(PROGRAMMING IN C -1)

Unit : 1
Introduction to Programming

Concept of Program

- **A Program is a set of statements / Instructions.**

- *A set of coded instructions that a computer can understand to solve a problem or produce a desired result.*

- **Two basic types of computer programs are**

- (1) **An operating system**, which provides the most fundamental instructions a computer uses in its operations.

- (2) **An application program**, which runs on the operating system and does a specific job.

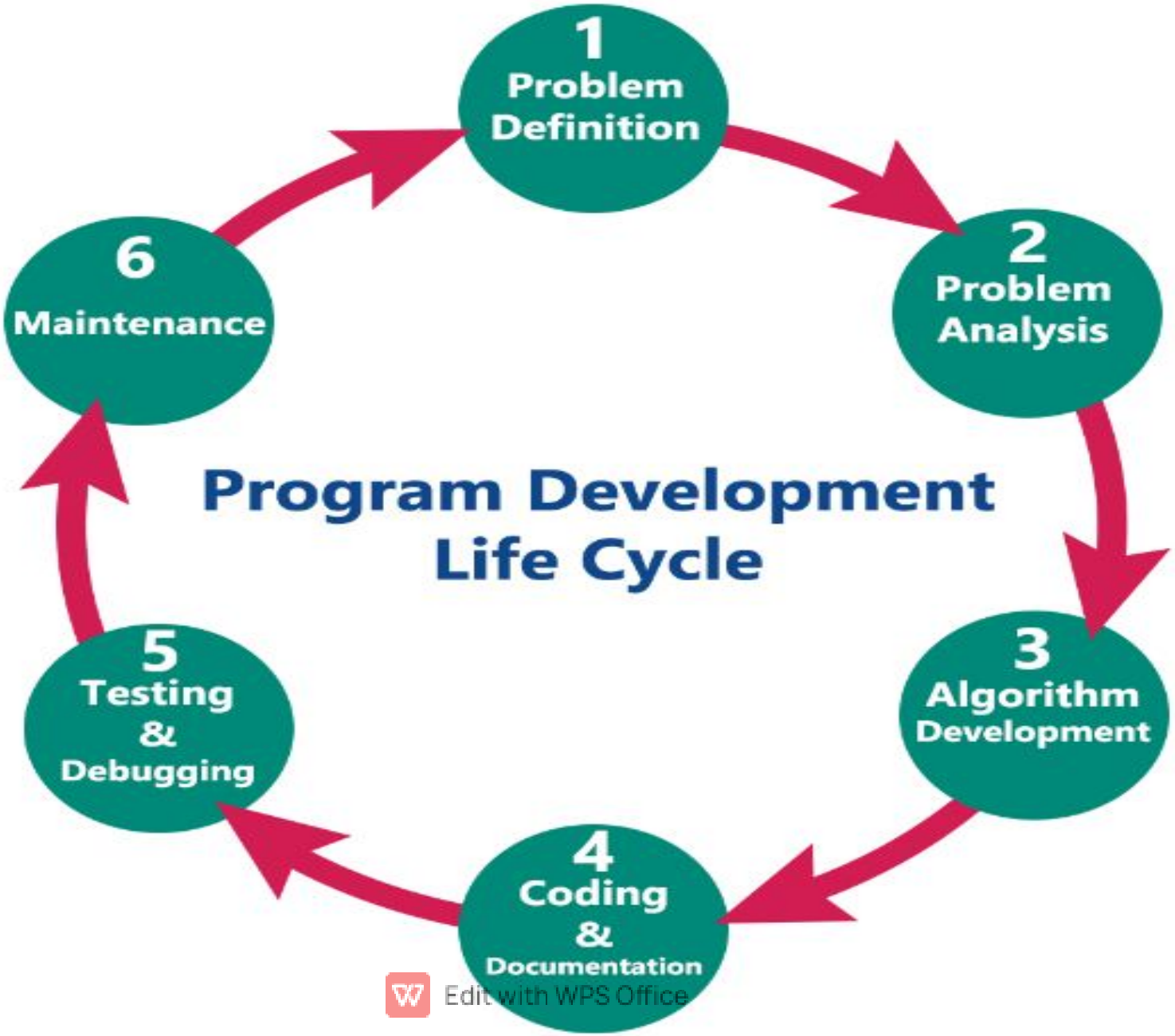
Program Development Cycle

Q: Write short note on Program development life cycle.

- **When we want to develop a program using any programming language, we follow a sequence of steps. These steps are called phases in program development.**
- The program development life cycle is a set of steps or phases that are used to develop a program in any programming language.
- **Generally, program development life cycle contains 6 phases, they are as follows....**

1) Problem Definition
2) Problem Analysis
3) Algorithm Development
4) Coding & Documentation
5) Testing & Debugging
6) Maintenance

Program Development Cycle



Program Development Cycle

1. Problem Definition

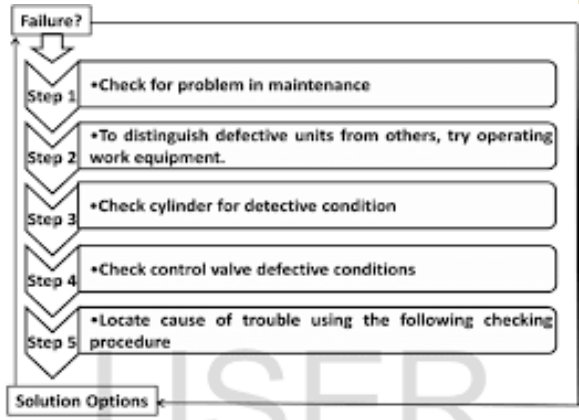
- In this phase, we define
  - The problem statement and we decide the boundaries of the problem.
  - We need to understand the problem statement, what is our requirement,
  - What should be the output of the problem solution. These are defined in this first phase of the program development life cycle.

Program Development Cycle

2. Problem Analysis

- In phase 2, we determine the requirements like variables, functions, etc. to solve the problem.
- That means we gather the required resources to solve the problem defined in the problem definition phase.

Program Development Cycle



3. Algorithm Development

- During this phase, we develop a step by step procedure to solve the problem using the specification given in the previous phase.
- This phase is very important for program development.
- That means we write the solution in step by step statements / instructions.

