Visual Basic for Applications

Control statements

designing of algorithms

conditional statement

collection processing, loop statement

Materials
http://staff.uz.zgora.pl/ipajak
http://staff.uz.zgora.pl/gpajak

Summary of the Lecture02

- Procedures (procedural macros), header

```
[Private|Public] Sub <name>([arg1, arg2, … , argN])
```

- Functions (functional macros), header

```
[Private|Public] Function <name>([arg1,… , argN]) As type
```

- User program stored data in variables, declaration scheme

```
Dim <name> As type
```

- To modify any element of program (e.g. variable) assigning statement „=” is used

```
[Set] element = value
```

- Arithmetic operators

```
+, -, *, /, \, ^, mod, ( )
```

- Error handling

```
On Error GoTo <label>

On Error Resume Next
```

Designing of the algorithms

Algorithm – a set of steps leading to the solution of a specific task; a set of commands specifying the method of processing a dataset with the order of their execution.

The implementation of each program should be followed by a design process. Its important step is planning the method of data processing. The result (designed algorithm) should be written in a form independent of the target programming tool, that allows to formulate a method for solving a problem.

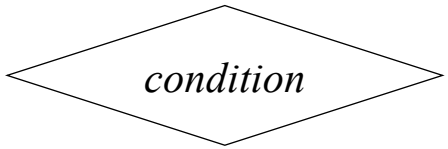
Basic symbols

beginning of the algorithm

end of the algorithm

operation or process

control flow



conditional block

Example

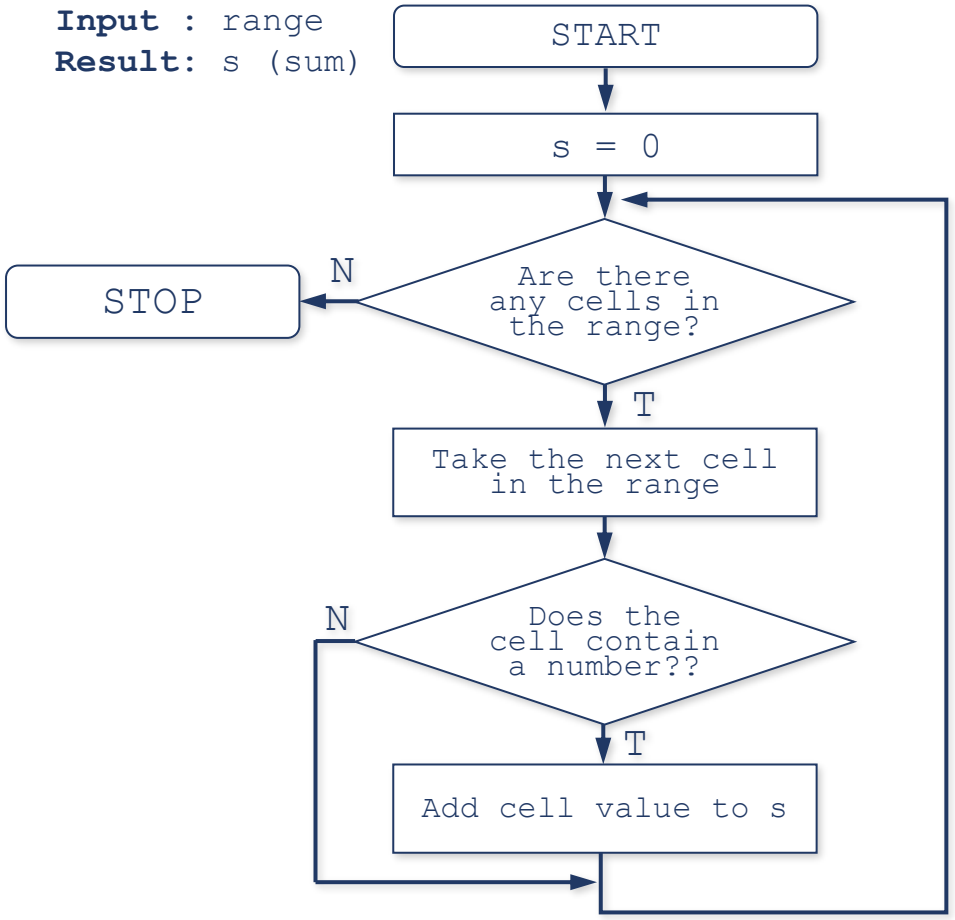
Task: design an algorithm that calculates the sum of numeric values in a given range of the worksheet.

