Inheritance : Extending Classes

→ The mechanism of deriving a new class from an old one is called inheritance (or derivation).

→ The old class is called base class. and the new one is called the derived class or sub class.

→ Inheritance provides the concept of reusability.

→ A new class reuses the properties of existing class.

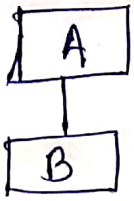
→ In C++, inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically.

→ In such way, you can reuse, extend or modify the attributes and behaviors which are defined in other class.

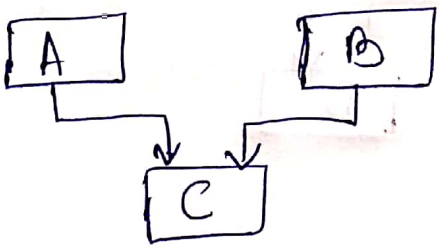
→ is a relationship

Types of Inheritance :-

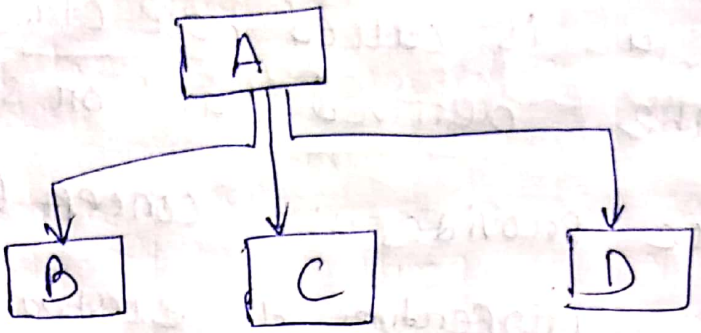
1. Single Inheritance :- A derived class with only one base class is called single inheritance.



2. Multiple Inheritance :- A derived class with several base classes is called multiple inheritance.

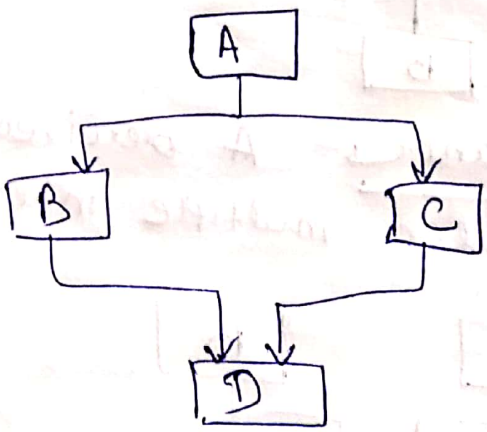


3. Hierarchical inheritance:- One class may be inherited by more than one class is called hierarchical inheritance.



4. Multilevel inheritance:- The mechanism of deriving a class from another derived class is known as multilevel inheritance.

5. Hybrid inheritance



Defining derived classes :-

→ A derived class can be defined by specifying its relationship with the base class in addition to its own details.

syntax :-

```
class derived-class-name : visibility_mode base-class name
{
    ... // members of derived class
}
```

Ex :-

```
class ABC : private XYZ
{
    ....
}
class ABC : public XYZ
{
    ....
}
```

* The colon (:) → indicates that derived class is derived from the base class name.

* The visibility mode is optional and it present may be either private, protected, and Public.

* The default visibity mode is private.

* The visibity mode specifies whether the features of the base class are privately derived or publicly derived.

Private inherited :-

→ When a base class is privately inherited by a derived class, public members of the base class become private members of the derived class.

→ The public members of the base class can only be accessed by the member functions of the derived class.

→ They are inaccessible to the objects of the derived class.

→ No member of the base class is accessible to the objects of the derived class.

Publicly inherited :-

→ When the base class is publicly inherited, public members of the base class become public members of the derived class.

→ They are accessible to objects of the derived class.

→ In both cases, the private members are not inherited and therefore the private member of a base class will never become the member of its derived class.