Program Development Cycle

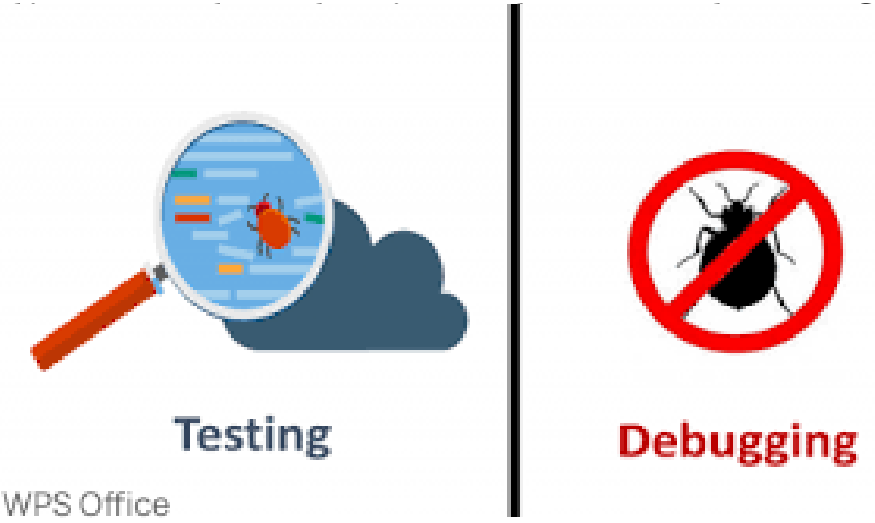
4. Coding & Documentation

- This phase uses a programming language to write or implement actual programming instructions for the steps defined in the previous phase.
- In this phase, we construct actual program. That means we write the program to solve the given problem using programming languages like C, C++, Java

Program Development Cycle

5. Testing & Debugging

- During this phase, we check whether the code written in previous step is solving the specified problem or not.
- That means we test the program whether it is solving the problem for various input data values or not.
- We also test that whether it is providing the desired output or not.
- **Debugging** is a process of fin
errors (or defects) in a comput



Program Development Cycle

6. Maintenance

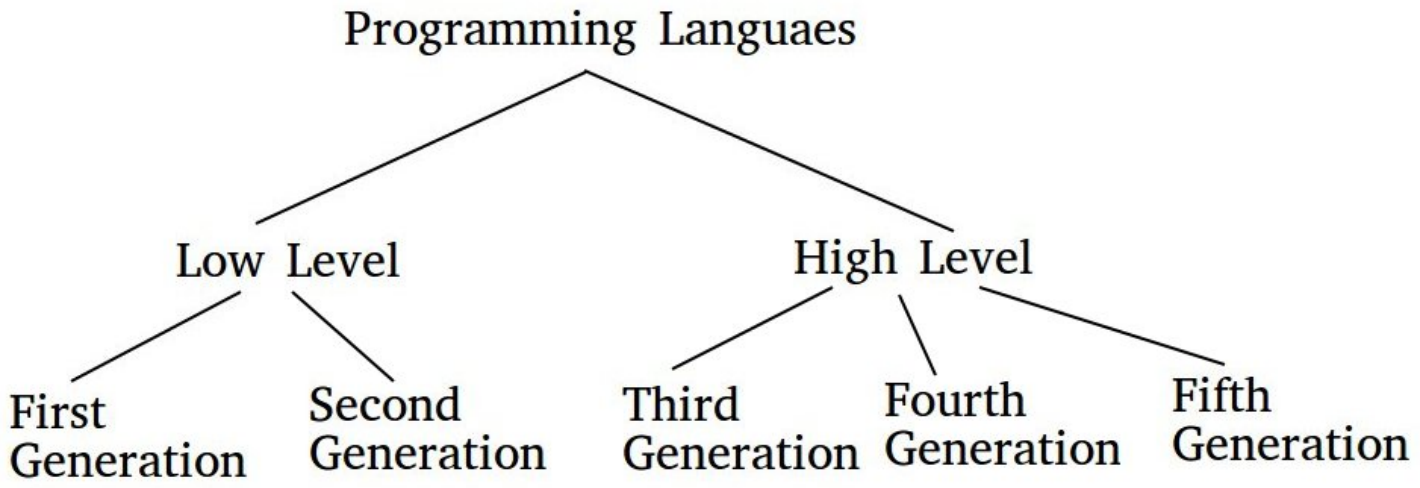
- During this phase, the program is actively used by the users.
- If any enhancements found in this phase, all the phases are to be repeated again to make the enhancements.

Concept of Programming language

- **Programming languages are used to create programs.**
- The Basic purpose of Programming Language is to develop software which ultimately solves a given problem.
- Such languages includes BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal ,etc.
- While programming languages are easy for the humans to read and understand, the computer understands the machine language that consists of numbers only that is binary language.

Q:- Explain five generation of computer programming languages.

- The concept of generations of programming languages also known as “Levels”
- There are five generations of Programming languages.



Overview of Programming Language (For Ref.)

