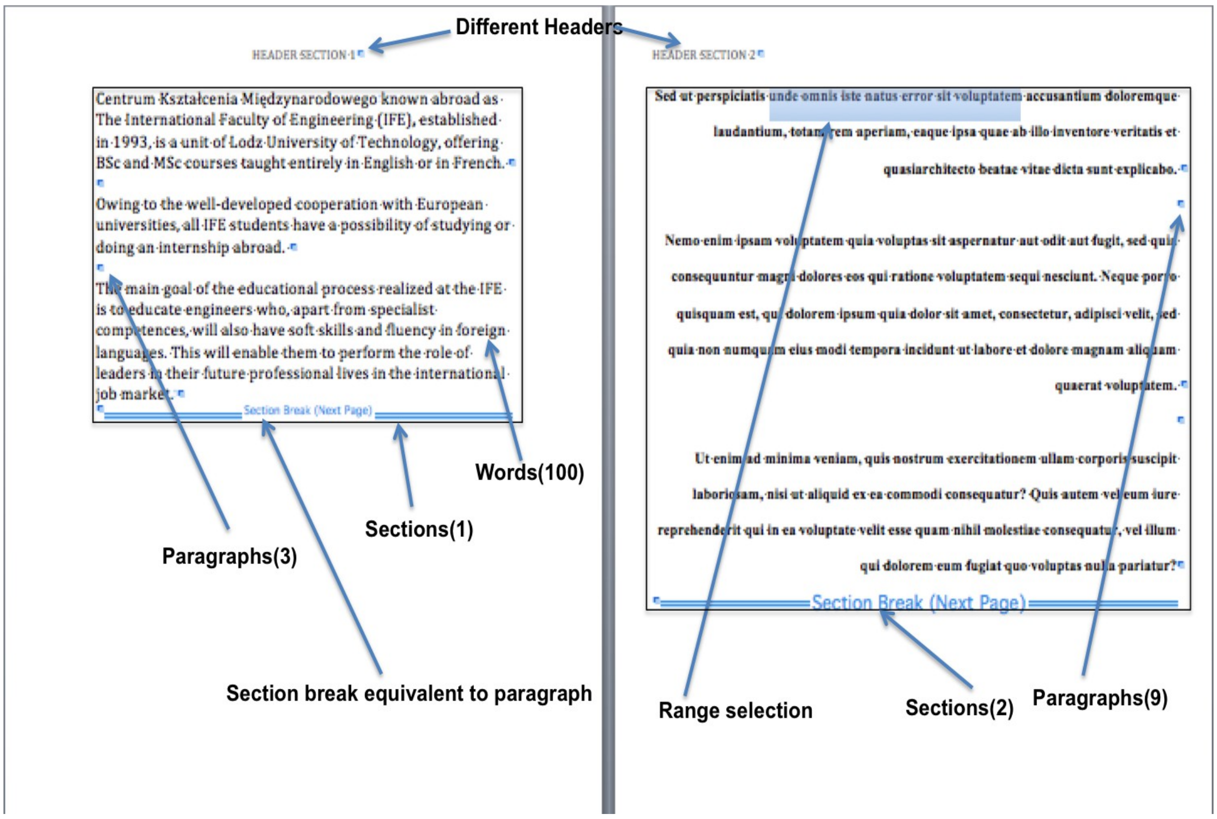
The aim of this lab is to get better acquainted with the VBA language syntax. Loops, conditions, variable declaration and so on will be presented and complete programming notions already presented in the previous lab. Also, MS Word is used as the primary application platform for this lab to complete your knowledge about VBA. A brief introduction to allow communication between Word and Excel using VBA is also presented. The MSDN library is good information website to get deeper in VBA language:
https://msdn.microsoft.com/enus/library/office/ee814735(v=office.14).aspx

INTRODUCTION: VBA for Word

You have been acquainted with VBA in Excel environment, but of course VBA can be used for Word, PowerPoint or even Access. The main thing that change between the applications in the object structure and its hierarchy. Objects in Word may have common attributes and methods as the one in Excel, but also their own. The standard (basic?) object hierarchy is the following:
Application (Word) $\rightarrow$ **Paragraphs** $\rightarrow$ **Range** and a possible example could be:
`ActiveDocument.Paragraph(4).Range.Words(4).Characters(1).` This means I consider the 1st letter of the 4th word of the paragraph 4 in the current Word document (here **Words** and **Characters** are actually properties of the the **Range** object). But the following is also possible: **`ActiveDocument.Range.Words(4)`**,which means accessing the 4th word of the whole document. Another important object which does not appear in the hierarchy is the **Sections** object, as it is usually associated with page break (we can break a document in different sections) and usually linked with page setup (e.g. page margins, footer/header distance) or access to all words (and characters) in the Section (e.g. **`ActiveDocument.Section(2).Range.Words(2000)).`**
Below is an illustration of different objects of a MS Word document (boxes just mark Sections).



