CUC春季学期C++整理12.0

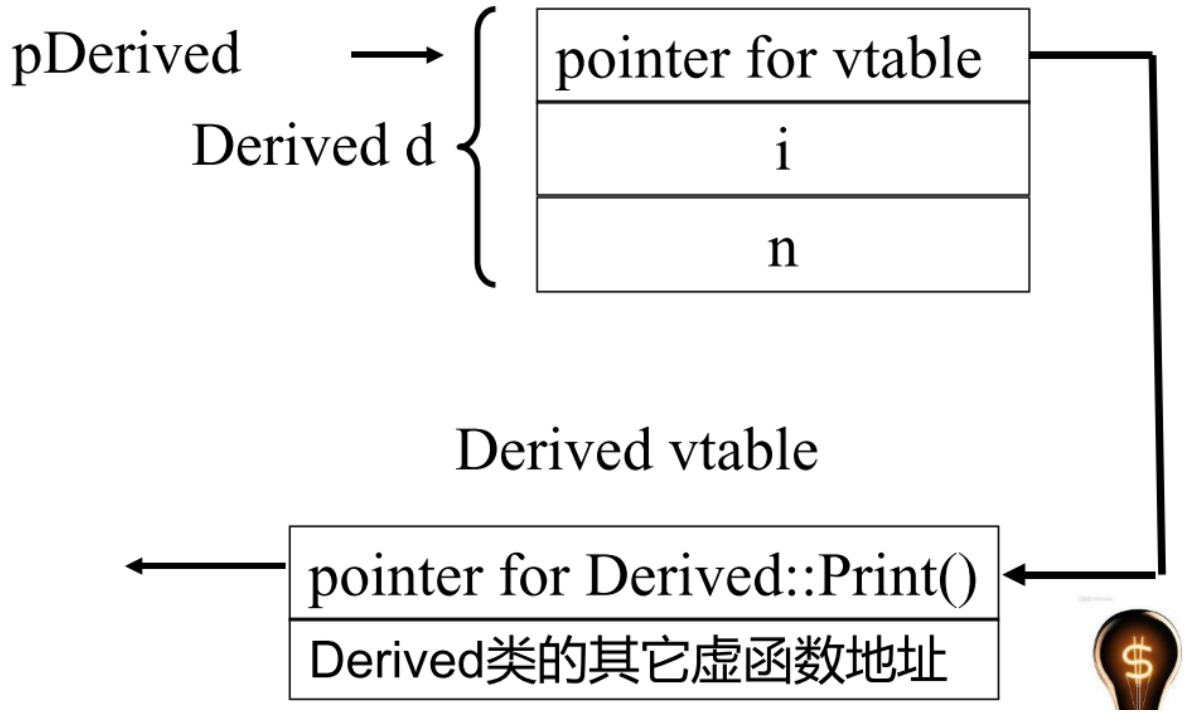
一、多态的实现原理

给出一个例子：

```
#include<bits/stdc++.h>
using namespace std;
class Base{
    public:
        int i;
        virtual void Print(){
            cout << "Base:Print" << endl;
        }
};
class Derived : public Base{
    public:
        int n;
        virtual void Print(){
            cout << "Drived:Print" << endl;
        }
};
int main()
{
    Derived d;
    cout << sizeof(Base) << ',' << sizeof(Derived);
    return 0;
 }
//输出
//8 12
```

所以可以看出，输出的结果比我们想象中的多了4个字节，所以到底是怎么引起的呢？

其实，每一个**有虚函数的类**，或者说有虚函数的**类的派生类**，都有一个虚函数表，在该类的任何对象中都存放着虚函数表的指针。虚函数表中列出了**该类的虚函数地址。多出的4个字节就是用来放虚函数表的地址的。**



如图所示，类的对象中的第一个空间存放的就是指向虚函数表的指针。

多态的函数调用语句被编译成一系列根据**基类指针所指向的（或者基类引用所引用的）对象**中存放的虚函数表的地址，在虚函数表中查找虚函数的地址，并调用虚函数的指令。

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
class A{
    public:
        virtual void Print(){
            cout << "A::Func" << endl;
        }
};
class B{
    public:
        virtual void Print(){
            cout << "B::Func" << endl;
        }
};
int main()
{
    A a;
    A * pa = new B();
    pa->Print();
    ll * p1 = (ll *) &a;
    ll * p2 = (ll *) pa;
    *p2 = *p1;//这里是把p1指向的地址全部赋给p2,所以此时p2的第一项地址就是存的A类的虚函数表
    pa->Print();
    return 0;
}
//输出
B::Func
A::Func
```

二、虚析构函数

通过基类的指针删除派生类对象时，**通常情况下只调用基类的析构函数**，但是，删除一个派生类的对象时，**应该先调用派生类的析构函 数，然后调用基类的析构函数。**

解决办法：把基类的析构函数声明为`virtual`。

- **派生类的析构函数可以`virtual`不进行声明。**
- 通过基类的指针删除派生类对象时，**首先调用派生类的析构函 数，然后调用基类的析构函数**

一般来说，**一个类如果定义了虚函数，则应该将析构函数也定义成虚函数。或者，一个类打算作为基类使用，也应该将析构函数定义成虚函数。**

注意：不允许以虚函数作为构造函数

```
#include<bits/stdc++.h>
using namespace std;
class son{
    public:
        virtual ~son(){
            cout << "bye from son" << endl;
        }
};
class grandson{
    public:
        ~grandson(){
            cout << "bye from grandson" << endl;
        }
};
int main()
{
    son *pson;
    pson = new grandson();
    delete pson;
    return 0;
}
//输出
bye from grandson
bye from son
```

包含纯虚函数的类叫抽象类

- 抽象类只能作为基类来派生新类使用，不能创建独立的抽象类的对象
- 抽象类的指针和引用可以指向由抽象类派生出来的类的对象

`A a;` // 错，A 是抽象类，不能创建对象

`A * pa;` // ok,可以定义抽象类的指针和引用

`pa = new A;`//错误, A 是抽象类，不能创建对象

- 在抽象类的成员函数内可以调用纯虚函数，但是**在构造函数或析构函数内部 不能调用纯虚函数。**
- 如果一个类从抽象类派生而来，那么**当且仅当它实现了基类中的所有纯虚函 数，它才能成为非抽象类。(就是说必须重写同名同参数表的函数)**

```
#include<bits/stdc++.h>
```

```cpp
using namespace std;
class A {
    public:
        virtual void f() = 0; //纯虚函数
        void g( ) { this->f( ) ; //ok
    }
    A( ){ //f( ); // 错误 }
};
class B:public A{
    public:
        void f(){cout<<"B:f()"<<endl; }
};
int main()
{
    B b;
    b.g();
    return 0;
}
```