INTRODUCTION TO VBA PROGRAMMING

LESSON3

dario.bonino@polito.it

Agenda

- Language Basics
  - Comments
  - Variables
  - Datatypes
  - Operators
  - Constants
  - Math Functions

Language Basics

The basic syntax of VBA

Comments

- Every program must be
  - Well structured
    - Address each sub-problem in an easy to spot and specific program part
  - Well commented
    - Allow others to easily understand and/or modify the program code
- Comments
  - Begins with the character ‘
    - `‘ this is a comment`
  - Can be on the same line of the instructions
    - `MsgBox(“hey!”) ‘ this is a comment`

Comments

```
Private Sub CommandButton1_Click()
  ‘ ask the first number
  x = InputBox("Insert the first number, please...")
  ‘ ask the second number
  y = InputBox("Insert the first number, please...")
  ‘ compute the difference
  result = x – y
  ‘ show the result
  MsgBox ("The result of " & x & "-" & y & " is " &
  result)
  End
End Sub
```


Variables

- Containers for data
  - (Wikipedia def.)
- Variable names
  - Case-insensitive (upper and lower case letters are the same)
    - Sample == sAmPLe == SAMPLE
  - Must begin with a letter
  - Can contain letters, digits and the “_ “ sign
  - Example: myVariable, Variable1, HELLO_1

Variables

- Variable names (continued…)
  - Should be long and meaningful
    - To easily remember what they are meant for
    - To keep the program code understandable
    - To allow easier documentation
- Variable have a Type
  - Type indicates what kind of data is contained by the variable
  - May be implicit or explicit (better)

Datatypes

- Visual Basic for Applications defines many datatypes
  - Numeric
  - Alphanumeric
  - Boolean
  - Others…
- Variable types are defined through the `Dim-As` expression
  - `Dim` *`variable-name`* `as` *`Type`*
    - *`Dim x As Integer`*
    - *`Dim y As String`*
    - *`Dim z As Boolean`*

Numeric Types

- Designed for holding numeric values
- Can be
  - **Integers**
    - Represent signed integer numbers on 16 bits
    - Values range from -32768 and + 32767
    - Numbers greater than 32767 or smaller than -32768 cannot be represented (overflow)
  - **Long** integers
    - Represent signed integer numbers on 32 bits
    - Values range from -2147483648 to 2147483647
    - Overflow can occur but with much bigger numbers

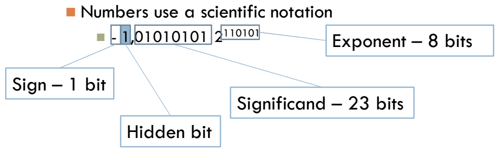
Overflow

- Try this program

```
Sub overflow()
    Dim x As Integer ' set x as Integer (16bit)
    x = 32767 ' assign x the maximum Integer value
    MsgBox ("x is" & x) ' show x
    x = x + 1 ' add 1 to x (out of the range)
    ' never reached, overflow occurs!
    MsgBox ("x is" & x)
End Sub
```


Numeric Types

- Can be (continued...)
  - Floating point (**Single** precision)
    - Represent real numbers on 32 bits
    - Numbers use a scientific notation



    - Range from (+/-) $1 \cdot 10^{-45}$ to $3.4 \cdot 10^{38}$

Numeric Types

- Can be (continued...)
  - Floating point (**Double** precision)
    - Represent real number on 64 bits
      - Significand 52 bits
      - Exponent 11 bits
      - Sign 1 bit
    - Range from (+/-) $4.9 \cdot 10^{-324}$ to $1.7 \cdot 10^{308}$

Numeric Types

- Integer vs Long vs Single vs Double
  - Floating point operations are slower than Integer operations
  - Floating point numbers require more memory than integers
  - Integers cannot be used when real numbers are needed
- In conclusion
  - Choose always the most suited datatype depending on the problem you have to solve

Boolean and String types

- Boolean
  - Represent numbers that can only assume two values
  - E.g. Logical truth values
  - Allowed values: true, false
- Strings (next lesson)
  - Hold alphanumeric values
    - E.g. "1,2 3, ... Hello World!"

