Unit 1:

1 Differentiate C Language and C++ Language. OR procedure oriented and object oriented programming language.
OR Write characteristics of POP and OOP.

Procedure Oriented Programming (POP)	Object Oriented Programming (OOP)
Emphasis is on doing things not on data	Emphasis is on data rather than procedure
Main focus is on the function and procedures that operate on data	Main focus is on the data that is being operated
Top Down approach in program design	Bottom Up approach in program design
Large programs are divided into smaller programs known as functions	Large programs are divided into classes and objects
Most of the functions share global data	Data is tied together with function in the data structure
Data moves openly in the system from one function to another function	Data is hidden and cannot be accessed by external functions
Adding of data and function is difficult	Adding of data and function is easy
Concepts like inheritance, polymorphism, data encapsulation, abstraction, access specifier are missing	Concepts like inheritance, polymorphism, data encapsulation, abstraction, access specifier are available and can be used easily
Examples: C, Fortran, Pascal, etc…	Examples: C++, Java, C#, etc…

Basic Concepts of OOP.

Various concepts present in OOP to make it more powerful, secure, reliable and easy.

Object

- An object is an instance of a class.
- An object means anything from real world like as person, computer, bench etc...
- Every object has at least one unique identity.
- An object is a component of a program that knows how to interact with other pieces of the program.
- An object is the variable of the type class.

Class

- A class is a template that specifies the attributes and behavior of things or objects.
- A class is a blueprint or prototype from which objects are created.
- A class is the implementation of an abstract data type (ADT). It defines attributes and methods which implement the data structure and operations of the ADT, respectively.

Data Abstraction

- Just represent essential features without including the background details.
- Implemented in class to provide data security.

Encapsulation

- Wrapping up of a data and functions into single unit is known as encapsulation.

Inheritance

- Inheritance is the process, by which class can acquire the properties and methods of another class.
- The mechanism of deriving a new class from an old class is called inheritance.
- The new class is called derived class and old class is called base class.
- The derived class may have all the features of the base class and the programmer can add new features to the derived class.

Polymorphism

- Polymorphism means the ability to take more than one form.
- It allows a single name to be used for more than one related purpose.
- It means ability of operators and functions to act differently in different situations.

Dynamic Binding

- Binding means linking of procedure call to the code to be executed in response to the call.
- It is also known as late binding, because it will not bind the code until the time of call at run time.
- It is associated with polymorphism and inheritance.

Message Passing

- A program contain set of object that communicate with each other.
- Basic steps to communicate
  1. Creating classes that define objects and their behavior.
  2. Creating objects from class definition
  3. Establishing communication among objects.

Benefits of OOP.

- We can eliminate redundant code though inheritance.
- Saving development time and cost by using existing module.
- Classes
- Build secure program by data hiding.
- It is easy to partition the work in a project based on object.
- Object oriented systems can be easily upgraded from small to large.
- Software complexity can be easily managed.

Applications of OOP.

- We can use OOP to develop various type of application of different areas.
  1. Real-time systems
  2. Simulation and modeling
  3. Object oriented database
  4. Hypertext, hypermedia, and expertext
  5. Artificial intelligence and expert systems
  6. Neural networks and parallel programming
  7. Decision support and office automation
  8. CIM/CAM/CAD systems.

Structure of C++ Program.

- C++ is an object oriented programming language.
- It is a superset of C language and also called as extended version of C language.
- It was developed by Bjarne Stroustrup at AT&T Bell lab in Murray Hill, New Jersey, USA in the early 1980's.
- Structure of C++ program is as follow.

Include Files
Class Declaration or Definition
Member functions definitions
Main function

- In any program first write header files like as iostream.h, conio.h, string.h etc..as per requirement of program.
- After header file write class declaration or definition as per your planning.
- After class, define all member functions which are not define but declare inside the class.
- In last write main function without main function program execution is not possible.
- For Example:

```
#include <iostream.h> //Header File
#include  <conio.h>  //Header  File
class Demo  //Class
{
      int a;
   public:
      void getdata()
      {
            a=10;
      }
      void putdata();
};
void Demo::putdata()     //Member Function Definition
{
      cout<<"The value of a="<<a;
}
void main()    //Main Function
{
      Demo D;
      clrscr();
      D.getdata();
      D.putdata();
      getch();
}
```

Write steps to create, compile and execute C++ program.