	A	B	C
1	1		1
2	2		2
3	a		a
4	01.03.2022		01.03.2022
5	b		b
6	3		3
7	4		4
8	44631		10

=SUM(A1:A7)

=SumOfNumbers(C1:C7)

Note: standard function SUM interprets dates as numeric values.



Control statements

Control statement – element of programming language determining the order in which the statements included in program code are executed.

Control statements in VBA

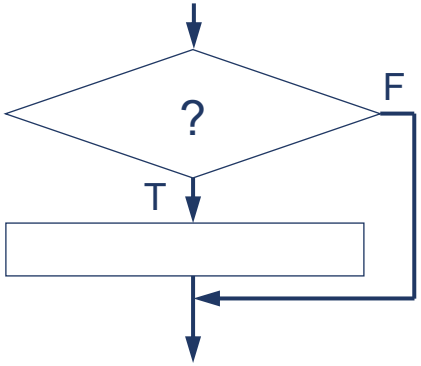
- **Conditional statement** (**`If-Then-Else`**) – introduces a branching in program code creating two alternative sequences of statements.
- **Selection statement** (**`Select Case`**) – introduces a branching in program code creating any number of alternative sequences of statements.
- **Loop statements** (**`For-Each`**, **`For-Next`**, **`Do-Loop`**) – repeat a set of statements a certain number of times.

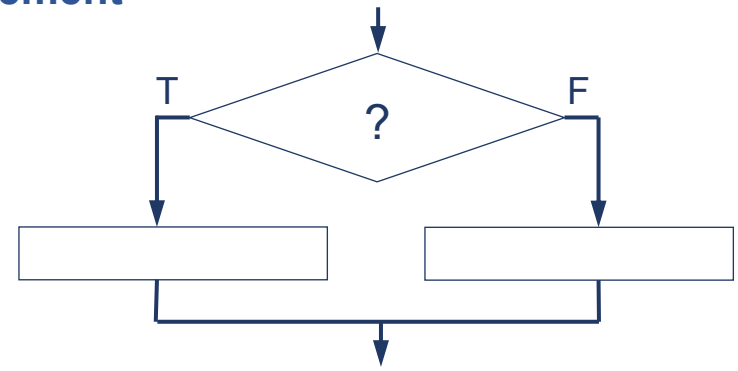
Additionally

- **Functions for selecting a value from a list** (**`Choose`**, **`Switch`**) – choose a value from predefined list, in a specific case, they allow for a substitution of series of conditional statements or selection statement.

Control statements in algorithms

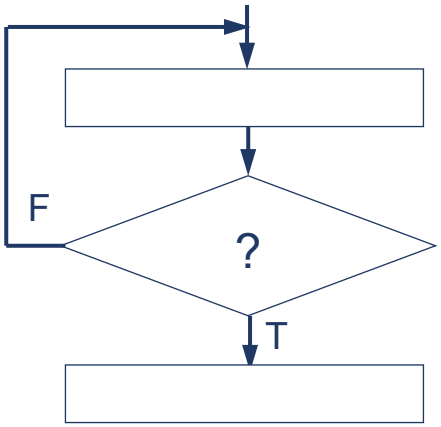
Conditional statement

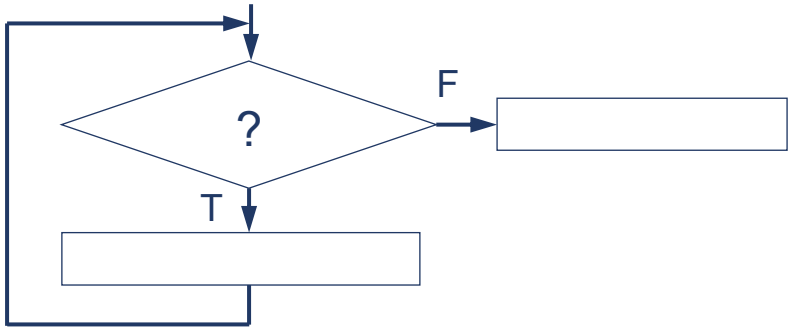




In the case of conditional statements, all flows are forward.

