OOP Short Note

0. Class and Object:

Class: A class is a user-defined data type in C++ that encapsulates data members and member functions and it works as an object constructor or a "blueprint" for creating objects.

Object: An object is a data structure that is an instance of a class

Constructor: A constructor is a special member function of a class that is called when an object is created. It initializes the object's data members to their initial values.

Two types of constructors:

a) Default Constructor: This takes no parameter and sets the value of object members to some fixed values

b) Parameterized Constructor: This takes parameters and sets the values of object members accordingly.

Destructor: A destructor is a special member function of a class that is called when an object is destroyed. It cleans up any resources that the object was using.

Virtual destructor: A virtual destructor is used to free up the memory space allocated by the derived class object or instance while deleting instances of the derived class using a base class pointer object. A base or parent class destructor use the virtual keyword that ensures both base class and the derived class destructor will be called at run time, but it called the derived class first and then base class to release the space occupied by both destructors.

1. Encapsulation:

Encapsulation: Encapsulation is a process of combining member functions and data members in a single unit called a class. The purpose is to prevent access to the data directly from outside.

2. Inheritance:

Inheritance: Inheritance is a mechanism that allows a class to inherit the properties and behaviors (methods) of another class.

In C++, there are five types of inheritance:

Single inheritance: A derived class inherits its property from only one base class.

Multiple inheritance: More than one base class is used to create a derived class.

Hierarchical inheritance: Here, One base class inherits its properties to more than one derived class.

Multilevel inheritance: When one derived class inherits its property from another derived class.

Hybrid inheritance: The combination of any of the above four types of inheritance.

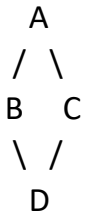
Ambiguity in Multiple Inheritance: In multiple inheritance, we derive a single class from two or more base or parent classes. So, it can be possible that both the parent classes have the same-named member functions. While calling via the object of the derived class (if the object of the derived class invokes one of the same-named member functions of the base class), it shows ambiguity. To avoid this we can call the function this way, derived_obj.base_class_name::function_name();

Diamond problem in Multiple Inheritance: It occurs when a class inherits from two or more classes that have a common base class, resulting in a diamond-shaped inheritance hierarchy.

-> Class A has a function named, introduce()

-> Class B and C derive from Class A.

-> Class D derives from both B and C



The problem is, in Class D the introduce() function of class A is copied two times which results in ambiguity. To avoid that we have to inherit Class A virtually.

Virtual function : To call a derived class function using base class pointer you have to make the base class function virtual.