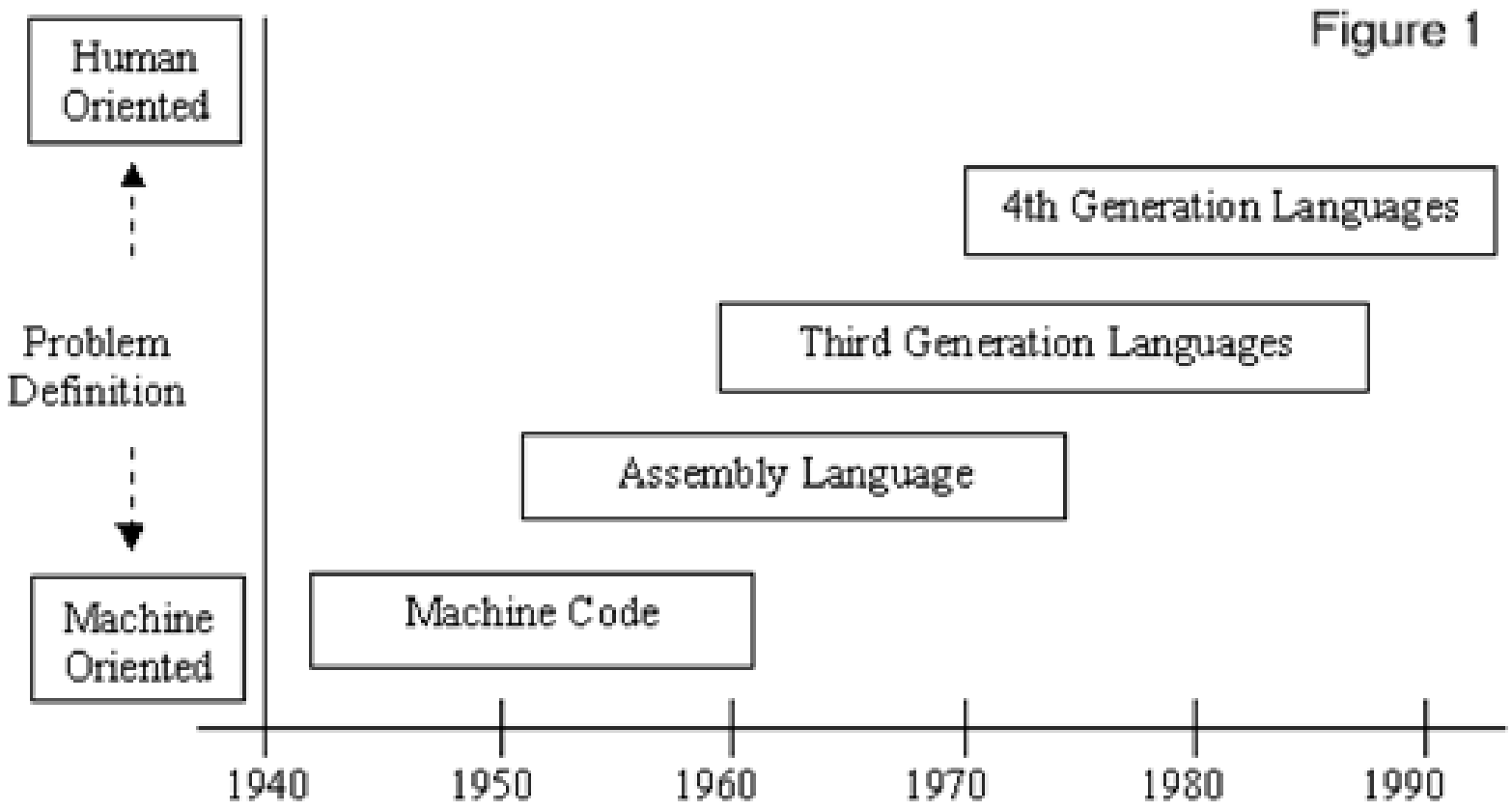


Figure 1

Overview of Programming Language (For Ref.)

"Generations" of Programming Languages

- First Generation
  - Machine language
- Second Generation
  - Assembly language
- Third Generation
  - "High-level" languages such as Pascal, C, COBOL, Fortran
- Fourth Generation
  - Scripting languages such as SQL, Applescript, VBScript
- Fifth Generation?
  - Natural language? Automatic code generation? Object-oriented languages?

(1) First Generation Language (1GL) [Machine / Binary Language]

- Machine (or low level) language was used to program the first stored-program computer systems.
- It is the only language that the computer understands.

- **All the commands and data values are expressed using 1s and 0s.**

- In 1950s each computer had its own native language, and programmers had to combine numbers to represent instructions such as *add* and *subtract*. (For example)

Machine code	Meaning
0011 1100	Load A Register with 5 value
0000 0101	
0000 0110	Load B Register with 4 value
0000 0100	
0011 1100	A <- A + B
0000 1001	Calculate sum of Register A value and Register B value and assign to A register

- Although there were similarities between each of the machine language but a computer could not understand programs written in another machine language.

- **The main advantage of machine language** is that the code can run very fast and efficiently, since it is directly executed by the CPU.

- **However, this language is difficult to learn** and is far more difficult to edit if errors occur.
- **Also code written in machine language is not portable.**

Advantages & Disadvantages of 1GL

ADVANTAGES OF BINARY LANGUAGE

- The program execution speed is fast
- Less memory usage
- No translation is required.

DISADVANTAGES OF BINARY LANGUAGE

- Difficult to use & remember:
- Machine dependent Language
- Difficulty in Error
- Difficult debugging process

(2) Second Generation Language [2GL] (Assembly Language)

The second generation includes the assembly language which is a symbolic language that use symbols to represent machine-language instructions.

These languages are closely connected to machine language and the internal architecture of the computer system on which they are used.

An assembly language statement consists of a label and an operation code.

However, like the machine language, assembly language is also machine-dependent.

Programs written in assembly language need a translator often known as the assembler to convert them into machine language.

For example :

Assembly code	Meaning
LD A, 5	Load register A with 5
LD B, 4	Load register B with 4
ADD A,B	A <-- A + B : Add value of A with value of B register and store result in register A
LD (100), A	Save the result in the main memory location 100
HALT	Halt process

ADVANTAGES OF ASSEMBLY LANGUAGE

- It is more convenient in writing program than code writing in machine language.
- Assembly code is more readable and under stable because it is written with the help of symbolic words or predefine words.

DISADVANTAGES OF ASSEMBLY LANGUAGE

- It is machine dependent language.
- It is again difficult and time consuming.
- Program execution speed is slow as compare to machine language.

(3) Third Generation Language [3GL]

- In these languages, the program statements are not closely related to the internal characteristics of the computer and is therefore often referred to has high-level languages.

- 3GLs made programming easier, efficient and less prone to errors.

- 3GL includes languages like FORTRAN and COBOL and C.

- ***A translator is needed to translate the instructions written in high level language into computer-executable machine language.*** Such translators are commonly known as **interpreters** and **compilers**.

- 3GLs makes it easier to write and debug a program and gives the programmer more time to think about its overall logic.

- The programs written in such languages are portable between machines.

(4) Fourth Generation Language [4GL]

- 4GLs are nonprocedural languages.

- When writing code using a procedural language, the programmer has to tell the computer how a task is done.

- While using a nonprocedural language the programmers define only what they want the computer to do, without supplying all the details of how it has to be done.

- **For example : SQL (Structured Query Language)**

Characteristics of 4GLs include:

- The code is written in English-like sentences. The code is easier to maintain.