Loop statements



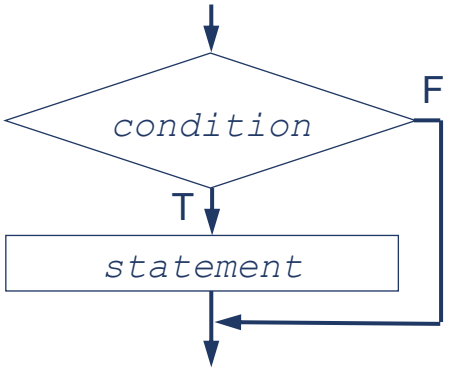


In the case of loop statements, there is always backward flow

Conditional statement

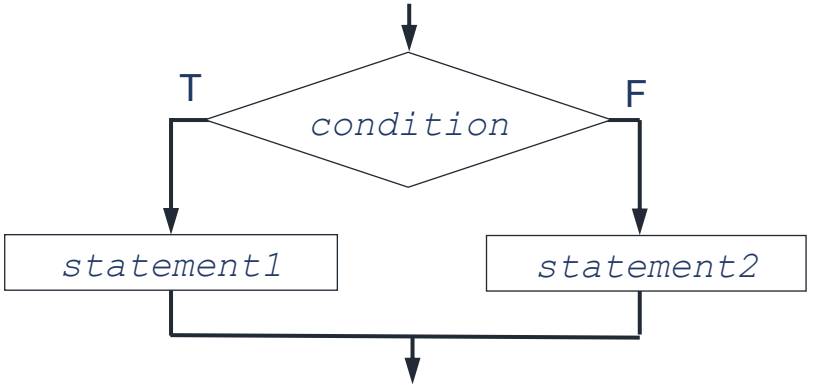
If *condition* is true execute *statement* (group of statements).

```
If condition Then statement

If condition Then
  statement1
  …
  statementN
End If
```



If *condition* is true execute *statement1* (first group of statements) otherwise execute *statement2* (second group of statements).

```
If condition Then statement1 Else statement2

If condition Then
  statement1-1
  …
  statement1-N
Else
  statement2-1
  …
  statement2-N
End If
```



Relational and logical operators

Relational operators

`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`<>`	not equal to
`=`	equal to

Logical operators

`And`	conjunction
`Or`	alternative
`Not`	negation

x	y	x And y	x Or y	Not x
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Examples (x is numeric variable)

- checking if x is a positive number : `x > 0`
- checking if x is not equal to 0: `x <> 0`
- checking if $x \in [\,-3,\ 5\,]$: `x >= -3 And x <= 5`
- checking if $x \in (\,-\infty, -2\,] \cup (\,10,\ +\infty\,)$: `x <= -2 Or x > 10`

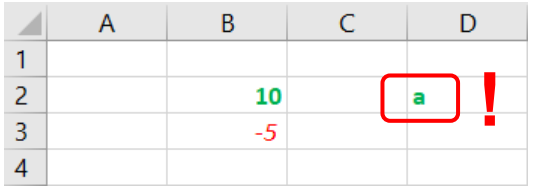
Examples

Procedure sets green font in cells with positive and zero values, red in cells with negative values.

```
Public Sub Color1()
  If ActiveCell.Value >= 0 Then ActiveCell.Font.Color = vbGreen _
  Else ActiveCell.Font.Color = vbRed
End Sub
```

Procedure sets green and bold font in cells with positive and zero values, red and italic font in cells with negative values.

```
Public Sub Color2()
  If ActiveCell.Value >= 0 Then
    ActiveCell.Font.Color = vbGreen
    ActiveCell.Font.Bold = True
  Else
    ActiveCell.Font.Color = vbRed
    ActiveCell.Font.Italic = True
  End If
End Sub
```



Note: text is interpreted as positive numeric value.

Source codes are available on the website

Checking the data type

Selected information functions in VBA

- `IsDate(exp)` – checks if *exp* can be converted to `Date`,
- `IsEmpty(exp)` – checks if *exp* is empty/uninitialized (only for variant type),
- `IsNumeric(exp)` – checks if *exp* is a number (empty cell is a number),
- `IsObject(exp)` – checks if *exp* is an object.