Centrum Kształcenia Międzynarodowego known abroad as The International Faculty of Engineering (IFE), established in 1993, is a unit of Lodz University of Technology, offering BSc and MSc courses taught entirely in English or in French.

Owing to the well-developed cooperation with European universities, all IFE students have a possibility of studying or doing an internship abroad.

The main goal of the educational process realized at the IFE is to educate engineers who, apart from specialist competences, will also have soft skills and fluency in foreign languages. This will enable them to perform the role of leaders in their future professional lives in the international job market.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasiarchitecto beatae vitae dicta sunt explicabo.

Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem.

Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Another important object is also the **Selection** object, which corresponds to the cursor position or any text portion selected in the document. Indeed, unlike Excel that is set around a grid-like domain, it is a bit harder to navigate between letters or words in the document and therefore it can be not trivial to know the current position of the cursor. However, because we can access to any character of words, then the call of the **Select** method at the level of a **Paragraphs**, **Words** or even **Characters** is possible.

Exercise 1: prepare macros

Prepare your Word document so as it looks like the one above:

- Copy/Paste first text fragment (frg1.txt)
- Insert a Section break ("next page" choice)
- Copy/Paste second text fragment (frg2.txt) in the next section (so after the page break)
- Insert a new Section break
- Create different headers for both sections
- Save you file as VBA2_*yourname*.docm (use this extension since Macros will be created)
- Open Visual Basic and copy/paste the code below

```
Sub selecttext()
dim rng As Range
Set rng =  ActiveDocument.Range(Start:=ActiveDocument.Paragraphs(8).Range
.Words(4).Start, End:=ActiveDocument.Paragraphs(8).Range.Words(10).End)
rng.Select
end sub
```

Add also another procedure called `main` that you will leave empty for the moment.

Warning: for some reasons, it may be possible that the paragraph #8 does not contain words. So play later around with the value (try 6, 7 or 9) and see if words are selected after executing the macro.

Task1.1: try to modify the previous procedure to include input variables (decide which one you believe could be parsed). Remember that you need to call the macro from another procedure where input values are set (for instance in `main`).

VBA VARIABLES

So far you have been introduced with variables (standards such as string, byte/integers and also object variables). In what follows, we treat the case of arrays declaration and also variable scope.

Exercise 2: array

In the previous exercise you have selected 7 words from the text. A question may arise: where can we keep them? In which type of data? The answer: an array. The way you declare array in VBA is very simple: you need to define its size. Example: `Dim myarray(1 To 10) As Byte`. This is the default declaration. Of course, your array can be N-dimensional. If you want a 2D array, easy: `Dim myarray(1 To 10,1 To 4) As String.` Also very easy to assign value to an array cell or use the array value to be assigned to another variable, e.g. :

```
myarray(3)=Selection.Words(3).Characters.Count-1 'why -1 by the way?
Msgbox(myarray(6))
```

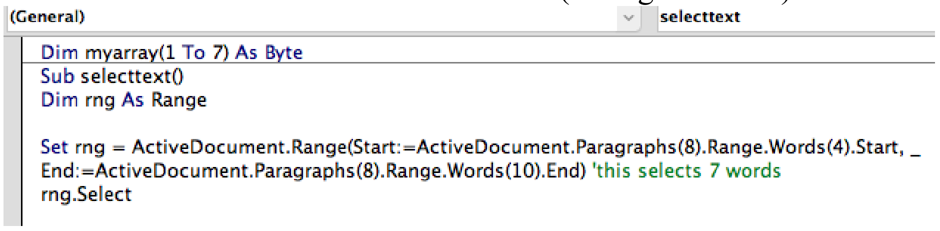
Task 2.1: In **selecttext** macro, declare one array of integer of size 7 (index: 1 to 7) and use it to store the number of letters of all words in the Selection. Display the results in message box. What you have above may be useful!

Now, what you have above is called a static array, that means you declare the size of the array from the start. What happen if you don't know the size of the array (for instance you don't know how many words the user wants to select) . The solution: write `Dim myarray() As Byte`. So keep content of brackets empty. However, at some point in the sub body, before the array is used for the first time, you will need to initialize its size using the **Redim** statement, e.g.:
`Redim my array(1 To 100)` or more generally, `Redim my array(1 To nbcell)`, where `nbcell` is a variable with a known value.