- 4GL code enhances the productivity of the programmers as they have to type fewer lines of code to get something done.

(5) Fifth Generation Language [5GL]

- 5GLs are centered on solving problems using constraints given to the program, rather than using an algorithm written by a programmer.

- They are widely used in artificial intelligence research. **Typical examples of a 5GL include Prolog and Mercury.**

- Another aspect of a 5GL is that it contains visual tools to help develop a program.

Features of a good programming language

Q:- List out various features of good programming language.

1) The language must allow the programmer to write simple, clear and concise programs.

2) The language must be simple to use so that a programmer can learn it without any explicit training.

3) The language must be platform independent. That is, the program developed using the programming language can run on any computer system.

4) The Graphical User Interface (GUI) of the language must be attractive, user-friendly, and self-explanatory.

Solution of ATM process

Solutions : 1
(Textual form)

- Step : 1 Insert ATM Card
- Step : 2 Enter PIN Number
- Step : 3 Select Withdrawal Opt
- Step : 4 Collect Money
- Step : 5 Remove ATM Card

Solutions : 2
(Graphical Form)

Algorithm

Q:- Write detail short note on Algorithm with examples

- "Algorithm" : It is a step by step descriptive procedure for solving a particular problem.

- The algorithm gives logic of the program, that is, a step-by-step description of how to arrive at a solution.

Algorithm

- A sequence of instructions must process the following characteristics:
  - Instructions must be clear
  - Instructions must be effective.
  - Not even a single instruction must not be repeated infinitely.
  - After the algorithm gets terminated, the desired result must be obtained

Algorithm

- Every algorithm always starts with START and terminate with STOP keywords.
- In other way we can write algorithm using BEGIN and terminate with END keywords.
- For input and output data we can use READ / INPUT for input and WRITE / PRINT for output.
- For each Step We use Step-Number.

KEY FEARTURES OF AN ALGORITHM

Any algorithm has a finite number of steps and some steps may involve decision making, repetition. Broadly speaking, an algorithm having three key features that can be given as:

1. Sequence
2. Decision
3. Repetition

KEY FEARTURES OF AN ALGORITHM

1. Sequence

Sequence means that each step of the algorithm is executed in the specified order.

Ex. Input Two Number and Perform Addition

STEP 1 : START
STEP 2 : DECLARE / INITIALIZE A←0,B←0,SUM←0
STEP 3 : PRINT "ENTER FIRST NUMBER:"
STEP 4 : INPUT A
STEP 5 : PRINT "ENTER SECOND NUMBER :"
STEP 6: INPUT VALUE B
STEP 7: SUM ← A+B
STEP 8: PRINT SUM
STEP 9: STOP

KEY FEARTURES OF AN ALGORITHM

- 2. Decision

Decision statements are used when the outcome of the process depends on some condition.

For Example, if A>B then print “A is Greater” otherwise “B is Greater”

General Form of If as follows

```
if condition
   then process 1
else
   process 2
```

KEY FEARTURES OF AN ALGORITHM

- EX. INPUT TWO NUMBER AND FIND MAXIMUM

STEP 1 : START

STEP 2 : DECLARE A←0, B←0

STEP 3 : INPUT A

STEP 4 : INPUT B

STEP 5 : IF A>B THEN

PRINT “A IS MAX”

STEP 6 : ELSE

PRINT “B IS MAX”

STEP 7: STOP

KEY FEARTURES OF AN ALGORITHM

EX. INPUT A NUMBER AND CHECK WHETHER IS IT ODD OR EVEN.

STEP 1 : START

STEP 2 : DECLARE A←0

STEP 3 : PRINT "ENTER ANY NUMBER"

STEP 4 : INPUT A

STEP 5 : IF A % 2 = 0THEN

PRINT "NUMBER IS EVEN"

STEP 6 : ELSE

PRINT "NUMBER IS ODD"

STEP 7: STOP

KEY FEARTURES OF AN ALGORITHM

3. Repetition

Repetition which involves executing one or more steps for a number of times can be implemented using constructs like the while, do-while and for loops.

These loops executed one or more steps until some condition is true.

KEY FEARTURES OF AN ALGORITHM

- EX. PRINT 1 TO 10 NUMBERS USING LOOP.

STEP 1: START

STEP 2: INITIALIZE SET I←1, N←10

STEP 3: REPEAT STEP WHILE I<=N

PRINT I

I ← I + 1

GOTO STEP 3

STEP 4: STOP

KEY FEARTURES OF AN ALGORITHM

- EX. PRINT SUM OF FRIST 10 NUMBERS

STEP 1: START

STEP 2: INITIALIZE SET I ← 1, N ←10, S ← 0

STEP 3: REPEAT STEP WHILE I<=N

S ← S + I

I=I+1

GOTO STEP 3

STEP 4 : PRINT S

STEP 5: STOP

KEY FEARTURES OF AN ALGORITHM

- EX. PRINT SQUARE OF FRIST 10 NUMBERS

STEP 1: START

STEP 2: INITIALIZE SET I=1, N=10,S

STEP 3: REPEAT STEP WHILE I<=N

PRINT I * I

I=I+1

GOTO STEP 3

STEP 4: STOP

ADVANTAGE OF ALGORITHM

- A programmer can know identification of the processes, major decision statements and required variables for solving the problem at the development time.

Algorithm represents a process of an entire solution in a specific sequence so it is better way to understand program.

ALGORITHM ADVANTAGE AND DISADVANTAGE

- **Advantages**
- Easy to write.
- Human readable techniques to understand logic.
- Algorithms for big problems can be written with moderate efforts.

- **Disadvantages**
- Difficult to debug.
- Difficult to show branching and looping.
- Jumping (goto) makes it hard to trace some problems.

FLOWCHART

Q:- Explain flowchart with standard symbols.

- ***A flow chart is a graphical or symbolic representation of a process.***

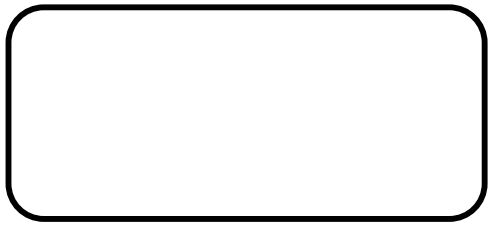
- They are basically used to design and document complex processes to help the viewers to visualize the logic of the process

- Each step in the process is represented by
- different shapes and displays a short detail of the
- process step. Every symbol is connected together
- with arrows showing the process flow direction.

