



SNS COLLEGE OF TECHNOLOGY



Coimbatore-36.

An Autonomous Institution

**Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A+’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai**

COURSE NAME : 23CST101-PROBLEM SOLVING & C PROGRAMMING

I YEAR/ I SEMESTER

UNIT – I INTRODUCTION TO PROBLEM SOLVING TECHNIQUES

Topic: Notation (Pseudo Code, Flow Chart, and Programming Language)

Mrs.Papithasri K

Assistant Professor

Department of Computer Science and Engineering



Programming Language



A **programming language** is a set of symbols and rules for instructing a computer to perform specific tasks. The programmers have to follow all the specified rules before writing program using programming language. The user has to communicate with the computer using language which it can understand.

Types of programming language

1. Machine language
2. Assembly language
3. High level language



Programming Language



Machine language:

- The computer can understand only machine language which uses 0's and 1's. In machine language the different instructions are formed by taking different combinations of 0's and 1's.

Advantages:

1. Translation free
2. High speed

Disadvantage:

1. Hard to find errors
2. Time consuming
3. Machine dependent



Programming Language



Assembly language:

- An assembly language is developed which is logically equivalent to machine language but it is easier for people to read, write and understand.
- Assembly language is symbolic representation of machine language that uses symbolic notation to represent machine language instructions.
- Its also called low level language because it closely related to the machines.

Assembler:

- Assembler is the program which translates assembly language instruction in to a machine language.

Advantage:

1. Easy to understand and use.
2. It is easy to locate and correct errors.

Disadvantage:

1. Machine dependent
2. Hard to learn
3. Less efficient



Programming Language



High level language

- High level language contains English words and symbols. The specified rules are to be followed while writing program in high level language. The interpreter or compilers are used for converting these programs in to machine readable form.

Translating high level language to machine language

Compiler:

- Compiler reads the whole program written in high level language and translates it to machine language.
- If any error is found it display error message on the screen.

Interpreter

- Interpreter translates the high level language program in line by line manner.
- The interpreter translates a high level language statement in a source program to a machine code and executes it immediately before translating the next statement.
- When an error is found the execution of the program is halted and error message is displayed on the screen.



Programming Language



Advantages

1. Readability
2. Machine independent
3. Easy debugging

Disadvantages

1. **Less efficient** - The translation process increases the execution time of the program. Programs in high level language require more memory and take more execution time to execute.



Categories Programming Language



Programming language are divided into following categories:

1. Interpreted programming languages
2. Functional programming languages
3. Compiled programming languages
4. Procedural programming languages
5. Scripting programming language
6. Markup programming language
7. Concurrent programming language
8. Object oriented programming language



Categories Programming Language



Interpreted programming languages

- An interpreted language is a programming language and it executes instructions directly, without previously compiling a program into machine language instructions.
- The interpreter executes the program directly translating each statement into a sequence of one or more subroutines already compiled into machine code.

Examples:

1. Pascal
2. Python

Object oriented programming language:

- Object oriented programming is a programming paradigm based on the concept of objects which may contain data in the form of procedures often known as methods.

Examples:

1. Java
2. Kotlin



Categories Programming Language



Functional programming language:

- Functional programming language defines every computation as a mathematical evaluation. They focus on the programming languages are bound to mathematical calculations.

Examples:

1. Clean
2. Haskell

Compiled Programming language:

- A compiled programming is a programming language whose implementation are typically compilers and not interpreters.
- It will produce a machine code from source code.

Examples:

1. C, C++, JAVA, C#



Categories Programming Language



Procedural programming language:

- Procedural (imperative) programming implies specifying the steps that the programs should take to reach to an intended state.
- Procedures help in the reuse of code.
- Procedural programming makes the programs structured and easily traceable for program flow.

Examples:

1. Hyper talk
2. MATLAB

Scripting language:

- Scripting language are programming languages that control an application.
- Scripts can execute independent of any other application.
- control and are used to automate frequently executed tasks like communicating with external program.

Examples:

1. VB script



Categories Programming Language



Markup languages:

- A markup language is an artificial language that uses annotations to text that define how the text is to be displayed.

Examples:

1. HTML
2. XML

Concurrent programming language:

- Concurrent programming is a computer programming technique that provides for the execution of operation concurrently, either within a single computer or across a number of systems.

