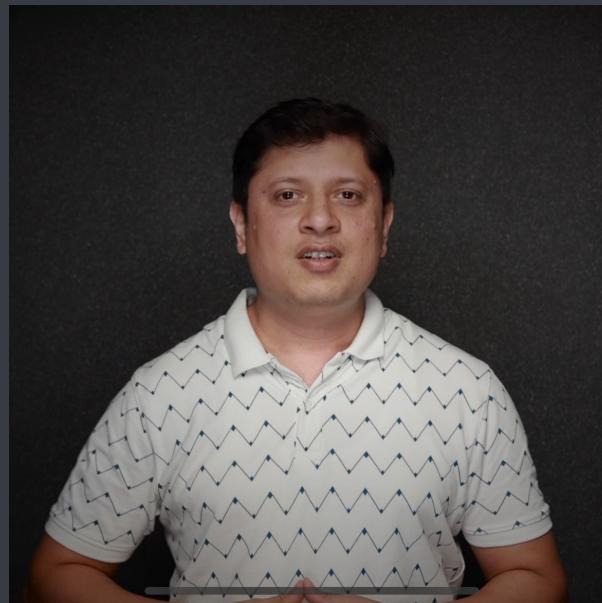


C, C++, DSA in depth

Runtime polymorphism



Saurabh Shukla (MySirG)

Agenda

- ① Base Pointer
- ② Problem with overriding
- ③ virtual function
- ④ Runtime Polymorphism

Base Pointers

Pointer of a class can point to an object of any of the descendant class

class A
{

};

class B : public A

{

};

A *P;
A O1;
P = &O1;

B O2;
P = &O2;

Base class pointer can point to an object of any descendant class but can access only base class members. Why??

Due to Early Binding

Early Binding

Base class pointer can point to an object of any descendant class but can access only base class members. Why??

Answer is Early Binding

Problem in overriding

Function overriding ensure appropriate call of function with respect to caller object.

but when function is called with the help of base class pointer, which points to the object of derived class, base class version of function is invoked instead of derived class version.

This problem in overriding occurs due to early binding.

As in early bnd my decision of
looking function definition is based
of the type of pointer which is
used to call an overridden method.

${}^*\sigma = \text{new } \beta;$

Solution of the problem

Use late binding of overridden methods.

Virtual Function

A function in a class can be qualified as virtual function by prefixing function declaration with the keyword **virtual**.

Virtual function → Late binding.

Virtual function should be a member of a class.

Overridden methods of virtual function are also virtual.

```
class A
{
public:
    virtual void f1(){} }
```

```
};

class B : public A
{
public:
    void f1() {} }
```

}

It is not mandatory to override a virtual function.

Class A

```
{ public:  
    virtual void f1() { ... }  
    virtual void f2() { ... }
```

};

class B : public A

```
{ public:  
    void f1() { }
```

}

Runtime Polymorphism

```
class User
{
public:
    virtual void show() { ... }

};

class Admin : public User
{
public:
    void show() { ... }

};

class Learner : public User
{
public:
    void show() { ... }

};
```

User *ptr = getUser();
ptr->show();

ptr = new Learner();
ptr->show(); //Learner
...
--

ptr = new Admin();
ptr->show(); //Admin

```

class A * -vptr
{
public:   -vtable
          static array of
          ↓ function pointers
          [f1 f2]
virtual void f1() { ... }
virtual void f2() { ... }
void f3() { ... }
void f4() { ... }
};

class B : public A
{
public:   -vtable
          [f1 f2]
void f1() { ... }
void f3() { ... }
void f5() { ... }
};

```

A*P;

P = new A; P
 P->f1(); LB//A
 P->f2(); LB//A
 P->f3(); EB //A
 P->f4(); EB //A
 P->f5(); EB //error

delete P;

P = new B; P
 P->f1(); LB//B
 P->f2(); LB//A
 P->f3(); EB //A
 P->f4(); EB //A
 P->f5(); EB //error