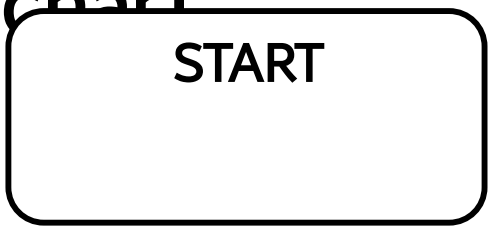
FLOWCHART

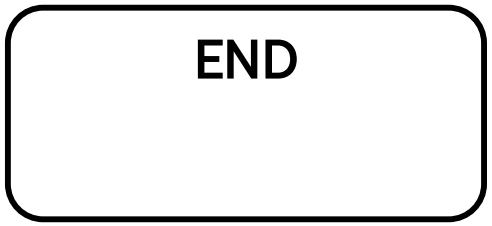
Various Symbols of Flowchart is as Follows:

1. Start and end symbols



It is also known as the terminal symbol is always the first and the last symbol in a flowchart



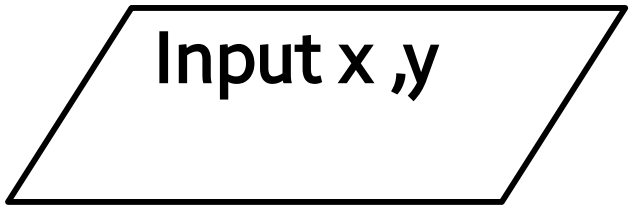


FLOWCHART

2. Input/Output Symbols

This *Symbol* is used to get input from users or display the results to them

For example :

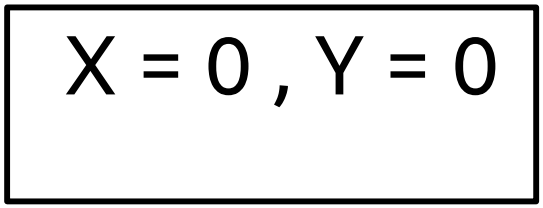


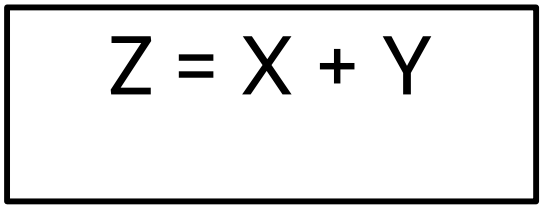
FLOWCHART

- 3. Generic processing

Generic processing step also called an activity is represents instructions like add a to b, save the result.

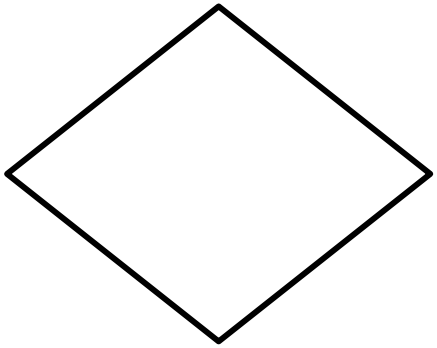
For example :



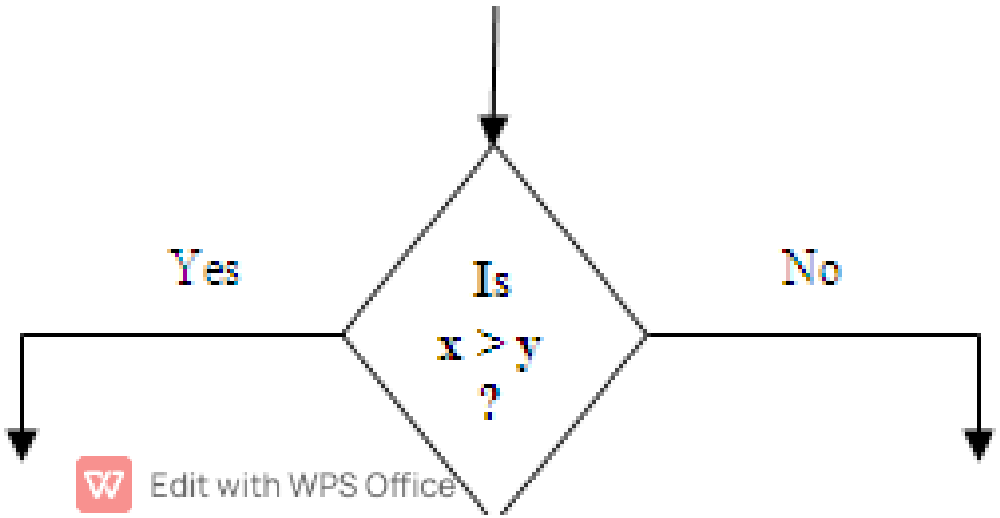


FLOWCHART

4. Conditional or decision symbol

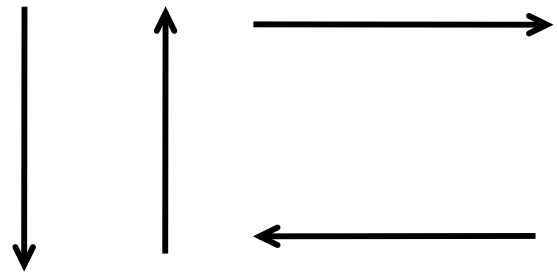


- It is represented using a diamond.
- It is used to depict a Yes/No question or True/ False test.
- The Arrow should always be labeled.



FLOWCHART

- 5. Arrow

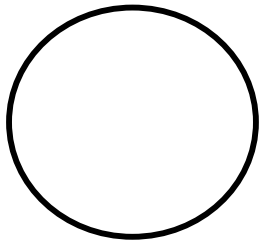


- Arrows depict the flow of control of the program.

- They illustrate the exact sequence in which the instructions are executed.