Other types

- Variant
  - Special, hybrid, type
  - Automatically assigned when the type of a variable is not specified
  - Can hold Integers, Real numbers, Strings, etc.
  - Does not behave as if the variable was explicitly typed
    - Neither resembling a number nor a string
    - Try to change the – to + in our simple calculator example

Working with numbers

- Numerical expressions
  - myVar = x + y – z * 25 / var7

Variable

Operator

Operators

- VBA provides many operators for working with numbers
  - + → sum
  - - → subtraction
  - * → multiplication
  - / → division
  - \ → integer division
  - Mod → remainder of a integer division
  - ^ → power
  - = → assignment

Example

- We want to write a program that, given a certain amount of seconds, computes the corresponding number of minutes and hours
  - nSeconds = 5275
  - nHours = ?
    - → compute the integer division of the number of second by 3600 (seconds in 1 hour)
    - `nHours = nSeconds \ 3600`
  - nMinutes = ?
    - → compute the integer division of the hour remainder by 60 (seconds in 1 minute)
    - `nMinutes = (nSeconds Mod 3600)\60`

Example - solution

```
Sub operators()
  Dim nSeconds As Integer
  Dim nHours As Integer
  Dim nMinutes As Integer
  'get the number of seconds
  nSeconds = InputBox("Insert the amount of seconds to convert")
  'compute the hours
  nHours = nSeconds \ 3600
  'compute the minutes
  nMinutes = (nSeconds Mod 3600) \ 60
  'compute remaining seconds
  nSeconds = (nSeconds Mod 3600) Mod 60
  'show the result
  MsgBox (nHours & ":" & nMinutes & ":" & nSeconds)
End Sub
```


Operator precedence rule

- Whenever combined together in a numeric expression, operators have different precedence
- In VBA operator precedence almost reflects the standard Mathematical precedence rule
  - Parentheses
  - Power
  - Multiplication and division
  - Integer division
  - Remainder
  - Sum and subtraction
- Operators at the same level are executed side by side
  - A+B-C+D = (((A+B)-C)+D)

Examples

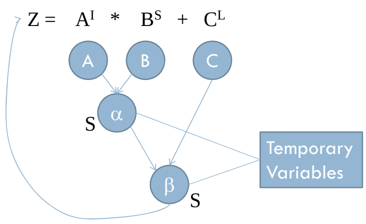
- `r = 2+3*4+3^2 = 2+3*4+9 = 2+12+9 = 23`
- `r = (2+3)*4+3^2 = 5*4+3^2 = 5*4+9 = 20+9 = 29`
- `r = 12 Mod 5 * 3 = 12 mod 15 = 12`
- `r = (12 Mod 5)*3 = 2*3 = 6`

Mixed Type operations

- What happens when different numeric types are involved in a single numeric expression?
  - Dim A as Integer
  - Dim B as Single
  - Dim C as Long
  - Z = A*B+C → which type will have Z?

Mixed Type operations

- Anatomy of an expression

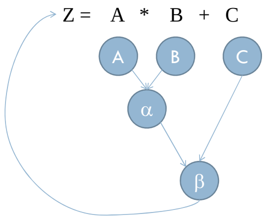


Mixed Type operations

- Rules:
  - The result of a computation between 2 values of a given Type has the same type
    - Integer + Integer = Integer
    - Long + Long = Long
  - The result of a computation between 2 values of different Type…
    - Depends…

Mixed Type operations

- Anatomy of an expression



Mixed Types

- The result of a computation between 2 values of different Type
  - On the right of the equal
    - If two values have different types, the smaller one is converted (promoted) temporarily to the larger type
  - On the left of the equal sign
    - The result of the operation is casted to the declared type
      - May generate errors
        - A Long result may be larger than an Integer
      - May introduce imprecision
        - A single result looses the fractional part when it is casted to an integer

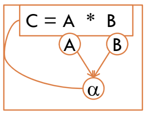
Example

```
Sub mixedTypes()
  Dim A As Integer
  Dim B As Single
  Dim C As Long
  Dim Z1 As Integer

  A = 10
  B = 12.5
  C = 1000000
Z1 = B
MsgBox ("Z1 = " & Z1)
Z = A * B + C
MsgBox ("Z = " & Z)
Z1 = A * B + C
MsgBox ("Z1 = " & Z1)
End Sub
```

Mixed types

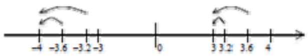
- Errors can also happen if variables have the same type
  - Dim A as Integer, Dim B as Integer, Dim C as Long
  - A=25677
  - B=20



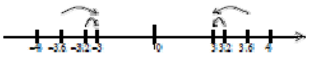
- In this case overflow occurs
  - Both A and B are integers → no conversion
  - A*B = α (Integer) >32767 →overflow (even if C can contain the result)