Examples:

1. Joule
2. Limbo

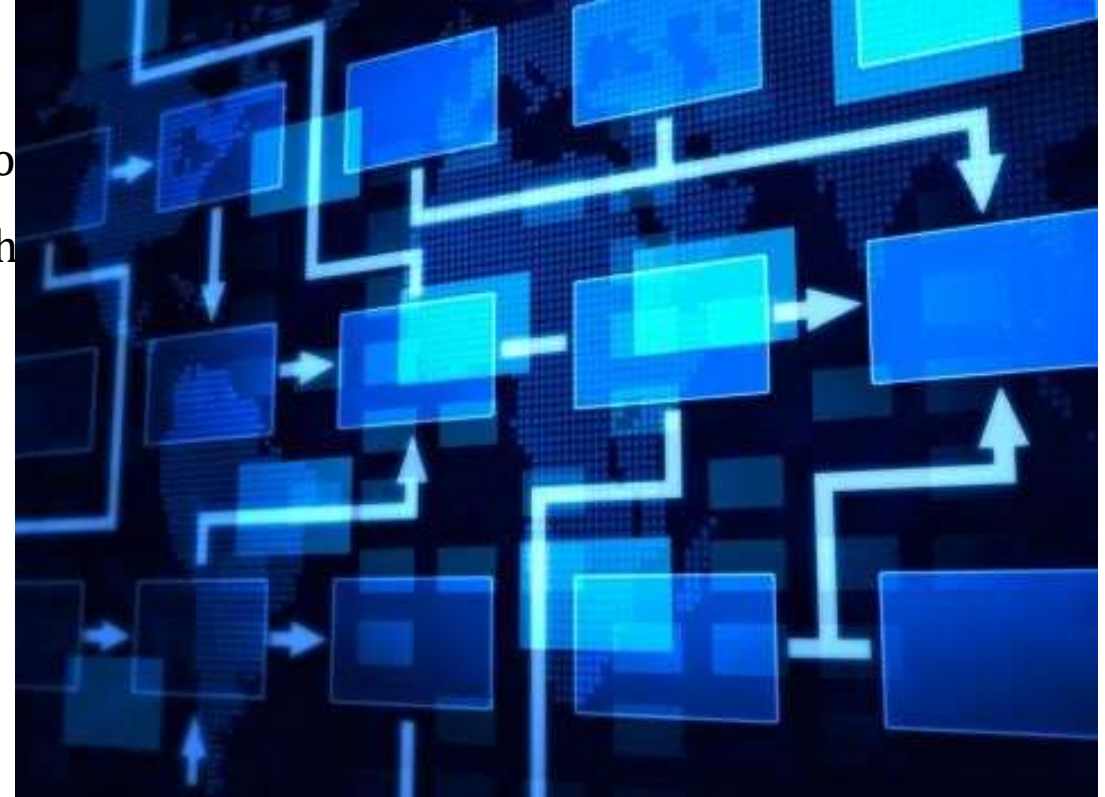


Flow Chart










What is Flow Chart?

- Flow chart is defined as graphical representation of the logic for problem solving.
- The purpose of flowchart is making the logic a visual representation.





Flow Chart

Symbol	Name	Function
	Process	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.



Flow Chart



Rules for drawing a flowchart

- The flowchart should be clear, neat and easy to follow.
- The flowchart must have a logical start and finish.
- Only one flow line should come out from a process symbol.
- Only one flow line should enter a decision symbol.
- two or three flow lines may leave the decision symbol
- Only one flow line is used with a terminal symbol.
- Intersection of flow lines should be avoided.



Flow Chart



Advantages of flowchart:

1. Communication
2. Effective analysis
3. Proper documentation
4. Efficient Coding
5. Proper Debugging
6. Efficient Program Maintenance

Disadvantages of flowchart:

1. Complex logic
2. Alterations and Modifications
3. Reproduction
4. Cost



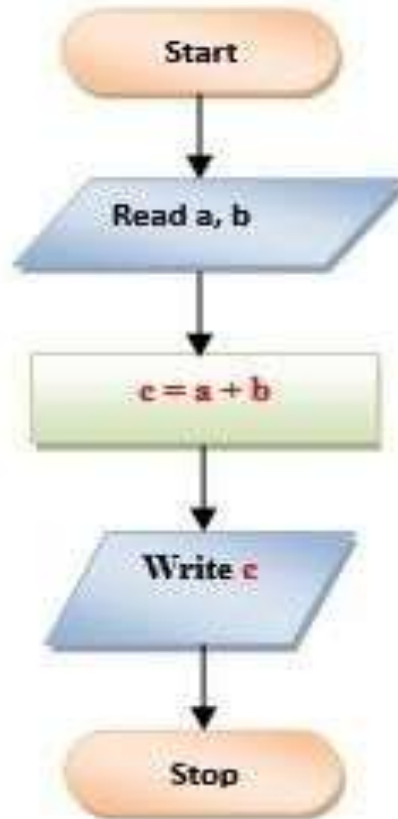
Flow Chart

To find sum of two numbers

Algorithm

1. Start
2. Read a, b
3. $c = a + b$
4. Print or display c
5. Stop

Flowchart



Program

```
#include<stdio.h>

int main()
{
    int a, b, c;

    printf("Enter value of a: ");
    scanf("%d", &a);

    printf("Enter value of b: ");
    scanf("%d", &b);
    c = a+b;

    printf("Sum of given two numbers is: %d", c);

    return 0;
}
```



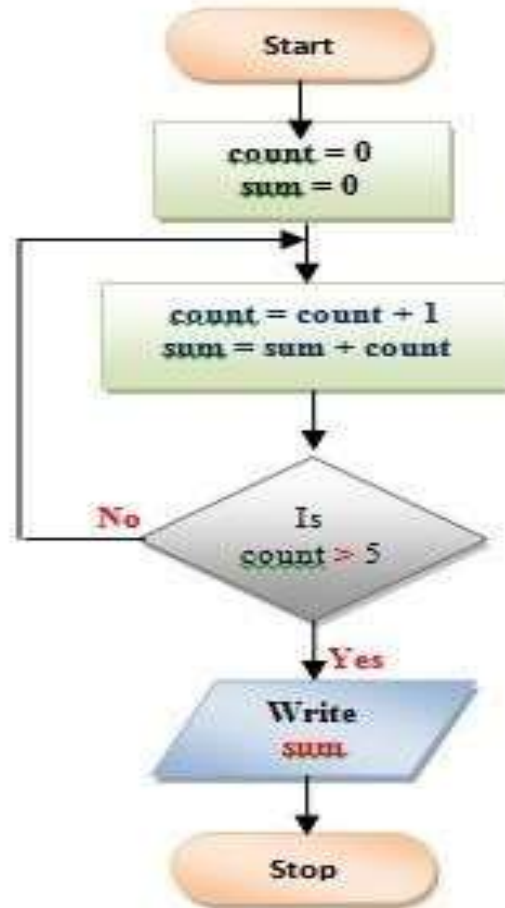

Flow Chart

Find the Sum of First Five Natural Numbers

Algorithm

1. Start
2. Initialize count = 0, sum = 0
3. count = count + 1
4. sum = sum + count
5. Repeat steps 3,4 until count > 5
6. Print sum
7. Stop

Flowchart



Program

```
#include<stdio.h>

int main()
{
    int count, sum;
    sum = 0;

    for (count = 1; count<=5; count++)
    {
        sum = sum +count;
    }

    printf("Sum of 1st 5 numbers is: %d", sum);
    return 0;
}
```



Pseudo Code



What is Pseudo Code?

- Pseudo code consists of short, readable and formally styled English languages used for explain an algorithm.
- It does not include details like variable declaration, subroutines.
- It is easier to understand for the programmer or non programmer to understand the general working of the program.
- It is not a machine readable
- Pseudo code can't be compiled and executed.
- No standard syntax.



Pseudo Code

Guidelines for writing pseudo code:

- Write one statement per line
- Capitalize initial keyword
- End multiline structure
- Keep statements language independent

Common keywords used in pseudocode

begin ... end: These keywords are used to start and finish pseudocode.

Begin is the first line and end is the last line of pseudocode.

accept: This keyword is used to obtain an input from a user.

display: This keyword is used to present a result or an output.

if ... else... endif: These keywords are used in decision-making.