TO Create

- Open editor(Any Text Editor)
- Open new file
- Type your program
- Save file with .cpp extension

To Compile

- We can compile program by pressing alt+f9 or by selecting ‘compile’ option from compile menu.
  - During compilation it will give syntax error, if any syntax is wrong.
  - After compilation it will generate .obj file
  - This object file contains machine code, the native language of computer.

To Link

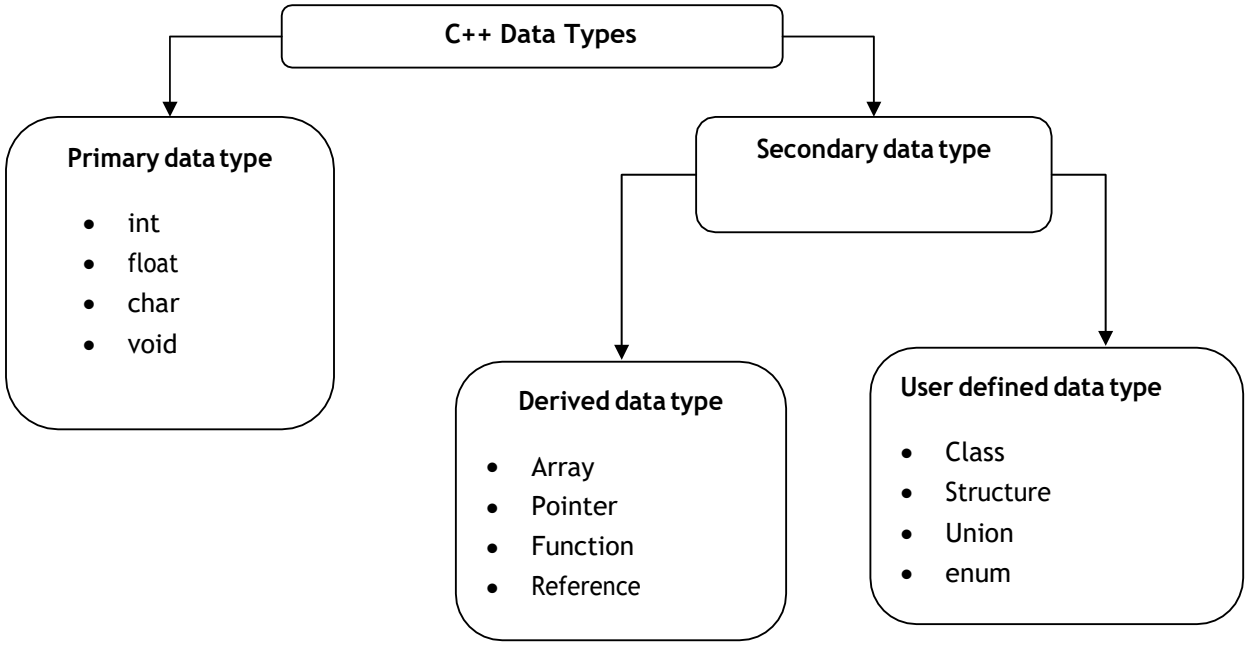
- Linking process is done by Linker.
- The Linker takes any object code file compiled from your source code and links them with special execution code and with any C++ library code required by your program.

To Execute

- We can execute program by pressing ctrl+f9 or by selecting ‘run’ option from run menu.
  - It will execute .exe file of program and give output.

Datatypes.

- C++ provides following data types.
- We can divide data types into three parts
  1. Primary datatype
  2. Derived datatype
  3. User defined data type



Primary Datatype

- The primary data type of C++ is as follow.

Data type	Size (bytes)	Range
Char	1	-128 to 127
Unsigned char	1	0 to 255
Short or int	2	-32.768 to 32.767
Unsigned int	2	0 to 65535

Long	4	-2147483648 to2147483647
Unsigned long	4	0 to4294967295
Float	4	3.4 e-38 to 3.4 e+308
Double	8	1.7 e-308 to 1.7 e+308
Long double	10	3.4e-4932 to 1.1 e+4932

Derived Datatype

- Following derived data types.
  1. Arrays.
  2. Function.
  3. Pointers.
- We cannot use the derived data type without use of primary data type.
- **Array**: An array is a fixed-size sequenced collection of elements of the same data type.
- **Pointer**: Pointer is a special variable which contains address of another variable.
- **Function**: A Group of statements combined in one block for some special purpose.