Note: functions **`Is`**... return logical value **`True`**/**`False`**.

Specification of the name/code of the data type

- `TypeName(var)` – returns name of variable type (as string),
- `VarType(var)` – returns code of variable type (as number).

Code	Name
0	Empty
1	Null
2	Integer
3	Long
4	Single
5	Double
6	Currency
7	Date

Code	Name
8	String
9	Object
10	Error
11	Boolean
12	Variant
13	Object
14	Decimal
17	Byte

Example

Modification of procedure `Color2`. An additional condition prevents modification of the font settings when the value in the cell is not a number (condition `.Vallue>=0` is only checked when the cell is not empty and contains a number).

```
Public Sub Color3()
  With ActiveCell
    If Not IsEmpty(.Value) And IsNumeric(.Value) Then
      If .Value >= 0 Then
        .Font.Color = vbGreen
        .Font.Bold = True
      Else
        .Font.Color = vbRed
        .Font.Italic = True
      End If
    End If
  End With
End Sub
```

	A	B	C	D
1				
2		10		a
3		-5		
4				

Source codes are available on the website

Nesting control statements

Nested control statement – a control statement that is contained within another control statement.

Example

```
If condition1 Then                    'first conditional statement
   If condition2 Then statement1      'second conditional statement
Else
   If condition3 Then                 'third conditional statement
      statement2
   Else
      statement3
   End If
End If
```

The second conditional statement is executed when `condition1` is true, so the execution of `statement1` depends on `condition1` and `condition2`.

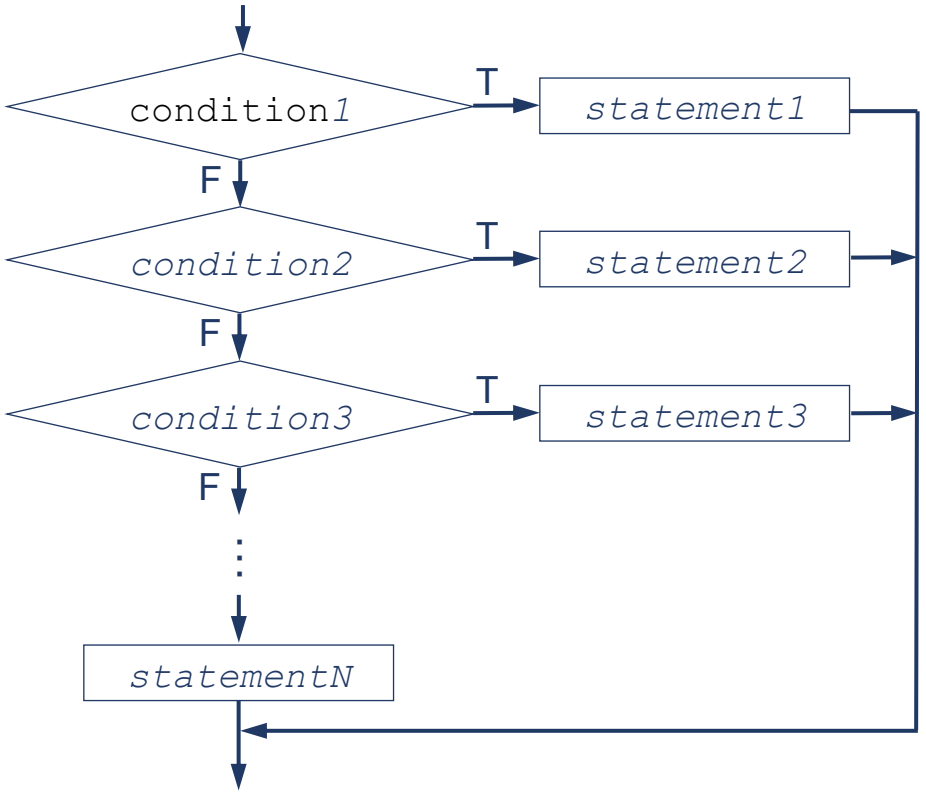
The third conditional statement is executed when `condition1` is false, so the execution of `statement2` and `statement3` depend on `condition1` and `condition3`.

