The lecture material, i.e. notes, task sheets and solutions can be found on the webpage:

http://www.staff.city.ac.uk/~fring/ExcelVBA/index.html

2

Excel & Visual basic applications (VBA)

- Create and manipulate tables of data
- Present data in diagrams

- Automatize the manipulation of tables
  - modify existing routines
  - create new applications

- Applications: Widespread use in Industry, Finance & Engineering

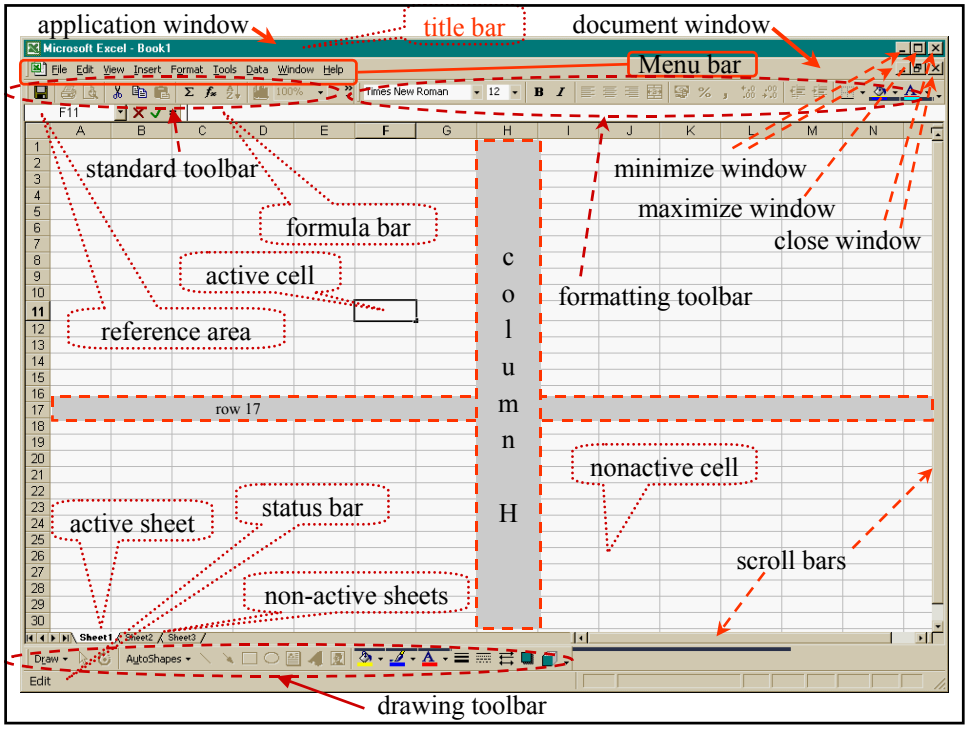
- Excel 2000 is part of the Office 2000 Package.
  Besides Excel it contains:
  - Word for creating text documents
  - Powerpoint for creating presentations
  - Access for creating databases

3

- Literature
  - *Excel 2000, An Introductory Course for Students,* J. Muir (Leaning Matters Ltd, 2001)
  - *Excel 2000 VBA, A Programmer's Reference,* J. Green (Wrox Press Ltd, 2000)
  - *Excel 2002, with Visual Basic Applications,* L. Friedrichsen (Thomson Course Technology, 2002)

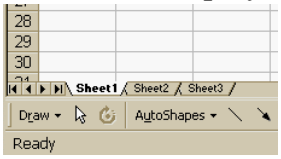
- Getting Started
  - Log into the City University system
  - *Select Excel in the following way*
    → *Start*
    → *Programs*
    → *B: Spreadsheets and Databases*
    → *Microsoft Office Excel 2003* ⇒

4



Basic Excel terminology and features

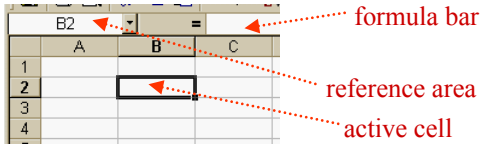
- The data are collected in a table called a **worksheet** (WS).
  - WS have names which can be changed, such as “shell1“, “task“, “income“, ... The names are displayed at the bottom of the page.



  - The **active worksheet** is the one highlighted, e.g. above it is “Sheet1“. A new sheet can become the active sheet by clicking on its name, e.g. “Sheet2“ or “Sheet3“.
  - One can insert new sheets, delete, rename, move and copy existing ones by right clicking on the name.
  - WS consist of vertical columns labeled by letters A,B,C... and horizontal rows labeled by numbers 1,2,3,...

6

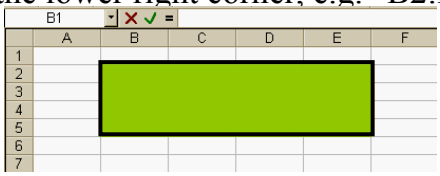
- The intersection of a row and a column is called a **cell** referred to by the letter of the column and number of the row, e.g. B5, AF1,...
  - An **active cell** is the cell currently in use. It is marked by a bold black frame.



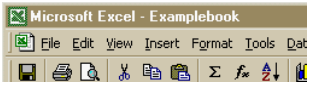
  - The **reference area** indicates the position of the active cell, e.g. “B2“
  - The **formula bar** displays the content of the active cell.
  - An active cell can be de-activated by clicking on another cell, which then becomes the new active cell.
  - Data are manipulated on the WS in the active cell.

7

- A collection of cells is called a **range**.
  - It is referred to by the name of the cell in the top left corner and the cell in the lower right corner, e.g. “B2:E5“ are 16 cells.

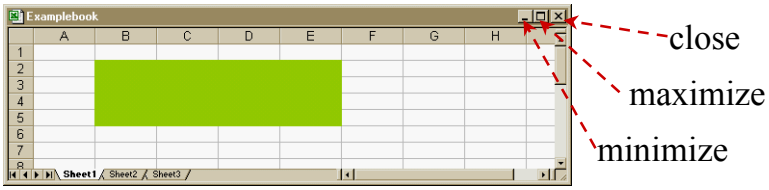


  - One can select a range by enlarging the active cell area. This is done by placing the screen pointer into the active cell, pressing the left mouse button and dragging the pointer down to the right to define the lower right corner of the range. Releasing the left mouse button will leave a bold frame surrounding the range.
- A collection of worksheets is called a **workbook** (WB).
  - The name of the workbook is displayed in the **title bar**, e.g. “Examplebook“



8

- The Excel window consists of two windows, the document window and the application window.
- The **document window** contains the worksheets, scroll bars, ...



  - the **scroll bars** allow to move to parts of the window which are currently invisible, e.g. the range K4:P25.
  - the **minimize button** skrinks the document window
  - the **maximize button** enlarges the window to the full size of the application window
  - the **close button** closes the document window
- The **application window** contains the Excel commands, menues...
  - **menu bars** contain the main Excel commands, File, Edit,..., which by themselves contain a wide range of commands

9

- **control windows** (minimize, maximize, close) allow to resize move, close and restore the application window
- **standard toolbars** offer various options such as copying, printing, cutting, pasting, etc.
- **formatting toolbars** offer various options to change the format of the individual cell and the entire WS
- **drawing toolbars** give options which allow to include various geometrical shapes and change the colour of the sheet, cells, text.
- the **status bar** displays the progress of the commands or operations which are currently executed, e.g "Edit" when text is being edited or "Ready" when no comman is executed
- one can make more toolbars visible by selecting
  → *View*
  → *Toolbars*
  → *Chart, Clipboard, Forms, Visual Basic, etc*

10

Basic Operations

▸ Creating a **workbook**:

• The first action should always be to give your workbook a name and save it on your computer.
Go for this to the menu bar and select by left mouse click (LC):

→ File → Save As

→ In the window which then opens enter a file name:

→ Save as "Microsoft Excel Workbook"

• Excel files have the extension "xls" on the disk, e.g. the above example will be stored (saved) as "Firstworkbook.xls".

• Organize your Excel files (in fact like all other ones) in folders.

• While editing save your file once in a while just by LC on save , this way you do not loose data in a crash.

11

▸ Entering data onto a **worksheet**:

• Data can be entered onto the worksheet either by typing into the active cell or by typing into the formula bar.

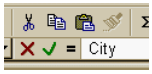
• Cell entries may be completed in various ways, e.g. complete C8.

a) Enter → moves to the next cell in the same column, e.g. C9.

b) Shift+Enter → moves to previous cell in the column, e.g. C7.

c) Cursors ←,→ ↑,↓ → move to the cell in the direction indicated, e.g. B8, D8, C6, C9.

d) Enter button → completes but does not move to a new cell.



e) Esc → does not move and cancels all modification done after the last completion of the type a), b), c) or d).

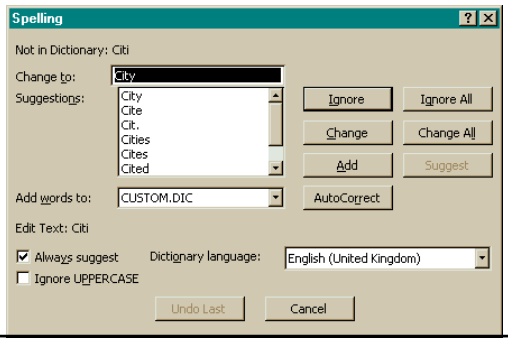
f) Cancel button → has the same effect as e).(Edit →Undo,)

Do not finish the entry by clicking with the pointer onto a new cell as this will produce wrong results for formulae.

12

▸ Modifying entered data:

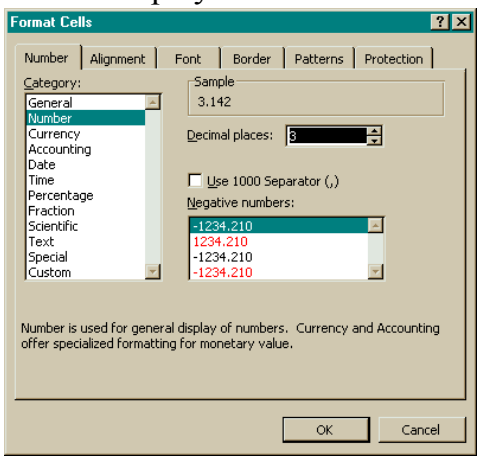
- Use “delete“ or “backspace“ (⟵) to delete right or left from the insertion point, respectively.
- LC on a cell (or formula bar) with some old entry and overtype it.
- Edit → Clear → All (deletes all informations related to the cell)
  → Contents (deletes only the content of the cell)
- Use the spelling tool to correct mistakes: Tools → Spelling
  Expl.: Assume a cell has the entry “Citi“, Tools → Spelling →



13

▸ Formatting the cell entries:

- For presentational reasons one can change the format in which the cell entries are displayed: Format → Cells →



- One can change the category (type of data) and its associated properties.

14

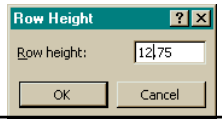
- Alignment allows to change the horizontal or vertical position and the orientation of the text.
- Font gives options to change the typeface and the colour of the displayed entry.
- Border provides possibilities to change the style of the frame surrounding a cell.
- Pattern changes the background of the cell.
- Protection allows to protect cells from being changed.

▸ Formatting the cell size:

• Position the pointer on the dividing line between the name of the row/column (e.g. 5|6 / E|F) and drag the line to the desired size.

• Alternatively use the menu bar:

Format → Row → Height or Format → Column → Width and change the numerical value, e.g.



