C++ Programming

Trainer : Rohan Paramane

Email: rohan.paramane@sunbeaminfo.com

Agenda

- Mode of Inheritance
- Diamond Problem
- Virtual Keyword
- Virtual Function
- Abstract Class
- Friend Function

Mode of inheritance

- If we use private, protected and public keyword to manage visibility of the members of class then it is
- called as access specifier.
- But if we use these keywords to extends the class then it is called as mode of inheritance.
- C++ supports private, protected and public mode of inheritance. If we do not specify any mode, then default mode of inheritance is private.

Mode Of inheritance

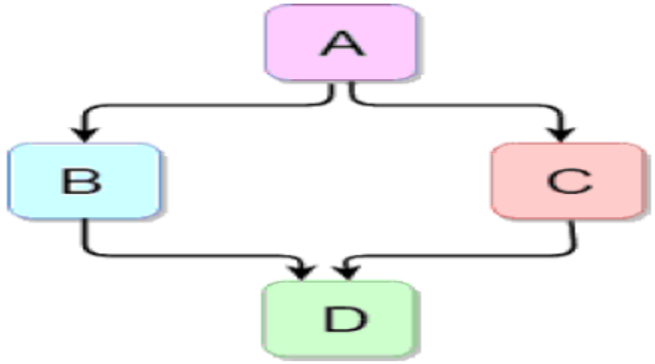
Public Mode of inheritance					
Access Specifier	**Same class**	**Derived class**	**Indirect Derived class**	**Friend function**	**Non- member function**
Private	A	NA	NA	A	NA
Protected	A	A	A	A	NA
Public	A	A	A	A	A
Private Mode of inheritance					
Access Specifier	**Same class**	**Derived class**	**Indirect Derived class**	**Friend function**	**Non- member function**
Private	A	NA	NA	A	NA
Protected	A	A	NA	A	NA
Public	A	A	NA	A	A : Base NA : Derived

Mode Of inheritance Cont….

Protected Mode of inheritance					
Access Specifier	**Same class**	**Derived class**	**Indirect Derived class**	**Friend function**	**Non- member function**
Private	A	NA	NA	A	NA
Protected	A	A	A	A	NA
Public	A	A	A	A	A: Base NA: Derived

Diamond Problem

- As shown in diagram it is hybrid inheritance. Its shape is like diamond hence it is also called as diamond inheritance.
- Data members of indirect base class inherit into the indirect derived class multiple times. Hence it effects on size of object of indirect derived class.
- Member functions of indirect base class inherit into indirect derived class multiple times. If we try to call member function of indirect base class on object of indirect derived class, then compiler generates ambiguity error.
- If we create object of indirect derived class, then constructor and destructor of indirect base class gets called multiple times.
- All above problems generated by hybrid inheritance is called diamond problem.



Solution to Diamond Problem– Virtual Base Class

- If we want to overcome diamond problem, then we should declare base class virtual i.e. we should derive class B & C from class A virtually. It is called virtual inheritance. In this case, members of class A will be inherited into B & C but it will not be inherited from B & C into class D.

```
class A { };
class B : virtual public A
{ };
class C : virtual public A
{ };
class D : public B, public C
{ };
```

Virtual Keyword

- Virtual functions allow us to create a list of base class pointers and call methods of any of the derived classes without even knowing kind of derived class object.
- **Early Binding**
- When we use Base class's pointer to hold Derived class's object, base class pointer or reference will always call the base version of the function.
- **Late Binding**
- **Using Virtual Keyword in C++**
- We can make base class's methods virtual by using **virtual** keyword while declaring them. Virtual keyword will lead to Late Binding of that method.
- On using Virtual keyword with Base class's function, Late Binding takes place and the derived version of function will be called, because base class pointer pointes to Derived class object.
- **Points to note**
  - **Only the Base class Method's declaration needs the Virtual Keyword, not the definition.**
  - If a function is declared as **virtual** in the base class, it will be virtual in all its derived classes.
  - The address of the virtual Function is placed in the **VTABLE** and the compiler uses **VPTR**(vpointer) to point to the Virtual Function

Program Demo

Early Binding

create a class Base and Derived (void show() in both classes)

create base *bptr;

bptr=&d;

bptr->show()

Late Binding

create a class Base and Derived (void show() in both classes one as virtual in base class)

create base *bptr;

bptr=&d;

bptr->show()

Abstract Class

- Sometimes implementation of all function cannot be provided in a base class because we don't know the implementation. Such a class is called abstract class.

```
classShape {
public:
virtualintArea() = 0; // Pure virtual function is declared as follows.
// Function to set width.
voidsetWidth(int w) {
width = w;
}
// Function to set height.
voidsetHeight(int h) {
height = h;
}

protected:
intwidth;
intheight;
};
 // A rectangle is a shape; it inherits shape.
// A triangle is a shape too; it inherits shape.
```

```
intmain() {
Rectangle R;
Triangle T;

R.setWidth(5);
R.setHeight(10);

T.setWidth(20);
T.setHeight(8);

cout <<"The area of the rectangle is: "<<
R.Area() <<endl;
cout <<"The area of the triangle is: "<< T.Area()
<<endl;
}
```

Friend function

- If we want to access private members inside derived class
  - Either we should use member function(getter/setter).
  - Or we should declare a facilitator function as a friend function.
  - Or we should declare derived class as a friend inside base class.

Thank You