Type Conversion

- To avoid promotion and cast problems
  - Explicit conversion is recommended!!
- Type Conversion can be achieved through conversion functions
  - Int(A)



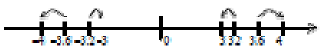
    - Truncates to the left side (floor)
  - Fix(A)



    - Truncates the fractional part

Type Conversion

- Conversion functions (continued...)
  - CInt(A) – rounds to the nearest integer



  - CLng(A) – converts to a Long (with the same semantics of Cint)
  - CSng(A) – converts to Single
  - CDbl(A) – converts to Double
- Implicit conversions use the Cxxx functions
  - CInt, CLng, CSng, CDbl

Exercise 1

- Write a program that asks the user to enter 4 integer values (Integer or Long), and then calculates and prints their average (the result must have the fractional part).

Exercise 2

- Write a program that asks for a temperature value (of an integer type) expressed in Fahrenheit degrees, and calculates and prints the corresponding values expressed in Celsius and Kelvin degrees (both with fractional part). [C=5/9*(F–32); K=C+273.15].

Math Functions

- VBA supports natively a set of common Math functions including
  - Sin(A) - sine of A (in radians)
  - Cos(A) - cosine of A (in radians)
  - Tan(A) - tangent of A (in radians)
  - Atn(A) - arc tangent of A (in radians)
  - Log(A) - natural logarithm of A
  - Log10(A) - common (base 10) logarithm of A
  - Exp(A) - e raised to A
  - Abs(A) - absolute value of A
  - Sqr(A) - square root of A
  - Sgn(A) - sign of A: –1 if negative, 0 if zero, +1 if positive

Constants

- Sometimes it may be useful to defined fixed values
  - If they have to be used in several computation
  - If they represent intrinsically constant values
    - π - PI number
    - G – Gravitation constant
    - …
- Can either be
  - Numeric
    - E.g. 1, 124, 32
  - Named
    - Use the keyword Const

Constants

- Examples
  - 12% → Integer numeric constant
  - 253& →Long numeric constant
  - 1.2345! → Single numeric constant
  - 1.2345# →Double numeric constant
  - Const PI As Single = 3.14.15 → Named numeric constant
    - No expressions allowed in this case!!
    - Const PI As Single = 4 * Atan(1) →WRONG!!

Exercise 3

- An object moving with speed v near to light speed c ($2.99793 \cdot 10^8$ m/s) shortens along the moving direction and gets heavier by a factor γ (less than 1). Write a program that asks for the length and the mass of a still object and calculates their *variation* at a speed requested from the user (in km/s). $\gamma = \sqrt{1 - \left(\frac{v}{c}\right)^2}$
- Suggestion

$$x' = \gamma x \rightarrow \Delta x = x - x' = x - \gamma x = x(1-\gamma)$$

$$m' = \frac{m}{\gamma} \rightarrow \Delta m = m' - m = m\left(\frac{1}{\gamma} - 1\right)$$

Exercise 4

- Write a program to calculate the shortest distance between two points on the surface of the Earth, given their geographic coordinates. The program requests the latitude and longitude values (in degrees) of the two points, and displays the distance between them. To compute the distance, use the following formula (remember that North and East coordinates are positive values, South and West negative, and that trigonometric functions use radians):

$$d = \arccos(p1 + p2 + p3) \cdot r$$

- where:
  - *p1* = cos(*lat1*)*cos(*lon1*)*cos(*lat2*)*cos(*lon2*)
  - *p2* = cos(*lat1*)*sin(*lon1*)*cos(*lat2*)*sin(*lon2*)
  - *p3* = sin(*lat1*)*sin(*lat2*)
  - *lat1* is the latitude in degrees of the first point
  - *lon1* is the longitude in degrees of the first point
  - *lat2* is the latitude in degrees of the second point
  - *lon2* is the longitude in degrees of the second point
  - *r* is the average Earth radius (6372.795 km or 3441.034 NM, this approximation results in an error of up to about 0.5%)
- The inverse cosine can be calculated by the following formula:

$$\arccos(x) = \arctan\left(\frac{-x}{\sqrt{1-x^2}}\right) + \frac{\pi}{2}$$

Exercise 4

- Calculate the distance between Turin International Airport (TRN, Italy, 45.02° N, 07.65° E)
- and
- Los Angeles International Airport (LAX, USA: 33.94 ° N, 118.40° W). [Answer: 9692.702 km or 5233.640 NM]