User Defined Datatype.

- We have following type of user defined data type in C++ language.
  1. Structure
  2. Class
  3. Union
  4. Enumeration
- The user defined data type is defined by programmer as per his/her requirement.
- **Structure:** Structure is a collection of logically related data items of different data types grouped together and known by a single name.
- **Union:** Union is like a structure, except that each element shares the common memory.
- **Class:** A class is a template that specifies the fields and methods of things or objects. A class is a prototype from which objects are created.
- **enum:** Enum is a user-defined type consisting of a set of named constants called enumerator.
  - Syntax of enumeration: enum enum_tag {list of variables};
  - Example of enumeration: enum day-of-week {mon=1,tue,wed,thu,fri,sat,sun};

Explain Constant in C++.

- Like variables, constants are data storage locations. But variables can vary, constants do not change.
- You must initialize a constant when you create it, and you can not assign new value later, after constant is initialized.

  Defining constant using #define

  #define is a preprocessor directive that declares symbolic constant.

  Example:

```
#define PI 3.14
```

  Every time the preprocessor sees the word PI, it puts 3.14 in the text.

```
#include<iostream.h>
   #include<conio.h>
   #define PI 3.14
   int main()
   {
         int r,area;
         cout<<"Enter Radius";
         cin>>r;
         area=PI*r*r;
         cout<<"Area of Circle"<<area;
         getch();
         return 0;
```

constant using const keyword

- **'const'** keyword is used to declare constant variable of any type.
- We cannot change its value during execution of program.
- Syntax: const DataType Variable_Name=value;
- Example: const int a=2;

  Now **'a'** is a integer type constant;

operators available in C++

An operator is a symbol that tells the compiler to perform certain mathematical or logical operation.

1. **Arithmetic Operators**

   Arithmetic operators are used for mathematical calculation. C++ supports following arithmetic operators

+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

2. **Relational Operators**

   Relational operators are used to compare two numbers and taking decisions based on their relation. Relational expressions are used in decision statements such as if, for, while, etc…

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	is equal to

!=	is not equal to

3. Logical Operators

Logical operators are used to test more than one condition and make decisions

&&	logical AND (Both non zero then true, either is zero then false)
||	logical OR (Both zero then false, either is non zero then true)
!	logical NOT (non zero then false, zero then true)

4. Assignment Operators

Assignment operators are used to assign the result of an expression to a variable. C++ also supports shorthand assignment operators which simplify operation with assignment

=	Assigns value of right side to left side
+=	a += 1 is same as a = a + 1
-=	a -= 1 is same as a = a - 1
*=	a *= 1 is same as a = a * 1
/=	a /= 1 is same as a = a / 1
%=	a %= 1 is same as a = a % 1

5. Increment and Decrement Operators

These are special operators in C++ which are generally not found in other languages.

++	Increments value by 1. a++ is postfix, the expression is evaluated first and then the value is incremented. Ex. a=10; b=a++; after this statement, a= 11, b = 10. ++a is prefix, the value is incremented first and then the expression is evaluated. Ex. a=10; b=++a; after this statement, a= 11, b = 11.
--	Decrements value by 1. a-- is postfix, the expression is evaluated first and then the value is

	decremented. Ex. a=10; b=a--; after this statement, a= 9, b = 10. --a is prefix, the value is decremented first and then the expression is evaluated. Ex. a=10; b=--a; after this statement, a= 9, b =9.

6. Conditional Operator

A ternary operator is known as Conditional Operator.

exp1?exp2:exp3 if exp1 is true then execute exp2 otherwise exp3

Ex: x = (a>b)?a:b; which is same as

```
if(a>b)
    x=a;
else
    x=b;
```

7. Bitwise Operators

Bitwise operators are used to perform operation bit by bit. Bitwise operators may not be applied to float or double.

&	bitwise AND
|	bitwise OR
^	bitwise exclusive OR
<<	shift left (shift left means multiply by 2)
>>	shift right (shift right means divide by 2)

8. Special Operators

&	Address operator, it is used to determine address of the variable.
*	Pointer operator, it is used to declare pointer variable and to get value from it.
,	Comma operator. It is used to link the related expressions together.
sizeof	It returns the number of bytes the operand occupies.
.	member selection operator, used in structure.
->	member selection operator, used in pointer to structure.