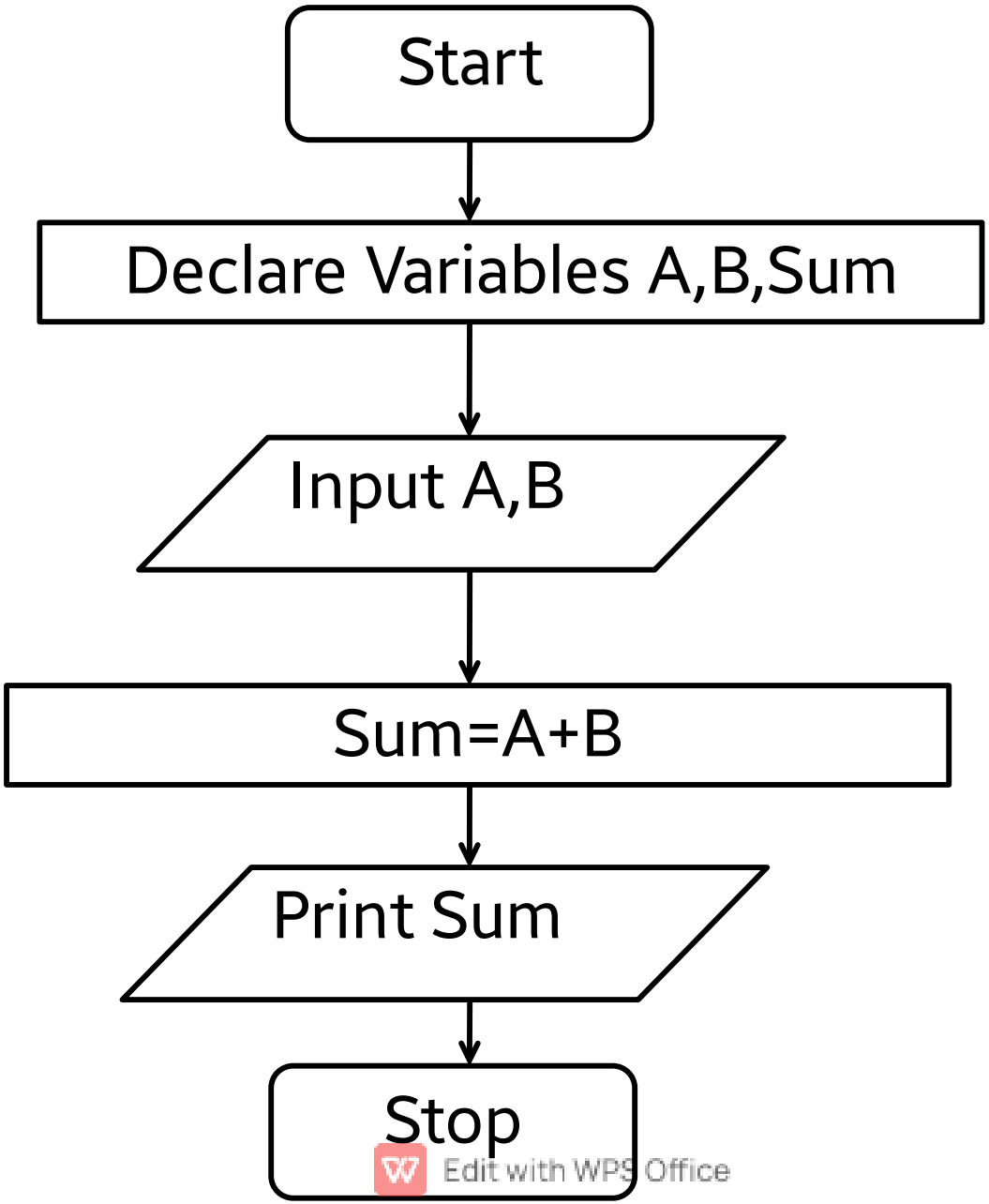
FLOWCHART

6. Labeled connectors

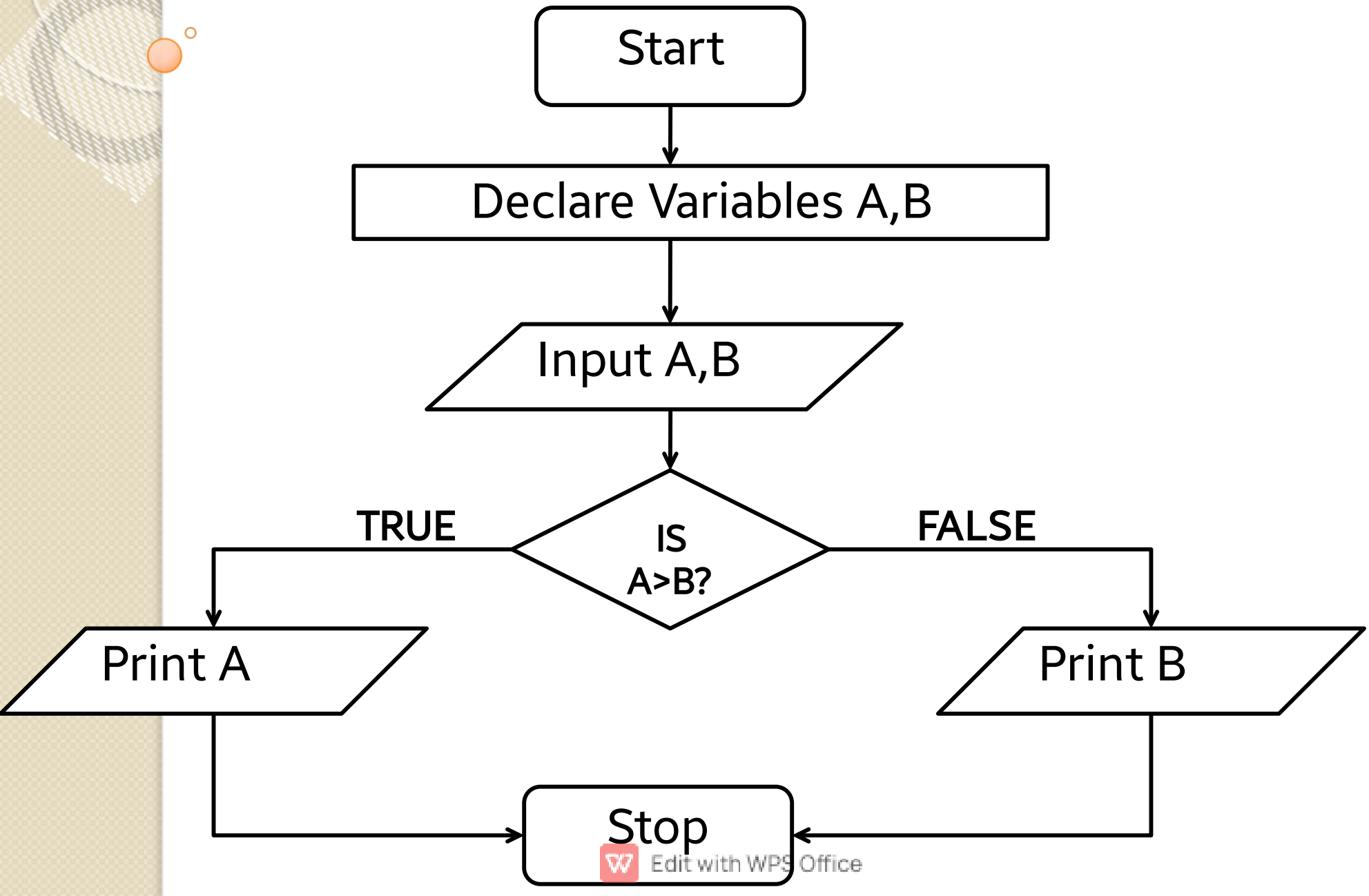


- *connectors* represented by an identifying label inside a circle are used in complex or multi-sheet diagrams to substitute for arrows.

Example of Flowchart



Example of Flowchart



Advantages of Flowcharts

- A flowchart is a diagrammatic representation that illustrates the sequence of steps that must be performed to solve a problem.

- They are usually drawn in the early stages of formulating computer solutions to facilitate communication between programmers and business people.

- Flowcharts help programmers to understand the logic of complicated and lengthy problems.

- They help to analyze the problem in a more effective manner

Limitations of using Flowcharts

- Drawing flowcharts is a laborious and a time consuming activity.

- Flowchart of a complex program becomes complex.

- At times, a little bit of alteration / Modification in the solution may require complete re-drawing of the flowchart.

FLOWCHART ADVANTAGE AND DISADVANTAGE (For Ref.)

- **Advantages**
- Easy to draw.
- Easy to understand the logic.
