



Sukkur IBA University Kandhkot Campus



C++ LAB MANUAL

Department of Computer Science & Software Engineering



S.NO	CONTENT
1	Basic Terms (Program, Programming Language)
2	Introduction to computer programming language
3	Levels of Programming languages
4	Introduction to C++ Programming language
5	Historical background of C++
6	Why C++?
7	Structure of C++ program
8	Syntax
9	Data Types
10	Types of data-types
11	Simple Program to find the size of data-types
12	Operators
13	Types of operators
14	Loops and their types
15	Nested loops
16	Jumping out of loop
17	Jump statement
18	Break statement
19	Flow chart
20	Conditions
21	Functions
22	Arrays (One dimensional and 2-dimensional)
23	Pointers and their Usage

PROGRAM

A programming language is a formal language which comprises a set of instructions used to produce various kinds of output. Programming languages are used to create programs that implement specific algorithms

COMPILER

00000 10011110

Computers understand only one language and that language consists of sets of instructions made of ones and zeros. This computer language is appropriately called machine language.

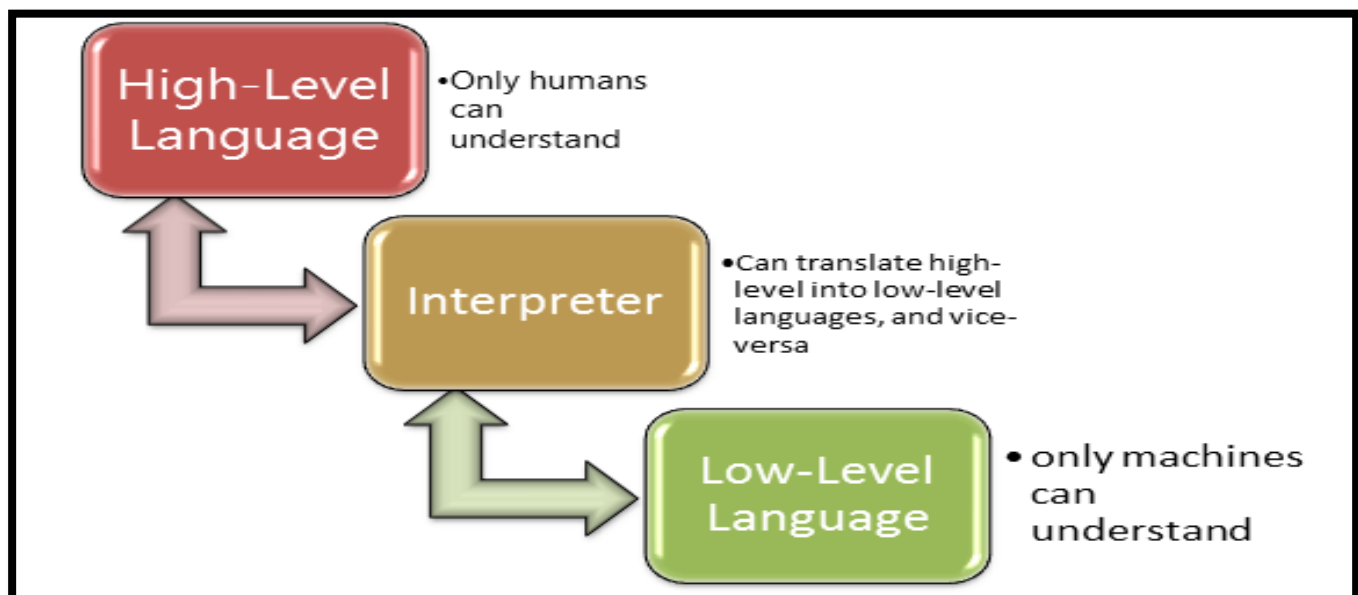
A single instruction to a computer could look like this:

ASSEMBLY LANGUAGE

An assembly language, often abbreviated asm, is any low-level programming language, in which there is a very strong correspondence between the assembly program statements and the architecture's machine code instructions