//: Comment

Do ... while, for ..., repeat ... until: Represent loop



Pseudo Code

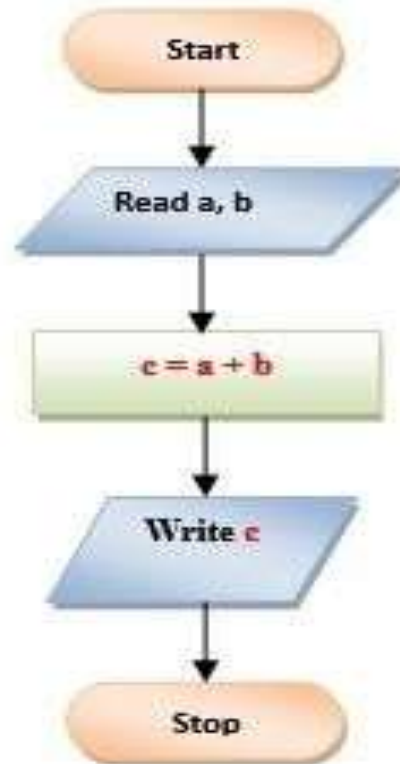
Example for Sequence Method:

To find sum of two numbers

Pseudo code

BEGIN
GET a,b
ADD $c=a+b$
PRINT c
END

Flowchart



Program

```
#include<stdio.h>

int main()
{
    int a, b, c;

    printf("Enter value of a: ");
    scanf("%d", &a);

    printf("Enter value of b: ");
    scanf("%d", &b);
    c = a+b;

    printf("Sum of given two numbers is: %d", c);

    return 0;
}
```



Pseudo Code

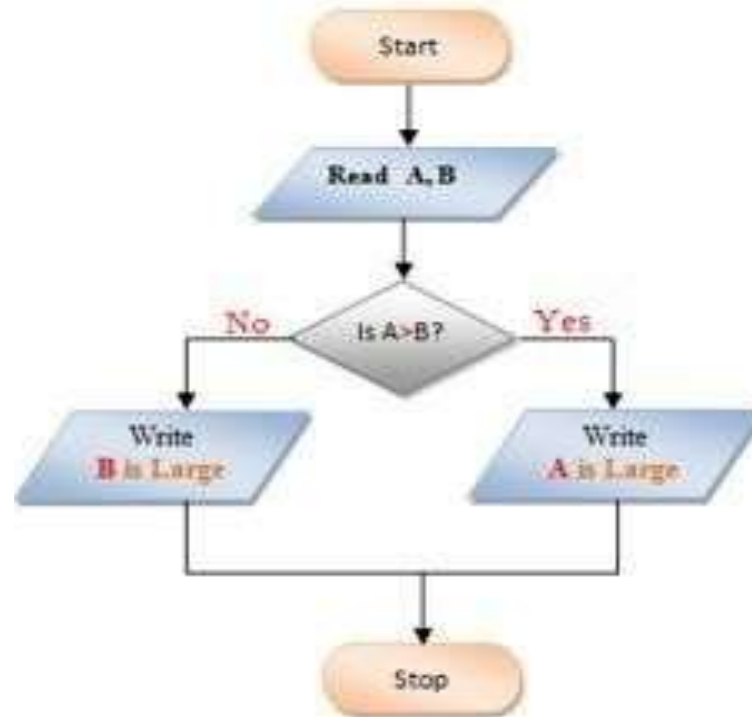
Example for Selection Method:

Greatest of two numbers

Pseudocode

```
PROGRAM PrintBiggerOfTwo:  
  Read A;  
  Read B;  
  IF (A>B)  
    THEN Print A;  
    ELSE Print B;  
  ENDIF;  
END.
```

Flowchart



Program

```
#include<stdio.h>  
  
int main()  
{  
  int A, B;  
  
  printf("Enter values of A, B: ");  
  scanf("%d %d", &A, &B);  
  
  if (A>B)  
    printf("A is Larger");  
  else  
    printf("B is Larger");  
  
  return 0;  
}
```



