C++ in depth

Runtime polymorphism

Saurabh Shukla (MySirG)

Agenda

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

Base Pointer

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??

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 binding decision of looking function definition is based on the type of pointer which is used to call an overridden method.

```
*p = new B;
```

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 bindmy.

Virtual function must 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.

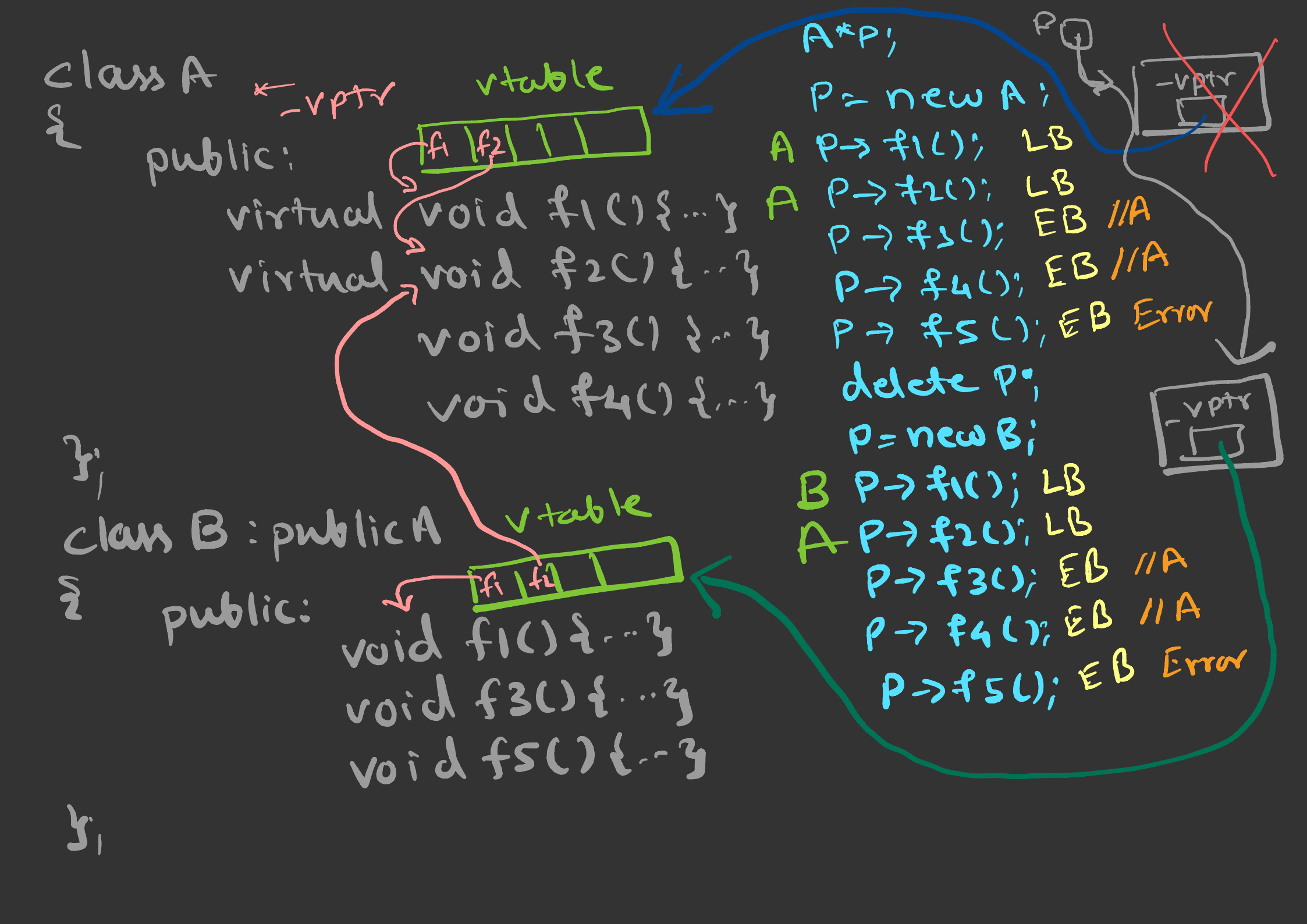
```
class A
{
    public:
      virtual void f1() {...}
      virtual void f2() {...}

};
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
{
public:
virtual void f1(){...}
virtual void f2(){...}
void f3(){...}
void f4(){...}
};
class B : public A
{
public:
void f1(){...}
void f3(){...}
void f5(){...}
};
_vptr
vtable
f1
f2
vtable
f1
f2
A*p;
p = new A;
A p->f1(); LB
A p->f2(); LB
p->f3(); EB //A
p->f4(); EB //A
p->f5(); EB Error
delete p;
p = new B;
B p->f1(); LB
A p->f2(); LB
p->f3(); EB //A
p->f4(); EB //A
p->f5(); EB Error
p
_vptr
_vptr