VBA in Excel 3

Arrays in VBA

1. In VBE insert a new module **Module1**
2. Add the following code

```
Option Explicit

Function TestRange(rng As Range)
  Dim Arr()
  Dim R As Integer, C As Integer
  Dim i As Integer, j As Integer
  Dim OneDimension As Boolean, Vertical As Boolean

  R = rng.Rows.Count
  C = rng.Columns.Count

  OneDimension = False
  If R > 1 Then
      If C > 1 Then
          ReDim Arr(1 To R, 1 To C)
      Else
          ReDim Arr(1 To R)
          OneDimension = True
          Vertical = True
      End If
  Else
      ReDim Arr(1 To C)
      OneDimension = True
  End If

  If OneDimension Then
      If R > 1 Then
          For i = 1 To R
              Arr(i) = rng(i)
          Next i
      Else
          For i = 1 To C
              Arr(i) = rng(i)
          Next i
      End If
  Else
      For i = 1 To R
          For j = 1 To C
              Arr(i, j) = rng(i, j)
          Next j
      Next i
  End If

  If OneDimension Then
    If Vertical Then
      ' Returns a vertical vector As the input using TRANSPOSE
      TestRange = WorksheetFunction.Transpose(Arr)
    Else
      TestRange = Arr
    End If
  Else
     TestRange = Arr
  End If
End Function
```

3. Check your code by launching **Compile VBAProject** from the **Debug** menu
4. Activate the Excel sheet and add a few values as in the figure below:

	A	B	C	D
1	1	2	3	
2	4	5	6	
3	7	8	9	
4	10	11	12	
5				
6				

5. Insert the function **TestRange** in cell **D1** with the argument **A1:C4**
6. Transform the range **D1:F4** in an array formula as shown in Lab 3.
7. Add a breakpoint at the beginning of the code and execute the function for a horizontal one line range, for a vertical one line range and for a rectangular range and follow the execution of the code. Check the values of the variables **OneDimension** and **Vertical**.

Functions that return a single value.

Program to evaluate the sum of an array

1. Add the following code to **Module 1**

```
Function SumArray(rng As Range) As Double
    Dim Arr()
    Dim Sum As Double
    Dim R As Integer, C As Integer
    Dim i As Integer, j As Integer

    R = rng.Rows.Count
    C = rng.Columns.Count

    Arr = rng

    Sum = 0
    For i = 1 To R
        For j = 1 To C
            Sum = Sum + Arr(i, j)
        Next j
    Next i
    SumArray = Sum
End Function
```

2. Activate the Excel sheet and insert the function `SumArray` into an empty cell with the arguments **A1:C4**:
3. Check the returned value against the library function **SUM**

REMARK: You'll notice that reading the range into an array is easier in VB by using a variant variable. There's no need to loop over the number of rows and columns to read the individual values of the range and assign them to the VB array. Instead through a simple assignment operation we can read the whole range into an array on the line:

```
    Arr = rng
```

Let’s change our function to evaluate the sum of elements above the main diagonal of the array

1. Change the loop about j as follows:

```
        For j = i To C
```

REMARK: By changing the inner loop, when the variable that counts the rows (i) is 1, the inner loop will go from 1 to the number of columns (C). When i is 2, the inner loop will go from 2 to C, when the current row i is 3, j will go from 3 to C and so on. Basically with this small change. We sum up only the values above the main diagonal of the array.

REMARK: You’ll notice that the function does not check if the array is square (having the same number of rows and columns)

Let’s change our function to check if the range is square. If not, a message should be returned. We will need to change the data type of the function first from Double to Variant.

2. Change the definition of our function as follows:

```
Function SumArray(rng As Range) As Variant
```

3. Add the following code to our function after reading the rows and columns:

```
    If R <> C Then
        SumArray = "Not square"
        Exit Function
    End If
```

4. Re-execute the function **SumArray** in our sheet by pressing **Ctr+Alt+F9**
5. Change the function arguments of the function from **A1:C4** to **A1:C3**

HOMEWORK

Use the **`SumArray`** function to write a VBA function that finds out the maximum element over the main diagonal of a square matrix. Name this function **SumMain**

Functions that return an array.

Function that scales a square array

1. Add the folowing code

```
Function ScaleArr(rng As Range, ByVal sc As Double) As Variant
    Dim Arr()
    Dim R As Integer, C As Integer
    Dim i As Integer, j As Integer

    R = rng.Rows.Count
    C = rng.Columns.Count

    If R <> C Then
        ScaleArr = "Not square"
        Exit Function
    End If

    Arr = rng

    For i = 1 To R
        For j = 1 To C
            Arr(i, j) = Arr(i, j) * sc
        Next j
    Next i
    ScaleArr = Arr
End Function
```

2. Check your code by launching **Compile VBAProject** from the **Debug** menu
3. Activate the Excel sheet and insert the function **ScaleArr** into an empty cell with the arguments **A1:C4** and **2**:
4. Change the arguments of the function to **A1:C3** and **2**
5. Select a range 3 by 3 starting from the cell in which you've inserted the **ScaleArr** function and turn it into an array formula
6. Change the value of the scale factor in the inserted function