Pseudo Code

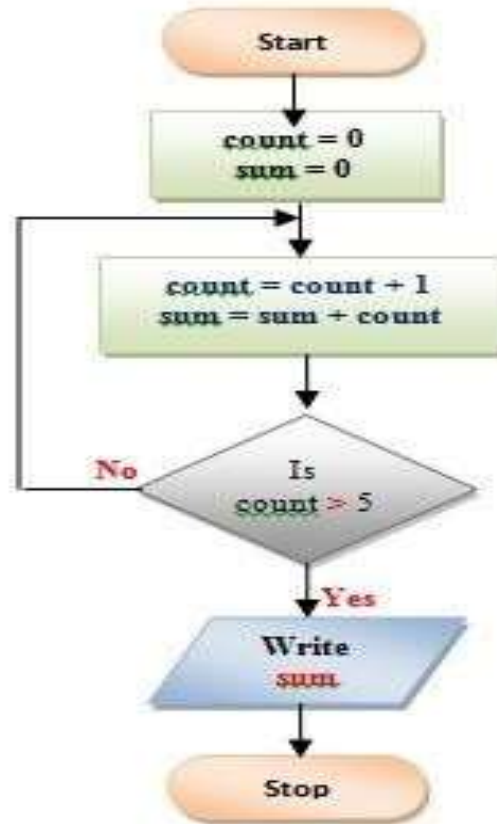
Example for Iteration Method:

Find the Sum of First Five Natural Numbers

Pseudo code

```
BEGIN
NUMBER counter, sum=0
FOR counter=1 TO 100 STEP 1 DO
    sum=sum+counter
ENDFOR
OUTPUT sum
END
```

Flowchart



Program

```
#include<stdio.h>
int main()
{
    int count, sum;
    sum = 0;
    for (count = 1; count<=5; count++)
    {
        sum = sum +count;
    }
    printf("Sum of 1st 5 numbers is: %d", sum);
    return 0;
}
```



Comparisons



Algorithm	Flowchart	Pseudo code
An algorithm is a sequence of instructions used to solve a problem	It is a graphical representation of algorithm	It is a language representation of algorithm.
User needs knowledge to write algorithm.	not need knowledge of program to draw or understand flowchart	Not need knowledge of program language to understand or write a pseudo code.



Pseudo Code



What is Pseudo Code?

- Pseudo code consists of short, readable and formally styled English languages used for explain an algorithm.
- It does not include details like variable declaration, subroutines.
- It is easier to understand for the programmer or non programmer to understand the general working of the program.
- It is not a machine readable
- Pseudo code can't be compiled and executed.
- No standard syntax.



Pseudo Code

Guidelines for writing pseudo code:

- Write one statement per line
- Capitalize initial keyword
- End multiline structure
- Keep statements language independent

Common keywords used in pseudocode

begin ... end: These keywords are used to start and finish pseudocode.

Begin is the first line and end is the last line of pseudocode.

accept: This keyword is used to obtain an input from a user.

display: This keyword is used to present a result or an output.

if ... else... endif: These keywords are used in decision-making.

//: Comment

Do ... while, for ..., repeat ... until: Represent loop



Pseudo Code

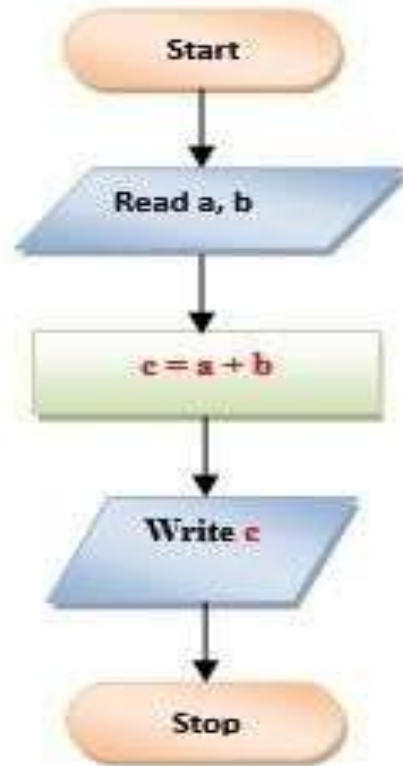
Example for Sequence Method:

To find sum of two numbers

Pseudo code

BEGIN
GET a,b
ADD $c=a+b$
PRINT c
END

Flowchart



Program

```
#include<stdio.h>

int main()
{
    int a, b, c;

    printf("Enter value of a: ");
    scanf("%d", &a);

    printf("Enter value of b: ");
    scanf("%d", &b);
    c = a+b;

    printf("Sum of given two numbers is: %d", c);

    return 0;
}
```