15

▸ Merging cells:

• One can merge a range of cells into one single cell. This is useful for instance for large titles. To do this select the range and then LC on the "Merge and Center" button in the formatting toolbar

▸ Adding comments to a cell:

• Sometimes it is useful to add some additional information to a cell which should not be visible on the WS for some reason. This is done by: Insert → Comment → a window open into which one can type a comment.

• A cell which has a comment attached to it is marked in the top right corner by a red triangle.

• The comment is made visible by pointing on top of the cell.

• The comment is removed by Edit → Clear → Comments.

16

▸ The Autofill function:

• The Autofill function determines automatically the entries of some cells given some starting values. Avoids lots of typing!

· Expl.: Fill the column C1-C20 with 50-1000 with step 50, i.e.
50 → C1, 100 → C2, 150 → C3, ... 1000 → C20

- fill in some starting values: 50 → C1, 100 → C2
- select the range of the starting values C1:C2
- while on top of the selected area the cursor will be ✚
- move the cursor to the lower right corner of the selection, until the cursor changes from ✚ to +
- drag the "fill handle" down (or to the right) and the new cells will be filled based on the initial selection, e.g. 150 → C3,...
- verify that Excel really filled in the sequence you wanted!!!

· Alternatively write just 50 into C1. Use Edit → Fill → Series with "Step value"=50, "Stop value"=1000

17

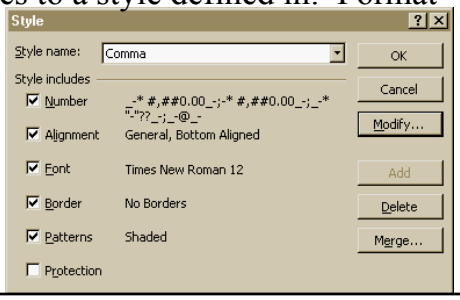
▸ Different types of data:

• **text** is left adjusted by default, use the format toolbar to change the adjustment, text format, indent or colour (find out yourself)

• **numbers** are right adjusted by default, use the format toolbar to change the format as for text

- in/decrease decimal → in/decreases the number of decimal points *displayed*, but not the cell value, e.g. 3.141 → 3.1415/3.14

- comma style → changes to a style defined in: Format → Style



18

- percentage style % → changes to a style pre-defined in the "percentage" style (similarly as above)
- currency style → changes to a style pre-defined in the "currency" style (similarly as above). It adds a currency sign such as $, £, €,...

• **formulae** are expressions which tell Excel to perform operations

- All formulae begin with an "="-sign followed by some arithmetic expression.
- The expression may contain numeric values, cell references and arithmetic operators. It is important to note the difference between the formula inside a cell and the numerical value displayed on the WS. With Ctrl+` you can change the display.
- Expl.: In the cell C5 write "=A1 +A2+A3". This will add the three cells A1, A2 and A3 and displays the result in C5. When you alter A1-A3 C5 will change accordingly.

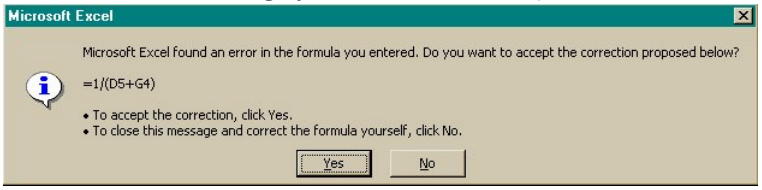
19

- In Excel and other major programming languages expressions are evaluated following a specific order of precedence for the arithmetic operators.
- The order is:
  · negation: "-"
  · exponentiation: "^"
  · multiplication and division: "*", "/"
  · addition and subtraction: "+", "-"
- The order of precedence can be overwritten by parentheses.

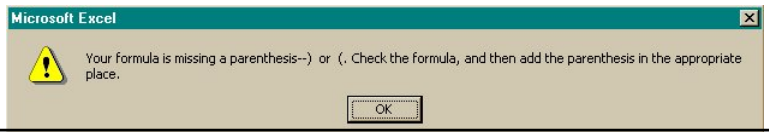
Expl.:
-4^2 → 16
-(4^2) → -16
3*(5+6) → 33
3*5+6 → 21
3^2+ 7 → 16
3^(2+7) → 19683

20

- Formulae can be entered by means of pointing, that means instead of typing the cell reference one can simply LC on the appropriate cell.
- Expl.: Enter the formula =1/(D5+G4) into B4
  → type "=1/(" into B4 → LC on D5 → type "+"
  → LC on G4 → type ")" → complete the entry
  · in case you entered a formula incorrectly Excel offers a corrected version, e.g. you entered "=1/(D5+G4"→



  · in case you select "No " Excel tells you your mistake



21

▸ Relative, Absolute and Mixed Cell References

- There are several default assumptions made by Excel when you enter a cell reference:

a) Excel assumes the cell is on the same WS and in the same WB as the cell in which you enter the formula.

b) Excel assumes the reference is a relative reference, that means the cell reference changes when you copy the cell.

- You copy a cell or a range by selecting the range or the cell Edit → Copy → select the destination cell(s) → Edit → Paste

- A column or a row can be "fixed" by adding a "$"-symbol: There are four possibilities:

= A1	≡	changeable column and row (relative reference)
= A$1	≡	changeable column, fixed row (mixed reference)
= $A1	≡	fixed column, changeable row (mixed reference)
= $A$1	≡	fixed column and row (absolute reference)

22

• Examples:

copy cell reference	paste cell reference	relative difference	formula being copied	final formula pasted cell
C5	D6	add one column add one row	=F4 =$F$4 =$F4	=G5 =$F$4 =$F5
C5	D3	add one column subtract 2 rows	=K7*B$7 =A3+$B7	=L5*C$7 =B1+$B5
C5	F11	add 3 columns add 6 rows	f(A1:B5) f(A$3:A7)	f(D7:E11) f(D$3:D13)
C5	F1	add 3 columns subtract 4 rows	=A3 =Z5	=#REF! =AC1

- f(...) indicates some function see below

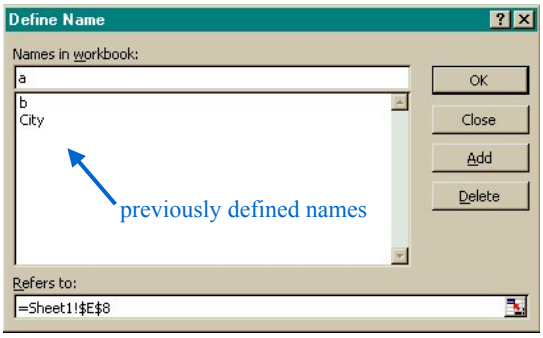
=#REF! is an error message ≡ cell reference not valid

23

▸ Naming cells or ranges:

• You can attach a name of your choice to a cell or a range and then use it as variable in a formula instead of a lengthy references

Menu bar: Insert → Name → Define →



-The name will also appear in the reference area. You can also type there directly to give a name to the cell.

Expl.: · cell A1 is called "b" and cell E8 is called "a"

=(A1+E8)^2 ≡ =(a+b)^2

· the range B2:H8 is called "City"

=f(B2:H8) ≡ =f(City)

24

Excel Functions

- Excel is equipped with over 300 built-in functions.
  - They are divided into 10 groups: mathematical and trigonometric, logical, statistical, date and time, database, financial, text, informational, lookup and reference, engineering (Toolpack)
    - A function computes "something" and returns a value.
  - Syntax: =name(argument)
    - "name" is the name of the function

      "argument" is a list of cells, ranges, other functions or formulae
      - the number of arguments can vary, e.g.

        zero arguments: =PI() → 3.1415926535898....

        one argument: =SQRT(B5) ≡ √B5 → 2 for B5=4

        two arguments: =ROUND(PI(),3) → 3.142

        variable number: =SUM(C1:C10,B12,B5) → sums up the values of the cells C1,C2,...,C10,B12,B5

25

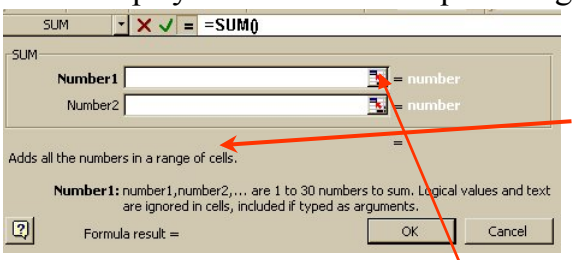
- When the number is variable, the maximum number of allowed arguments is 30 and the maximum number of characters is 1024. A range counts as one argument.

- You can either type the entire syntax or use:

  Insert → Function → (or LC)

  → gives a list of functions organized into the mentioned groups

  → selecting a particular function opens a dialog window which helps you to fill in the required arguments, e.g.



You also get the information about what the function does.

→ either type in the data or LC on , which allows you to select cells by pointing (see above).

26

- There are some special functions called array functions which need to be entered in a particular way (see below).
- There are various ways to make errors when using functions.

Excel will give the following error messages:

#DIV/0! ≡ division by zero
#NAME? ≡ a formula contains an undefined variable or function name, or the function syntax is not valid
#N/A ≡ value is not available, that is when formulae refer to cells which do not contain the appropriate data
#NULL! ≡ a result has no value
#NUM! ≡ numerical overflow, e.g. SQRT(A1) for A1 is -5
#VALUE! ≡ invalid argument type, e.g. SQRT(A1) for A1 containing text
#REF! ≡ invalid cell reference
circular error ≡ a formula contains a reference to its own location

27

- Examples for functions:
  - Mathematical &Trigonometric Functions
    - These are standard functions you also find on a calculator, e.g.

    =SIN(x) x is an angle in radians, e.g. SIN(PI()/2) → 1

    =COS(), =TAN(), =ACOS(), =EXP(), LN(), =COSH(), =ABS(), =PRODUCT(x,y,z,..) , =FACT(x) (x!), ...
  - Statistical Functions
    - These are functions from statistics & probability, e.g.

    =AVERAGE(A1:B7) ≡ computes the arithmetic mean

    =MAX(A1:B7) ≡ returns the largest number in A1:B7

    VAR, POISSON,SLOPE,TREND,HYPGEODIST,.....
  - Date & Time Functions
    - These are functions which deal with time, e.g.

    =TODAY() ≡ returns todays date

    =NOW() ≡ returns todays date and the current time

28

- Text Functions
  - These are functions which manipulate text strings and data, e.g.
    =EXACT(text1,text2) ≡ returns "TRUE" if text1=text2 and "FALSE" if text1≠text2 (case sensitive)
    =UPPER(text) ≡ converts all characters of text to upper case
- Financial Functions
  - These are functions with some financial applications, e.g.
    =FV(rate,np,pmt,pv,type) ≡ future value of an investment
    rate ≡ interest rate per period
    np ≡ total number of payment per year
    pmt ≡ is the payment made each period
    pv ≡ initial lump-sum, (optional, default is 0)
    type ≡ indicates when payments are due, 1 at the beginning of the period and 0 at the end of the period (optional, default is 0)

29

Expl.: You deposit £1,500 into a savings account at a monthly interest rate of 0.6%. You plan to deposit £150 at the beginning of every month for the next 2 years.
How much money will be in the account after 2 years?
FV(0.6%, 24, -150, -1500, 1) → £5,614.42

- Information Functions
  - These are functions which return informations about the cell data, format etc, e.g.
    =TYPE(A1) ≡ returns a number which stands for the data type contained in cell A1: 1 = number, 2 = text, 4 = logical value, 16 = error value, 64 = array
- Logical Functions
  - These functions handle boolean values, i.e. TRUE or FALSE. There are 6 functions of this type, IF, NOT, AND, OR, FALSE() and TRUE().