- Easy to identify mistakes by the non-computer person.
- Easy to show branching and looping

- **Disadvantages**
- Time-consuming.
- Difficult to modify.
- Very difficult to draw a flowchart for big or complex problems.

Difference between Flowchart & Algorithm (For Ref.)

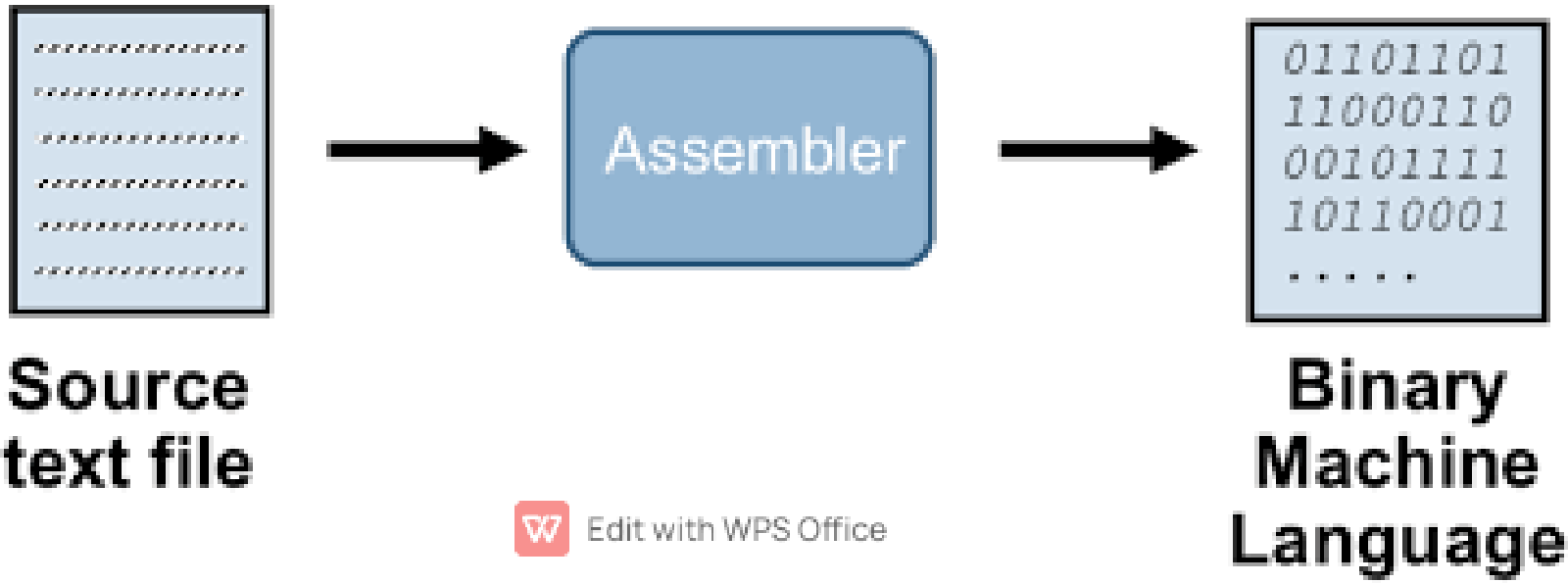
❖Flowchart	❖Algorithm
❖Block by block information diagram representing the data flow.	❖Step by step instruction representing the process of any solution.
❖It is a pictorial representation of a process.	❖It is step wise analysis of the work to be done.
❖Solution is shown in graphical format.	❖Solution is shown in non computer language like English.
❖Easy to understand as compared to algorithm.	❖It is somewhat difficult to understand.
❖Easy to show branching and looping.	❖Difficult to show branching and looping
❖Flowchart for big problem is impractical	❖Algorithm can be written for any problem
❖Difficult to debug errors.	❖Easy to debug errors.

Assembler

- An assembler is a program that takes basic computer instructions (of assembly language) and converts them into a pattern of bits that the computer's processor can use to perform its basic operations.

