in c++ ->

:: is called scope resolution operator.

Multiple inheritance:-

A B

C

Multi level inheritance:-

A

B

c

Hirarecle inheritance:-

C

A B

Base class pointer can point to child class object but vice versa not allowed.

i.e child class pointer cannot point to Base class object.

Through Base class pointer we can only access properties of child object which are also properties of parent class.

polymorphism compile time:-

function overloading: - fun(a,v),fun(a),fun()

operator overloading:- int a + int b,char a + char b,double a + double b

function overiding:- decided in run time

vehicle v parent has print()

car c child has print()

v.print() call print() of vehicle.

c.print() call print() of car.

\*v1 pointer of vehicle type pointing to vehicle v

\*v2 pointer of vehicle type pointing to car c.

v1->print will call print() of vehicle v.

v2->print will also call print() of vehicle v.It will not check whom it is pointing to.Just call print of class vehicle of object c.

Virtual funtions:- present in base class and overridden in the derived class

polymorphism runtime:-

\*v1 pointer of vehicle type pointing to vehicle v

\*v2 pointer of vehicle type pointing to car c.

print() is virtual function in parent class "vehicle" and overriden in child class "car".

v2->print will call print of vehicle v but since print of vehicle v is virutal function in parent class(vehicle class) so it will check type of object v2 is pointing.

v2 is pointing to car class type so it will call print of car c object.

if print not present in the child class car c then complier will go to parent class(vehicle class) where print is defined as virtual and will print the

print() function of Vehicle v.

if print() function not present in vehicle class then v2->print() will give error as we can only access properties of parent class since pointer is of type vehicle.

Usecase:-

org has diff emp like hr,manager,engineer,other

all have different salary calculation.

Employees -> virtual CalculateSalary() function

HR Manager engineer others

1. (3) (20) (4)

All object have CalculateSalary() function

Employee \*\*e=new Employee\*[28]

e[i]->CalculateSalary()will call CalculateSalary() function from HR,Manager,engineer,others class based on the type of object e[i] is pointing to.

**Advantage** ->On only one traversal we can calculate salary of all employees.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HR | M | M | M | M | E | E | E | E | E |  |  |  |  |  |  |  |  |  |  |  |  |  |  | O | O | O | O |

HR Manager engineer OTHERS

Pure Virtual functions - which have no defintion.

ex:- virtual void print()=0;

**Abstract class** :- Any class which contains any pure virtual functions becomes an **abstract class.**

Note: - We cannot make an object of an abstract class.

Vehicle(Abstract class)

Implement all the pure virtual functions.

Option 1

Car(child class)

Option 2

Become an abstract class

Use case:-

org has diff emp like hr,manager,engineer,other

all have different salary calculation.

Employees -> virtual CalculateSalary() function

HR Manager engineer others

1. (3) (20) (4)

All object have CalculateSalary() function

Employee \*\*e=new Employee\*[28]

e[i]->CalculateSalary()will call CalculateSalary() function from HR,Manager,engineer,others class based on the type of object e[i] is pointing to.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HR | M | M | M | M | E | E | E | E | E |  |  |  |  |  |  |  |  |  |  |  |  |  |  | O | O | O | O |

HR Manager engineer OTHERS

**Advantages:-**

1.we can make CalculateSalary() as Pure Virtual function in Employee class and make it as a abstract class by which we can enforce child class like HR,Manager,Engineer,Others to write CalculateSalary() in their respective class as we cannot decide salary of an employee which knowing its category.

2.if we want to enforce our child class to make a particular function compulsorly then we can use concept of abstract class by making that function as pure virtual function(i.e by making the base class as abstract class)

**Friend Function**

Truck Bus

Print can access x if Print is Friend function of Truck

Print(){

}

Private x

Protected y

Public z

friend void Bus::print();

Note:-

1.Friend function should always be defined outside the class. Cannot be defined inside the class

2.Friend function don’t have access of **this pointer** because they not the member of any class.

3.Access modifier has no role on friend function. We can add friend function against any access modifier, there behviour remains the same.

4. friend class Bus;

By writing the **friend class Bus;** inside **Truck class** that means every member function of bus class is now friend function of Truck class and they can access every property of Truck class and now it can access all the private and protected properties.

5. **friend class Bus;**Here Bus is called friend class

If global function wants to access private properties of a class make that global function as a friend function of that class.

friend void test();