

## INTRODUCTON TO PROGRAMMING LANGUAGE

In this module will discuss about topics like,

- Introduction to problem solving.
- Introduction to algorithm and flow chart.
- Why programming language is needed.

### PROBLEM SOLVING AND DECOMPOSITION

- A problem is assigned to solve it, that problem must be solved efficiently. So, the issue in hand at most of the computer science is the **problem solving**.
- Unless we understand the program solving efficiently by following some steps, it is difficult to understand **The Programming Methodology and Data Structures** because, these are derivatives of system of problem solving.
- So ,the program is done on a computer which has the specific architecture and particular structure, on the other hand problem solving is not only the programming done on a computer, to get the efficient result, there are some prior steps to it to be followed i.e., in **natural way** of representation.



For example :- If a problem like addition of two numbers is given to solve , then the first step is the problem posed in **natural language** to a **computer language** which will solve the problem.

So, if we have the technology for computer which understands the natural language, then there is no requirement of some series of steps before programming.

- Since, we have restricted form of language which computer understands and we have natural language. Our task is to make convert the natural language into the computer understandable language. This is the core concept of **problem solving**.
- Next task is to understand the techniques of problem solving.
- In this case there are many issues raised in case of problem solving. One of them is that, **inheritance property of architecture of the computer**.
- As we discussed before that, problem solving has been done in natural way, so natural form means \*it may be in pictorial format. The architecture of the computer will not be able to read the picture form and analyze the data. To have a natural form of problem solving then series of steps should be formed which makes easy to write a program in later cases.
- So ,these series of steps are formed using
  - Algorithm design.
  - Flowchart design.
  - Programming methodology.
  - Data structuring, are related to convert a given problem in sequence of steps to be executed in a computer.
- The natural way of representation of a problem means is as *body of knowledge*.
- It is not easy task to find that *body of knowledge*. That becomes easy to form by practice.
- If we consider a beginner then, there is need to develop the style of thinking which we called problem decomposition.
- In problem decomposition, the given problem is divided into parts and each part can be expressed in natural way and then later merging the entire

concept formed in natural way makes the total steps required to write a program.

If we consider an example like,

Finding the maximum of n numbers then,

Let the numbers are  $t = t_1, t_2, t_3, t_4, t_5, t_6, \dots, t_n$ .

Then,

**Problem decomposition means,**

Max (t)

{

$L = \{t_1, t_2, t_3, t_4, \dots, t_n\}$

Make the L sets as two disjoint sets.

L1 and L2.

$a = \text{MAX}(L1)$ .

$b = \text{MAX}(L2)$ .

$\text{MAX} = \text{MAX}(a, b)$ .

Return/print the MAX value

}

## REQUIRED STEPS FOR PROBLEM SOLVING

**1.** Initial solution generation.

- a. Contains the set of actual potential programs.
  - i. Here Correctness test is done.

2. Initial solution refinement.
  - a. Efficiency
    - i. Time efficiency.
    - ii. Space efficiency.
3. Algorithm description a data structuring.
4. Final solution
5. Program

So we see in detail what each steps explains,

### 1. Initial solution generation:-

There must be solution for the given problem in your mind. This solution in your mind is vague and to make the idea concrete the decomposition of the problem in some way which inherently makes the series of steps. So, the initial solution is **Problem decomposition**.

In this step correctness of the program is tested.

#### Note:-

This correctness test result is true then it must be maintained throughout the remaining steps.

### 2. Initial solution refinement

Here in this step status of the **correctness** obtained in before step is maintained and then the extra features like **efficiency** of the given program is monitored in terms of **time efficiency** and **space efficiency**.

The terms like **algorithm description and data structuring concepts** are the features of the **programming language**. These two factors will organize to get the **final solution**.

3. This **final solution** is converted to a **program** expressed in the language which the computer understands.

That language is formed with the help of some syntax. These languages supports the high level features so that the given input will be in the form of **partial English sentences which includes some syntax** and forms like **structured oriented series of steps**, these steps. That language is called **vehicle language**.

## FEATURES OF THE VEHICLE LANGUAGE

If series of steps in natural way is formed then you are ready to write the program using these vehicle languages. The vehicle languages examples are C, C++, JAVA, etc.

The features of the vehicle languages are,

- a. **Data Structures**

Several Data types and operations used to write a program. Example of the data types are arrays, structures integers etc.

Procedure encapsulation is one of the examples of object oriented programming languages.

Dynamic data structuring is also one of the example of the data structuring.

- b. **Algorithm design**

Control constructs comes under this category. This control constructs includes, loops concepts (while loops, for loops etc). For **decomposition method**, functions and recursions concept is used.

**Note:** - In this module the **vehicle language** considered is **C**

## Algorithms

- A sequential solution of any program that written in human language or natural language, called algorithm.
- The algorithm represents the solution for a problem. Here, we not at all introducing any programming struts or syntax to solve the problem. In algorithm only general English is used to write the logically placed instructions.
- **So first step to solve a problem is to write an algorithm.**

## Example of Algorithm:-

**Make an algorithm to find whether a number is even or odd.**

**Step 1:-** start

**Step 2:-** input a number n

**Step 3:-**remainder = n mod 2

**Step 4:-**if remainder is equal to zero then

**Print "number even"**

**Step 5:-**if remainder is not equal to zero then

**Print "number is odd"**

**Step 6:-** stop

**Flow chart:** -Pictorial representation of sequence of instructions was using some predefined symbols.

The symbols used to form the flow chart is,





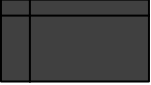








<b>Process</b> 	<b>Decision</b> 	<b>Document</b> 
<b>Merge</b> 	<b>Internal Storage</b> 	<b>Alternate Process</b> 
<b>Connector</b> 	<b>Display</b> 	<b>Delay</b> 
<b>Sort</b> 	<b>Manual Input</b> 	<b>Terminator</b> 
<b>Preparation</b> 		

Figure: Symbols used to form the flow chart.