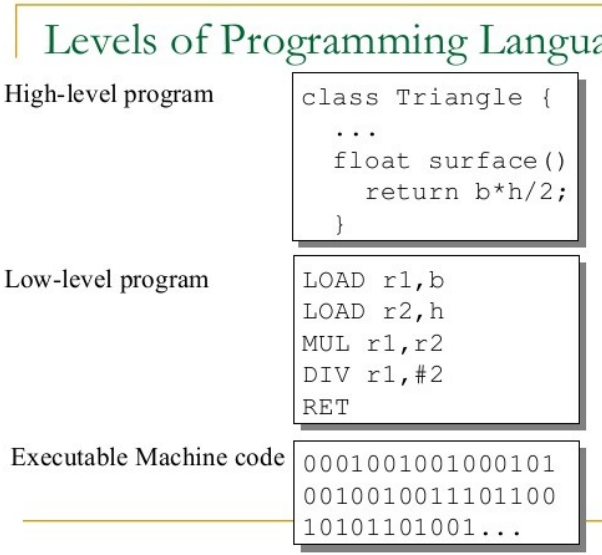
- This is used in 2GL. [Assembly Language]

- Some people call these instructions assembler language and others use the term as



Interpreter

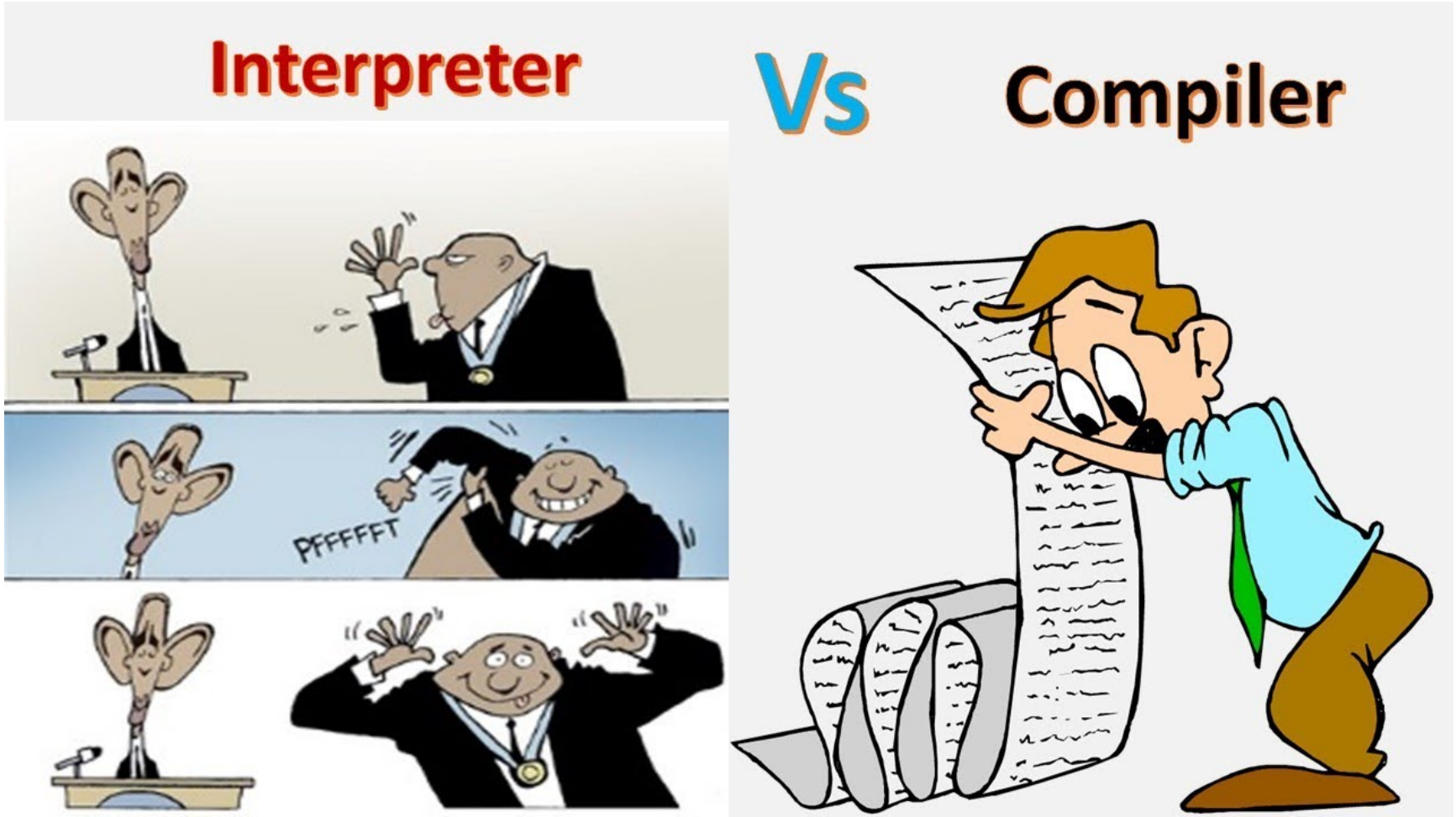
- *Translates program one statement at a time.*
- It takes less amount of time to analyze the source code but the overall execution time is slower.
- No intermediate object code is generated, hence are memory efficient.
- Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.
- **Programming language like PHP, Perl,Python, Ruby uses interpreter.**



Compiler

- *Scans the entire program and translates it as a whole into machine code.*
- It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
- Generates intermediate object code which further requires linking, hence requires more memory.
- It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
- **Programming language like C, C++ , JAVA use compiler.**

Concept of Compiler & Interpreter



Difference between Compiler & Interpreter

COMPARISON	COMPILER	INTERPRETER
Input	It takes an entire program at a time.	It takes a single line of code or instruction at a time.
Output	It generates intermediate object code.	It does not produce any intermediate object code.
Working mechanism	The compilation is done before execution.	Compilation and execution take place simultaneously.
Speed	Comparatively faster	Slower
Memory	Memory requirement is more due to the creation of object code.	It requires less memory as it does not create intermediate object code.
Errors	Display all errors after compilation, all at the same time.	Displays error of each line one by one.
Error detection	Difficult	Easier comparatively
Pertaining Programming languages	C, C++, C#, Scala, typescript uses compiler.	PHP, Perl, Python, Ruby uses an interpreter.

THANK YOU