**Computer program and Programming Languages**

Computer program is a set of instructions that, when executed, causes the computer to behave in a predetermined manner. Without programs, computers are useless and can do nothing. Computer cannot understand human natural languages like English or Nepali. To instruct a computer to perform a certain job, we need languages which can be understood by the computer. The languages which are used to instruct the computer to do certain jobs are called **computer programming languages.**

Types or Levels of Programming Languages

There are two **types of programming languages**, which can be categorized into the following ways:

**1. Low level language**

* Machine language (1GL)
* Assembly language (2GL)

**2. High level language**

* Procedural-Oriented language (3GL)
* Problem-Oriented language (4GL)
* Natural language (5GL)

Low Level Language:

This language is the most understandable language used by computer to perform its operations. It can be further categorized into:

**a) Machine Language (1GL)**

Machine language consists of strings of binary numbers (i.e. 0s and 1s) and it is the only one language, the computer directly understands. Machine language has a merit of very fast execution speed and efficient use of primary memory.

Example:

To add two numbers 3 and 6 in machine language program may be like:

11 10111001 110

3 Machine code for addition(say) 6

**Merits:**

* It is directly understood by the computer so has faster execution time since the programs written in this language need not be translated.
* It doesn’t need larger memory.

**Demerits:**

* It is very difficult to program using 1GL since all the instructions are to be represented by 0s and 1s.
* Use of this language makes programming time consuming.
* It is difficult to find errors and to debug.
* It can be used by experts only.

**b) Assembly Language**

Assembly language is also known as low-level language because to design a program programmer requires detailed knowledge of hardware specification. This language uses mnemonics code (symbolic operation code like ‘ADD’ for addition) in place of 0s and 1s. The program is converted into machine code by the assembler.

Example:

To add two numbers 3 and 6 in machine language program may be like:

11 ADD 110

Some of the commonly used mnemonics are:

**CODE Meaning**

ADD Addition

SUB Subtraction

INR Increase

DCR Decrease

CMP Compare

**Merits:**

* It makes programming easier than 1GL since it uses mnemonics code for programming. Eg: ADD for addition, SUB for subtraction, DIV for division, etc.
* It makes programming process faster.
* Error can be identified much more easily compared to 1GL.
* It is easier to debug than machine language.

**Demerits:**

* Programs written in this language is not directly understandable by computer so translators should be used.
* It is hardware dependent language so programmers are forced to think in terms of computer’s architecture rather than to the problem being solved.

High level language

Instructions of this language closely resembles human language or English like words. It uses mathematical notations to perform the task. The high level language is easier to learn. It requires less time to write and is easier to maintain the errors. The high level language is converted into machine language by one of the two different languages translator programs; **interpreter or compiler.**

High level language can be further categorized as:

**a) Procedural-Oriented language (3GL)**

Procedural Programming is a methodology for modeling the problem being solved, by determining the steps and the order of those steps that must be followed in order to reach a desired outcome or specific program state. These languages are designed to express the logic and the procedure of a problem to be solved.

It includes languages such as Pascal, COBOL, C, FORTAN, etc.

**b) Problem-Oriented language (4GL)**

It allows the users to specify what the output should be, without describing all the details of how the data should be manipulated to produce the result. This is one step ahead from 3GL. These are result oriented and include database query language.

Eg: Visual Basic, C#, PHP, etc.

**c) Natural language (5GL)**

Natural language are still in developing stage where we could write statements that would look like normal sentences

**Problem Solving Approaches**

A computer is a very powerful and versatile machine capable of performing a multitude of different tasks, yet it has no intelligence or thinking power. The intelligence Quotient (I.Q) of a computer is zero. A computer performs many tasks exactly in the same manner as it is told to do. This places responsibility on the user to instruct the computer in a correct and precise manner, so that the machine is able to perform the required job in a proper way. A wrong or ambiguous instruction may sometimes prove disastrous. Therefore computer needs to be instructed correctly and for that the user must have clear understanding of the problem to be solved.

Problem solving technique (Procedure for solving problem):

In order to solve a problem by the computer, one has to pass through certain stages or steps. They are

1. Understanding the problem

2. Analyzing the problem

3. Developing the solution

4. Coding and implementation.

1. Understanding the problem:

Here we try to understand the problem to be solved in totally. Before with the next stage or step, we should be absolutely sure about the objectives of the given problem.

2. Analyzing the problem:

After understanding thoroughly the problem to be solved, we look different ways of solving the problem and evaluate each of these methods. The idea here is to search for an appropriate solution to the problem under consideration.

3. Developing the solution:

Here an overview of the sequence of operations that was the result of analysis stage is expanded to form a detailed step by step solution to the problem under consideration.

4. Coding and implementation:

The last stage of the problem solving is the conversion of the detailed sequence of operations into a language that the computer can understand. Here each step is converted to its equivalent instruction or instructions in the computer language that has been chosen for the implantation.

Algorithm:

Algorithm can be defined as the sequence of precise instructions that leads to a solution. It may be possible to solve the problem in more than one ways, resulting in more than one algorithm. The choice of various algorithms depends on the factors like reliability, accuracy and easy to modify. The most important factor in the choice of algorithm is the time requirement to execute it.

Steps involved in algorithm development:

Step 1: Identification of input

The input for the problem to be solved needs to be identified at first.

Step 2: Identification of output

The expected output of the problem is identified.

Step 3: Identification of processing operations

All the calculations to be performed in order to lead to output from the input is identified.

Step 4: Processing definiteness

It should be ensured that the algorithm is clear and doesn’t posses any ambiguity.

Step 5: Processing finiteness

It should be ensured that the program terminates after certain steps and the terminating condition should be specified.

