第六章 继承和派生类

基类和派生类

派生类的定义

派生类的定义格式：

```
class 派生类名:继承方式1 基类名1，继承方式2 基类名2,...
派生类类体
派生类类体中未定义的成员函数的实现
```

继承的类型

按照继承次数有：单重继承、多重继承（一带多）、多级继承（连续继承）

按照继承方式分有如下三种

1.public公有继承

- 公有继承中，直接基类的公有成员和保护成员均分别作为派生类中的公有成员和保护成员
- 派生类的**成员函数**可以发访问基类的**公有成员和保护成员**
  不能直接访问基类的私有成员，只能通过基类的共有成员函数和保护成员函数间接访问。
- 派生类的**对象**只能直接访问基类的公有成员

示例：

```
class A
{
public:
        int k1;void f1();
protected:
        int j1;void g1();
private:
        int i1;void h1();        //不能被派生类直接访问
};
class B :public A
{
```

```
public:
        int k2;void f2();
protected:
        int j2;void g2();
private:
        int i2;void h2();
}
```

类B的成员函数f2()、g2()和h2()中可以直接访问类A的k1、f1()、j1和g1()，但不能直接访问类A的i1和h1()
类B的对象只能直接访问类A的k1()和f1()

2.private私有继承

- 私有继承中，直接基类的公有成员和保护成员均作为派生类的私有成员
- 派生类的**成员函数**可以发访问基类的**公有成员和保护成员**
  不能直接访问基类的私有成员，只能通过基类的公有成员函数和保护成员函数间接访问。
- 派生类的对象不能直接访问直接基类的**任何成员**

注意：缺省继承方式为private

3.protected保护继承

- 保护继承中，直接基类的公有成员和保护成员均作为派生类的保护成员
- 派生类的**成员函数**可以发访问基类的**公有成员和保护成员**
  不能直接访问基类的私有成员，只能通过基类的公有成员函数和保护成员函数间接访问
- 派生类的对象不能直接访问直接基类的**任何成员**

派生类的构造函数和析构函数

基类的构造函数而析构函数均不能被派生类继承！！

派生类的构造函数

派生类构造函数的一般形式为：

```
派生类构造函数名（总参数表）:直接基类初始化表，子对象初始化表，
派生类自身数据成员初始化表
派生类构造函数体
```

注意：派生类构造函数名与本派生类名相同；总参数表包含其后各初始化表所需要的全体参数

调用时机：在创建派生类对象的时候，系统自带调用派生类构造函数对派生类对象进行初始化

初始化顺序：**直接**基类构造函数⟶子对象构造函数(子对象的创建顺序)⟶派生类自身数据成员和自身构造函数

代码示例：

```
#include<iostream>
using namespace std;

//基类People
class People{
protected:
    char *m_name;
    int m_age;
public:
    People(char*, int);
};
People::People(char *name, int age): m_name(name), m_age(age){}

//派生类Student
class Student: public People{
private:
    float m_score;
public:
    Student(char *name, int age, float score);
    void display();
};
//People(name, age)就是调用基类的构造函数
Student::Student(char *name, int age, float score): People(name, age),
m_score(score){ }
void Student::display(){
    cout<<m_name<<"的年龄是"<<m_age<<", 成绩是"<<m_score<<"。"<<endl;
}

int main(){
    Student stu("小明", 16, 90.5);
    stu.display();

    return 0;
}
```

People(name, age)就是调用基类的构造函数，并将 name 和 age 作为实参传递给它，m_score(score)是派生类的参数初始化表

派生类的析构函数

没有什么特殊的地方，不用显示地调用各直接基类、各子对象相应类的析构函数，由系统自动完成

执行顺序：与构造函数相反
派生类的析构函数⟶子对象的析构函数(子对象创建的相反顺序)⟶直接基类的析构函数

赋值兼容规则

下面规则只对公有继承有效

- 可以用派生类对象给基类对象赋值，反之不行
- 可以用派生类对象初始化基类对象或基类的引用
- 可以把派生类对象的指针赋值给基类对象的指针，反之不行
- PS1:对象指针之间的赋值并没有拷贝对象的成员，也没有修改对象本身的数据，**仅仅是改变了指针的指向**
- PS2：将派生类指针赋值给基类指针时，通过基类指针只能使用派生类的成员变量，但不能使用派生类的成员函数

```
base b,*pb;  derived d,*pd;
pb=&d;   //正确
pd=&b;   //错误
pb=pd;   //正确
pd=pb;   //错误
```

总结：**编译器通过指针来访问成员变量，指针指向哪个对象就使用哪个对象的数据；编译器通过指针的类型来访问成员函数，指针属于哪个类的类型就使用哪个类的函数。（针对非虚函数！）**

抽象类