PROGRAMMING CONCEPTS

A programming code is made up of algorithms that are implemented within a programming language.

Syntax: the rules on how words are used within a given language

Data and Variables

Memory address: a specific location in memory where instructions or data are stored

Data are stored either as constants or as variables:

Constant: an item of data whose value does not change (VB: Const instead of Dim)

Variable: an item of data whose value could change while the program is being run

Debug: the process of finding and correcting errors in programs

Declaration: the process of defining a variables and constants in terms of their name and data type.

VB: Dim Name As String

Assignment: the process of giving a value to a variable or constant

VB: Name = "Gabriele"

Data type: determines what sort of data are being stored and how it will be handled by the program. Examples are: integer, float/real, character, pointer/reference, text/string, array, date/time and boolean.

Sequence, selection and iteration

- Sequence: principle of putting the correct instructions in the right order within a program
- Selection: principle of choosing what action to take based on some criteria
- Iteration: principle of repeating processes
    - *Definite iteration:* a process repeat a set number of times
    - *Indefinite iteration:* a process repeats until a certain condition is met

Subroutines

Subroutine: a named block of code designed to carry out a specific task. Even called procedure, subprogram or routine.

Module: a number of subroutines that form a part of a program

Top-down design: split a problem (and so the program) in subproblems (modules: modular design) and divide these in others subproblems (procedure / function)

Procedural language: is a characteristic of a programming language that allows to spit a program in subroutines.

Function: a subroutine that returns a value

There can be multiple definition of subroutines with the same name (overloading), but they cannot have the same parameters.

Parameters: data being passed to a subroutine → argument: actual value being passed to a subroutine. A parameter can be passed by reference or value (default value if not specified):

- Reference parameters: if changed inside a function remain changed, gives another name at the value
- Value parameters: are the copy of a variable therefore they don't change their value outside the function

Reference parameters can be used to return more than a value from a function. In the variable table if a variable is a value parameter it creates a new variable with a new memory address instead a reference creates a new variable with the same memory address of the first one.

A parameter can be *optional*, if so it must have a default value.

There are two types of variables:

- Local variables: can be accessed just inside a particular block of the program
- Global variables: can be accessed from the position where they are declared to the end of the program. They are declared outside any block of code.

If I have a global variable I can have another local variable with the same name, this will create holes in the scope: a block of code when the global variable can't be reference by name.

When a subroutine is called, the *stack frame* is stored (return addresses, parameters and local variables).

Exception handling: the process of dealing with events that cause the current subroutine to stop. The exception handling code (or catch block) is executed when the expected exception occurs.

Structured programming

Hierarchy chart: a diagram that shows the design of a system from the top down

Structure chart: similar to a hierarchy chart with the addition of showing how data are passed around the system.

Top-down approach: when designing systems it means that you start at the top of the process and work your way down into smaller and smaller sub-processes

System flowchart: a diagram using standard symbols that describes a process or system.