30

· The IF-function is used when you want the function to return a different result depending on the value of a certain condition.
Syntax: =IF(*condition*, value for true, value for false)

condition = expression1 *comparison operator* expression2

comparison operators: = ≡ equal to
<> ≡ not equal to
> ≡ greater than
>= ≡ greater than or equal to
< ≡ less than
<= ≡ less than or equal to

Expl.: - =IF(B3>0, "positive", "negative")
returns the text value "positive" if the value in the cell B3 is positive and otherwise the text "negative" .
- =IF((A1-B2)>=0, SQRT(A1-B2), "complex value")
- =IF(SUM(A1:A9)>0, 1, 0)
- =IF(D6, "true", "false")

31

· IF-functions can be nested up to seven times, which means that inside the argument of an IF-function (as condition or returned value) you can have further IF-functions.
Expl.: - =IF(A1>-5, IF(A1<=5,1,0) , 0) produces the function:

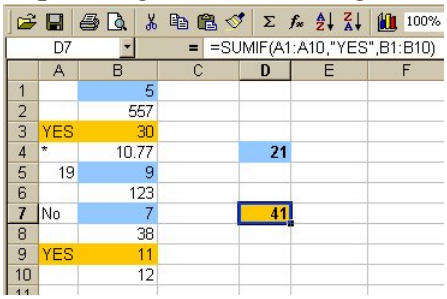
$$f(A1) = \begin{cases} 0 & \text{for } A1 \le -5 \\ 1 & \text{for } -5 < A1 \le 5 \\ 0 & \text{for } A1 > 5 \end{cases}$$

· Several Excel functions contain implicit IF- statements,e.g.
=SUMIF(range,condition,sum_range)
range ≡ The range to be evaluated.
condition ≡ A criterium which select out certain values.
sum_range ≡ The range which will actually be evaluated
It is optional. When omitted it corresponds to range.

32

· Expl.: - =SUMIF(B1:B10,"<10") sums up all values in the cells B1:B10 which are smaller than 10.

- =SUMIF(A1:A10,"YES",B1:B10) sums up all the values in the cells B1:B10 when YES is written in the corresponding cell of the range A1:A10.

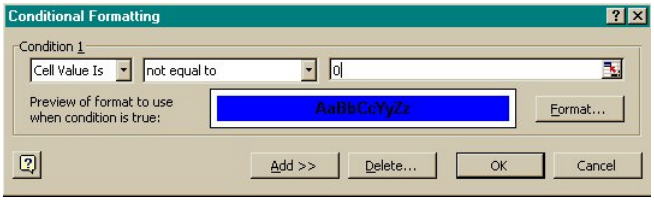


	A	B	C	D	E	F
1		5				
2		557				
3	YES	30				
4	*	10.77		21		
5	19	9				
6		123				
7	No	7		41		
8		38				
9	YES	11				
10		12				

· Also Formats can contain an implicit IF-statement. This is called conditional Formatting. It is for instance useful when some cells only contain data under certain circumstances.

33

You invoke it by: Format → Conditional Formatting →



In the example only when the content of the conditionally formatted cell(s) is not equal to zero the cell will be coloured in blue and the text will be displayed in bold. (see Lab-Session 2)

· The AND-function can be used to produce more complex tests. It returns the logical value TRUE if **all** conditions in its argument are true.

Syntax: =AND(condition1,condition2,condition3,...)

Expl.: - =AND(A1>5,A2>5) returns TRUE for A1>5 and A2>5

- =IF(AND(A1>-5, A1<=5) ,1 , 0) produces the same function as the example for the nested IF-function

34

· The OR-function can also be used to produce more complex tests. It returns the logical value TRUE if **at least one** condition in its argument is true.

Syntax: =OR(condition1,condition2,condition3,...)

Expl.: - =OR(A1>5, A2>5, B1>5, D11>5)

Returns TRUE if any of the values in A1, A2,B1,D11 is greater than 5 and otherwise FALSE.

- =IF(OR(A1<=-5, A1>5) ,0 , 1)

Produces the same function f(A1) as the example for the nested IF-function.

· The NOT-function reverses the values of its logical argument, i.e. TRUE is changed into FALSE and vice versa.

Syntax: =NOT(condition)

Expl.: - =IF(NOT(OR(A1<=-5, A1>5)) ,1 , 0)

Produces again the function f(A1).

35

· There are useful combinations of AND, OR and NOT:

NAND : =NOT(AND(A,B)) ≡ not both are true

NOR: =NOT(OR(A,B)) ≡ neither is true

XOR: = OR(AND(A, NOT(B)), AND(B, NOT(A)))

≡ only one is true

· The boolean values TRUE or FALSE can be entered as: TRUE, =TRUE, =TRUE() and similar for FALSE.

• Lookup & Reference Functions

Lookup functions can be used for various purposes. They can be used to retrieve information from a reference list of data and use them in some other part of the WS or WB. In general they are equivalent to some combination of multivalued IF-functions. Reference functions return informations about the cell reference as text values, such as the entire address, the row or column.

36

Syntax:

=VLOOKUP(lookup_value, table_array, column_index,match)

=HLOOKUP(lookup_value, table_array, row_index,match)

lookup_value ≡ The value to be located in the first column of a vertical table (or the first row of a horizontal table). It can be numeric, text or a cell reference.

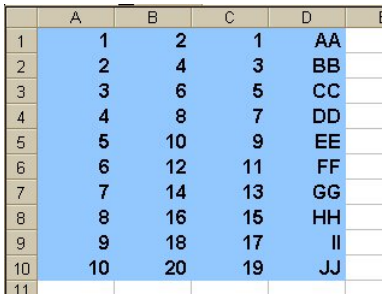
table_array ≡ The range reference or name of the lookup table.

column(row)_index ≡ The column (row) of the table from which the value is to be returned.

match ≡ Is a logical value, i.e. TRUE or FALSE, which specifies whether you want an exact or approximate value. It is optional with default value TRUE. In that case the functions returns the next largest value which is less than the lookup value. For FALSE it only returns exact matches. If there is no exact match → #N/A

37

Examples: Consider the following table



	A	B	C	D
1	1	2	1	AA
2	2	4	3	BB
3	3	6	5	CC
4	4	8	7	DD
5	5	10	9	EE
6	6	12	11	FF
7	7	14	13	GG
8	8	16	15	HH
9	9	18	17	II
10	10	20	19	JJ

- =VLOOKUP(6,A1:D10,2) → 12
- =VLOOKUP(4,A1:D10,3) → 7
- =VLOOKUP(8,A1:D10,4) → HH
- =VLOOKUP(3.2,A1:D10,3) → 5
- =VLOOKUP(16,A1:D10,2) → 20
- =VLOOKUP(16,A1:D10,2,FALSE) → #N/A
- =VLOOKUP(8,A1:D10,5) → #REF!
- =VLOOKUP(8,A1:E10,5) → 0
- =VLOOKUP(8,B1:D10,2) → 7
- =VLOOKUP(F1,A1:D10,2) → 6 for F1=3
  → 18 for F1=9
  → #N/A for F1≠1,...,10

38

· Improve the currency conversion table of Lab-session 1 task 2.

Euro	Pounds	Dollar
€ 0.10	£0.0688	$ 0.1238
€ 0.20	£0.1377	$ 0.2476
€ 0.30	£0.2065	$ 0.3715
€ 0.40	£0.2754	$ 0.4953
€ 0.50	£0.3442	$ 0.6191
€ 0.60	£0.4130	$ 0.7429
€ 0.70	£0.4819	$ 0.8667
€ 0.80	£0.5507	$ 0.9906
€ 0.90	£0.6196	$ 1.1144
€ 1.00	£0.6884	$ 1.2382
€ 2.00	£1.3768	$ 2.4764
€ 3.00	£2.0652	$ 3.7146
€ 4.00	£2.7536	$ 4.9528
€ 5.00	£3.4420	$ 6.1910
€ 6.00	£4.1304	$ 7.4292
€ 7.00	£4.8188	$ 8.6674
€ 8.00	£5.5072	$ 9.9056
€ 9.00	£6.1956	$ 11.1438
€ 10.00	£6.8840	$ 12.3820
€ 20.00	£13.7680	$ 24.7640
€ 30.00	£20.6520	$ 37.1460
€ 40.00	£27.5360	$ 49.5280
€ 50.00	£34.4200	$ 61.9100
€ 60.00	£41.3040	$ 74.2920
€ 70.00	£48.1880	$ 86.6740
€ 80.00	£55.0720	$ 99.0560
€ 90.00	£61.9560	$ 111.4380
€ 100.00	£68.8400	$ 123.8200

Into the cells I20:I22 we put:

```
0.7
=VLOOKUP(I20,$D$6:$F$33,2)
=VLOOKUP(I20,$D$6:$F$33,3)
```

Into the cells J20:J22 we put:

```
8
=VLOOKUP(J20,$D$6:$F$33,2)
=VLOOKUP(J20,$D$6:$F$33,3)
```

Into the cells K20:K22 we put:

```
50
=VLOOKUP(K20,$D$6:$F$33,2)
=VLOOKUP(K20,$D$6:$F$33,3)
```

This will produce:

€ 58.70	€ 0.70	€ 8.00	€ 50.00
£40.41	£0.48	£5.51	£34.42
$72.68	$0.87	$9.91	$61.91

39

· A geologist wants to grade some ore samples found on four different sites based on their rare metal content. Ore with a rare metal content of 50-59 ppm is given a low grade, 60-79 ppm is medium grade, 80-99 ppm is high grade and anything greater or equal 100 ppm is very high grade.

The following worksheet performs this task.

	A	B	C	D	E
1		Quality			
2	ppm	50	60	80	100
3	grade	low	medium	high	very high
4					
5	site	ppm	grade		
6	A	55	low		
7	D	111	very high		
8	C	60	medium		
9	B	77	medium		
10	A	44	#N/A		
11	B	88	high		
12	C	99	high		
13	C	56	low		
14	D	102	very high		

- The lookup_values are in row B6:B14.
- The lookup_table is the range B2:E3.
- The values to be selected depending on the grade are in the column B3:E3.
- The HLOOKUP functions are in the column C6:C14.

Produce this WS in Lab-session 2.

40

· Other lookup functions are MATCH, INDEX etc.

· Use the help option to find out how reference functions work.

▸ Protecting and hiding worksheet informations:

· When writing workbooks or worksheets you may want to protect parts of them to make sure that your work will not be changed by accident (or deliberately). Possibly some of the informations on the WS might be confidential and should only be visible to certain users.

You set a protection by: Tools → Protection → Protect Sheet

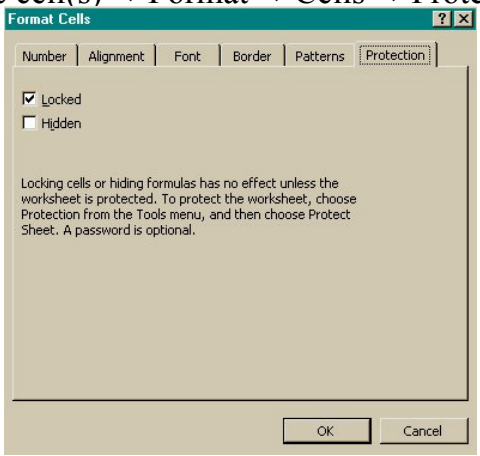
You can choose now which type of date you want to protect either contents, scenarios or objects on the WS. Optionally you can type a password, such that only with the use of this password the entire WS will be unprotected.

Unlock a protection by: Tools → Protection → Unprotect Sheet → Password

41

You can also just protect or hide parts of the worksheet:

Select some cell(s) → Format → Cells → Protection →



Expl.: The Excel file for the solutions of Lab-session 2 will be on the course webside. The sheets are protected with “Hidden“ so that you can get an idea what the solution should look like without being able to see the formulae.