ASSEMBLER

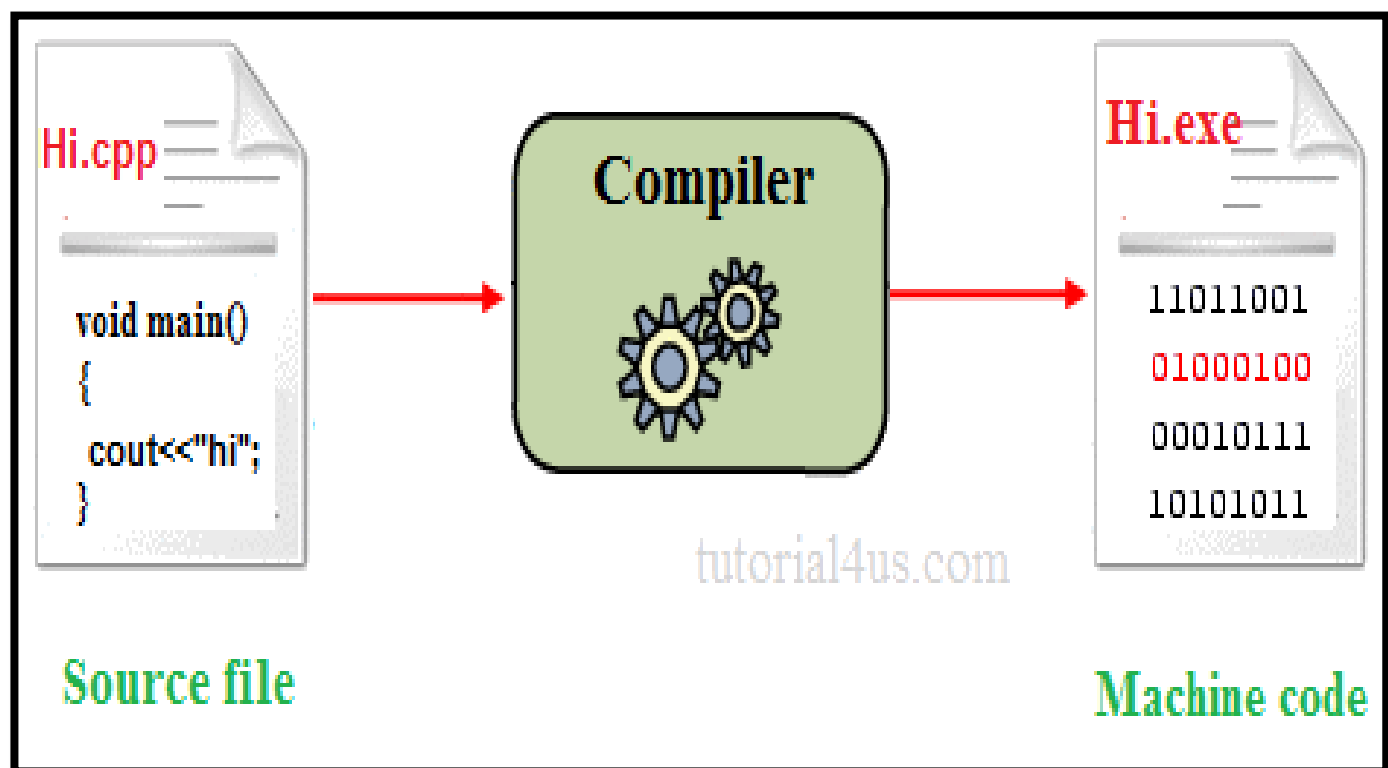
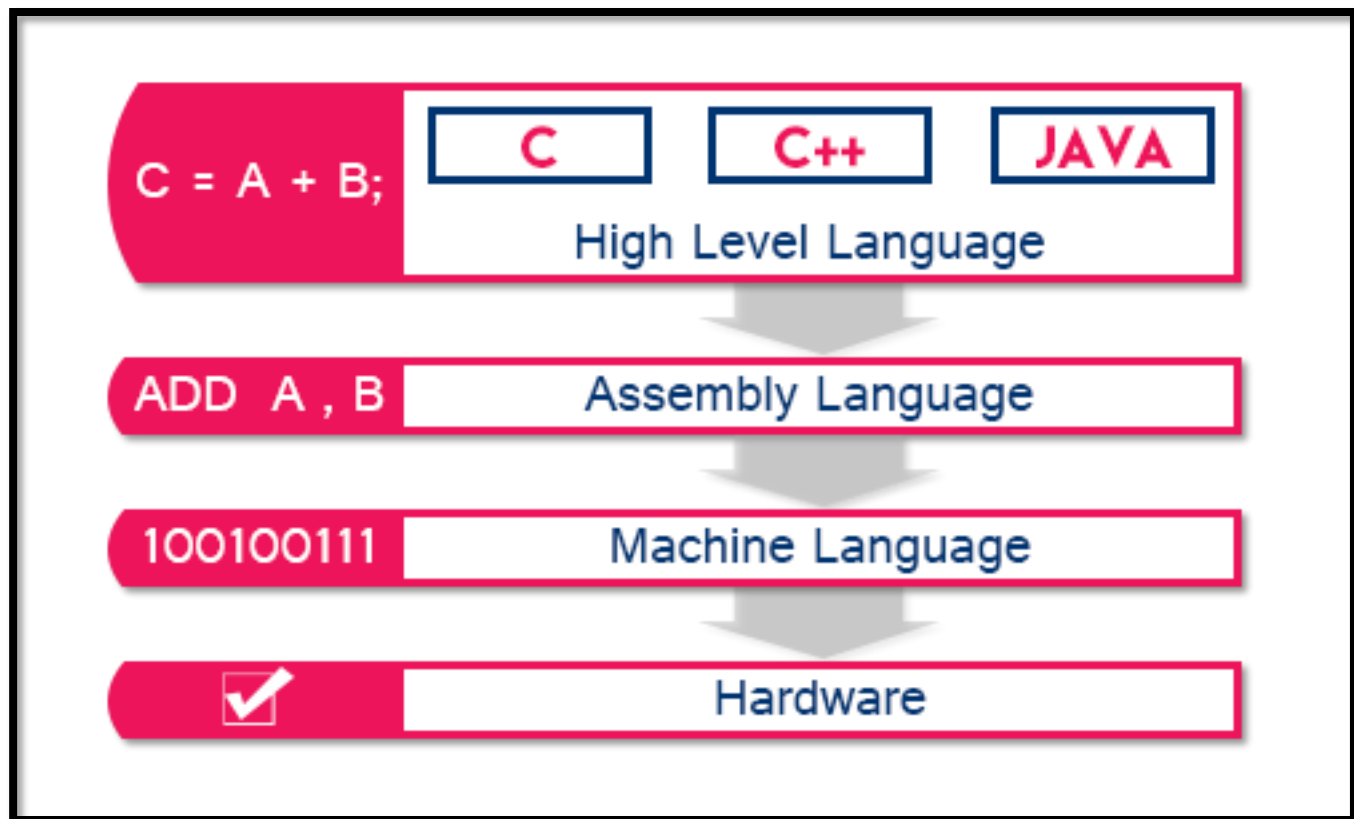
An assembler is a program that takes basic computer instructions and converts them into a pattern of bits that the computer's processor can use to perform its basic operations. Some people call these instructions assembler language and others use the term assembly language.



High Level Language	Low Level Language
These are Interpreted	Direct memory management
They have open classes and message-style methods which are known as Dynamic constructs	Hardware has extremely little abstraction which is actually close to having none.
Poor performance	Much fast than high level
Codes are Concise	Statements correspond directly to clock cycles
Flexible syntax and easy to read	Superb performance but hard to write
Is object oriented and functional	Few support and hard to learn
Large community	

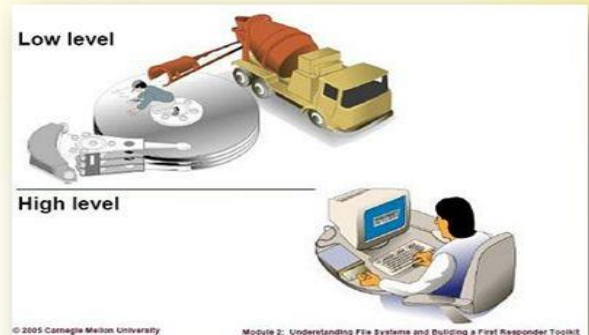
Computer programming

- ❑ Computer Programming is defined as the process of creating computer software using a programming Language. Computer programs are written by Human individuals(Programmers)
- ❑ A **computer program** is a step by step set of instructions that a computer has to work through in a logical sequence in order to carry out a particular task. The computer executes these instructions (obeys the instructions) when told to do so by the user.