Step 6: Processing effectiveness

It should be ensured that the instructions in the algorithm are practicable and can be carried out easily.

Algorithm should posses following properties:

1. Finiteness:

An algorithm must terminate in a finite number of steps.

2. Definiteness:

Each step of the algorithm must be precisely and unambiguously stated.

3. Effectiveness:

Each step must be effective, in the sense that it should be primitive easily convert able into program statement) can be performed exactly in a finite amount of time.

4. Generality:

The algorithm must be complete in itself so that it can be used to solve problems of a specific type for any input data.

5. Input/output:

Each algorithm should take one or more input and produce relevant output. An algorithm can be written in English like sentences or in any standard representation. Sometimes, algorithm written in English like languages are called Pseudo Code .

Example:

Suppose we want to find the average of three numbers, the algorithm is as follows:

Step 1: Start

Step 2:Read the numbers a, b, c .

Step 3 :Compute the sum of a, b and c.

Step 4 :Divide the sum by 3 .

Step 5 :Store the result in variable d.

Step 6: Print the value of d .

Step 7: End of the program .

Step 8: End

Example 2: Algorithm to check prime number.

Step 1: Start

Step 2:Read integer variable A from user.

Step 3: Divide the variable A with (A-1 to 2)

Step 4: If A is divisible by any value (A-1 to 2) it is not prime

Step 5:Else it is prime.

Step 6:End

Example 3: Algorithm to find the greatest among 3 numbers

Step 1:Start

Step 2:Read three numbers A,B & C

Step 3:If A>B,then go to step 6

Step 4:If B>C,then print B & go to step 8

Step 5:print C is greatest & go to step 8

Step 6:If A>C,then print A is greatest & go to step 8

Step 7:Print C is greatest

Step 8:End

**Flow Chart:**

A flow chart is a step by step diagrammatic representation of the logic paths to solve a given problem. Or A flowchart is visual or graphical representation of an algorithm.

Advantages of Flowcharts

1. The flowchart shows the logic of a problem displayed in pictorial fashion which facilitates easier checking of an algorithm.

2. The Flowchart is good means of communication to other users. It is also a compact means of recording an algorithm solution to a problem.

3. The flowchart allows the problem solver to break the problem into parts. These parts can be connected to make master chart.

4. The flowchart is a permanent record of the solution which can be consulted at a later time.

Flow Chart Symbols

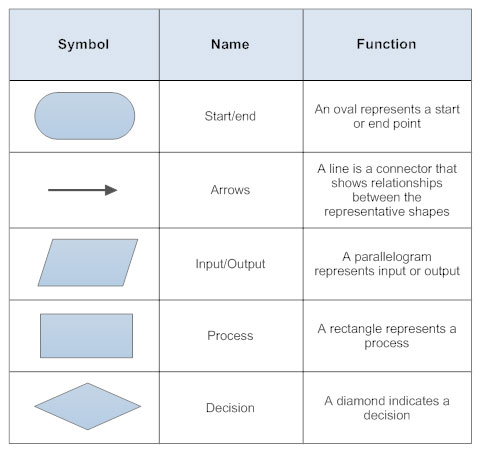
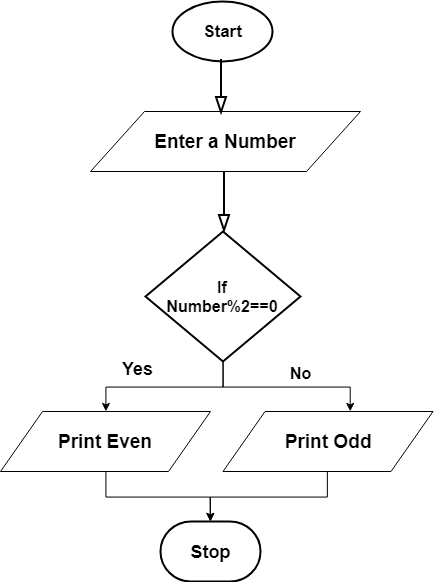


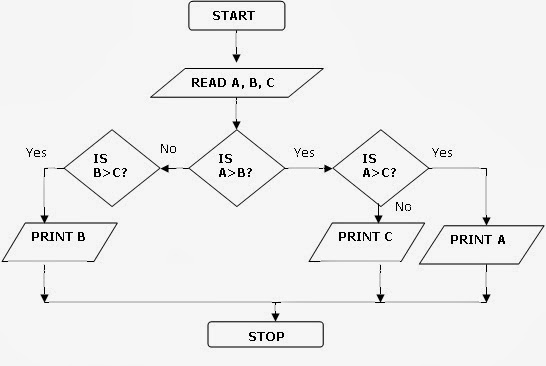
Fig: Flow Chart Symbols

Example: Check whether the number is even or odd.



Example:

Flowchart to find the largest among three numbers



**Compiler vs Interpreter**

A high level source program must be translated first into a form that machines can understand. This is done by software called a compiler or interpreter.

The compiler takes the source code as input and produces the machine language code (Object code) for the machine on which it is to be executed as output. An interpreter, like compiler is also a translator which translates a high level language into machine level language line by line.

The basic difference between compiler and interpreter are as follows:

|  |  |
| --- | --- |
| Compiler | Interpreter |
| Compiler scans the entire program before translating it into machine code. | Interpreter translates and executes the program line by line. |
| Syntax errors are found only after the compilation of complete program. | Syntax error can be trapped after translation of every line. |
| It takes less execution time. | It takes more execution time. |
| Memory requirement is more. | Memory requirement is less. |
| Intermediate code is generated. | Intermediate code is not generated. |
| Example: C, C++,Java, FORTRAN etc. | Example: QBASIC , PERL, PHP, ASP etc. |