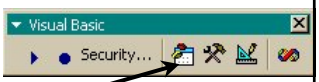
42

User-defined (Custom) Functions

- Excel is equipped with the powerful programming language Visual Basic for Applications (VBA). VBA allows you to write your own programs, such as user-defined functions (UDF) and subroutines (see later in the course).
  - What is a UDF?
    Just like a built-in function, a UDF is a pre-defined formula which can be executed in the same way. The difference is that you design the definition exploiting the flexibility of VBA.
  - When and why do you use a UDF?
    You use a UDF for the same reason as a built in function, namely to make calculations (operations) which are repeated more efficient.
  - Before writing a UDF make sure that it or parts of it do not already exist as built-in Excel functions.

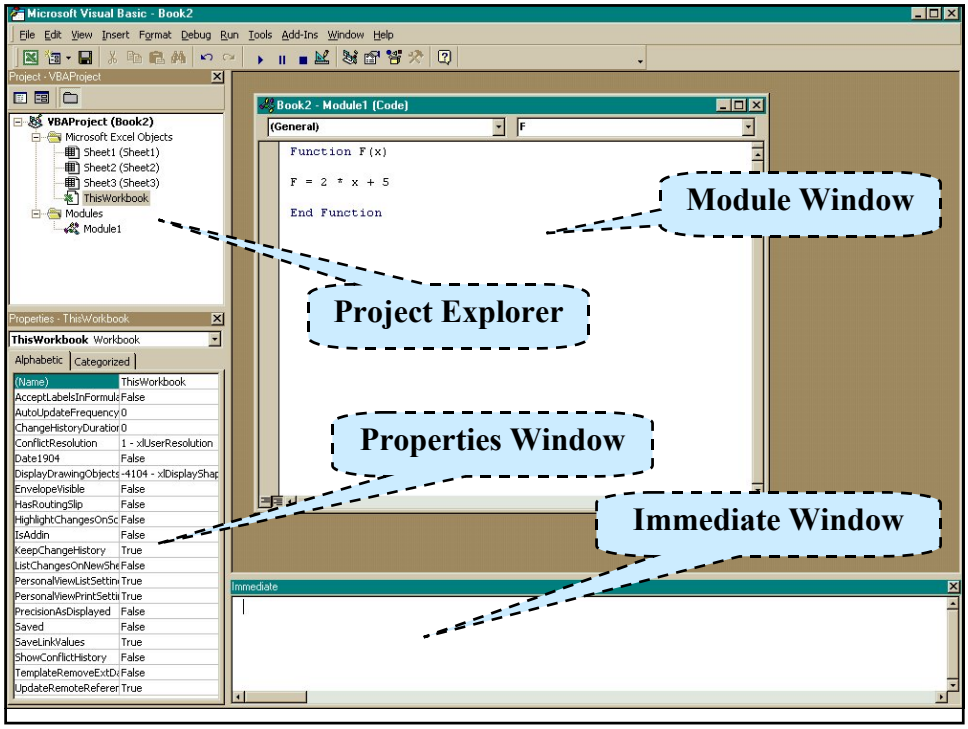
43

- Creating a UDF
  - First activate the Visual Basic Editor (VBE)
    - Tools → Macro → Visual Basic Editor (LC)
    - or use the keyboard shortcut Alt + F11
    - When you program it is useful to include the Visual Basic toolbar into your menu:
      View → Toolbars → Visual Basic (LC) →
      Now you can also activate the VBE by LC on



  - The anatomy of the VBE is like most other applications. It is equipped with a menu and a toolbar at the top of the window and has several subwindows:
    - The Project Explorer displays the hierachical structure of projects.
    - The Properties Window displays the properties of the projects.
    - The Module Window contains the VBA-code of your project.
    - The Immediate Window displays compiling messages.

44



- The Module Window might not be visible when you open VBE.
  VBE menu bar: Insert → Module (LC)
- The Immediate Window is made visible by
  VBE menu bar: View → Immediate Window (LC)

• You return to the Excel window by:

- LC on the Excel icon in the windows toolbar.
- LC on the Excel icon in the VBE toolbar.
- Use the keyboard shortcut Alt+F11.

• Writing any kind of computing program consists of three basic principal steps:

i) Design an algorithm which will perform the task you want.
ii) Translate the algorithm into a computer language (code) with a certain syntax, e.g. VBA in our case.
iii) Test (debug) your program thoroughly.

These steps are not carried out just once in consecutive order! 46

- The UDF syntax:
  Function name [(arguments) [**As** type]] [**As** type]
  [statements]
  [name = expression]
  [**Exit Function**]
  [statements]
  [name = expression]
  End Function
  - name ≡ the name of the function
  - arguments ≡ a list of input value (just like for built-in functions)
  - type ≡ the data type which will be returned by the function
  - statements ≡ valid VBA commands
  - expression ≡ an arithmetic expression assigned to the function name, which will be returned
- Everything in **bold** has to be typed exactly as above.
- Everything in squared brackets [...] is optional.

47

- Each statement has to begin in a new line.
- In case the statement is longer than the line you can split it by typing " _" (i.e. space and underscore). You can not split VBA commands this way!
- A program (function) is read from top to bottom, that is each line is executed after the next. There might be branches, loops etc which you can design.
- When **End Function** or **Exit Function** is reached the calculation terminates and the value last assigned to the function´s name is returned.
  - An assignment is done by an equation, which has to be read from the right to the left, i.e. the value on the right hand side of the equation is assigned to the name on the left hand side
- The arguments are the Input and the function name contains the Output.

48

• Examples:

a) Function F(x)
F = 2 * x + 5
End Function
- You can now use this function on an Excel WS in the same way as you use a built-in function, e.g. "=F(5)"→ 15

b) Function FF(x)
h = 2 * x
FF = h + 5
End Function
- The variable h only exists temporarily inside the function FF.
- Note: F(x) is the same function as FF(x)

c) Function G(x,y,z)
G = y*x + z
End Function
- As for built-in functions you can have more than one input variable (argument).
- Note: G(x,2,5) gives the same as F(x)

d) Function Q(a,b,c,x)
' quadratic equation
Q = a*x^2 + b*x +c
End Function
- You can add comments to enhance the readability. VBA does not execute text following a single quote.

"=Q(2,3,10,2)" → 24

49

e) Function S(x, y, z)
S = 2 * Application.WorksheetFunction.SUM(x, y, z)
End Function
- You can use Excel built-in functions inside UDF by Application.WorksheetFunction.FunctionName, e.g. FunctionName = SUM. "=S(1,2,3)" → 12

f) Function Squareroot(x)
Squareroot = 2*Sqr(x)
End Function
- Some built-in functions can be used in VBA under slightly different names, e.g. SQRT =Sqr. "=Squareroot(9)" → 6
- Other functions are: Abs, Atn, Cos, Exp, Fix, Int, Log, Mod, Rnd, Sgn, Sin, Tan (For a list with explanations use the help function and search for "Math Functions". You also get a list for "derived Math Functions" such as Hsin,...)

50

- Comments on the names of UDF
  - The first character in the name has to be a letter.
  - The names are not case sensitive.
  - Names are not allowed to contain spaces, @, $, #,... or be identical to VBA commands.
- A few comments on debugging
  - Inevitably you will make some mistakes either just typos or structural ones and you need some strategy to eliminate them.
  - Some mistakes block the entire WS, e.g. suppose you type:

```
Function Err(x)
    Err = 2 * Sqr        (Here the brackets are missing in Sqr)
End Function
```

  - Call this function on the WS (Recalculation of the WS is F9) → an error message will be displayed → LC on OK → the mistake will be highlighted → Unlock with "Reset" ≡ ■

51

- Declaration of the variable type
  - Recall: **Function** name [(arguments) [**As** type]] [**As** type]
  - The first type refers to the variable type of the arguments and the second type to the variable type of the function.
  - You can also declare variables used inside the program:
    - Syntax: Dim variable_name as type
  - When you do not declare the type it will be "variant" by default.
  - Why is is useful to declare the type?
    - Declaring the type avoids that different types of data get mixed up. You can trace systematically mistakes in long programs.
    - The variant type takes more space than properly defined variables. Your program will run faster when you declare the types.

52

- There are the following types of variables:
  - integer ≡ integer numbers 0, ± 1, ± 2, ± 3, ... ±32767, -32768
    Expl.: Dim a as integer
    a = 32768 → gives an error
    a = 11.3 → a = 11
  - single ≡ 32 bit (4 byte) floating point number between
    -3.402823E38 to - 1.401298E-45 and
    1.401298E-45 to 3.402823E38
  - double ≡ 64 bit (8 byte) floating point number between
    -1.79769313486231E308 to -4.94065645841247E-324
    4.94065645841247E-324 to 1.79769313486232E308
  - string ≡ can contain up to 2 billion (2^31) characters
  - boolean ≡ 16 bit (2 byte) number which is "true" or "false"
  - variant ≡ 16 byte with numerical value
    (here you see the disadvantage)

53

  - date ≡ 64-bit (8-byte) number representing
    dates from 1-st January 100 to 31-st December 9999
    and times from 0:00:00 to 23:59:59.

▸ Working with dates and times

- VBA handles dates as numbers where
  1-th of January 1900 = 1
  2-nd of January 1900 = 2
  .....
  25-th of October 2005 = 38650
- Some Date and time related VBA-functions:
  - Month(date) → a number between 1 and 12 representing the month
  - Weekday(date) → a number between 1 and 7 representing the day
  - Year(date) → a number between 1000 and 9999 for the year
  - Hour(date) → a number between 0 and 23 for the hour
  - Minute(date) → a number between 0 and 59 for the minute
  - Second(date) → a number between 0 and 59 for the second

54

- Examples:

a) Write a UDF which computes the weekday for a date

```
Function DD(da As Date)
    DD = Weekday(da)
End Function
```

· Format the cell A1 as date and enter 25/10/2005

· "=DD(A1)" → 3

b) Write a UDF which calculates the age in years given the birthdate.

```
Function age(birthdate As Date)
        age = Int((Now() - birthdate) / 365)
End Function
```

· (Now() - birthdate) ≡ the age in days

· Int(x) ≡ extracts the integer part of x

· age ≡ the age in integer numbers of years

55

► Declaration of constants

• Constants are variables which do not change their value during the execution of the program (UDF).

· Constants are used to keep the programming structure clear and to avoid tedious re-typing or time consuming re-calculations.

· You can declare constants

i) such that they are only available inside the program or

ii) such that they are available in the entire worksheet.

Syntax: i) **Const** name [**as** type] = value

ii) **Public Const** name [**as** type] = value

Function .....

It is important to do the **Public Const** statement before the **Function** statement.

· Expl. a) Const Pihalf = 1.570796327

b) Const Errmess as string = "Division by zero!!!"

c) Public Const Errmess as string = "Division by zero!!!"