Task 2.2: replace the static array with a dynamic array in your previous task. Link the size of the array with the number of words that may have been passed as an input variable of the procedure.

Exercise 3: variable scope

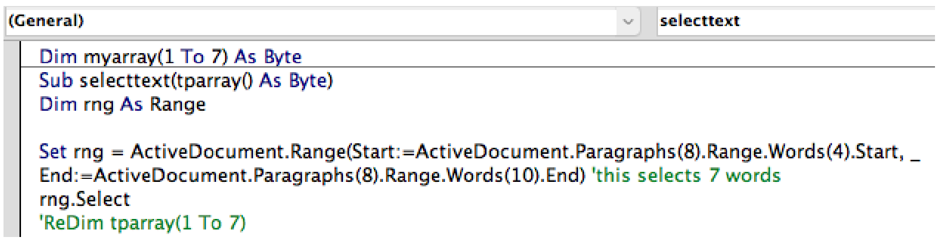
An important aspect of programming is the scope of a variable, that is where it can be used. Indeed, depending where it will be declared, a variable might only exists in a procedure, or shared by many procedures or even been accessible anywhere in the program. In VB/VBA, there are three scoping levels: **procedure level**, **module level** and **public level**. This is important to know from the start if a variable will be parsed to different procedures, can be modified in different modules or just used locally, because this impacts the place where it will be declared in the code.
For example, let take again the array from the previous exercise. What it its scoping level? **Procedure level** of course, because it is declared in the body of the procedure **selecttext** (by the way, if it was declared in a function, we would also say that it is a procedure-level variable).
How to make it become a **module level** variable? Quite simple: you declare it at the top of the module space, before any procedure/macro is defined. You see that a separation line will appear, marking a new area of module-level variable declaration (see figure below).



```
Dim myarray(1 To 7) As Byte
Sub selecttext()
Dim rng As Range

Set rng = ActiveDocument.Range(Start:=ActiveDocument.Paragraphs(8).Range.Words(4).Start, _
End:=ActiveDocument.Paragraphs(8).Range.Words(10).End) 'this selects 7 words
rng.Select
```

Now, in order for **selecttext** to recognize **myarray** as a module level variable and not any more as a procedure-level variable, an empty array of same datatype as to be declared as input <u>but</u> don't need to be redimensioned in the procedure body (ask the teacher why), as below:


(General) selecttext

```
Dim myarray(1 To 7) As Byte
Sub selecttext(tparray() As Byte)
Dim rng As Range

Set rng = ActiveDocument.Range(Start:=ActiveDocument.Paragraphs(8).Range.Words(4).Start, _
End:=ActiveDocument.Paragraphs(8).Range.Words(10).End) 'this selects 7 words
rng.Select
'ReDim tparray(1 To 7)
```

Task 3.1: Do the module-level declaration of the array, assign to the first cell any value of your choice. In **selectext** body, declare an array of same size and datatype as **myarray** in which the length of the 1st selected word will be assigned to its 1st cell and display the result in a message box. Then, after the call of **selectext** in **main** procedure, add another message box to display the value of **myarray(1)**. Are the displayed messages showing the same value?

Task 3.2: now, modify the previous code so as the previous procedure-level array is set as an input for **selecttext** macro (see above hints). What can you say about the values displayed by the message boxes?

We will continue visiting the characteristics of variable scope in the next lab, especially module-level and public level.

CONTROL FLOW IN VB

This means all language structures that fulfill certain conditions to take decision on what is executed by the program, i.e. Decision structures (IF...THEN...ELSE, SELECT...CASE) or Loop structures (FOR...NEXT, WHILE...WEND, DO...LOOP). Of course, there exist in other programming languages, but declared in another way. The difficulty here is to remember how the declaration is done.

Exercise 4: Decision structure

The first decision structure we look at is the **If...Then...Else...End If** one. Better than words, an example:

```
If rng.Bold = True Then
rng.Bold = False 'bold -> normal
Else rng.Italic = True  'normal -> italic
End If
```

Note that the **Else** can be replace by **ElseIf** if another condition has to be set (embedded If).

Task 4.1: include the code above in a macro to test if you can change the font of the 1st word of the 1st paragraph. Then do another macro to test if you can change the text of the **Header** of the Section 2 so as it is the same as the one of Section 1 (**LinkToPrevious** property set to **True**). The **strcomp** function can be helpful to check the value of the two headers (`strcomp(text1,text2)=0` if equal, `<>0` if different).