Pseudo Code

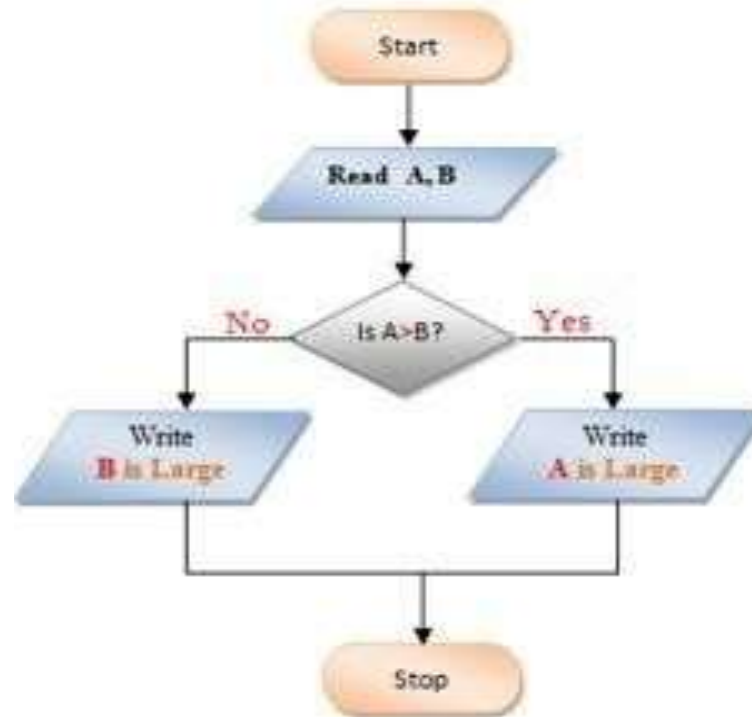
Example for Selection Method:

Greatest of two numbers

Pseudocode

```
PROGRAM PrintBiggerOfTwo:  
  Read A;  
  Read B;  
  IF (A>B)  
    THEN Print A;  
    ELSE Print B;  
  ENDIF;  
END.
```

Flowchart



Program

```
#include<stdio.h>  
  
int main()  
{  
  
  int A, B;  
  
  printf("Enter values of A, B: ");  
  scanf("%d %d", &A, &B);  
  
  if (A>B)  
    printf("A is Larger");  
  else  
    printf("B is Larger");  
  
  return 0;  
}
```



Pseudo Code

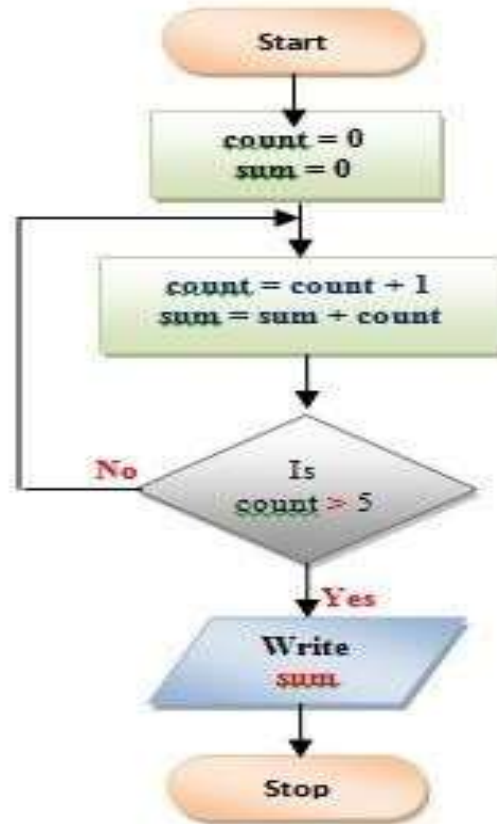
Example for Iteration Method:

Find the Sum of First Five Natural Numbers

Pseudo code

```
BEGIN
NUMBER counter, sum=0
FOR counter=1 TO 100 STEP 1 DO
    sum=sum+counter
ENDFOR
OUTPUT sum
END
```

Flowchart



Program

```
#include<stdio.h>
int main()
{
    int count, sum;
    sum = 0;
    for (count = 1; count<=5; count++)
    {
        sum = sum +count;
    }
    printf("Sum of 1st 5 numbers is: %d", sum);
    return 0;
}
```



Comparisons



Algorithm	Flowchart	Pseudo code
An algorithm is a sequence of instructions used to solve a problem	It is a graphical representation of algorithm	It is a language representation of algorithm.
User needs knowledge to write algorithm.	not need knowledge of program to draw or understand flowchart	Not need knowledge of program language to understand or write a pseudo code.