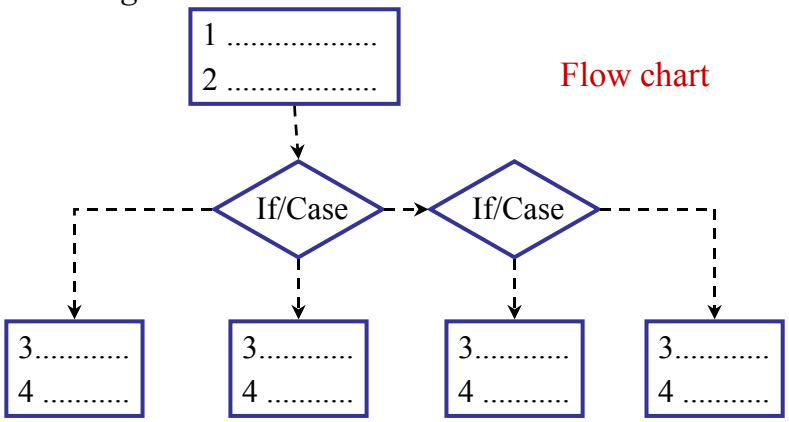
56

Program Structures

- So far we have only seen *sequential structures* (line by line)

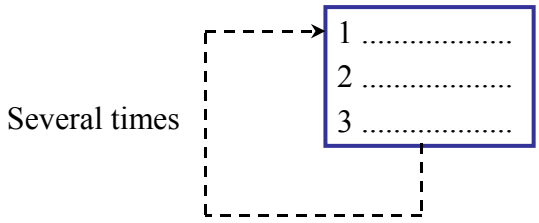
1 .....
2 .....
3 .....

- You can change this way of execution by *control structures*
  - *branching* or *decision structures*

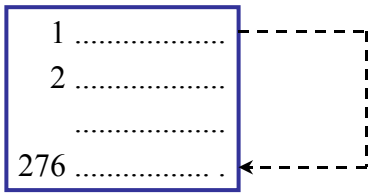


57

  - *looping (repetition structures)*



  - *controlled GOTO*



- It is useful to draw flow charts in order to keep track of the logic of the program structure. You do not need to write all comments in detail, but it suffices to write general statements in words.

58

▸ The IF-structure

• The IF-structure allows you to change the flow of your program depending on various conditions. The logic of this structure is very similar to the discussed Excel built-in IF-function.

Syntax1: **If** condition **Then**
[statements]
[**ElseIf** condition´ **Then**
[elseifstatements]]... } can be repeated many times
[**Else**
[elsestatements]]
End If

- condition(´) ≡ expressions which are true of false
- statements ≡ valid VBA commands
- elseifstatements ≡ executed when condition´ is true
- elsestatements ≡ executed when no previous condition is true

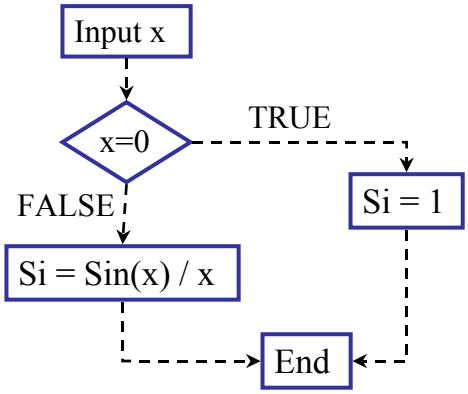
59

- Examples:

a) Write a UDF which produces the function

$$\mathrm{Si}(x) = \begin{cases} \frac{\sin x}{x} & \text{for } x \in \mathbb{R}\setminus 0 \\ 1 & \text{for } x = 0 \end{cases}$$

```
Function Si(x)
    If x = 0 Then
        Si = 1
    Else
        Si = Sin(x) / x
    End If
End Function
```



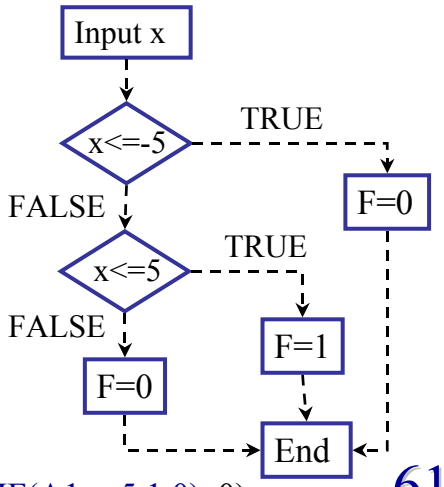
· Recall from Lab 1 Task 3 that this function also can be produced by using Excel built-in functions as
=IF(x=0,1,SIN(x)/x)

60

b) Write a UDF which produces the function

$$F(x) = \begin{cases} 0 & \text{for } x \le -5 \\ 1 & \text{for } -5 < x \le 5 \\ 0 & \text{for } x > 5 \end{cases}$$

```
Function F(x)
    If x <= -5 Then
       F = 0
    ElseIf x <= 5 Then
       F = 1
    Else
       F = 0
    End If
End Function
```



· Recall page 32: =IF(A1>-5, IF(A1<=5,1,0) ,0)

61

c) Write a UDF which determines whether a certain date falls on a weekend or not!

```
Function WE(x As Date) As String
    Dim temp As Integer
    temp = Weekday(x)
    If temp = 1 Or temp = 7 Then
       WE = "That day falls on a weekend."
    Else
       WE = "That day is a weekday."
    End If
End Function
```

· Format the cell A1 as date and enter 01/11/2005 or Now()

· "=WE(A1)" → That day is a weekday.

· Note that we declared all variable types.

62

Syntax2: **If** condition **Then** [statement1] : [statement2]: ...

- Just one line! The VBA statements are carried out when the condition is TRUE. Several statements are separated by ":".
- Expl.: The function F(x) can also be produced by:

```
Function Ftwo(x as single) as integer
    Ftwo = 1
    If x <= -5 Then Ftwo = 0
    If x > 5 Then Ftwo = 0
End Function
```

Syntax3: **IIf** (*condition*, value for true, value for false)

- Same syntax as for built-in functions with IF → IIF
- Expl.: The function in Expl. a) can also be produced by:

```
Function Fthree(x)
    Fthree = IIf(x = 0, 1, Sin(x) / x)
End Function
```

63

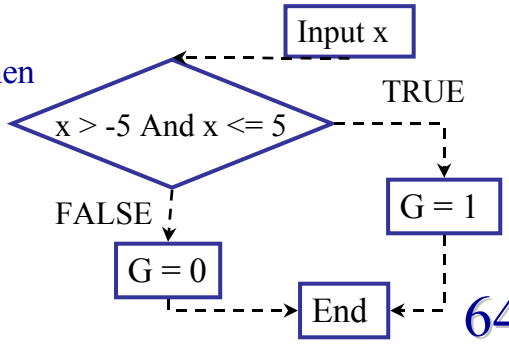
▸ Boolean operators

- Just as for built-in functions one can use boolean operators to create more complex conditions.

  Syntax: condition1 **And** condition2 **And** condition3
  condition1 **Or** condition2 **Or** condition3

- The logic is the same as for built-in functions.
- Expl.: The function F(x) can also be produced by

```
Function G(x)
  If x > -5 And x <= 5 Then
      G = 1
  Else
      G = 0
  End If
End
```

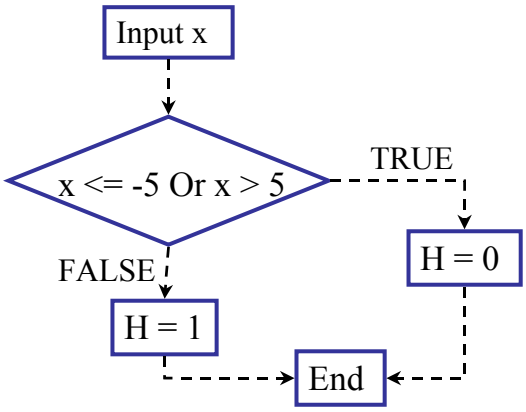


64

- We need to call it differently when it is on the same WS, e.g. G(x).

- Using "Or" F(x) can be produced by

```
Function H(x)
 If x <= -5 Or x > 5 Then
       H = 0
 Else
       H = 1
 End If
End
```



- You can also use "Not" and produce the same logical structures as with built-in functions, e.g.

```
Function Fnot(x)
      Fnot = IIf(Not (x <= -5 Or x > 5), 1, 0)
End Function
```

65

➤➤➤ The progress test is between:

2nd-13th of January 2006

Find out in time the room where you have to go!

- The test is open book, that means you can take all your notes, Lab-sheets and solutions.
- You will have a computer at your disposal, which you can use to verify and develop your answer.
- You have to write down your answer into an answer booklet which will be provided to you.
- There are two of these tests (exams) each one counting 50% towards your final mark for this course module.
- The questions will be similar in style to the ones of the Lab-sessions.

66

▸ An example tutorial (involving UDF, IF-structures, VLOOKUP):

a) Write a user-defined function which computes the body mass index according to the formula

body mass index = weight in kilograms /(height in meters)^2.

Declare all your variables. Write two types of functions one using the function ROUND giving an answer to a precision of one digit and one returning an integer value.

- How to start? Think first about the general structure.

- How many and which input variables do you need?
  Two: weight and height

- Program first the rough structure: (Do not progam all at once)

```
Function bmi(weight, height)
    bmi = weight / (height) ^ 2
End Function
```

67

- Test the rough structure on the Excel sheet:
  =bmi(70, 1.71) → 23.93898977

· What if nothing happens or something strange?
  Check if you typed in the correct place, i.e. the module.
  Check your spelling and other possible typos, e.g.

```
Function bmi(weight, height)
    bm = weight / (height) ^ 2
End Function
```

=bmi(70, 1.71) → 0

```
Function bmi(w, h)
    bmi = v / (h) ^ 2
End Function
```

=bmi(70, 1.71) → 0

```
Function bmi(weight height)
    bmi = weight / (height) ^ 2
End Function
```

=bmi(70, 1.71) → crash

68

- Implement the other tasks:
  · Declare the variables:
    weight and height are of type Single
    bmi is of type Double when working with ROUND
    bmi is of type Integer when working to integer precision

```
Function bmi(weight as Single, height as Single) as Single
    bmi = weight / (height) ^ 2
End Function
```

Test your function
=bmi(70, 1.71) → 23.93898964
there is a small difference in the last two digits 77 → 64

```
Function bmi(weight as Single, height as Single) as Double
    bmi = Round( weight / (height) ^ 2 ,1)
End Function
```

=bmi(70, 1.71) → 23.9 =bmi(70, 1.71) → 23.9

69

- Now integer precision:

```
Function bmi(weight as Single, height as Single) as Integer
    bmi = Round( weight / (height) ^ 2 )
End Function
```

or

```
Function bmi(weight as Single, height as Single) as Integer
    bmi = weight / (height) ^ 2
End Function
```

=bmi(70, 1.71) → 24

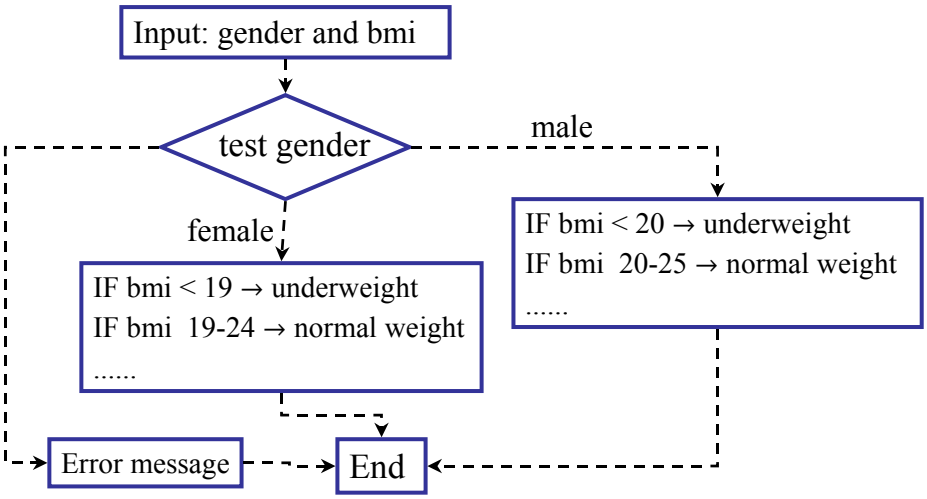
- Test your function with some more values to make sure that the answer was not accidental.
- Try to judge whether the output makes sense at all. Do you expect very small numbers 0.1, 0.0001 or very large numbers 653542.2? This information is not given yet.

