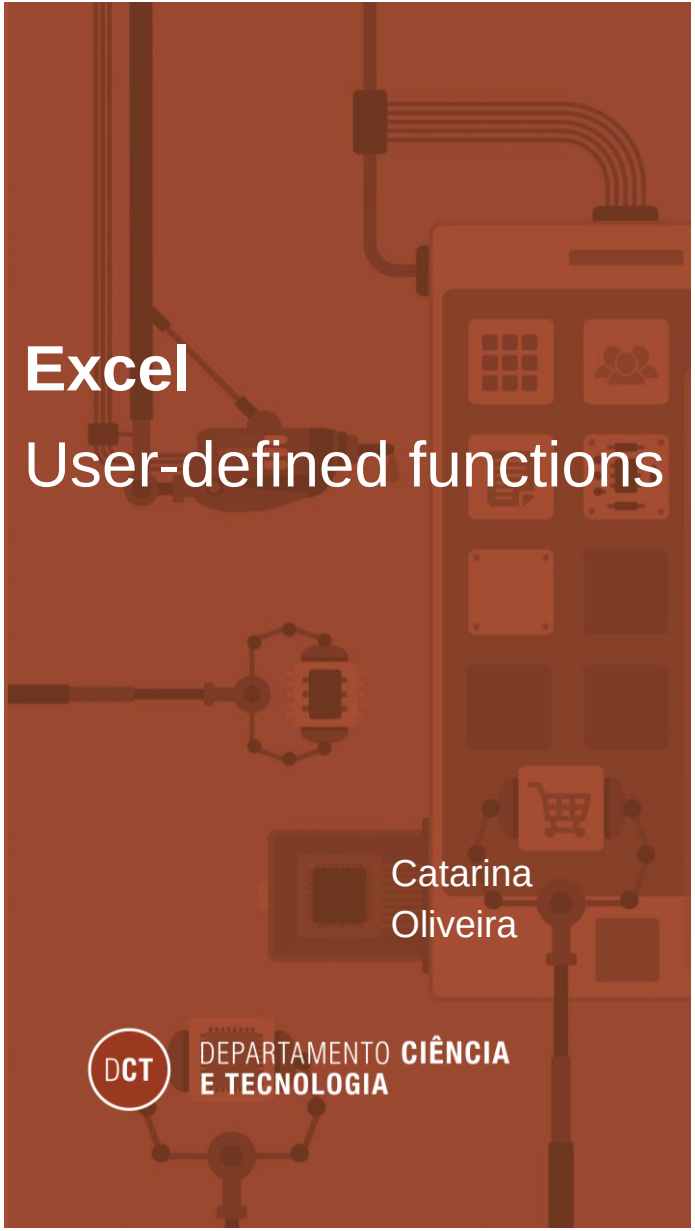

UNIVERSIDADE PORTUCALENSE

CONTENT

1. User-defined functions
2. Visual Basic editor
3. Visual Basic editor's most common features
4. VBA modules
5. VBA language
    1. Function structure
    2. Variables and types of data
    3. VBA operators and functions
    4. Control structures
        1. Conditional structures
        2. Loop structures

User-defined functions

Creating User Defined Functions (FDU):

- Allows you to reduce the complexity of a spreadsheet
    - Complex calculations can be integrated into a function
        - Which is then used more simply (like standard Excel functions)
- Requires some knowledge of the language VBA (*Visual Basic for Applications*)
- Cannot be recorded as command macros.

Function: program or algorithm built with the objective of producing a result, executing a set of instructions and using a set of arguments provided when using it.

Example - function named DOLLARS, whose purpose is to calculate the value in Dollars (DOLLARS) from the value in Euros (EUROS) and the exchange rate (TxEURUSD)

```
Function DOLARES(EUROS As Single, TxEURUSD As Single)
      DOLARES = EUROS * TxEURUSD
End Function
```

Call the function:

- =DOLARES(1000;1,423)
- =DOLARES(B5;1,487)

Visual Basic editor

Open the editor:

- ALT + F11
- Tab “Developer” > Group “Code” > Command “Visual Basic”

Visual Basic editor's most common features

Standard toolbar: contains the most used editor commands, from which they stand out:

- **Insert Options:** decide the type of object to insert/create in VBA, namely modules, class modules, forms or procedures
- **Run Macro (Run):** Run and test a command or function macro
- **Suspend Macro Execution (Break):** Momentarily suspend/stop the execution of a macro
- **End macro execution (Reset):** definitively end the execution of a macro

- **Project Window:** essential for navigating between the different objects of VBA projects, namely to locate and open a certain module; this window lists all the workbooks open in Excel and, for each of them, the objects they contain: sheets, modules (modules) and forms (forms), among others

- **Properties Window:** displays a list of properties and respective values of the selected project object

- **Module windows:** editor location where the VBA code for command and function macros is introduced and edited. For each of the newly created or opened modules, the editor provides its own window

VBA modules

The Visual Basic Editor allows the creation of two types of modules :

- “Standard” modules (module) ← simpler and most frequently used
- Class modules

How to (create a standard module):

1. In the "Project Window" select the book where you want to create/insert a new module
2. In the "Standard Toolbar", in the "Insert Options" select the "Module" option

A new "Module Window" should appear in the editor with a predefined module name (eg Module1, Module2, …) and in the "Project Window", "Hanging" in the chosen book, this module should also appear.

To change the default name: select the module, change the value of the “Name” field in “Properties Window”

In a single module, several functions or command macros can be inserted.