Contrast

- **Low Level Languages**
 - Very close to machine language
 - Concentrate on machine architecture
- **High Level Languages**
 - Machine-independent programming language
 - Concentrate of the logic of problem



**High-level
programming language**

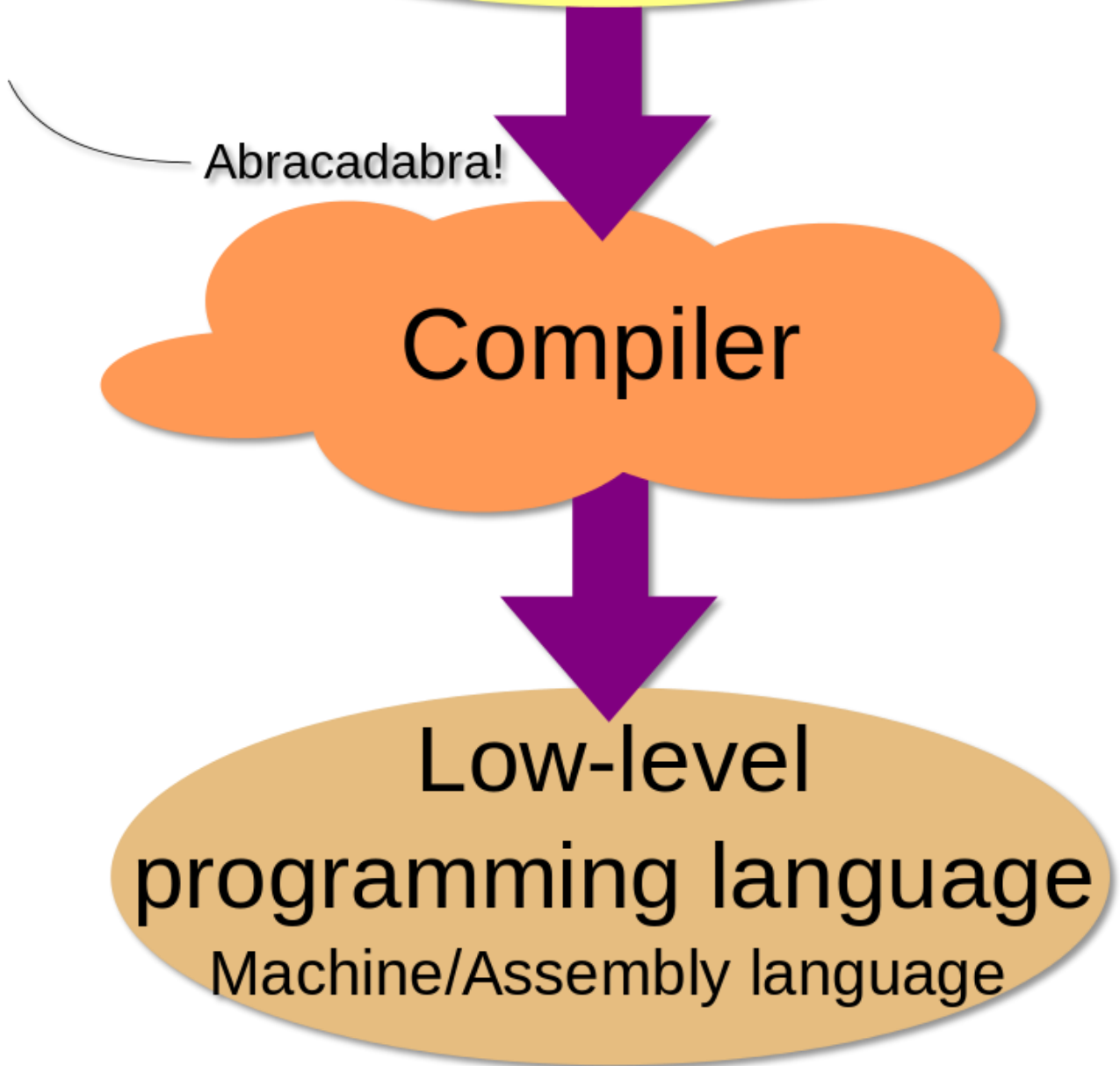
C, Pascal, Java, Python...

Abracadabra!

Compiler

**Low-level
programming language**

Machine/Assembly language



High Vs. Low Level Languages

Human languages

E.g., English, French, Spanish,
Chinese, German, Arabic etc.

High level
programming
language

E.g., Python, Java, C++

for (i = 1; i <= 10; i++)

Low level
programming
language

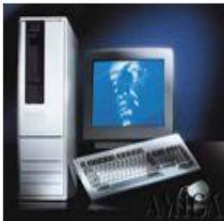
Assembly

MOV #10, R0

Machine language

Binary

10100000 1010 00



Computer hardware

High
level

Low
level

High-Level Languages
(C++, Java, FORTRAN, BASIC, etc.)

Assembly Language

Machine Language

Hardware

Programming Languages

- When computers were first invented they had to be programmed in their own natural language, i.e. *binary machine code*.
- Later, programmers developed more sophisticated systems to make programming easier and more efficient.
- **Assembly languages** were developed as the **2nd generation** of languages.
- **High-level languages** (3rd & 4th generation) later made programming even more productive.

Assembly Languages

Advantages

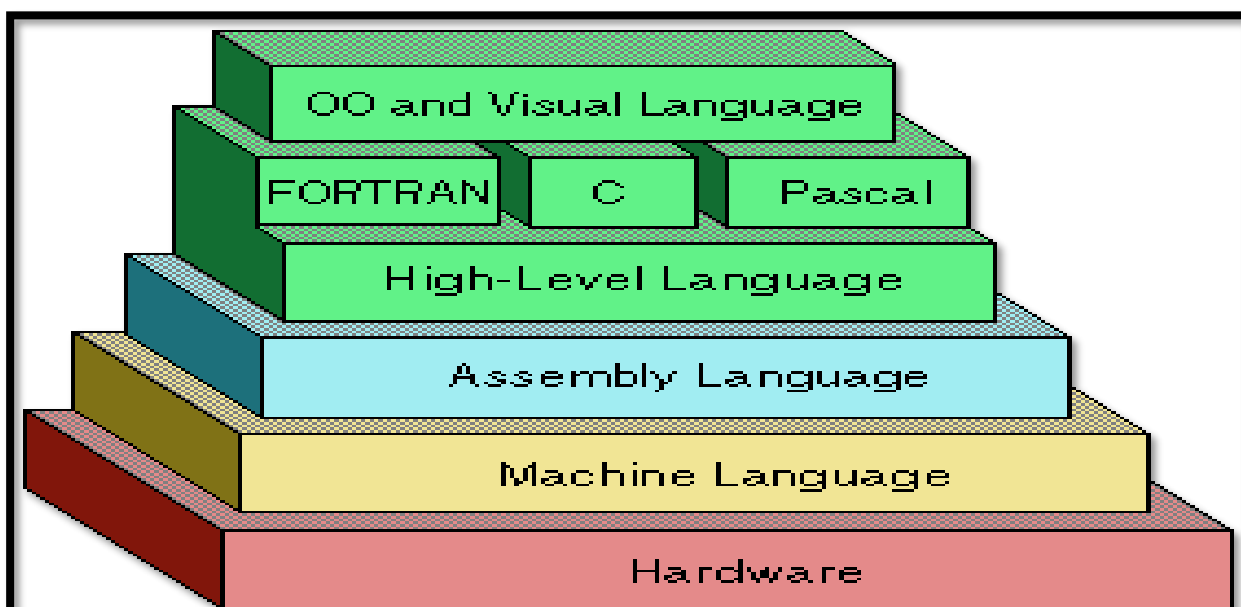
- More standardize and easier than machine languages
- Operate efficiently not so efficient than machine languages
- Easy to debug programs