70

b) Write a user-defined function which gives a meaningful interpretation for the body mass index according to the table:

male	female	
<20	<19	underweight
20-24.9	19-23.9	normal weight
25-29.9	24-28.9	overweight
30-39.9	29-38.9	obese
≥40	≥ 39	extreme obese

- How many and which input variables do you need?
  Two: the gender and the body mass index
- What should be the output?
  The meaning of the bmi, that is an entry from the last column of the table depending on the value of bmi.
- Think first about the general outline, draw a flow chart. 71



- You don´t have to be more specific.
- Start again with the rough structure.
- Program first the outer IF-structure. 72

```
Function bmimean(bmin, mf)
   If mf = "male" Then
      If (bmin < 20) Then
         bmimean = "underweight"
      ElseIf (bmin >= 20 And bmin < 25) Then
         bmimean = "normal weight"
      .....
      Else
         bmimean = "extreme obese"
      End If
   ElseIf mf = "female" Then
      If (bmin < 19) Then
         bmimean = "underweight"
      ElseIf (bmin >= 19 And bmin < 24) Then
         bmimean = "normal weight"
      .....
      Else
         bmimean = "extreme obese"
      End If
   Else
      bmimean = "Specify gender!"
   End If
End Function
```

73

- Test your function:

=bmimean(19.5, "male") → underweight

=bmimean(19.5, "female") → normal weight

=bmimean(19.5, "e-mail") → Specify gender!

There are 11 cases to be tested.

c) Use a VLOOKUP table to produce the same function as in b).

Enter the following table into the Excel WS:

	A	B	C	D	E
1		male	female		
2		0	0	underweight	
3		20	19	normal weight	
4		25	24	overweight	
5		30	29	obese	
6		40	39	extreme obese	

On the WS we could produce for instance:

=VLOOKUP(23,B2:D6,3) → normal weight

74

d) The ideal body mass index is 21 and 22 for female and male, respectively. Given the height of a person in meters and the gender write a UDF which computes the ideal weight in kilograms to a precision of one digit. Declare all your variables.

```
Function Idealweight(height As Single, mf As String) As Double
    If mf = "male" Then
        Idealweight = Round(22 * height ^ 2, 1)
    ElseIf mf = "female" Then
        Idealweight = Round(21 * height ^ 2, 1)
    Else
        Idealweight = "Specify gender!"
    End If
End Function
```

Formula:
- BMI= w/h^2
- BMI = 21 (22)
⇒ w = 21(22) h^2

75

- keep the "outer" If-structure

```
Function bmitab(bmin, mf)
    If mf = "male" Then
        bmitab = WorksheetFunction.VLookup(bmin,[b2:d6], 3)
    ElseIf mf = "female" Then
        bmitab = WorksheetFunction.VLookup(bmin, [c2:d6], 2)
    Else
        bmitab = "Specify gender!"
    End If
End Function
```

· Note the change of the range for the two tables.

· Note that ranges in VBA are of the format [c2:d6].
Using c2:d6 or (c2:d6), as possible on the WS, will not work.

76

➤➤➤ The progress test will be:

11-th of January 2006 14:30-16:00

Find out in time the room where you have to go!

- The test is open book, that means you can take all your notes, Lab-sheets and solutions.
- You will have a computer at your disposal, which you can use to verify and develop your answer.
- You have to write down your answer into an answer booklet which will be provided to you.
- There are two of these tests (exams) each one counting 50% towards your final mark for this course module.
- The questions will be similar in style to the ones of the Lab-sessions.

77

e) Produce a table which labels columns by heights from 1.55m to 1.95m in steps of 5 cm and rows by weights from 50kg to 95kg in steps of 5kg. At each intersection compute the corresponding body mass index. Write then a UDF which uses this table as a Vlookup table to determine the body mass index from a given height and weight.

- The table should look like:

(Use the autofill function to produce it. Only type row 5.)

	A	B	C	D	E	F	G	H	I	J
2										
3										
4		1.55	1.6	1.65	1.7	1.75	1.8	1.85	1.9	1.95
5	50	20.8	19.5	18.4	17.3	16.3	15.4	14.6	13.9	13.1
6	55	22.9	21.5	20.2	19	18	17	16.1	15.2	14.5
7	60	25	23.4	22	20.8	19.6	18.5	17.5	16.6	15.8
8	65	27.1	25.4	23.9	22.5	21.2	20.1	19	18	17.1
9	70	29.1	27.3	25.7	24.2	22.9	21.6	20.5	19.4	18.4
10	75	31.2	29.3	27.5	26	24.5	23.1	21.9	20.8	19.7
11	80	33.3	31.2	29.4	27.7	26.1	24.7	23.4	22.2	21
12	85	35.4	33.2	31.2	29.4	27.8	26.2	24.8	23.5	22.4
13	90	37.5	35.2	33.1	31.1	29.4	27.8	26.3	24.9	23.7
14	95	39.5	37.1	34.9	32.9	31	29.3	27.8	26.3	25

e.g. D6 contains =bmi($A6,$D$4)

78

```
Function bmitable(weight As Single, height As Single) As Single
   Dim x As Integer
   If height >= 1.55 Then x = 2
   If height >= 1.6 Then x = 3
   If height >= 1.65 Then x = 4
   If height >= 1.7 Then x = 5
   If height >= 1.75 Then x = 6
   If height >= 1.8 Then x = 7
   If height >= 1.85 Then x = 8
   If height >= 1.9 Then x = 9
   If height >= 1.95 Then x = 10
   bmitable = WorksheetFunction.VLookup(weight, [a5:j14], x)
End Function
```

This function gives wrong results for heights >2.00 and heights <1.55.

79

f) Produce two tables which label columns and rows in the same way as in e). At each intersection compute the meaning for the body mass index for male and female in the tables. Write then a UDF which uses either of these tables as a Vlookup table to determine the meaning of the body mass index from a gives height, weight and gender.

- The tables (part of them) should look like:

29	female	1.55	1.6	1.65	1.7	1.75	
30	50	normal weight	normal weight	underweight	underweight	underweight	u
31	55	normal weight	normal weight	normal weight	normal weight	underweight	u
32	60	overweight	normal weight	normal weight	normal weight	normal weight	u
33	65	overweight	overweight	normal weight	normal weight	normal weight	n
34	70	obese	overweight	overweight	overweight	normal weight	n
35	75	obese	obese	overweight	overweight	overweight	n
36	80	obese	obese	obese	overweight	overweight	o

You can either compute the body mass index or use the table from e) to look up the values.

In the latter case D32 contains =bmimean($D7,"female")

80

```
Function BT(weight As Single, height As Single, mf As String) As String
   Dim x As Integer
   If height >= 1.55 Then x = 2
   If height >= 1.6 Then x = 3
   If height >= 1.65 Then x = 4
   If height >= 1.7 Then x = 5
   .....
   If height >= 1.95 Then x = 10
   If mf = "female" Then
      BT = WorksheetFunction.VLookup(weight, [a30:j39], x)
   ElseIf mf = "male" Then
      BT = WorksheetFunction.VLookup(weight, [a18:j27], x)
   Else
      BT = "Specify gender!"
  End If
End Function
```

Decide here which table to take:

81

▸ The SELECT CASE-structure

• The SELECT CASE structure is another branching structure provided by VBA. It is a more elegant and transparent version of an IF-structure, which tests always the same variable.

For instance (determine the sign of a number):

```
Function sig(x As Single) As String
    If x > 0 Then
      sig = "positive"
    ElseIf x < 0 Then
      sig = "negative"
    Else
      sig = "zero"
    End If
End Function
```

=

```
Select Case x
    Case Is > 0: sig = "positive"
    Case Is < 0: sig = "negative"
    Case Else:   sig = "zero"
End Select
```

82

Syntax: **Select Case** testvariable

[**Case** expressionlist
[statements]]... } can be repeated many times

[**Case Else**
[elsestatements]]

End Select

- testvariable ≡ a numeric or string expression
- expressionlist ≡ list of one or more expressions separated by a comma
  - · expression
  - · expression **To** expression
  - · **Is** comparisonoperator expression
- statements ≡ executed when one condition from expressionlist is true
- elsestatements ≡ executed when no previous condition is true

83

• Examples (Select case):

a) Function si(x)

```
Select Case x
  Case 0:      si = 1
  Case Else: si = Sin(x) / x
End Select
End Function
```

$$Si(x) = \begin{cases} \frac{\sin x}{x} & \text{for } x \in \mathbb{R}\backslash 0 \\ 1 & \text{for } x = 0 \end{cases}$$

b) Function F(x As Single) As Single

```
Select Case x
  Case Is < 0:    F = 0
  Case 0 To 4:   F = 3 * x
  Case Else:      F = 12
End Select
End Function
```

$$F(x) = \begin{cases} 0 & \text{for } x < 0 \\ 3x & \text{for } 0 \le x \le 4 \\ 12 & \text{for } x > 4 \end{cases}$$

· Note that "a To b" means "a ≤ x ≤ b"

84

c) Function G(x As Single) As Single
    Select Case x
      Case -4 To 4: G = 1
      Case Else: G = 0
    End Select
  End Function

$$G(x) = \begin{cases} 1 & \text{for } -4 \le x \le 4 \\ 0 & \text{otherwise} \end{cases}$$

· Note that "a To b" means "a ≤ x ≤ b"

d) Function entry(age As Integer) As Variant
    Select Case age
      Case 0 To 5, Is > 65: entry = 0
      Case 6 To 15: entry = 2
      Case 15 To 65: entry = 5
      Case Else: entry = "Age not valid!"
    End Select
  End Function

85

e) Function price(product As String) As Variant
    Select Case product
      Case "Mangoes": price = 2.5
      Case "Bananas": price = 1.8
      Case "Pears", "Apples": price = 0.9
      Case Else: price = "Fruit not in price list!"
    End Select
  End Function

· Note that the test variable can also be of string type
· Note that price is of type Variant, as it could be a number or a string
· Note that the test is case sensitive, e.g.
  =price("mangoes") → "Fruit not in price list!"
· Note that when the "Case Else" line is dropped
  =price("Papayas") → 0

86

f) Function pricec(product As String, country As String) As Variant

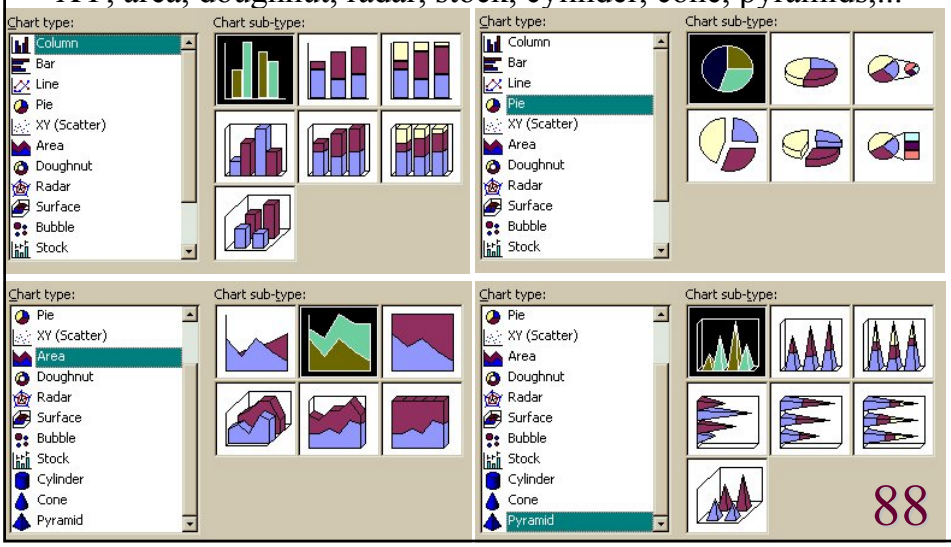
```
Select Case country
  Case "Brasil"
    Select Case product
      Case "Mangoes" , "Papayas":   pricec = 2.5
      Case "Bananas":   pricec = 1.3
      Case Else: pricec = "Fruit not in the list!"
    End Select
  Case "Thailand"
    Select Case product
      Case "Mangoes": pricec = 2.2
      Case "Papayas":   pricec = 2.8
      Case Else:  pricec = "Fruit not in the list!"
    End Select
  Case Else: pricec = "Country not the list!"
End Select
End Function
```