Programming Language



- A programming language is a **set of symbols and rules** for instructing a computer to perform specific tasks.
- The programmers have to follow all the specified rules before writing program using programming language.
- The user has to communicate with the computer using language which it can understand.

Program = Algorithm + Data

• Need for Programming Languages

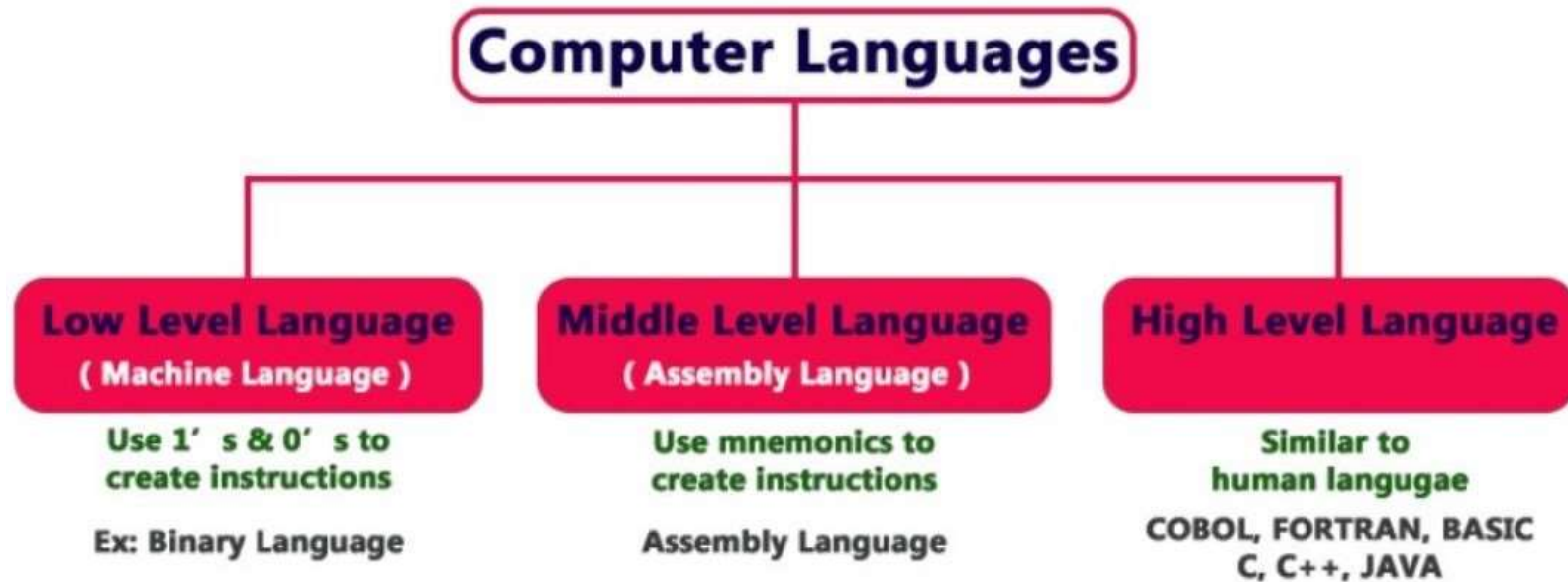
- Used to organize the computation.
- We can solve different problems.
- To improve the efficiency of the programs.



Programming Language

Types of programming language

- Low – level or Machine Language
- Intermediate or Assembly Language
- High – level or Programming language





Machine Language

Machine language:

- Machine language is the lowest-level programming language.
- Machine languages are the only languages understood by computers.
- It is also called as low level language.
- In machine language the different instructions are formed by taking different combinations of 0's and 1's.

Example code: 100110011

111001100

Advantages:

- Translation free
 - The program written in machine language can be executed directly on computer.
 - In this case any conversion process is not required.
- High speed
 - The conversion time is saved, the execution of machine language program is extremely fast.

Disadvantage:

- It is hard to find errors in a program written in the machine language.
- Writing program in machine language is a time consuming process.
- Machine dependent: According to architecture used, the computer differs from each other.