Disadvantages

- Very long
- Complex
- Dependant with machine languages

Levels/Generations of Programming Languages

- 1st Generation Programming language (1GL)
 - Machine Language: 0s or 1s
- 2nd Generation Programming language (2GL)
 - Assembly Language : Mnemonics
- 3rd Generation Programming language (3GL)
 - High-Level Languages ; (procedure oriented or Object Oriented)
- 4th Generation Programming language (4GL)
 - Very-High-Level Languages
- 5th Generation Programming Language
 - Natural Languages

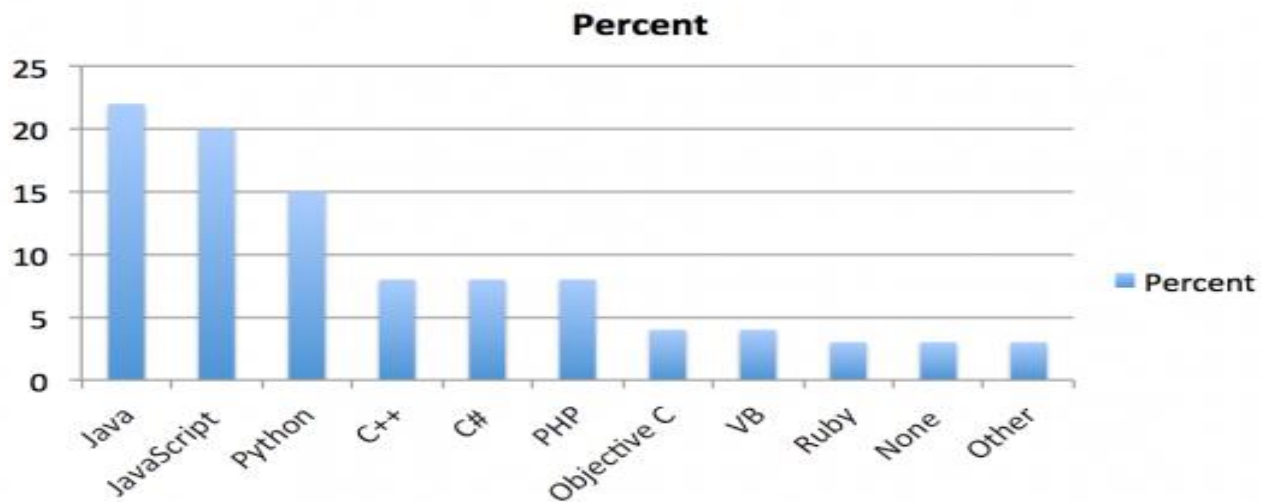


Programming Languages

- ▶ Two broad groups
 - Traditional programming languages
 - ▢ Sequences of instructions
 - ▢ First, second and some third generation languages
 - Object-oriented languages
 - ▢ Objects are created rather than sequences of instructions
 - ▢ Some third generation, and fourth and fifth generation languages



Most Common Programming languages tech writers know



www.technicalprogramming.com

Sr No	Compiler	Interpreter
1	Compiler Takes Entire program as input	Interpreter Takes Single instruction as input .
2	Intermediate Object Code is Generated	No Intermediate Object Code is Generated
3	Conditional Control Statements are Executes faster	Conditional Control Statements are Executes slower
4	Memory Requirement : More (Since Object Code is Generated)	Memory Requirement is Less
5	Program need not be compiled every time	Every time higher level program is converted into lower level program
6	Errors are displayed after entire program is checked	Errors are displayed for every instruction interpreted (if any)
7	Example : C Compiler	Example : BASIC

DISADVANTAGE OF LLL:-

- 1.LLL INSTRUCTION ARE DIFFICULT TO READ, WRITE AND UNDERSTAND.
- 2.DIFFICULT TO UPDATE THE INSTRUCTION.
- 3.DIFFICULT TO REMEMBER THE ADDRESS SEQUENCE OF INSTRUCTION.
- 4.PROGRAMMING METHODOLOGY VERY FROM MACHINE TO MACHINE.
- 5.ONLY SPECIALIST CAN BE ABLE TO OPERATE THESE LANGUAGES.





High level language

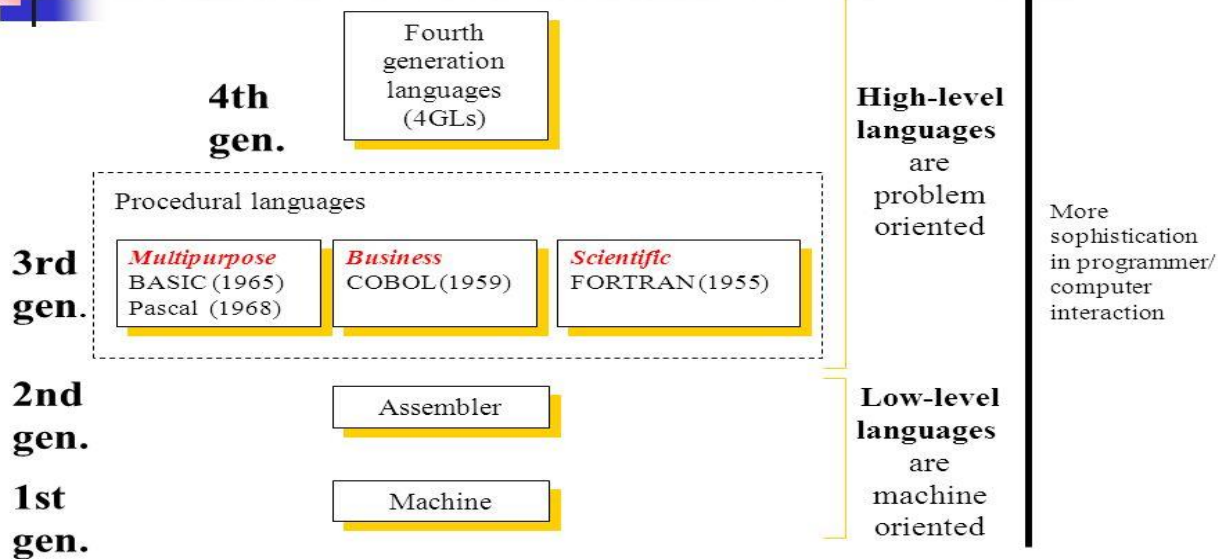
Advantage

- Easier to write
- Easier to read
- Easier to maintain
- Portable : can work across different CPU families
Supports a wide range of data types.