· One can also nest the SELECT structure similar to the IF-structure

87

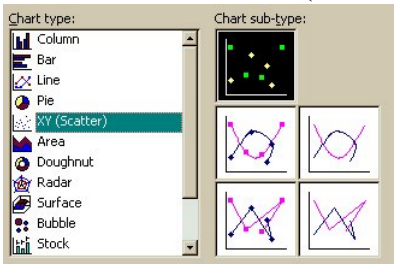
Charts (Graphs):

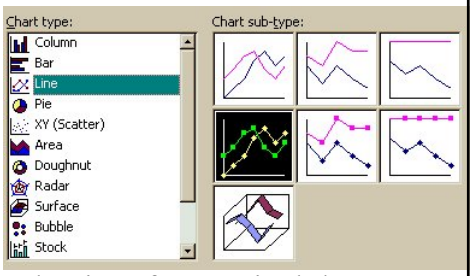
• Charts are ways to display data in a graphical way.
- Excel offers various types of charts, such as column, bar, pie, XY, area, doughnut, radar, stock, cylinder, cone, pyramids,...



88

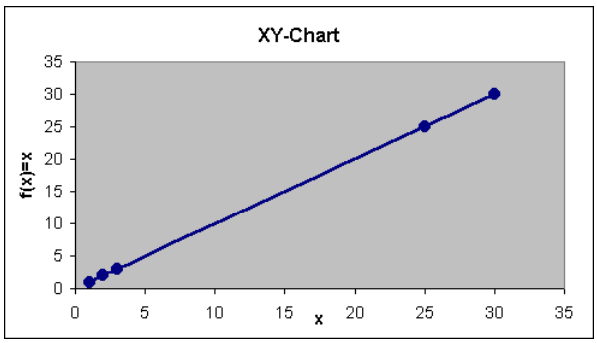
- Here we want to learn more about the most common types:
  XY-charts (scatter) and line charts.

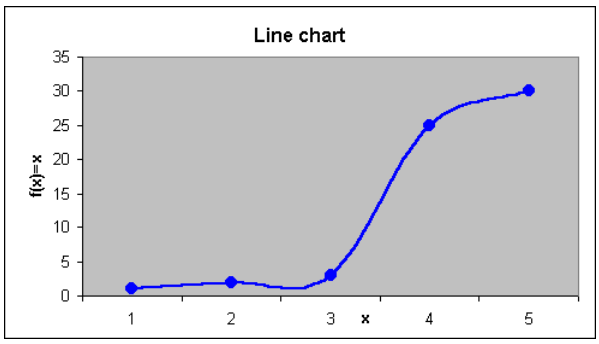




- XY charts are used to plot ordered pairs of numerical data, e.g. from a scientific experiment, mathematical functions, etc.
- Line charts are used when the x-values are textual, e.g. month of the year, names of people or companies, places, etc.
· These two types of charts should not be confused with each other, as their display is quite different, which is not suggested by their names
· Example:

89





We plot the data:

x	f(x)
1	1
2	2
3	3
25	25
30	30

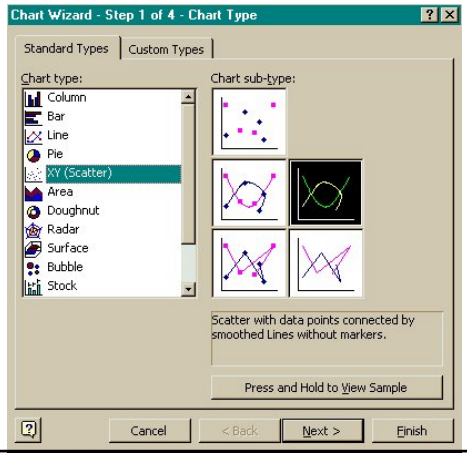
1) Creating an XY/line chart:

i) open a worksheet

ii) select the data you wish to display, e.g. cells A1:B30

· in particular we want to see here how to plot a function f(x), e.g. the x are in A1:A30 and the f(x) in B1:B30

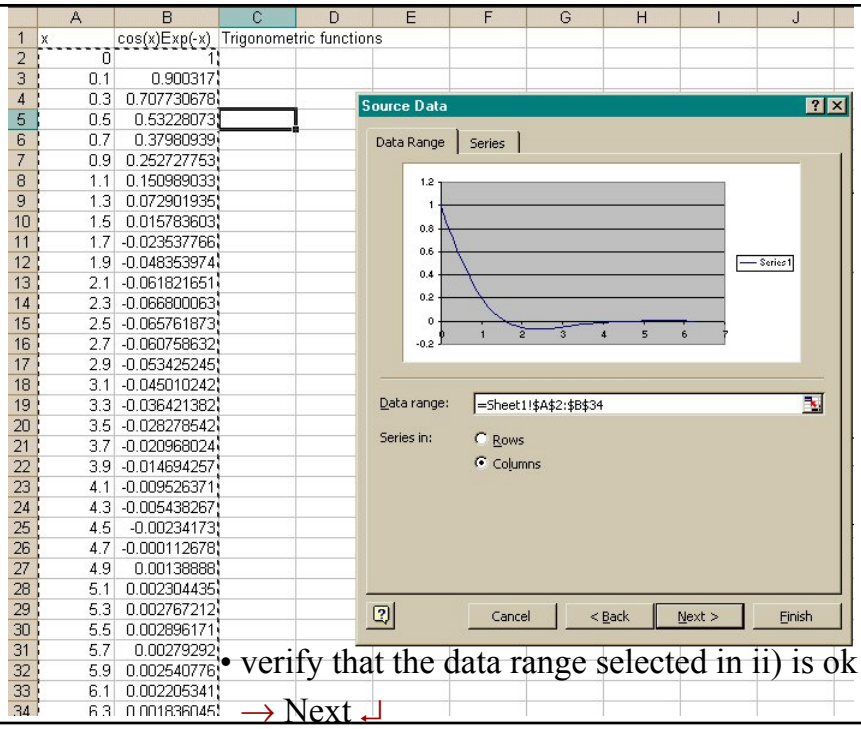
iii) open the chart wizard ⇒ a series of 4 dialog boxes open up



• specify the type and the sub-type of the chart

→ Next ↵

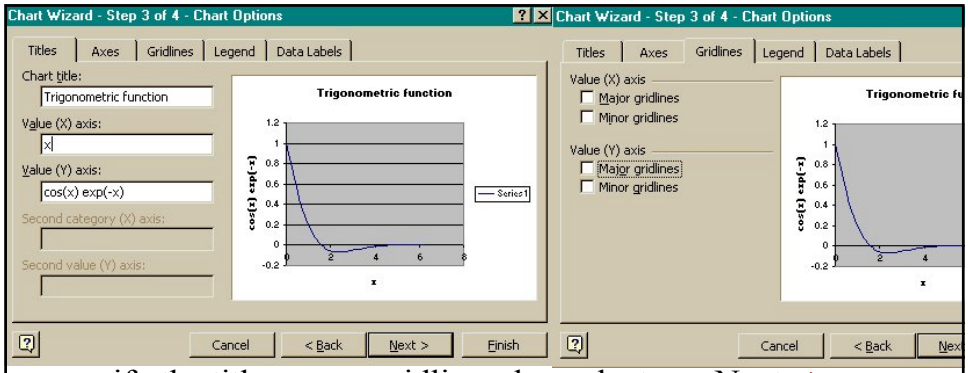
91



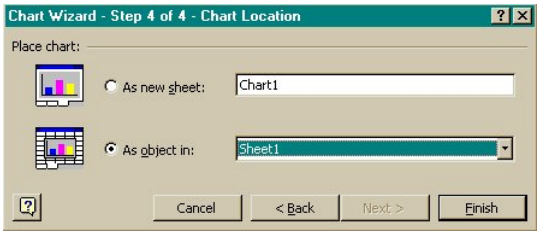
• verify that the data range selected in ii) is ok

→ Next ↵

92



• specify the titles, axes, gridlines, legend, etc → Next ↵

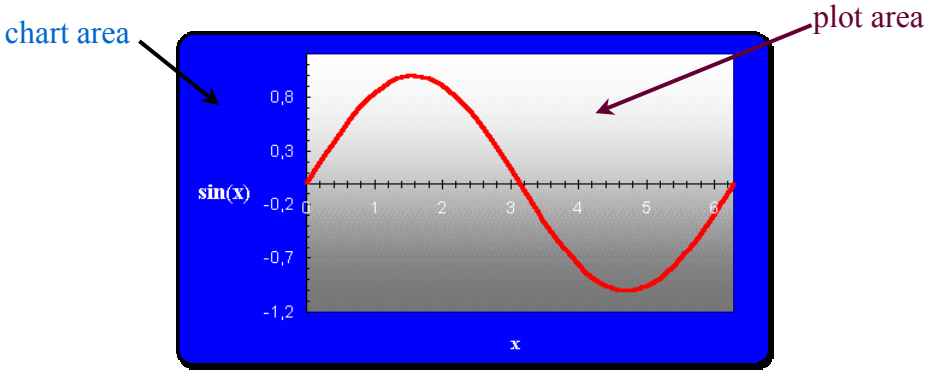


• specify the location where the chart should stored→ Finish ↵

⇒ a chart will appear in the location you specified

93

• For instance, if in some column (row) we had had some (densely enough) distributed x-values and in some other column (row) the corresponding values sin(x), we could have produced



• Most likely the design would not have been of this type, therefore ➔

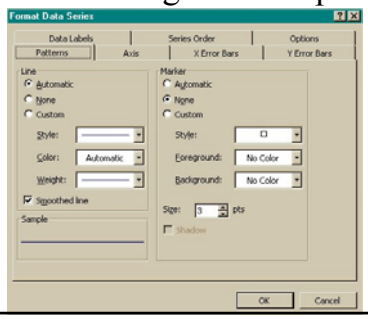
94

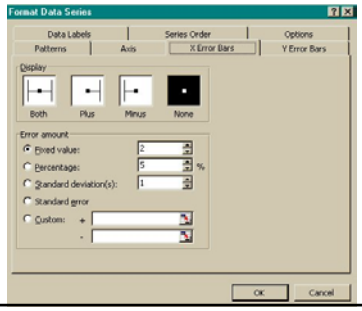
2) Modifying a chart:

- you can change the design of the presentation by selecting the objects you wish to modify

i) Formatting the plot area
- by default the plot area will be grey
- select the plot area ⇒ the "Format Plot Area" window opens
- use it to change the colours of the background, frame, etc.