For simplicity, it is advisable to distribute the various functions/macros into different modules

VBA language: structure of a function

```
Function FUNCTIONNAME(argument1, argument2, …)
      [Instructions]
       FUNCTIONNAME = [Expression or variable]
      [Instructions]
End Function
```

Function name:

- It must be original (you cannot repeat the names of Excel functions or other created functions
- It must clearly identify the purpose of the function.

Argumento

- Must be defined between two mandatory parentheses
- Number and type of arguments depend on the characteristics of the function to be created

Instructions

- Depend on the purpose and calculations that will be performed
- They are composed of variables, expressions, values, operators and control structures.
- Number of instructions depends on the characteristics of the problem
- **Mandatory instruction**: `FUNCTIONNAME = [Expression or variable]`
    - Defines the return value

VBA: Variables and data types

Variable: memory location to which a name is assigned and where the values used in processing functions (and macros in general) are temporarily stored

Each variable has a unique name and a data type that can be defined.

Type	*Size in Bytes*	*Description*
`Boolean`	2	Stores a value of `True` (0) or `False` (–1).
`Byte`	1	Contains a number in the range of 0–255.
`Currency`	8	Defines monetary values in the range of –922,337,203,685,477.5808 to 922,337,203,685,477.5807.
`Date`	8	Defines combination value that includes both date and time; although you certainly don't have to include both. Valid dates range from January 1, 100 to December 31, 9999.
`Decimal`	14	Defines a precise 28-digit number. You must use the `Variant` data type for this value. The number is in the range of ±79,228,162,514,264,337,593,543,950,335 with no decimal point and ±7.9228162514264337593543950335 with 28 places to the right of the decimal.
`Double`	8	Contains a real number in the range of –1.79769313486231 E308 to – 4.94065645841247 E-324 for negative values and 4.94065645841247 E-324 to 1.79769313486232 E308 for positive values.
`Integer`	2	Contains a number in the range of –32,768 to 32,767.
`Long`	4	Contains a number in the range of –2,147,483,648 to 2,147,483,647.
`Object`	4	Any object reference.
`Single`	4	Contains a real number in the range of –3.402823 E38 to –1.401298 E-45 for negative values and 1.401298 E-45 to 3.402823 E38 for positive values.
`String` (fixed length)	String length	Contains a set of characters from 1 to approximately 65,400 characters long.
`String` (variable length)	10 + string length	Contains a set of characters from 0 to approximately 2 billion characters long.
`Variant` (with characters)	22 + string length	Stores any valid non-numeric data type or data types larger than a `Double`.
`Variant` (with numbers)	16	Stores any valid numeric data type up to the size of a `Double`.

VBA: Variables and data types

In functions, variables can be used in different contexts:

- To represent and store argument values
- To save intermediate calculation values
- To save the result of the function itself

Declaration of variables

- Allows you to associate them with a name and a data type (contextualizes and conditions the use)
- In functions, the way they are declared depends on the usage type. :
    - Arguments: `VariableName As DataType`
    - Remaining variables: `Dim VariableName As DataType`

```
Function EXAMPLE(NUM1 As Integer, NUM2 As Integer)
      Dim DIVISION As Single
      DIVISION = NUM1 / NUM2
       EXAMPLE = DIVISION * 100
End Function
```

Function **EXAMPLE** (return the quotient multiplied by 100)

- Variables **NUM1** and **NUM2**: two integer arguments
- Variable **DIVISION**: auxiliary variable used to save the partial result
- Variable **EXAMPLE**: variable with the function name, that allows returning the result

VBA language: Operators

Operators	Type
`()`	parentheses
`^`	exponentiation
`+ -`	unary plus and minus
`* /`	multiplicative
`\`	integer division
`Mod`	modulus
`+ -`	additive
`&`	concatenation
`< <= > >= = <>`	relational and equality
`Not`	logical NOT
`And AndAlso`	logical AND
`Or OrElse`	logical inclusive OR
`Xor`	logical exclusive OR

VBA language: control structures

In programming, in order to control the flow of execution of instructions in an algorithm, three types of structures are usually used:

- **Sequential structures:** most basic type of structure. It translates into the sequential execution of a set of instructions, considering only their order. Example: the instructions of a recorded macro

- **Conditional structures (decision or selection):** statements are grouped and executed depending on the result of one or more conditions

- **Loop structures:** allow you to repeat the execution of a set of instructions while certain conditions are met

Conditional structures

```
If condition Then
     [instructions if condition is true]
Else
     [instructions if condition is false]
End If
```

```
If x > 0 Then
     result = “Positive”
Else
      result = “Negative or zero”
End If
```

```
Select Case expression
Case value1
     [instructions]
Case value2
     [instructions]
Case else
     [instructions]
End Select
```

```
Select Case day
Case 1
     result = “Sunday”
Case 2
     result = “Monday”
...
Case else
     result = “Invalid”
End Select
```

Loop structures

```
For counter = initialVal To finalVal
      [instructions]
Next [counter]
```

```
For cont = 1 To 3
      result = result + cont
Next cont
```

```
Do While condition
      [instructions]
Loop
```

```
Dim cont As Integer
Cont = 1
Do While cont <= 3
      result = result + cont
      cont = cont + 1
Loop
```

```
Do
      [instructions]
Loop While condition
```

```
Dim cont As Integer
Cont = 1
Do
      result = result + cont
      cont = cont + 1
Loop While cont <= 3
```

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.