Disadvantage

- Usually slower than lower-level languages, for example assembler is faster than C which is faster than C++ which is faster than c# or Java (on the same computer, of course)

The hierarchy of programming languages



Three types of computer languages

- High-level languages

- Similar to everyday English, use common mathematical notations
- Single statements accomplish substantial tasks
 - Assembly language requires many instructions to accomplish simple tasks
- Translator programs (compilers)
 - Convert to machine language
- Interpreter programs
 - Directly execute high-level language programs
- Example:

```
grossPay = basePay + overTimePay
```

3

TRANSLATOR

A translator or programming language processor is a computer program that performs the translation of a program written in a given programming language into a functionally equivalent program in another computer language (the target language), without losing the functional or logical structure of the original code

KEYWORDS

C Keywords. Keywords are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier.

Example:

break enum register typedef void if int string

INSTRUCTION

An instruction is an order given to a computer processor by a computer program. ... In assembler language, a macro instruction is one that, during processing by the assembler program, expands to become multiple instructions (based on a previously coded macro definition).

PROGRAMMING PROCESS

There are five main ingredients in the programming process:

- Defining the problem.
- Planning the solution.
- Coding the program.
- Testing the program.
- Documenting the program.

LOGIC

Logic programming is a computer programming paradigm in which program statements express facts and rules about problems within a system of formal logic. Rules are written as logical clauses with a head and a body; for instance, "H is true if B1, B2, and B3 are true." Facts are expressed similar to rules, but without a body; for instance, "H is true."

Some logic programming languages such as Data log and Answer Set Programming (ASP) are purely declarative — they allow for statements about what the program should accomplish, with no explicit step-by-step instructions about how to do so. Others, such as Prolog, are a combination of declarative and imperative — they may also include procedural statements such as "To solve H, solve B1, B2, and B3.

- Think to solve
- Practice

FLOWCHART

A flowchart is a type of diagram that represents an algorithm, workflow or process. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields

Introduction to C++

C++, as we all know is an extension to C language and was developed by Bjarne Stroustrup at Bell Labs. C++ is an intermediate level language, as it comprises a combination of both high level and low level language features. C++ is a statically typed, free form, multi-paradigm, compiled general-purpose language.

C++ is an Object Oriented Programming language but is not purely Object Oriented. Its features like Friend and Virtual, violate some of the very important OOPS features, rendering this language unworthy of being called completely Object Oriented. It's a middle level language.

Benefits of C++ over C Language

The major difference being OOPS concept, C++ is an object oriented language whereas C language is a procedural language. Apart from this there are many other features of C++ which give this language an upper hand on C language.

Background History:

- C++ Development started in 1979.
- During the creation of Ph.D. thesis, Bjarne Stroustrup worked with language called Simula.
- Simula is basically useful for the simulation work.
- Simula was first language to support object-oriented programming paradigm
- Bjarne Stroustrup identified that this OOP features can be included in the software development.
- After that Bjarne Stroustrup started working on the C language and added more extra OOP features to the classic C.
- He added features in such a fashion that the basic flavour of C remains unaffected.
- C++ includes some add-on features such as classes, basic inheritance, in-lining, default function arguments, and strong type checking

Basic History of C++

- During 1970 Dennis Ritchie created C Programming language.
- In the early 1980's, also at Bell Laboratories, another programming language was created which was based upon the C language.
- C++ is also called as C with classes
- Stroustrup states that the purpose of C++ is to make writing good programs easier and more pleasant for the individual programmer.
- C++ programming language is extension to C Language.
- In C we have already used increment operator (++). Therefore we called C++ as "Incremented C" means Extension to C.

How do Java and C# create cross-platform portable programs, and why can't C++ ?

- Why C++ is not cross platform dependent?
- Why C++ is not a portable?
- How Java is portable?
- What makes Java and C# portable?

Reason :

Consider Scenario of C++ Language

- Portability of any language depends on the object code created by compiler.
- C++ Compiler produces machine code which is directly executed by the CPU.
- C++ Code is thus machine dependent and tied to particular Operating System (OS).
- If we try to execute C++ program on the another Operating C++ then we must recompile program before executing program.

Consider Scenario of Java Programming Language

- Java is created by Sun Micro System. [See : What is Java ?]
- Java Compiler produces Intermediate code called Byte Code.
- Byte Code i.e intermediate code is executed by the Run Time Environment.
- In Java Run Time Environment is called as “JVM” [Java Virtual Machine]
- If we have JVM already installed on any platform then JVM can produce machine dependent Code based on the intermediate code.

Consider Scenario of C# Programming Language

- C# was created by Microsoft.
- C# compiler produces intermediate code called as MSIL. (MicroSoft Intermediate Language).
- MSIL i.e intermediate code is executed by CLR (Common Language Run Time)
- If we have CLR already implemented on any platform then CLR can produce machine dependent Code based on the intermediate code.

C++ Design Aims Why C++ was designed ?

- C++ makes programming more enjoyable for serious programmers.
- C++ is a general-purpose programming language that
 - is a better C
 - supports data abstraction
 - supports object-oriented programming

Following features of C++ makes it a stronger language than C,

1. There is Stronger Type Checking in C++.

2. All the OOPS features in C++ like Abstraction, Encapsulation, Inheritance etc makes it more worthy and useful for programmers.
3. C++ supports and allows user defined operators (i.e Operator Overloading) and function overloading is also supported in it.
4. Exception Handling is there in C++.
5. The Concept of Virtual functions and also Constructors and Destructors for Objects.
6. Inline Functions in C++ instead of Macros in C language. Inline functions make complete function body act like Macro, safely.
7. Variables can be declared anywhere in the program in C++, but must be declared before they are used.

Syntax and Structure of C++ program:

Here we will discuss one simple and basic C++ program to print "Hello this is C++" and its structure in parts with details and uses.