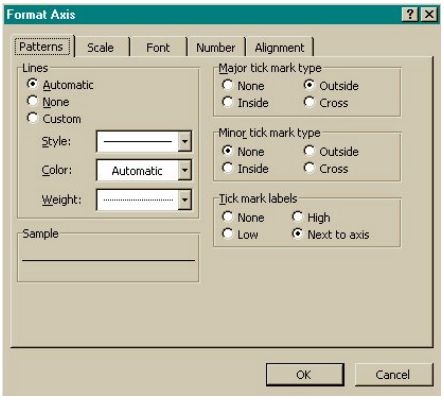
ii) Formatting the data series
- select the line ⇒ the "Format Data Series" window opens
- use it to change the line pattern, data labels, error bars etc.

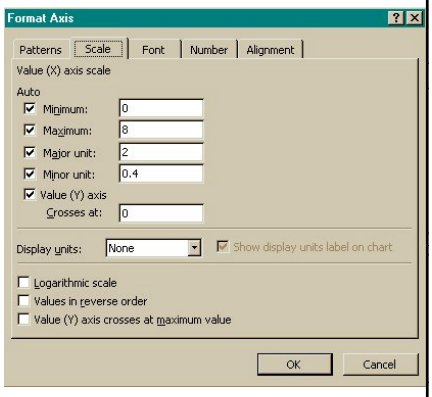




95

iii) Formatting the axis
- select the axis ⇒ the "Format Axis" window opens
- use it to change the axis pattern and scale





iv) Modifying the chart options
- right select the chart area ⇒ Chart Options ↵
- use it to change titles, axes properties, gridlines, legends and data labels

96

v) Dynamical titles and axis labels
- the data are already linked in a dynamical way to the chart, this means if you change them the plot will change as well
- you can also do this with the title and axis labels
  · type some text into a cell, e.g. "sin(x)" into F1
  · select the title or an axis label
  · type "=" into the Formular bar
  · select again the cell where you wrote the text, e.g. F1
  ⇒ in the Formular bar the location of your text appears, e.g. =Sheet1!F1
  · select the "✓" to complete the process
  ⇒ Now, whenever you update the selected cell, e.g. F1, the text inside the chart will change accordingly

vi) Changing the default setting
- you might have a preferred chart style and if you do not want to repeat the previous steps use this style as default
  · select the chart → Chart → Chart type ↵ Select as default ↵

97

3) Data input:
• There are various ways to fill in the cells with data:

i) You can fill in the data the pedestrian way by just typing them

ii) The data might be stored externally on some file resulting for instance as output from another program.
- Importing the data:
  · select a cell on your worksheet for the first value
  · select Data → Get External → Import Text File ↵
  ⇒ Text Import Wizard opens with a series of 3 dialog boxes
    · answer questions about data and file type
    · modify the field width
    · select the data format → Finish ↵
  · confirm the location where the data should be stored

iii) Use the fill function (recall page 17 and more see lecture on Macros in part II)

iv) Use a VBA program to fill in the data

98

```
Sub fill()
Const pi = 2 * 3.1415
Range("a1").Value = 0 * pi
Range("a2").Value = 0.1 * pi
Range("a3").Value = 0.2 * pi
      ……..
Range("a10").Value = 0.9 * pi
Range("a11").Value = pi
Range("b1").Value = f(Range("a1").Value)
Range("b2").Value = f(Range("a2").Value)
      ……
Range("b10").Value = f(Range("a10").Value)
Range("b11").Value = f(Range("a11").Value)
End Sub
```

```
Function f(x)
   f = Exp(-x) * Cos(x)
End Function
```

99

v) Use the autofill function (recall from page 17)

- The autofill function determines automatically the entries of some cells given some starting values:

· fill in some starting values
  e.g. 0 →A1, 0.1 →A2, =SIN(A1) →B1, =SIN(A2) →B2

· select the area of the starting values, e.g. A1:B2

· while you are on top of the selected area the cursor will be ✚

· move the cursor to the lower right corner of the selection, until the cursor changes from ✚ to +

· drag the fill handle down (or to the right) and the new cells will be filled based on the initial selection, e.g. 0.2 →A3, 0.3 →A4, =SIN(A3) →B3, =SIN(A4) →B4, etc.

· verify that Excel really filled in the sequence you wanted!!!

⇒ In our example we have now two column of data, which we can plot against each other in the chart

100

4) Data handling:

- Adding data to an existing chart:
  - plot area → Source data → Series → add → X/Y values, name → Ok ↵
- Data → sort ≡ arrange selected data alphabetically, by data or numerically in ascending or descending order
- Data → filter ≡ allows to filter out certain data based on their location
- Data → validation ≡ allows to filter certain data based on a criterion you define, e.g. a certain range
- Data → subtotals ≡ computes totals and subtotals for selected columns and inserts them into the sheet
- Data → text to columns ≡ allows to change the data type

101

There is a small change in style of this years progress test:

- This years progress test has four questions. Three of them correspond to full marks.
- You have 90 min to answer the questions.

102

▸ Examples, examples, examples:

1a) Compute the future value of an investment using the Excel built-in function FV. For an initial deposit of 2500 pounds in a savings account the bank pays an interest rate of 0.18%. For the next years you deposit 150 pounds at the beginning (end) of every month into the account. How much money is in the account after 5 years. Provide the exact command line for an Excel built-in function with all its arguments.

=FV(0.18%,60,-150,-2500,1) → £12,296.91 beginning
=FV(0.18%,60,-150,-2500,0) → £12,279.82 end

- do not forget the %-sign (or write 0.0018)
- even though you pay in you need to write -150 and -2500
- the 60 corresponds to 60 month from 5 years times 12 month

103

1b) Write down the command line for an Excel built-in function which produces the function

$$f(x) = \begin{cases} \sin x - 1/8 & \text{for } x \le 0 \\ x^3 - 7x & \text{for } x > 0 \end{cases}$$

Use your function to complete the table

	A	B	C	D	E	F
1	x	-12.24	-1.2	0	1.25	5.2
2	f(x)					

- The function can be produced with

=IF(x<=0, SIN(x)-1/8, x^3 -7 *x)

- Just type into the cell B2 and then use the autofill function

	A	B	C
1	x	-12.24	-1.2
2	f(x)	=IF(B1<=0, SIN(B1)-1/8, B1^3 -7 *B1)	=IF(C1<=0, SIN(C1)-1/8, C1^3 -7 *C1)

	A	B	C	D	E	F
1	x	-12.24	-1.2	0	1.25	5.2
2	f(x)	0.19561	-1.05704	-0.125	-6.797	104.208

104

1c) Write down the functions which are produced by the following combinations of Excel built-in functions

i) =IF(x<=0, SIN(x)-1/8, x^3 -7 *x)

ii) =IF(Not(AND(x<>1, x<>-2)),"infinity",1/(x-1)/(x+2))

i)

$$f(x) = \begin{cases} \sin x - 1/8 & \text{for } x \le 0 \\ x^3 - 7x & \text{for } x > 0 \end{cases}$$

ii)

$$f(x) = \begin{cases} \frac{1}{(x-1)(x+2)} & \text{for } x \neq 1,-2 \\ \text{infinity} & \text{for } x = 1,-2 \end{cases}$$

105

2) Write a user defined function with the name MinAv, which for an arbitrary number of input variables computes the minimum, the maximum, the average of the input and the sum of these three numbers. When the average plus 7 is smaller or equal than the sum, the function should return the sum and otherwise the average. Declare all your variables. Implement your function on an Excel spreadsheet to complete the following tables:

	A	B	C	D
1	x	y	z	MinAv
2	4	34	-11	
3	111	-5	12	
4	-1	-4	5	

	A	B	C	D	E
1	x	y	z	w	MinAv
2	4	34	-11	54	
3	34	-5	12	3	
4	-3	-4	5	2	

- As the number of input variables is arbitrary you have to call the function as MinAv(range) rather than MinAv(x,y,z)

106

```
Function MinAv(range) As Single
  Dim MA, MI, AV, SU As Integer
  MA  = WorksheetFunction.Max(range)
  MI   = WorksheetFunction.Min(range)
  AV   = WorksheetFunction.Average(range)
  SU = MA + MI + AV
  If AV + 7 <= SU Then
    MinAv = SU
  Else
    MinAv = AV
  End If
End Function
```

	A	B	C	D	E
1	x	y	z	w	MinAv
2	4	34	-11	54	63
3	34	-5	12	3	40
4	-3	-4	5	2	0

- Row 2: MI=-11, MA=34, AV=9, SU=32
- Row 3: MI=-5, MA=111, AV=39, SU=145
- Row 4: MI=-4, MA=5, AV=0, SU=1

	A	B	C	D
1	x	y	z	MinAv
2	4	34	-11	32
3	111	-5	12	145
4	-1	-4	5	0

107

3) Write a user defined function with the name MinMax, which for an arbitrary number of input variables computes the minimum and the maximum. When the minimum is negative the function should return the minimum plus 10 and otherwise the maximum. Declare all your variables.
Implement your function on an Excel spreadsheet to complete the following tables:

	A	B	C	D
1	x	y	z	MinMax
2	3	-5	2	
3	2	7	44	
4	0	4	66	

- You can compute this table easily not even writing the function

	A	B	C	D
1	x	y	z	MinMax
2	3	-5	2	5
3	2	7	44	44
4	0	4	66	66

108

```
Function MinMax(range) As Integer
  Dim x, y as Integer
  x = WorksheetFunction.Max(range)
  y = WorksheetFunction.Min(range)
  If y < 0 Then
    MinMax = y + 10
  Else
    MinMax = x
  End If
End Function
```

4) For the table below complete the command line and the output. Then write a function which uses the select case structure and choses for a country by means of an HLOOKUP table the capital, the number of inhabitants, the area or the birthrate depending on whether the second input parameter is “Capital“, “Inhabitants“, “Area“ or “Birth rate“. Declare all variables! 109

	A	B	C	D	E	F
1	**Country**	UK	Germany	India	China	Brasil
2	**Capital**	London	Berlin	New Delhi	Beijing	Brasilia
3	**Inhabitants**	60.3 Mio	82.4 Mio	1065 Mio	1298 Mio	184 Mio
4	**Area/km^2**	244820	357021	3287590	9595960	8511965
5	**Birth rate**	10.88	8.45	22.8	12.98	17.25
6						

=HLOOKUP("UK",A1:F5, *,FALSE) → 60.3 Mio * = 3

=HLOOKUP("New Delhi", * ,4,FALSE) → 22.8 * = A2:F5

=HLOOKUP("Great Britain", A1:F6,2) → * * = Brasilia

=HLOOKUP("Great Britain", A1:F6,2,False) → * * = #N/A

=HLOOKUP("1298 Mio", *) → 12.98 * = E3:F5,3

=VLOOKUP("Capital", A1:F5,3,FALSE) → * * = Berlin

=VLOOKUP(357021, *) → 9595960 * = C1:F5,3

=HLOOKUP(5000000, A4:F5,2)→ * * = 22.8

=HLOOKUP(*, A1:F5,3,FALSE) → 1065 Mio * = "India"

=HLOOKUP("London", A2:F6,*,FALSE)→ 10.88 * = 4 110

```
Function Cof(Co As String, command As String) As Variant
Select Case command
Case "Capital": Cof = WorksheetFunction.HLookup(Co, [A1:F5], 2, False)
Case "Inhabitants": Cof = WorksheetFunction.HLookup(Co, [A1:F5], 3, False)
Case "Area": Cof = WorksheetFunction.HLookup(Co, [A1:F5], 4, False)
Case "Birth Rate": Cof = WorksheetFunction.HLookup(Co, [A1:F5], 5, False)
Case Else: Cof = "Command not found"
End Select
End Function
```

111