C++ Programming

Trainer : Rohan Paramane

Email: rohan.paramane@sunbeaminfo.com

Upcasting , Downcasting & Object Slicing

- If we put the derived class object into base class pointer or reference then it is called as **Upcasting**
- If you want to access the members of derived class then you can convert the base class pointer/reference into derived class pointer/reference, this is called as **Downcasting**.
- At the time of downcasting explicit typecasting is mandatory.
- When upcasting is done then you can only call the base class members using the base class pointer or reference.
- you cannot access the derived class members using the base class pointer or reference.
- This is because of **Object Slicing**.

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.
- In such cases we can declare a function but cannot define it.
- Such functions are then made as pure virtual functions which must be implemented by the Derived class.
- Such a class where pure virtual function exists is called abstract class.
- We cannot create an object, but we can create pointer or reference of abstract class.

Thank You