程序设计实习

郭炜　微博 http://weibo.com/guoweiofpku
http://blog.sina.com.cn/u/3266490431
刘家瑛 微博 http://weibo.com/pkuliujiaying

虚函数和多态(P247)

虚函数

- 在类的定义中，前面有 virtual 关键字的成员函数就是虚函数。

```
class base {
        virtual int get() ;
};
int base::get()
{ }
```

- virtual 关键字只用在类定义里的函数声明中，写函数体时不用。

多态的表现形式一

- 派生类的指针可以赋给基类指针。

- 通过基类指针调用基类和派生类中的同名**虚函数**时：

（1）若该指针指向一个基类的对象，那么被调用是基类的虚函数；

（2）若该指针指向一个派生类的对象，那么被调用的是派生类的虚函数。

这种机制就叫做“多态”。

多态的表现形式一

```
class CBase {
        public:
        virtual void SomeVirtualFunction() { }
};
class CDerived:public CBase {
        public :
        virtual void SomeVirtualFunction() { }
};
int main()  {
        CDerived ODerived;
        CBase * p = & ODerived;
        p -> SomeVirtualFunction();  //调用哪个虚函数取决于p指向哪种类型的对象
        return 0;
}
```

多态的表现形式二

- 派生类的对象可以赋给基类引用

- 通过基类引用调用基类和派生类中的同名**虚函数**时：

（1）若该引用引用的是一个基类的对象，那么被调用是基类的虚函数；

（2）若该引用引用的是一个派生类的对象，那么被调用的是派生类的虚函数。

这种机制也叫做“多态”。

多态的表现形式二

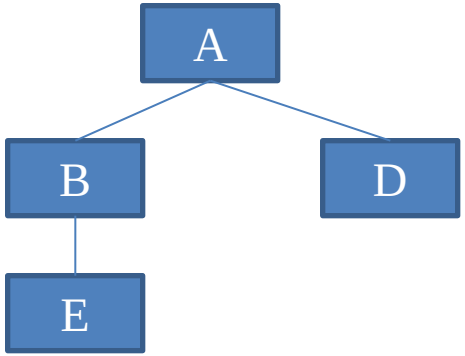
```
class CBase {
        public:
        virtual void SomeVirtualFunction() { }
};
class CDerived:public CBase {
        public :
        virtual void SomeVirtualFunction() { }
};
int main()  {
        CDerived ODerived;
        CBase & r = ODerived;
        r.SomeVirtualFunction();  //调用哪个虚函数取决于r引用哪种类型的对象
        return 0;
}
```

多态的简单示例

```
class A {
        public :
        virtual  void Print(  )
        {   cout << "A::Print"<<endl ; }
};
class B: public A {
        public :
        virtual  void Print(  ) {   cout << "B::Print" <<endl; }
};
class D: public A {
        public:
        virtual  void Print(  ) {   cout << "D::Print" << endl ; }
};
class E: public B {
        virtual  void Print(  ) {   cout << "E::Print" << endl ; }
};
```

```
int main()  {
   A  a; B  b;     E e; D d;
   A * pa = &a;  B * pb = &b;
   D * pd = &d ; E * pe = &e;

   pa->Print();     // a.Print()被调用，输出：A::Print
   pa = pb;
   pa -> Print();   //b.Print()被调用，输出：B::Print
   pa =  pd;
   pa -> Print();   //d. Print ()被调用,输出：D::Print
   pa = pe;
   pa -> Print();   //e.Print () 被调用,输出：E::Print
   return 0;
}
```



多态的作用

在面向对象的程序设计中使用多态，能够增强程序的可扩充性，即程序需要修改或增加功能的时候，需要改动和增加的代码较少。