First C++ program

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
cout << "Hello this is C++";
```

```
}
```

C++ Data Types:

All variables use data-type during declaration to restrict the type of data to be stored. Therefore, we can say that data types are used to tell the variables the type of data it can

Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data-type with which it is declared. Every data type requires different amount of memory.

Data types in C++ is mainly divided into two types:

Primitive Data Types: These data types are built-in or predefined data types and can be used directly by the user to declare variables. example: int, char , float, bool etc. Primitive data types available in C++ are:

- Integer
- Character
- Boolean
- Floating Point
- Double Floating Point
- Valueless or Void
- Wide Character

Abstract or user defined data type: These data types are defined by user itself. Like, defining a class in C++ or a structure.

This article discusses primitive data types available in C++.

Integer: Keyword used for integer data types is int. Integers typically requires 4 bytes of memory space and ranges from -2147483648 to 2147483647.

Character: Character data type is used for storing characters. Keyword used for character data type is char. Characters typically requires 1 byte of memory space and ranges from -128 to 127 or 0 to 255.

Boolean: Boolean data type is used for storing boolean or logical values. A boolean variable can store either true or false. Keyword used for boolean data type is bool.

Floating Point: Floating Point data type is used for storing single precision floating point values or decimal values. Keyword used for floating point data type is float. Float variables typically requires 4 byte of memory space.

Double Floating Point: Double Floating Point data type is used for storing double precision floating point values or decimal values. Keyword used for double floating point data type is double. Double variables typically requires 8 byte of memory space.

void: Void means without any value. void datatype represents a valueless entity. Void data type is used for those function which does not returns a value.


Wide Character: Wide character data type is also a character data type but this data type has size greater than the normal 8-bit datatype. Represented by wchar_t. It is generally 2 or 4 bytes long.

Datatype Modifiers: As the name implies, datatype modifiers are used with the built-in data types to modify the length of data that a particular data type can hold. Data type modifiers available in C++ are:

- Signed
- Unsigned
- Short
- Long

Below table summarizes the modified size and range of Built-in data-types when combined with the type modifiers:

Data Types - Summary		
Data Types	Length	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
enum	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	$3.4 * (10^{*-38})$ to $3.4 * (10^{*+38})$
double	64 bits	$1.7 * (10^{*-308})$ to $1.7 * (10^{*+308})$
long double	80 bits	$3.4 * (10^{*-4932})$ to $1.1 * (10^{*+4932})$

 Lectures on Busy Bee Workshop 11

Note : Above values may vary from compiler to compiler. In above example, we have considered GCC 64 bit.

We can display the size of all the data types by using the sizeof() function and passing the keyword of the datatype as argument to this function as shown below:

```
// C++ program to sizes of data types

#include<iostream> using namespace std;

int main()
{
    cout << "Size of char : " << sizeof(char) << " byte" << endl;
    cout << "Size of int : " << sizeof(int) << " bytes" << endl;
```

```

cout << "Size of short int : " << sizeof(short int) << " bytes" << endl;
cout << "Size of long int : " << sizeof(long int) << " bytes" << endl;
cout << "Size of signed long int : " << sizeof(signed long int) << " bytes" << endl;
cout << "Size of unsigned long int : " << sizeof(unsigned long int) << " bytes" << endl;
cout << "Size of float : " << sizeof(float) << " bytes" << endl;
cout << "Size of double : " << sizeof(double) << " bytes" << endl;
cout << "Size of wchar_t : " << sizeof(wchar_t) << " bytes" << endl;

return 0;
}

```

Output:

Size of char : 1 byte

Size of int : 4 bytes

Size of short int : 2 bytes

Size of long int : 8 bytes

Size of signed long int : 8 bytes

Size of unsigned long int : 8 bytes

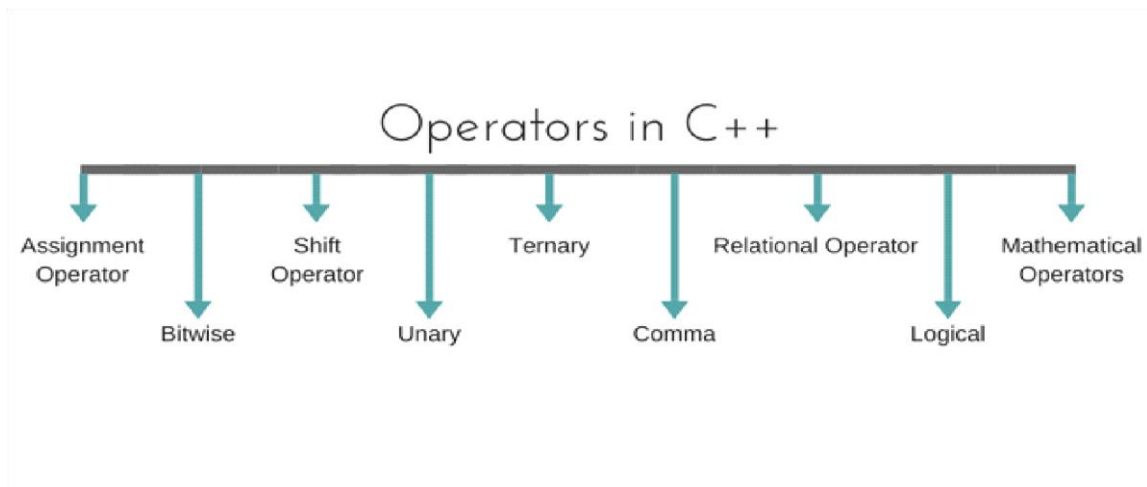
Size of float : 4 bytes

Size of double : 8 bytes

Size of wchar_t : 4 bytes

Operators in C++

Operators are special type of functions, that takes one or more arguments and produces a new value. For example : addition (+), subtraction (-), multiplication (*) etc, are all operators. Operators are used to perform various operations on variables and constants. *operators in C++*



Types Of Operators

1. Assignment Operator
2. Mathematical Operators
3. Relational Operators
4. Logical Operators
5. Bitwise Operators
6. Shift Operators
7. Unary Operators
8. Ternary Operator
9. Comma Operator

Assignment Operator (=)

Operates '=' is used for assignment, it takes the right-hand side (called rvalue) and copy it into the left-hand side (called lvalue). Assignment operator is the only operator which can be overloaded but cannot be inherited.

Mathematical Operators

There are operators used to perform basic mathematical operations. Addition (+) , subtraction (-) , division (/) multiplication (*) and modulus (%) are the basic mathematical operators. Modulus operator cannot be used with floating-point numbers.

C++ and C also use a shorthand notation to perform an operation and assignment at same type. Example,

`int x=10; x += 4 // will add 4 to 10, and hence`

`assign 14 to X. x -= 5 // will subtract 5 from 10`

`and assign 5 to x.`

Relational Operators

These operators establish a relationship between operands. The relational operators are : less than (<) , greater than (>) , less than or equal to (<=), greater than equal to (>=), equivalent (==) and not equivalent (!=).

You must notice that assignment operator is (=) and there is a relational operator, for equivalent (==). These two are different from each other, the assignment operator assigns the value to any variable, whereas equivalent operator is used to compare values, like in if-else conditions, Example

`int x = 10; //assignment operator x=5; // again assignment`

`operator if(x == 5) // here we have used equivalent relational`

`operator, for comparison`

```
{  
  
cout <<"Successfully compared";  
  
}
```

Logical Operators

The logical operators are AND (&&) and OR (||). They are used to combine two different expressions together.

If two statement are connected using AND operator, the validity of both statements will be considered, but if they are connected using OR operator, then either one of them must be valid. These operators are mostly used in loops (especially while loop) and in Decision making.

Bitwise Operators

They are used to change individual bits into a number. They work with only integral data types like char, int and long and not with floating point values.

Bitwise AND operators &

Bitwise OR operator |

And bitwise XOR operator ^

And, bitwise NOT operator ~

They can be used as shorthand notation too, &=, |=, ^=, ~= etc.

Shift Operators

Shift Operators are used to shift Bits of any variable. It is of three types,

Left Shift Operator <<

Right Shift Operator >>

Unsigned Right Shift Operator >>>

Unary Operators

These are the operators which work on only one operand. There are many unary operators, but increment ++ and decrement -- operators are most used.

Other Unary Operators : address of &, dereference *, new and delete, bitwise not ~, logical not !, unary minus - and unary plus +.

Ternary Operator

The ternary if-else ?: is an operator which has three operands.