Indentation – space at the beginning of a line code that indicates nesting of statements. It does not affect the execution of the program, but increases the readability of the code.

Block ElseIf

ElseIf is an optional component of **If-End If** statement. It can be repeated many times, each component introduces additional condition.

```
If condition1 Then
  statement1-1
   …
  statement1-N
ElseIf condition2 Then
  statement2-1
   …
  statement2-N
ElseIf condition3 Then
  statement3-1
   …
  statement3-N
ElseIf …
Else
  statementN-1
   …
  statementN-N
End If
```

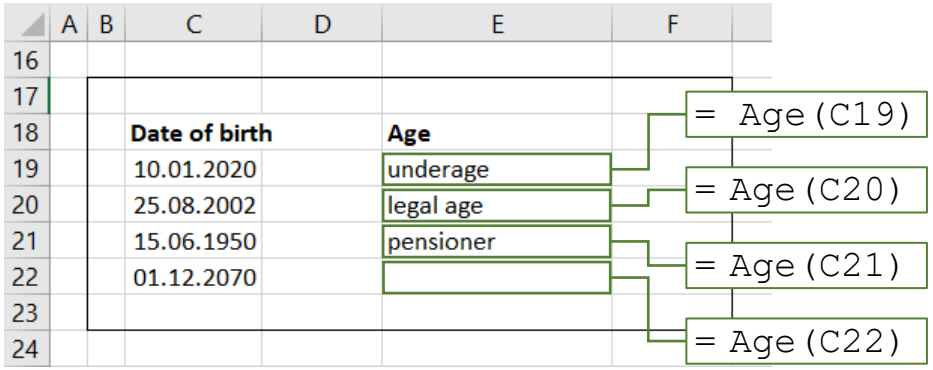


Note: condition *i+1* is checked if *i-th* condition is false. Only one group of statements may be executed in each run. The statements in the **Else** section (optional) is executed if all conditions are false.

Example – function Age

Function determines age based on date of birth (underage, legal age, pensioner).

```
Public Function Age(d As Date) As String
  If d > Date Then
    Age = ""
  ElseIf DateAdd("yyyy", 18, d) > Date Then
    Wiek = "underage"
  ElseIf DateAdd("yyyy", 65, d) > Date Then
    Wiek = "legal age"
  Else
    Wiek = "pensioner"
  End If
End Function
```



`Date` – determines current date.

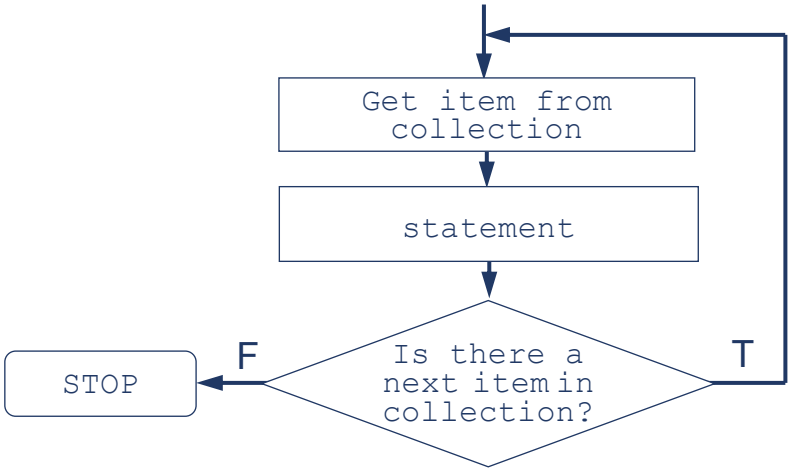
`DateAdd(`*`interval`*`,`*`number`*`,`*`date`*`)` – adds *`interval`* specified by *`number`* to *`date`*. Allowed time intervals: `yyyy` – year, `q` – quarter, `m` – month, `d` – day, `ww`- week, `h` – hour, `n` – minute, `s` – second.

Source codes are available on the website

Collection processing – `For Each`

Execute the statement (group of statements) for each item of *`collection`*.

```
For Each var In collection
   statement1
   statement2
   …
   statementN
Next
```



The loop repeats statements as many times as there are items in *`collection`*. In subsequent iterations *`var`* represents the next item of the *`collection`*.

