str() As String) As String
    Dim i As Integer, s As String
    s = ""
    For i = LBound(str) To (UBound(str) - 1)
        s = s & str(i) & " - - "
    Next i
    s = s & str(UBound(str))
    NewStringOne = s
End Function
```

```
Function NewStringTwo(str() As String) As String
    Dim i As Integer, s As String
    s = ""
    For i = UBound(str) To (LBound(str) + 1) Step -1
        s = s & str(i) & " - - "
    Next i
    s = s & str(LBound(str))
    NewStringTwo = s
End Function
```

## SOLUTION

```
Function AverageLength(str() As String) As Double
    Dim i As Integer, l As Integer, n As String
    l = 0
    n = 0
    For i = LBound(str) To UBound(str)
        l = l + Len(str(i))
        n = n + 1
    Next i
    AverageLength = l / n
End Function

Sub Main()
    Dim SetOfStrings(1 To 5) As String
    Dim i As Integer
    For i = 1 To 5
        SetOfStrings(i) = InputBox("Enter string " & i)
    Next i
    MsgBox ("New String One -> " & NewStringOne(SetOfStrings))
    MsgBox ("New String Two -> " & NewStringTwo(SetOfStrings))
    MsgBox ("Average Length -> " & AverageLength(SetOfStrings))
End Sub
```