Function that multiplies two arrays

Multiplication of two matrices is defined if and only the number of columns of the left matrix is the same as the number of rows of the right matrix. If **A** is an m-by-n matrix and **B** is an n-by-p matrix, then their matrix product **AB** is the m-by-p matrix whose entries are given by dot product of the corresponding row of **A** and the corresponding column of **B**:

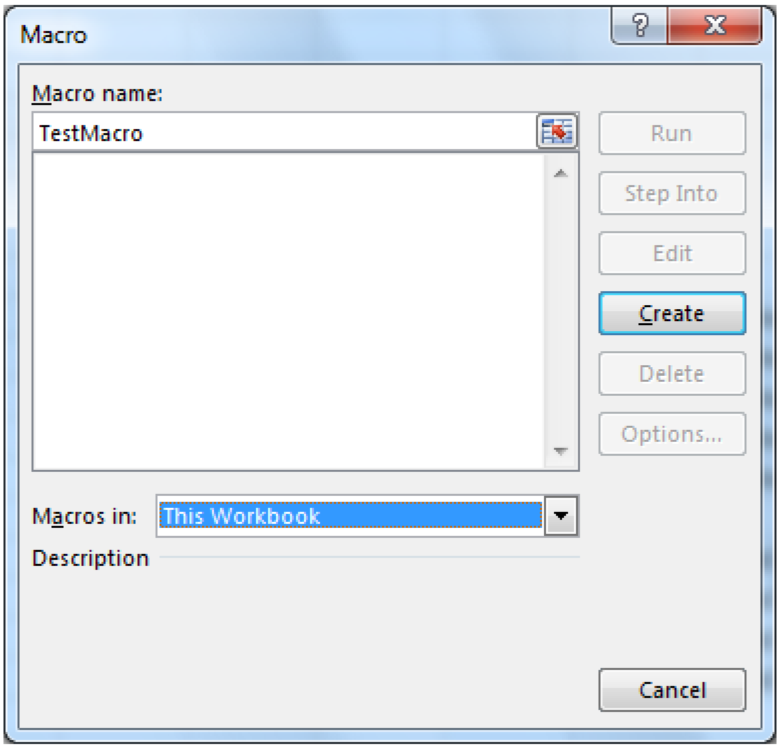
$$c_{i,j} = a_{i,1}b_{1,j} + a_{i,2}b_{2,j} + \cdots + a_{i,n}b_{n,j} = \sum_{k=1}^{n} a_{i,k}b_{k,j}$$

1. Add the following code

```
Function MatMult(rng1 As Range, rng2 As Range) As Variant
    Dim A(), B(), C() As Double
    Dim m As Integer, n As Integer, p As Integer
    Dim i As Integer, j As Integer, k As Integer

    m = rng1.Rows.Count
    n = rng1.Columns.Count
    p = rng2.Columns.Count

    If rng1.Columns.Count <> rng2.Rows.Count Then
        MatMult = "Invalid"
        Exit Function
    End If

    ReDim C(1 To m, 1 To p)

    A = rng1
    B = rng2

    For i = 1 To m
        For j = 1 To p
                C(i, j) = 0
                For k = 1 To n
                    C(i, j) = C(i, j) + A(i, k) * B(k, j)
                Next k
        Next j
    Next i
    MatMult = C
End Function
```

Macros in Excel

1. Activate the **DEVELOPER** tab and click **Macros**
2. In the opened dialog box type the name **TestMacro**, select **This Workbook** from the dropdown list at the bottom and click **Create**


Macro name: TestMacro

Macros in: This Workbook

Description

3. In the module window, add the code to the **TestMacro** subroutine

```
Sub TestMacro()
    Dim wks As Worksheet
    Dim Texts(1 To 3, 1 To 1) As String
    Dim Values(1 To 3, 1 To 1) As Integer

    Texts(1, 1) = "Water"
    Texts(2, 1) = "Heating"
    Texts(3, 1) = "Electricity"
    Values(1, 1) = 125
    Values(2, 1) = 350
    Values(3, 1) = 75

    ' Adds a new sheet in the current workbook
    Set wks = Application.ActiveWorkbook.Sheets.Add
    ' Inserts the text in the range A1:A3
    wks.Range("A1:A3").Value = Texts
    ' Inserts the values in the range B1:B3
    wks.Range("B1:B3").Value = Values
    ' Adds the text TOTAL in cell A4
    wks.Cells(4, 1).Value = "TOTAL"
    wks.Cells(4, 1).Font.Bold = True
    ' Adds the formula SUM in cell B4
    wks.Range("B4").Formula = "=SUM(B1:B3)"
```

```
End Sub
```

4. Activate the Excel sheet and launch the macro by clicking the command **Macros** in the **DEVELOPER** tab
5. Add a breakpoint in the beginning of the macro and execute step by step the macro using the key combination **SHIFT+F8**