Collections in VBA

- `Cells` – collection of cells,
- `Selection` – collection of cells in currently selected range,
- `Rows`, `Columns` – collection of rows/columns,
- `Workbooks` – collection of all opened documents,
- `Worksheets` – collection of worksheets.

Example – color of cells, ver. 1

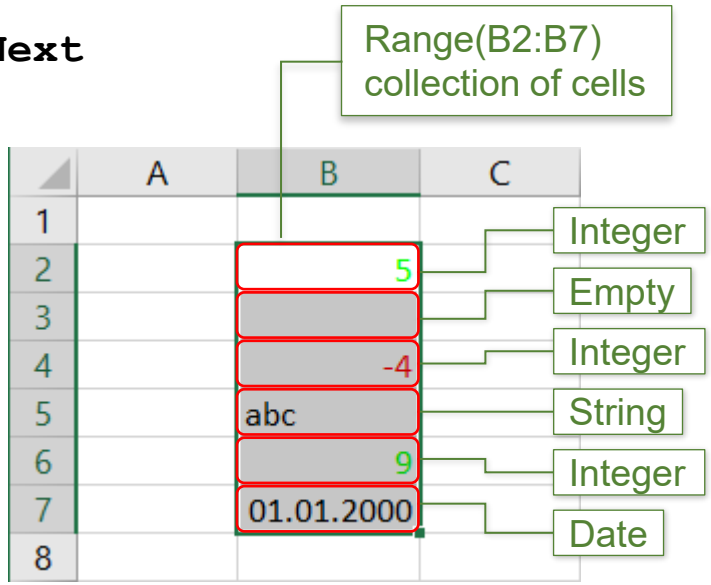
Procedure modifies color of all cells containing numerical values in specified range (it repeats the operation performed by procedure `Color3` (s.11) for each cell in range).

```
Public Sub CellsColor1()
  Dim rng As Range
  Dim c As Range
  On Error GoTo Cancel                                   range selection
  Set rng = Application.InputBox("Select range", Type:=8)
  For Each c In rng                                              loop
    If Not IsEmpty(c.Value) And IsNumeric(c.Value) Then
      If c.Value >= 0 Then c.Font.Color = vbGreen _
      Else c.Font.Color = vbRed
    End If                          color setting when cell contains a number
  Next
  Exit Sub
Cancel:
End Sub
```

Source codes are available on the website

Example – analysis

```
For Each c In rng
  If Not IsEmpty(c.Value) And IsNumeric(c.Value) Then
    If c.Value >= 0 Then
      c.Font.Color = vbGreen
    Else
      c.Font.Color = vbRed
    End If
  End If
Next
```



Analysis (rng=Range('B2:B7'))

1. c = Range('B2')
   non-empty value, number > 0
   Color = vbGreen
2. c = Range('B3')
   empty value
   the color is not modified
3. c = Range('B4')
   non-empty value, number < 0
   Color = vbRed
4. c = Range('B5')
   non-empty value, string
   the color is not modified
5. c = Range('B6')
   non-empty value, number > 0
   Color = vbGreen
6. c = Range('B7')
   non-empty value, date
   the color is not modified

Processing of large ranges

Problems

- Programmer has no influence on the range selected by the user.
- In the case of ranges containing empty cells or cells with values that are not processed, the loop executes not necessary (empty) iterations.
- Large range of cells leads to a significant increase in the macro execution time.

Example

Presented selection contains columns from B to D, each column contains 1 048 576 cells (Excel 2019). Execution of macro `CellsColor1` needs:

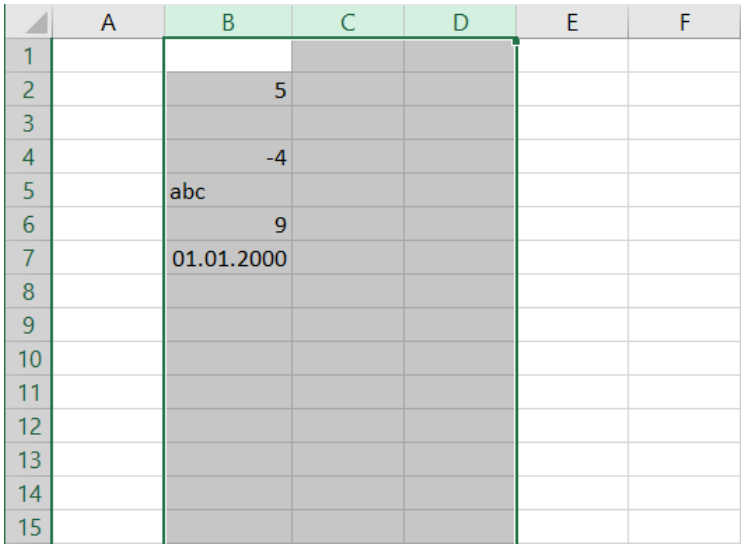
- 3 x 1 048 576 = 3 145 728 iterations,
- execution time (i7 3.4GHz) ~25s,

but there are only:

- 5 cells with values,
- 3 cells with numerical values,

so there are:

- 3 145 725 empty iterations.



Working with ranges

Selection of cells containing certain values (Range object method)

```
SpecialCells(Type As XlCellType, _
             Value As XlSpecialCellsValue) As Range
```

`Type` (required) specifies type of value:

- `xlCellTypeBlanks` – empty cells,
- `xlCellTypeConstants` – cells containing constant values,
- `xlCellTypeFormulas` – cells containing formulas,
- `xlCellTypeVisible` – visible cells.

`Value` (optional) specifies detailed type of value:

- `xlErrors` – cells containing errors,
- `xlLogical` – logical values,
- `xlNumbers` – numerical values,
- `xlTextValues` – text values.

Operations on ranges (Application object method)

```
Union(r1 As Range, r2 As Range, r3, r4, … r30)
Intersect(r1 As Range, r2 As Range, r3, r4, … r30)
```

The methods calculate sum and intersection of ranges `r1...r30` (at least two non-empty ranges are required).

Example – color of cells, ver. 2

Modification of procedure `CellsColor1` (s.16). The range selected by user is limited to cells containing numeric values only.

```
Public Sub CellsColor2()
  Dim rng As Range
  Dim c As Range
  On Error GoTo Cancel
  Set rng = Application.InputBox("Select range", Type:=8)
  Set rng = rng.SpecialCells(xlCellTypeConstants, xlNumbers)
  For Each c In rng
    If IsNumeric(c.Value) Then
      If c.Value >= 0 Then c.Font.Color = vbGreen _
      Else k.Font.Color = vbRed
    End If
  Next
  Exit Sub
Cancel:
End Sub
```

specifying a range and selecting cells containing numeric values

Note: in this case checking if the cell is non-empty is redundant, the range includes only numeric values (including dates).

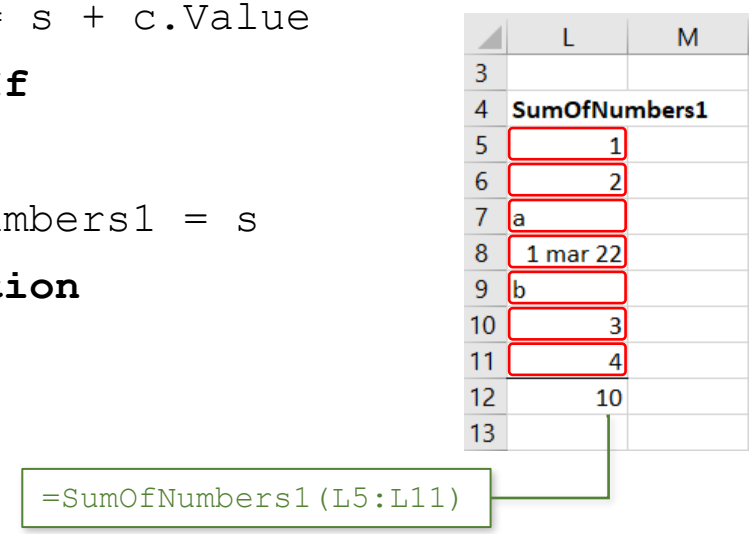
Source codes are available on the website

Function SumOfNumbers

Function calculates sum of numeric values (excluding dates) in a given range.

```
Public Function SumOfNumbers1(rng As Range) As Double
  Dim c As Range
  Dim s As Double
  s = 0
  For Each c In rng
    If Not IsEmpty(c.Value) And IsNumeric(c.Value) Then
      s = s + c.Value
    End If
  Next
  SumOfNumbers1 = s
End Function
```

Note: the function can be supplemented with preselection of cells containing numeric values. Both versions are available on the website.