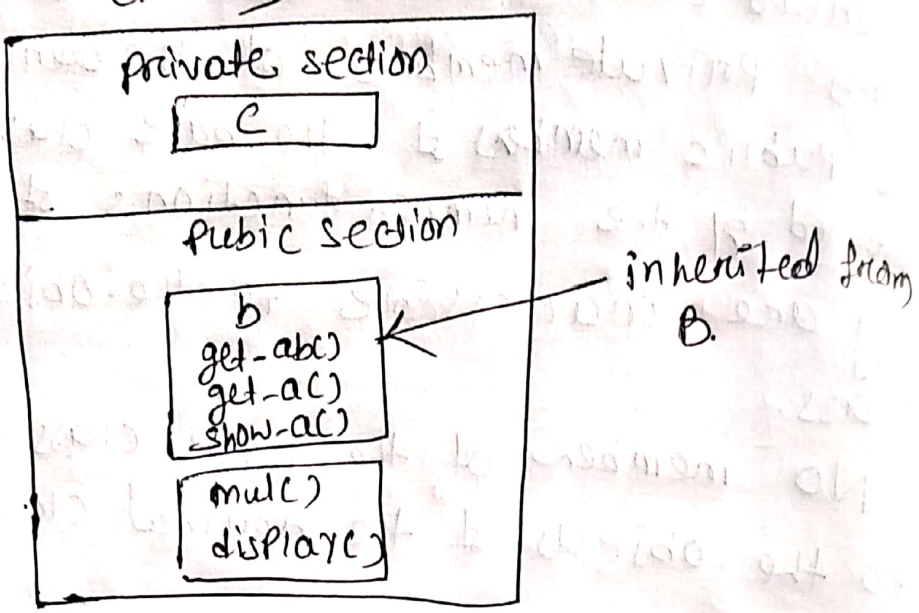
Example :-

```
class B
{
    int a;
  public :
      int b;
      void set_ab();
       int get_a();
       void show_a();
};
class D : public B
{
    int c;
  public:
      void mul();
      void display();
};
void B :: set_ab() { a=5; b=10; }
int B :: get_a() { return a; }
void B::show_a() { cout << a << endl; }
void D::mul() { c = b * get_a(); }
void D :: display()
{
  cout << get_a << endl;
  cout << b << endl;
  cout << c << endl;
}
int main()
{
  D d; d.set_ab();
  d.mul();
  d.show_a();
  d.display();
  d.b = 20;
  d.mul();
  d.display();
  return 0;
}
```

presents in class D

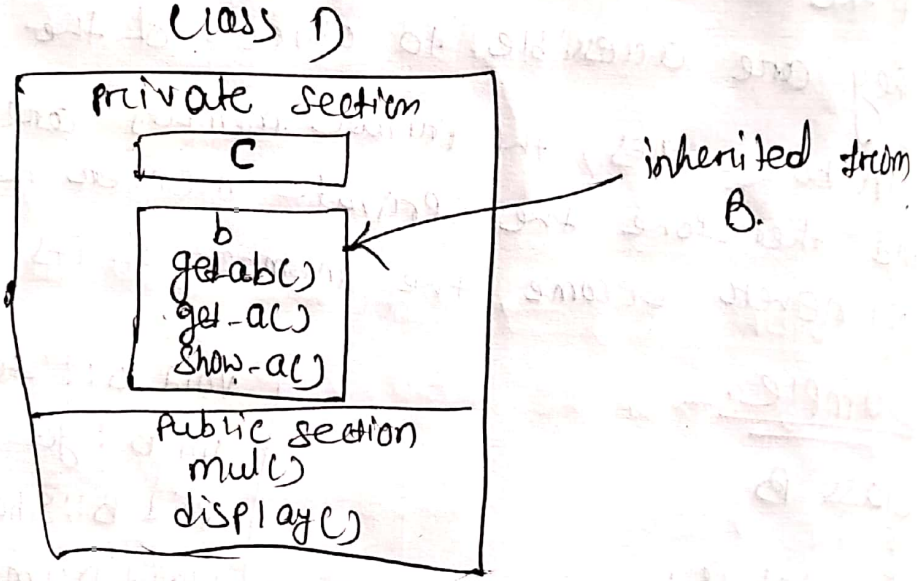
presents in class B

```
void mul()
{
  c = b * get_a();
}
```



If class D : private B

then



Making a private member inheritable

→ Let private data need to be inherited then we have to make its public. (by modifying the visibility limit of the private member by making it public).

→ This would make it accessible to all other functions of the program, thus taking away the advantage of data hiding.

→ C++ provide a third visibility modifier, protected which serve a limited purpose in inheritance.

→ A member declared as protected is accessible by the member functions within its class and any class immidiately derived from it.

→ It cannot be accessed by the functions outside these two classes.

```
Class ABC
{
    private:
        ....        // visible to member functions.
        ....        // within its class.
    protected:
        ....        // visible to member functions
                    of its own and derived class.
    public:
        ....        // visible to all functions
        ....        // in the program.
};
```