第五章 继承和派生

一、继承和派生的概念

1.基本概念

(1)继承和派生

继承：在定义一个新的类B时，如果该类与某个已有的类A相似（指的B拥有A的全部特点），那么就可以把A作为一个基类，而把B作为基类的一个派生类（也称子类）。

派生类是通过对基类进行修改和扩充得到的。在派生类中，可以扩充新的成员变量和成员函数。

派生类一经定义后，可以独立使用，不依赖于基类。

派生类拥有基类的全部成员函数和成员变量，不论是private，protected，public。

1）在派生类的各个成员函数中，不能访问基类中的private成员。

2）学籍管理程序：学号、姓名、性别、成绩为共同属性；是否该留级，是否该奖励为共同方法（成员函数）；大学生，研究生，导师等为不同的属性和方法等

(2)派生类的写法

```
class 派生类名：public 基类名
{
    ......

};
```

(3)派生类对象的内存空间

派生类对象的体积，等于基类对象的体积，再加上派生类对象自己的成员变量的体积。在派生类对象中，包含着基类对象，而且基类对象的存储位置位于派生类对象新增的成员变量之前。

```
class CBase
{
    int v1,v2;
};
class CDerived:public CBase
{
    int v3;
}
```

2.程序实例

下面看一个有两个类的简单学生管理程序:

```
#include<iostream>
#include<string>
using namespace std;
class CStudent
{
private:
  string name;
  string id;
  char gender;
  int age;
public:
  void PrintInfo();
  void SetInfo(const string&name_,const string & id_,int age_,char gender_);
  string GetName(){return name;}
};
class CUndergraduateStudent:public CStudent
{
private:
  string department;
public:
  void QualifiedForBaoyan(){
    cout<<"qualified for baoyan"<<endl;
  }
void PrintInfo() {
  CStudent::PrintInfo();
  cout<<"Department:"<<department<<endl;
}
```

```
void SetInfo(const string&name_,const string & id_,int age_,char gender_,const string&depart
{
   CStudent::SetInfo(name_,id_,age_,gender_);
   department=department_;
}
};
void CStudent::PrintInfo()
{
  cout<<"Name:"<<name<<endl;
  cout<<"ID:"<<id<<endl;
  cout<<"Age:"<<age<<endl;
  cout<<"Gender:"<<gender<<endl;
}
void CStudent::SetInfo(const string&name_,const string & id_,int age_,char gender_)
{
  name=name_;
  id=id_;
  age=age_;
  gender=gender_;
}
int main()
{
  CStudent s1;
  CUndergraduateStudent s2;
  s2.SetInfo("Harry Potter","118829212",19,'M',"Computer Science");
  cout<<s2.GetName()<<endl;
  s2.QualifiedForBaoyan();
  s2.PrintInfo();
  cout<<"sizeof(string)="<<sizeof(string)<<endl;
  cout<<"sizeof(CStudent)="<<sizeof(CStudent)<<endl;
  cout<<"sizeof(CUndergraduateStudent)="<<sizeof(CUndergraduateStudent)<<endl;
  return 0;
}
```

二、继承关系和复合关系

1.类之间的两种关系

(1)继承关系和复合关系

1.继承：“是”关系。

基类A，B是基类A的派生类。

逻辑上要求：“一个B对象也是个A对象”。

2.复合：“有”关系。

类C中“有”成员变量k，k是类D的对象，则C和D是复合关系；

一般逻辑上要求：“D对象是C对象的固有属性或组成部分”。

(2)继承关系的使用

在设计两个有关系的类时，要注意，并非两个类有共同点，就可以让他们成为继承关系。让类A继承类B，必须满足“类B所代表的事物也是类A所代表的事物”，这个命题从逻辑上是成立的。

几何形体的绘图程序，点类和圆类，两者的关系是复合关系：

```
class CCircle
{
    CPoint center;//圆心
    double radius;//半径
}
```

(2)复合关系的使用

写一个小区养狗管理程序，要有两个类，主人类和狗类：（注意不要循环定义：人中有狗，狗中有人）

正确的写法：是为狗类设一个主人类的指针成员变量，为主人类设一个狗类的对象数组。

```
class CMaster;
class CDog
{
    CMaster *pm;
};
class CMaster{
CDog *dogs[100];
};
```

三、protected访问范围说明符

基类的private成员：可以被下列函数访问

1.基类的成员函数 2.基类的友元函数

基类的public成员：可以被下列函数访问

1.基类的成员函数 2.基类的友元函数 3.派生类的成员函数 4.派生类的友元函数 5.其他的函数

基类的protected成员：可以被下列函数访问

1.基类的成员函数 2.基类的友元函数 3.派生类的成员函数可以访问当前对象的基类的保护成员

在基类中，一般都将需要隐藏的成员说明为保护成员而非私有成员。

四、派生类的构造函数

派生类的构造函数

1.派生类对象包含基类对象

2.执行派生类构造函数之前，先执行基类的构造函数

3.派生类交代基类初始化，具体形式：

```
构造函数表（形参表）：基类名（基类构造函数实参表）
{
        ......
}
```

4.派生类对象消亡时，先执行派生类的析构函数，再执行基类的析构函数

5.在创建派生类的对象时:

1）需要调用基类的构造函数：初始化派生类对象中从基类继承的成员

2）在执行一个派生类的构造函数之前，总是先执行基类的构造函数

6.调用基类构造函数的两种方式:

1）显示方式：派生类的构造函数中—¿基类的构造函数提供参数

2）隐式方式：派生类的构造函数中，省略基类构造函数时，派生类的构造函数，自动调用基类的默认构造函数

派生类的析构函数被执行时，执行完派生类的析构函数后，自动调用基类的析构函数

2.派生类的构造函数和析构函数调用顺序

```
#include <iostream>
using namespace std;
class Base{
 public:
  int n;
  Base(int i):n(i)
  {
    cout<<"Base "<<n<<" constructed"<<endl;//1
  }
  ~Base()
  {
    cout<<"Base "<<n<<" destructed"<<endl;//2
  }
};
class Derived:public Base{//派生类构造函数
 public:
  Derived(int i):Base(i)//基类构造函数
  {
    cout<<"Derived constructed"<<endl;//4
  }
  ~Derived()//基类的析构函数
  {
    cout<<"Derived destructed"<<endl;//3
  }
};
int main()
{
  Derived Obj(3);
  return 0;
}
```

1.创建派生类的对象时，执行派生类的构造函数之前：

1）调用基类的构造函数：初始化派生类对象中从基类继承的成员

2)调用成员对象类的构造函数：初始化派生类对象中的成员对象

2.执行完派生类的析构函数后：

1）调用成员对象类的析构函数

2）调用基类的函数

注意：析构函数的调用顺序与构造函数的调用顺序相反

五、public继承的赋值兼容规则

（1）派生类的对象可以赋值给基类对象；

（2）派生类的对象可以用来初始化基类引用；

（3）派生类对象的地址可以赋值给基类指针，亦即派生类的指针可以赋值给基类的指针。

以上三条反过来是不成立的。例如，不能把基类对象赋值给派生类对象。

在公有派生的情况下，可以说，派生类对象也是基类对象，任何本该出现基类对象的地方，如果出现的是派生类的对象，也是没有问题的。