Second step is to make a flow chart.

### Example of Flow chart

Make a flow chart to find whether a number is even or odd.

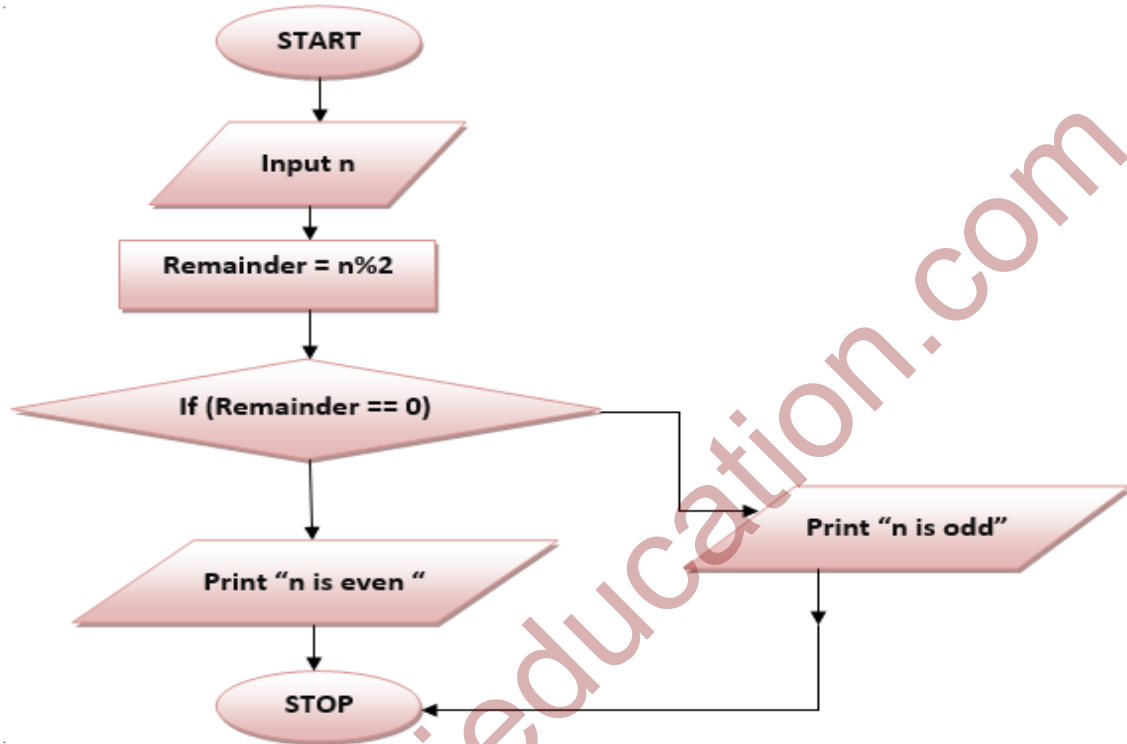


Figure: - Flow chart to find whether a number is even or odd.

Next step is to write a program. So the programming language is used.

The programming language is a communication channel between the computer and user. The user can instruct the computer to solve the problem.

### WHY THE PROGRAMMING LANGUAGE IS NEEDED?

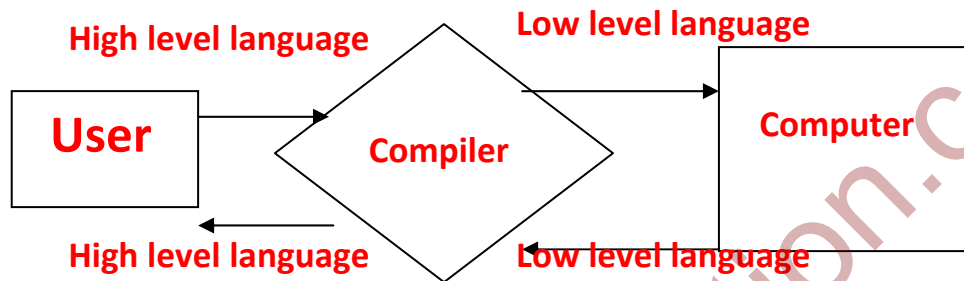
Why the programming channel is needed to communicate with the computer?

The solution is that, the computer can only understand is the machine level language. The human can understand high level language. The human can directly interact with the computer using the machine level language but it is a tedious



process. So the human uses the high level language to interact with the computer but, between the user and the computer there must be a translator which translates the high level language to machine level language. That translator is an interpreter or a compiler.

**Diagram:-**



**Programming language:** - Programming language is a procedure of writing the sequence of instructions logically using algorithm to solve the problem.

**Program:** - The sequence of instructions is called the program.

There two types of programming languages.

1. Low level languages.
2. Higher level languages.

### Low level languages

- Machine oriented language.
- The written code s directly interacted with the hardware.
- Difficult to learn.
- Instructions written are dependent on the machine circuit elements.
- For better machine working efficiency these programs are written.

Examples: - assembly language .machine language.

- Depending upon the architecture of the system the program instructions are written.

### High level language

- Here the program code written is machine independent. The code is not dependent on the architecture of the hardware.
- Easy to learn.
- Machine cannot understand the code directly so the interpreter or compiler is used to make the computer understand the instructions written in high level language by the user.