**- PROGRAMMING LANGUAGE:** Is a formal constructed language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behavior of a machine or to express algorithms.

**- ALGORITHM:** Is a self-contained step-by-step set of operations to be performed. Algorithms exist that perform calculation, data processing, and automated reasoning.

**- MACHINE LANGUAGES:** Are the only languages understood by computers. While easily understood by computers, machine languages are almost impossible for humans to use because they consist entirely of numbers.

**- LOW-LEVEL LANGUAGE:** A machine language or an assembly language. Low-level languages are closer to the hardware than are high-level programming languages, which are closer to human languages.

**-  HIGH- LEVEL LANGUAGE:** A programming language such as C, FORTRAN, or Pascal that enables a programmer to write programs that are more or less independent of a particular type of computer. Such languages are considered high-level because they are closer to human languages and further form machine languages. Ultimately, programs written in a high-level language must be translated into machine language by acompiler or interpreter. The first high-level programming languages were designed in the 1950s. Now there are dozens of different languages, including Ada, Algol, BSIC, COBOL, C, C++, FORTRAN, LISP, Pascal, and Prolog.

**- PSEUDOCODE:** An outline of a program, written in a form that can easily be converted into real programming statements.

**- CODE:** A set of symbols for representing something. For example, most computers use ASCII codes to represent characters. / Written computer instructions. The term code is somewhat colloquial. For example, a programmer might say: "I wrote a lot of code this morning" or "There's one piece of code that doesn't work". / Code can appear in a variety of forms. The code that a programmer writes is called source code. After it has been compiled, it is called object code. code that is ready to run is called executabe code or machine code.

**- COMPILER:** A [program](http://www.webopedia.com/TERM/P/program.html) that translates [*source code*](http://www.webopedia.com/TERM/S/source_code.html) into [*object code*](http://www.webopedia.com/TERM/O/object_code.html). The compiler derives its name from the way it works, looking at the entire piece of source code and collecting and reorganizing the [instructions](http://www.webopedia.com/TERM/I/instruction.html). Thus, a compiler differs from an [*interpreter*](http://www.webopedia.com/TERM/I/interpreter.html), which analyzes and [executes](http://www.webopedia.com/TERM/E/execute.html) each line of source code in succession, without looking at the entire program. The advantage of interpreters is that they can execute a program immediately. Compilers require some time before an executable program emerges. However, programs produced by compilers [run](http://www.webopedia.com/TERM/R/run.html) much faster than the same programs executed by an interpreter.

**- INPUT DEVICE:** Any machine that feeds data into a computer. For example, a keyboard is an input device, where as a display monitor is an output device.

**- OUTPUT DEVICE:** Any machine capable of representing information from a computer. This includes display screens, printers, plotters, and synthesizers.

**- CPU - CENTRAL PROCESSING UNIT:** CPU is the abbreviation for central processing unit (meaning the processor). The CPU is the brains of the computer where most calculations take place.

**- DATA TYPE:** In programming, classification of a particular type of information. It is easy for humans to distinguish between different types of data. We can usually tell at a glance whether a number is a percentage, a time, or an amount of money. We do this through special symbols --%; and $ -- that indicate the data's type. Similarly, a computer uses special internal codes to keep track of the different types of data it processes.

**- EXPRESSION:** In programming, an expression is any legal combination of symbols that represents a value. Each programming language and application has it own rules for what is legal and ilegal. For example, in the C language x+5 is an expression, as is the character string "MONKEYS".

**- OPERATOR:** A symbol that represents a specific action. For example, a plus sign (+) is an operator that represents addition.

**- OPERAND:** In all computer languages, expressions consist of two types of components: operands and operators. Operands are the objects that are manipulated and operators are the symbols that represent specific actions. For example, in the expression 5+x, x and 5 are operands and + is an operator. All expressions have at least one operand.

**- IDENTIFIER:** Same as name. The term identifier is usually used for variable names.

**- VARIABLE:** A symbol or name that stands for a value. For example, in the expression (x + y), x and y are variables.

**- CONSTANT:** In programming, a constant is a value that never changes. The other type of values that programs use is variables, symbols that can represent different values throughout the course of a program.

**- CU - CONTROL UNIT:** CU stands for control unit. Short for control unit, it is a typical component of the CPU that implements the microprocessor instruction set. It extracts instructions from memory and decodes and executes them, and sends the necessary signals to the ALU to perform the operation needed. Control Units are either hardwired (instruction register is hardwired to rest of the microprocessor) or micro-programmed.

**- ALU - ARITHMETIC LOGIC UNIT:** The part of a computer that performs all arithmetic computations, such as addition and multiplication, and all comparison operations. The ALU is one component of the CPU (central processing unit).

+

**- QUALITATIVE ALGORITH:** One in which no numerical calculations involved, and are always expressed in words. Algorithms are used to describe processes of everyday life . This type of algorithm is not strictly computable because raw largely subjective part of the objective.

**- QUANTITATIVE ALGORITHM:** One in which other involved numerical calculations. They are very objective algorithms, as to always correspond with mathematical problems demand the inclusion of numerical calculations for obtaining results.

**- LANGUAGES ALGORITHMIC GRAPHICS:** These are graphical representations of operations performed by an algorithm (flowchart).

**- NO GRAPHIC LANGUAGE:** Descriptively represents operations to be performed by an algorithm (pseudocode).

**- PROBLEM DEFINITION:** Refers to the understanding of the problem as such. This will define what is required to solve, which results to get , what you have, what is unknown, the conditions that give us the amounts given.

**- PROBLEM ANALYSIS:** The main objective of the analysis is thoroughly deepen understanding of the mechanics of the problem to be solved, fully describing the input and output specifications and computations to be performed. This is where the mathematical and logical approach to solving the problem becomes.

**- ALGORITHM DESIGN:** Description of logical steps that solve the problem hate gets the required results. The characteristics of a good algorithm are: A) Have a particular starting point. B) Be defined, must not allow double interpretations. C) Being generally supports most of the variants that may arise in the problem definition. D) Be finite in size and run time.

**- PERFORMANCE:** Convert the instructions written in the programming language high-level instructions written in machine language, so that this can understand , and so one by one the instructions are executed.

**- TEST DESK:** That which can detect possible errors committed by the programmer in designing the algorithm. This test involves the selection of different input data to the algorithm and in following the sequence of each of the stages to obtain the results.

**- TEST OR TREATMENT:**Process of identifying and eliminating errors.

**- TECHNICAL DESIGN TOP DOWN:** The process by which a problem is decomposed into a series of levels or successive steps.

-**BOTTOM UP TECHNICAL DESIGN:** Emphasis on programming and early testing, which can begin as soon as the first module specified. This approach has the risk of not knowing how to program things be connected to the rest of the system, and this connection may not be as easy as it was thought at first. One of the biggest benefits of this approach is code reuse.