```
int a = 10; a > 5 ? cout << "true" :
```

```
cout << "false"
```

Comma Operator

This is used to separate variable names and to separate expressions. In case of expressions, the value of last expression is produced and used.

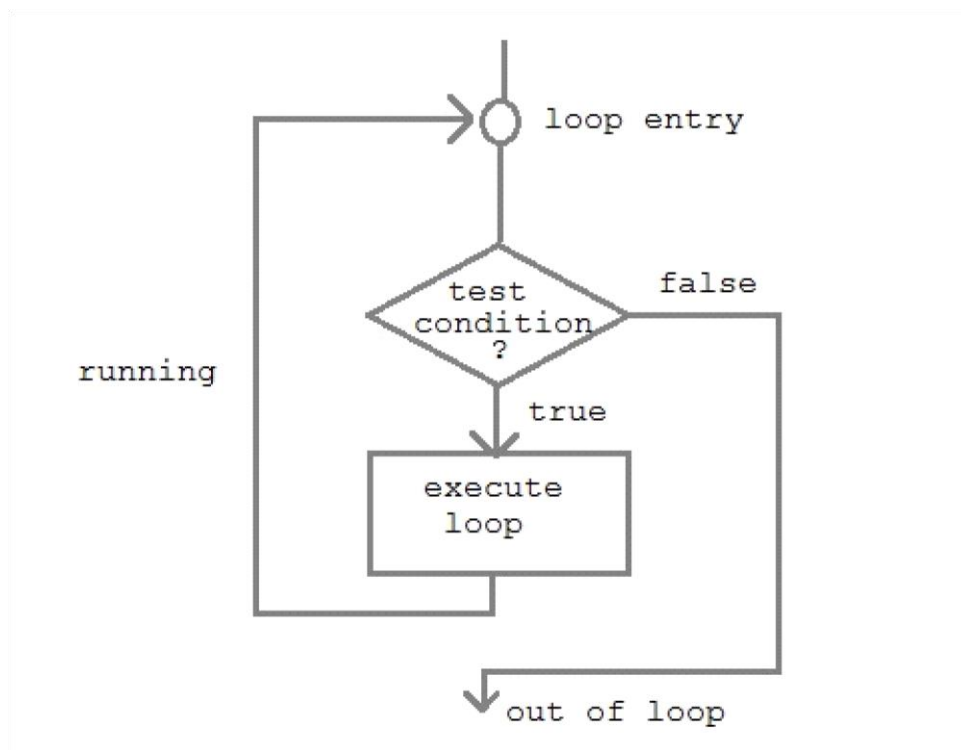
Example :

```
int a,b,c; // variables declaration using comma operator  
a=b++, c++; // a = c++ will be done.
```

Looping in C++

In any programming language, loops are used to execute a set of statements repeatedly until a particular condition is satisfied.

How it works loop flow diagram in C++



A sequence of statement is executed until a specified condition is true. This sequence of statement to be executed is kept inside the curly braces { } known as loop body. After every execution of loop body, condition is checked, and if it is found to be true the loop body is executed again. When condition check comes out to be false, the loop body will not be executed.

There are 3 type of loops in C++ language

1. while loop
2. for loop
3. do-while loop **while loop**

while loop can be address as an entry control loop. It is completed in 3 steps.

Variable initialization.(e.g int x=0;)

condition(e.g while(x<=10))

Variable increment or decrement (x++ or x-- or x=x+2)

Syntax :

variable initialization ;

while (condition)

{

statements ; variable increment or decrement ;

}

for loop

for loop is used to execute a set of statement repeatedly until a particular condition is satisfied. we can say it an open ended loop. General format is,

for(initialization; condition ; increment/decrement)

```
{  
  
    statement-block;  
  
}
```

In for loop we have exactly two semicolons, one after initialization and second after condition. In this loop we can have more than one initialization or increment/decrement, separated using comma operator. for loop can have only one condition.

Nested for loop

We can also have nested for loop, i.e one for loop inside another for loop.

Basic syntax is, For (initialization; condition; increment/decrement)

```
{  
  
    For (initialization; condition; increment/decrement)  
  
        { statement }  
  
}
```

do while loop

In some situations it is necessary to execute body of the loop before testing the condition. Such situations can be handled with the help of do-while loop. do statement evaluates the body of the loop first and at the end, the condition is checked using while statement. General format of do-while loop is,

```
do  
  
{ ....  
  