9. Extraction operator (>>)

Extraction operator (>>) is used with cin to input data from keyboard.

10. Insertion operator (<<)

Insertion operator (<<) is used with cout to output data from keyboard.

11. Scope resolution operator (::)

Scope resolution operator (::) is used to define the already declared member functions of the class.

Memory Management Operators of C++ with example.

- For dynamic memory management, C++ provides two unary operator 'new' and 'delete'.

 An object can be created by using new, and destroy by using delete, as and when required.
 Dynamic allocation of memory using new
 Syntax of **new :**

pointer_variable = new data_type;

 Here pointer_variable is a pointer of any data type.
 The new operator allocates sufficient memory to hold a data object.
 The pointer_variable holds the address of the memory space allocated.
 For example:

p=new int;
q=new float;

- Type of 'p' is integer and type of 'q' is float.

 We can combine declaration and initialization.

int *p=new int;
float *q=new float;

 We can dynamic allocate space for array, structure and classes by new.

int *p=new int[10];

 Allocates a memory space for an array of size 10.
 p[0] will refer location of p[1] and p[1] will refer location of [2] and so on.

 Release memory using delete
 When a data object is no longer needed, it is destroyed to release the memory space for reuse.
 Syntax of delete: delete pointer_variable;
 The pointer_variable is the pointer that points to a data object created with new.
 For example:

delete p;
delete q;

 To free a dynamically allocated array

delete [size] pointer_variable;
delete [10]p;

reference variable

 A reference variable provides an alias (alternative name) for a previously defined variable.

- Syntax: Data_type & reference_name = variable_name
- For example :

```
int a=100;
int &b=a;   //Now both a and b will give same value.
```

use of scope resolution operator (::) with example.

- The scope resolution operator is used to resolve or extend the scope of variable.
- C++ is block structured language. We know that the same variable name can be used to have different meaning in different block.
- The scope resolution operator will refer value of global variable from anywhere (also from inner block).
- Without scope resolution operator all variable will refer local value.
- We can better understand it by following example.

```
#include <iostream.h>
int m=10;
void main()
{
      int m=20;
      {
            int k=m;
            int m=30;
            cout<<"we are in inner block\n";
            cout<<"k="<<k<<"\n";
            cout<<"m="<<m<<"\n";
            cout<<"::m="<<::m<<"\n";
      }
      cout<<"we are in outer block\n";
      cout<<"m="<<m<<"\n";
      cout<<"::m="<<::m<<"\n";
}
```

member dereferencing operators.

- C++ provides three pointer to member operators to access the class member through pointer.

Operators	Function
::*	To declare a pointer to a member of a class.
.*	To access a member using object name and a pointer to that member.
->*	To access a member using a pointer to the object and a pointer to that member.

Manipulators.

- Manipulators are operators that are used to format the output that user wants to display.
- There are numerous manipulators are available in C++. The most commonly used manipulators are **endl** and **setw** and **setfill**.
- **Endl Manipulator**:
  - This manipulator does the same functionality as the '\n' newline character does.
  - Inserts the new-line character.
  - For example:

…….

```
cout<<"Hello World"<<endl;
cout<<"Good Morning"<<endl;
cout<<"Hi";
……..
```

Output:

```
Hello World
Good Morning
Hi
```

- **Setw Manipulator:**
  - This manipulator sets the minimum field width on output and right justified the number.
  - **Syntax:**

    setw(x)
  - Here setw causes the number or string that follows it to be printed within a field of x characters wide.
  - Iomanip.h header file must be included while using setw manipulator.
  - **For Example:**

```
#include<iostream.h>
#include<iomanip.h>
int main()
{
      int  m=12,n=123,p=1234;
      cout<<setw(5)<<"m="<<m<<endl;
      cout<<setw(5)<<"n="<<n<<endl;
      cout<<setw(5)<<"p="<<p;
      return 0;
}
```

Output:

```
m = 12
n = 123
p = 1234
```

- **Setfill Manipulator:**
  - This manipulator is used after setw manipulator.
  - The usage of the setfill is if a value does not entirely fill a field then the character specified in the setfill argument of the manipulator is used for filling fields.
  - **Example:**

```
……..
cout<<setw(10)<<setfill('$')<<123;
……..
```

Output:

```
$$$$$$$123
```