Unit II. (i) Computer Programming Languages

Need of a computer programming language:

A programming language is an artificial language designed to communicate instructions to a computer. Thousands of different programming languages have been created with many more being created every year. Most programming languages describe computation in an imperative style, i.e., as a sequence of commands, although some languages, such as those that support functional programming or logic programming, use alternative forms of description.

Computer Languages are generally classified into two categories

(i) Low – Level computer programming language and
(ii) High – Level computer programming language

Low level languages can either be (i) Machine language or (ii) Assembly language. They have the following characteristics.

- They are machine dependent,
- They require the programmers to have a good knowledge of the internal structure of the computer being used
- They are very difficult to learn
- They are not portable to other machine with different instruction set
- They are very fast to execute
- They more prone to errors

Examples of low level languages include machine language specific to each processor and assembly language specific to each processor.

High level languages allow much more abstraction than low level languages. This allows algorithms and functions to be written without requiring detailed knowledge of the hardware used in the computing platform.

In comparison to low-level programming languages, it may use natural language elements, be easier to use, or be from the specification of the program, making the process of developing a program simpler and more understandable with respect to a low-level language. The amount of abstraction provided defines how "high-level" a programming language is.

Examples of high level languages include Basic, Pascal, C, C++, Java, Visual Basic, etc.
Unit II. (ii) Steps in Programming

A program is a set of instructions that performs a desired task. Different programs are written with different languages. An editing program is made with a different programming "language" than one that uses graphics. Some well known programming languages are COBOL -business, BASIC -language, and - C which is used in science. Programming is complicated process, below is general overview of 5 steps involved in Programming.

1. Clarify Programming Needs (Understanding the problem.)
2. Design the Program
3. Code the Program
4. Test the Program
5. Document and Maintenance

1. Clarify Programming Needs (Understanding the problem)

Six mini steps are involved at this stage:

- Clarify objectives and users
- Clarify desired outputs
- Clarify desired inputs
- Clarify desired processing
- Double - check feasibility of implementing the program
- Document the analysis

2. Design the Program

Two mini steps are involved at this stage:

- Determine program logic through top down approach and modularization, using a hierarchy chart. A hierarchy chart, which represents top-down program design, explains the main purpose of the program.
- Design details using Pseudocode and/or flowcharts, preferably involving control structure. Pseudocode, a way of designing a program which uses normal language statements in order to describe the logic and the processing flow. Program flowcharts, graphically shows the detailed series of steps.

Three control structure:

- Sequence control structure
- Conditional/Selection control structure
- Loop control structure

3. Code the Program

Two mini steps are involved at this stage:

- Select the appropriate high-level programming language
- Code the program in that language following the syntax carefully
4. Test the Program

Testing the program comes in two phases, alpha and beta.

- **Alpha** testing is the process of reading through the program in search of errors in logic. The second step is to run a diagnostic program to search for syntax or input errors.
- **Beta** testing involves using the program in the real world to see if it contains any bugs or other deficiencies.

5. Document and Maintenance

Documentation should be ongoing from the very beginning because it is needed for those involved with the program now and future. Upon completion **User Documentation** for commercial use, **Operator Documentation** for people who run computer systems, and **Programmer Documentation** for programmers charged with maintenance.

Four mini steps are involved:

- Write user documentation
- Write operator documentation
- Write programmer documentation
- Maintain the program

**Unit II. (iii) Algorithms & Flow Charts**

**What is an Algorithm:**

An **Algorithm** is a stepwise sequence of logical operations performed to get the solution of a problem.

**What is a FLOWCHART?**

A flowchart is a graphical representation of an Algorithm. Flowcharts are generally drawn in the early stages of formulating computer solutions. Flowcharts facilitate communication between programmers and business people. These flowcharts play a vital role in the programming of a problem and are quite helpful in understanding the logic of complicated and lengthy problems.

Flowcharts are usually drawn using some standard symbols; however, some special symbols can also be developed when required. **Some standard symbols, which are frequently, required for flowcharting many computer programs are shown below.**
GUIDELINES FOR DRAWING A FLOWCHART

The following are some guidelines in flowcharting:

a. In drawing a proper flowchart, all necessary requirements should be listed out in logical order.
b. The flowchart should be clear, neat and easy to follow. There should not be any room for ambiguity in understanding the flowchart.
c. The usual direction of the flow of a procedure or system is from left to right or top to bottom.
d. Only one flow line should come out from a process symbol.
e. Only one flow line should enter a decision symbol, but two or three flow lines, one for each possible answer, should leave the decision symbol.
f. Only one flow line is used in conjunction with terminal symbol.
g. Write within standard symbols briefly. As necessary, you can use the annotation symbol to describe data or computational steps more clearly.
h. If the flowchart becomes complex, it is better to use connector symbols to reduce the number of flow lines. Avoid the intersection of flow lines if you want to make it more effective and better way of communication.

i. Ensure that the flowchart has a logical start and finish.

j. It is useful to test the validity of the flowchart by passing through it with a simple test data.

ADVANTAGES OF USING FLOWCHARTS

The benefits of flowcharts are as follows:

1. Communication: Flowcharts are better way of communicating the logic of a system to all concerned.
2. Effective analysis: With the help of flowchart, problem can be analysed in more effective way.
3. Proper documentation: Program flowcharts serve as a good program documentation, which is needed for various purposes.
4. Efficient Coding: The flowcharts act as a guide or blueprint during the systems analysis and program development phase.
5. Proper Debugging: The flowchart helps in debugging process.
6. Efficient Program Maintenance: The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part

LIMITATIONS OF USING FLOWCHARTS

1. Complex logic: Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy.
2. Alterations and Modifications: If alterations are required the flowchart may require re-drawing completely.
3. Reproduction: As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.
4. The essentials of what is done can easily be lost in the technical details of how it is done.

Assignment on Flowcharts

1. Draw a flowchart to (i) add 2 numbers (ii) multiply 2 numbers and (iii) subtract 2 numbers.

2. Draw a flowchart to calculate the area and circumference of a circle.
   (Area of circle = 3.14 * radius * radius, circumference = 2 * 3.14 * radius)

3. Draw a flowchart that converts the input Celsius degree into its equivalent Fahrenheit degree. Use the formula: F= (9/5) * C + 32

4. Draw a flowchart that converts the input Fahrenheit degree into its equivalent Celsius degree. Use the formula: C= (5/9)*F – 32

5. Draw a flowchart to find the largest of three arbitrary numbers.
6. Draw a flowchart to generate the following series
   1, 3, 5, 7, 9, .................................., 97, 99

7. Draw a flowchart to generate the following series
   2, 4, 6, 8, 10, .................................., 98, 100

8. Draw a flowchart to generate the table of an arbitrary number up to a given limit ‘n’.

9. Draw a flowchart to find the largest of n arbitrary numbers.

10. Draw a flowchart to find the sum of n arbitrary numbers.