.....  
}  
  
while(condition);
```

Jumping out of loop

Sometimes, while executing a loop, it becomes necessary to skip a part of the loop or to leave the loop as soon as certain condition becomes true, that is jump out of loop. C language allows jumping from one statement to another within a loop as well as jumping out of the loop.

break statement

When break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop.






continue statement




It causes the control to go directly to the test-condition and then continue the loop process. On encountering continue, cursor leave the current cycle of loop, and starts with the next cycle.

Flowchart is a diagrammatic representation of an algorithm. Flowchart are very helpful in writing program and explaining program to others.

Symbols Used In Flowchart

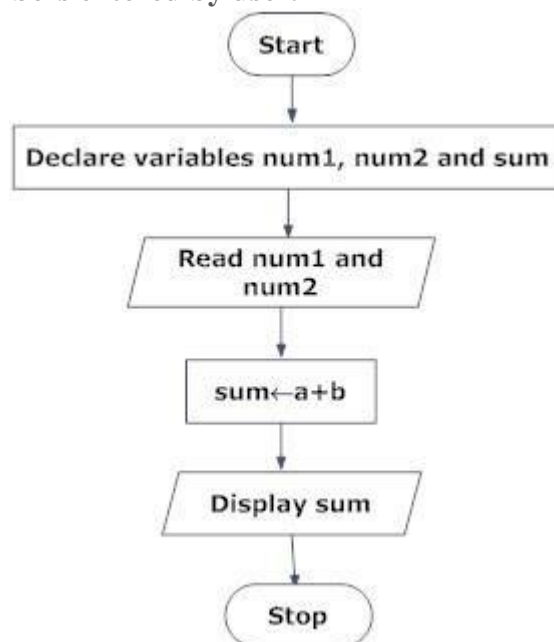
Different symbols are used for different states in flowchart, For example: Input/Output and decision making has different symbols. The table below describes all the symbols that are used in making flowchart

Symbol	Purpose	Description
	Flow line	Used to indicate the flow of logic by connecting symbols.
Symbol	Purpose	Description
	Terminal(Stop/Start)	Used to represent start and end of flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for airthmetic operations and data-manipulations.
	Desicion	Used to represent the operation in which there are two alternatives, true and false.

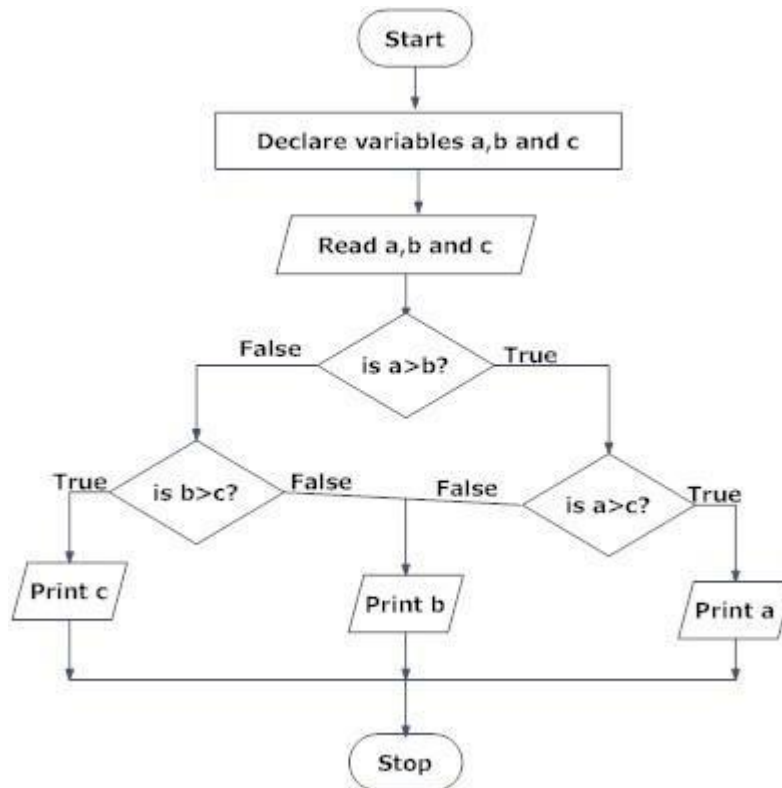
	On-page Connector	Used to join different flowline
	Off-page Connector	Used to connect flowchart portion on different page.
	Predefined Process/Function	Used to represent a group of statements performing one processing task.

Examples of flowcharts in programming

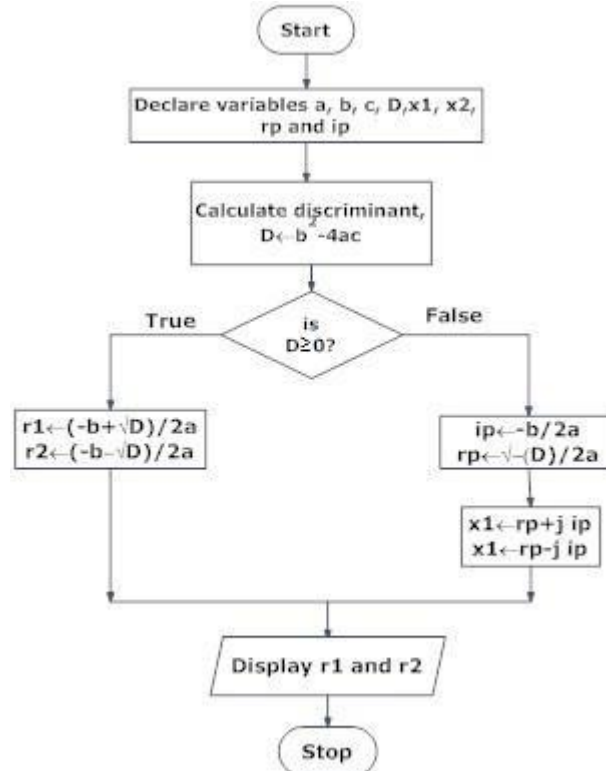
Draw a flowchart to add two numbers entered by user.



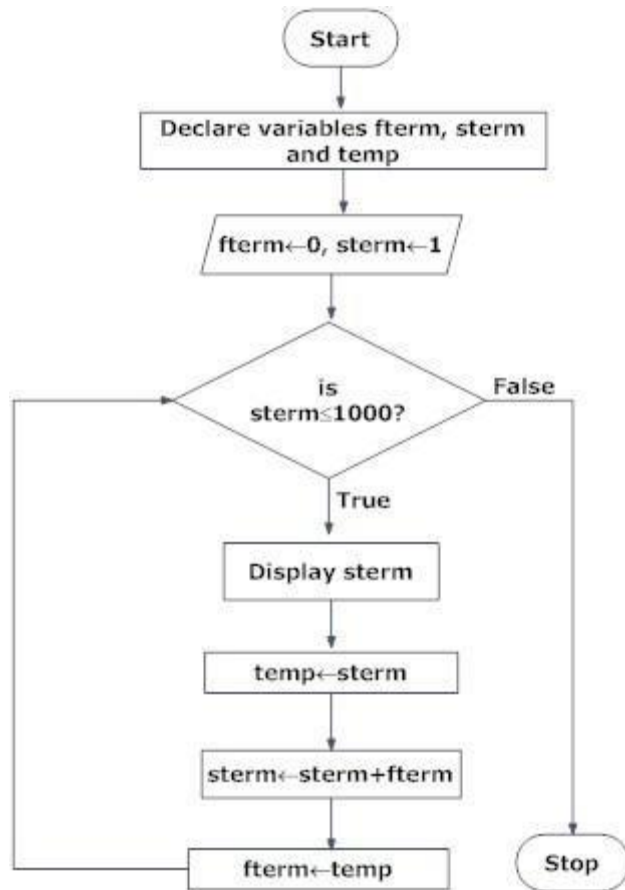
Draw flowchart to find the largest among three different numbers entered by user.



Draw a flowchart to find all the roots of a quadratic equation $ax^2+bx+c=0$



Draw a flowchart to find the Fibonacci series till term ≤ 1000 .



Though, flowchart are useful in efficient coding, debugging and analysis of a program, drawing flowchart in very complicated in case of complex programs and often ignored.