Instead of If, a set of choices can be declared and check by the program using the **Select...Case...End Select** decision structure. For instance, let assume you want to format a selection by changing font or style (bold, italic or underline). This can be programmed as follows:

```
Select Case choice 'a variable set as macro input
Case "size"
tpselection.Font.Size = 20
Case "bold"
tpselection.Font.Bold = True
Case "italic"
tpselection.Font.Italic = True
Case "underline"
tpselection.Font.Underline = True
End Select
```

Optionally, the statement can also take an **Else** case that considers any expression that does not match the defined clauses of the **Case** statements.

Task 4.2: create a new macro that control the font of a selected text (i.e. The one selected in the **selecttext** macro). This macro needs 2 inputs: the selected text (type **Selection**), the font choice (type **String**) and an optional input for the font size (simply write **Optional** before the input declaration). Call the procedure from **selecttext** and observe results for different test cases.

Note: you can see that input variables are not all mandatory and depend on the case to be treated.

Exercise 5: Loop structure

In VB, **For** loop has the following basic exemplar structure:

```
For i = 1 To 7
 tparray(i) = selection.Words(i).Characters.Count - 1
Next i
```

Optionally, iterations can be iterated with step different than 1 and also in reverse order. In that case, add step statement in header (e.g. `For j= 10 to 1 step -2`...)

Task 5.1: modify **selecttext** so as to copy the number of characters of the selected words in an array and after display their sum in a message box.

A useful alternative in the case of arrays is to use the **For Each...Of...Next** structure. Note that it has already been introduced in the previous lab (exercise 7).

Task 5.2: create a new macro that will **Copy/Paste** each word of a selected text range (again here set as procedure input) after the paragraph selection, with the condition that a paragraph is inserted between pasted words. Use the following statements to locate place for performing the **Paste** method:

```
selection.HomeKey  unit:=wdStory 'put cursor at beginning of document
selection.MoveDown  unit:=wdParagraph,  Count:=nbpara 'paragraph number
 'where text is selected
selection.HomeKey 'place cursor at beginning of paragraphs(nbpara+1)
```

Also, use `selection.InsertParagraph` and `selection.MoveDown unit:=wdParagraph` to insert paragraphs between pasted words.

The other popular Loop structure is the **While...Wend**. The Loop stops when a *condition* is satisfied, e.g. a certain number of words is reached. The while loop can be stopped earlier using Exit While if a particular *condition* set in the **While** body is satisfied. For instance:

```
While StrComp(word2find, obj.Words(i)) <> 0
i = i + 1
Wend
```

Usually a buffer value is used in the **While** body to control the flow (and so overcome endless loop), with the primary condition that it is initialized before the **While** structure (in the example above, it is obvious: the buffer is the variable *i*.

Task 5.3: create a new procedure that will count the position in an active document of the first instance of a word that is parsed as a procedure input. The value of the position is returned in a message box. It is advised to set **Range** variable directly linked to the **ActiveDocument** (so take the whole range of words in the document, regardless of 6paragraphs). An example of possible input is "foreign" and should give position 100, as shown on the 1st figure.

VBA built-in functions

You have already been introduced with some of them: **INPUTBOX**, **MSGBOX**, **STRCOMP**, **CONCATENATE** of even **MID**. There are many built-in functions but see below the ones I see as important in the case or VBA for Word.

TRIM(*string*): trim basically remove the leading and trailing spaces of a text value. If you have noticed, a **word** in Word is a set of characters followed by one space. To remove the space, use **trim**.

CHR(*ascii_value*): returns the character based on the ASCII value. One useful example is **Chr**(10) to add new line in displayed message. Use **&** to concatenate text fragments, e.g. `Chr("Hello students" & chr(10) & "how are you today after VBA classes?" & chr(10) & "Lost in translation?")`

ASC(*string*): returns the ASCII value of a character or the first character in a string.

STR(*number*): returns a string representation of a number.

VAL(*string*): accepts a string as input and returns the numbers found in that string. If string starts with number, return natural part of it. `Str("12.45")`→12,45 (if , is set as digit separator), `Str("12,45")` → 12, `Str("hello")`→ 0

Homework (apart finishing tasks):

1. Create a macro (or set of macros) that give(s) a summary result similar to the **Word Count** tool (accessible in Tools→Word Count...). Use message box to visualize summary result.
2. Create a macro (or set of macros) that perform(s) the Find/Replace action. Use inputboxes to asks for words to be found and replaced.