Analysis, SumOfNumbers1("L5:L11")

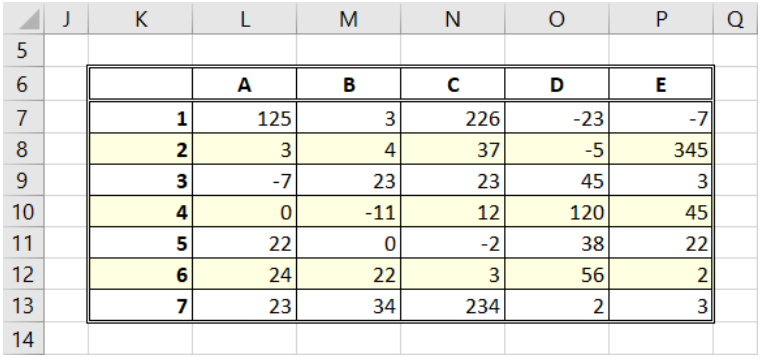
```
0. s=0
1. c=Range("L5"),   s=0+1=1
2. c=Range("L6"),   s=1+2=3
3. c=Range("L7"),   string
4. c=Range("L8"),   date
5. c=Range("L9"),   string
6. c=Range("L10"), s=3+3=6
7. c=Range("L11"), s=6+4=10
```

Source codes are available on the website

Example – processing collection of rows

The procedure sets the background color on the even rows in specified range.

```
Public Sub RowsColor()
  Dim rng As Range
  Dim r As Range
  Dim i As Integer
  On Error GoTo Cancel
  Set rng = Application.InputBox(…)
  i = 1
  For Each r In rng.Rows
    If i Mod 2 = 0 Then r.Interior.Color = RGB(255,255,225) _
    Else  r.Interior.ColorIndex = xlColorIndexNone
    i = i + 1
  Next
Cancel:
End Sub
```



	A	B	C	D	E
1	125	3	226	-23	-7
2	3	4	37	-5	345
3	-7	23	23	45	3
4	0	-11	12	120	45
5	22	0	-2	38	22
6	24	22	3	56	2
7	23	34	234	2	3

Note 1: loop performs operations for each row in selected range (collection **`r.Rows`**).

Note 2: variable **`i`** is incremented in each loop iteration (it is loop counter).

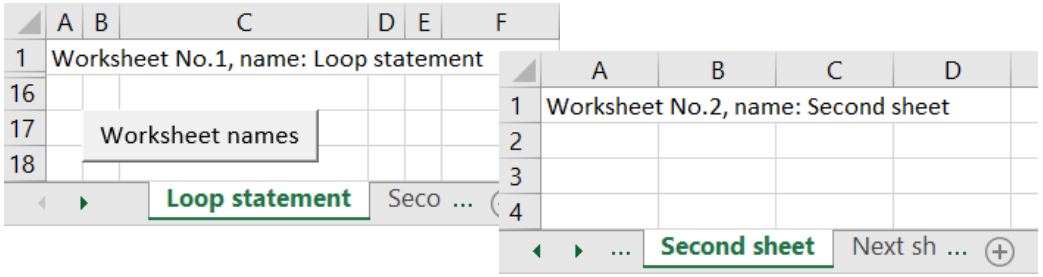
`i Mod 2 = 0` when the row number is even.

Source codes are available on the website

Example – processing collection of sheets

The procedure enters the sheet number and its name into the first cell of each worksheet in the current document.

```
Public Sub WorksheetNames()

  Dim sh As Worksheet

  For Each sh In ActiveWorkbook.Worksheets

    sh.Cells(1, 1).Value = "Worksheet No." & sh.Index & _

                           ", name: " & sh.Name

  Next

End Sub
```



Note 1: loop statement performs operations for each worksheet in the current workbook (collection **`ActiveWorkbook.Worksheets`**).

Note 2: operator „&" combines text values creating a single String value, e.g. **`"abc" & "def" = "abcdef"`**.

Source codes are available on the website