- 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 .....

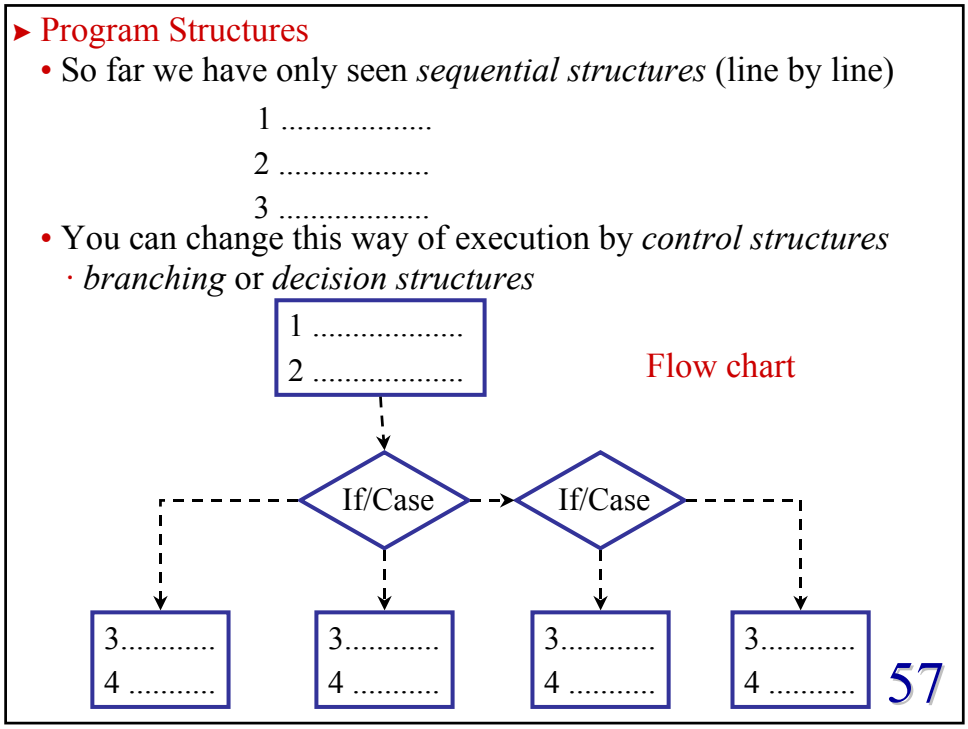
It is important to do the **Public Const** statement before the **Function** statement.

· 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
1 .....
2 .....
Flow chart
If/Case
If/Case
3.....
4 .....
3.....
4 .....
3.....
4 .....
3.....
4 .....
57