Assembly Language



Assembly Language:

- To overcome the issues in programming language and make the programming process easier, an assembly language is developed which is logically equivalent to machine language but it is easier for people to read, write and understand.
- Assembly language is symbolic representation of machine language.
- Assembly languages are symbolic programming language that uses symbolic notation to represent machine language instructions.
- They are called low level language because they are so closely related to the machines.
- An assembly language contains the same instructions as a machine language, but the instructions and variables have names instead of being just numbers.
- An assembly language consists of mnemonics, mnemonics that corresponds unique machine instruction.

Example code: start

Add x,y

Sub x,y



Assembly Language



Assembler:

- Assembler is the program which translates assembly language instruction into a machine language.
 - Easy to understand and use.
 - It is easy to locate and correct errors.

Disadvantages:

- **Machine dependent:**
 - The assembly language program which can be executed on the machine depends on the architecture of that computer.
- **Hard to learn:**
 - It is machine dependent, so the programmer should have the hardware knowledge to create applications using assembly language.
- **Less efficient :**
 - Execution time of assembly language program is more than machine language program.
 - Because assembler is needed to convert from assembly language to machine language.



High Level Language

High – level Language:

- High level language contains English words and symbols.
- The specified rules are to be followed while writing program in high level language.
- The **interpreter or compilers** are used for converting these programs into a machine readable form.
- A high-level language (HLL) is 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 from machine languages.
- Ultimately, programs written in a high-level language **must be translated into machine language** by a compiler or interpreter.

Example code: `printf("Hello World!")`



High Level Language



Translating high level language to machine language:

- The programs that translate high level language in to machine language are called [interpreter or compiler](#).

Interpreter:

- Interpreter translates the high level language program in line by line manner.
- The interpreter translates a high level language statement in a source program to a machine code and executes it immediately before translating the next statement.
- When an error is found the execution of the program is halted and error message is displayed on the screen.
- Ex :Pascal, Python

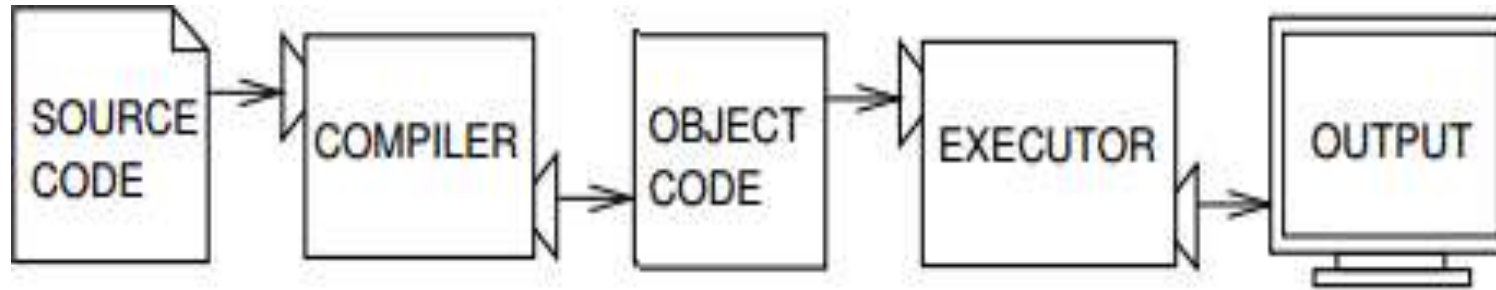




High Level Language

Compiler:

- A compiler is a program which translates the source code written in a high level language in to object code which is in machine language program.
- Compiler reads the whole program written in high level language and translates it to machine language.
- If any error is found it display error message on the screen.
- Ex: C, C++, JAVA





High Level Language

Advantages

•**Readability**

- High level language is closer to natural language so they are easier to learn and understand

•**Machine independent**

- High level language program have the advantage of being portable between machines.

•**Easy debugging**

- Easy to find and correct error in high level language

Disadvantages

•**Less efficient**

- The translation process increases the execution time of the program.
- Programs in high level language require more memory and take more execution time to execute.



High Level Language



High Level Language are divided into following categories:

Language Type	Example
1. Interpreted programming languages	Pascal, Python
2. Functional programming languages	Clean, Haskell
3. Compiled programming languages	C, C++, C#, JAVA
4. Procedural programming languages	Hyper talk, MATLAB
5. Scripting programming language	Apple script, VB script
6. Mark-up programming language	HTML, XML
7. Concurrent programming language	Joule, Limbo
8. Object oriented programming language	Lava, Moto, C++, JAVA



Thank
you

THANK YOU