第8章 取其精华 发挥优势—继承

1、模拟人的行走、听、说、写	2、为什么需要继承
3、派生类的定义	**4、基类与派生类**
5、三种继承方式	6、派生类的构造与析构函数
7、点、圆、圆柱体继承设计	8、从U盘到MP3继承设计

基类和派生类

- 采用已存在的类去定义建立新类
- 新类称为派生类（子类）
- 已存在的类称为基类（父类）
- 派生类与基类具有相对性
  - 人→学生→大学生
  - 水果→桃→水蜜桃→陕西水蜜桃

派生类的语法结构

class <派生类名>:<访问权限> <基类名1>,.....<访问权限> <基类名n>

{

private:

新增私有数据成员和成员函数

protected:

新增保护数据成员和成员函效

public:

新增公有数据成员和成员函效

};

派生类定义的变化

- 派生出新类时，可以做如下几种变化：
  1. 可以增加新的数据成员
  2. 可以增加新的函数成员
  3. 可以重新定义已有的函数成员
  4. 可以改变现有成员的数据值

派生类所起的作用

- 从基类接受成员
- 派生类对基类的扩充
- 派生类对基类成员的改造
- 系统的默认值就是私有继承

智能手机类的设计

- 先设计普通移动电话(手机)类
  - 电话号码、型号、价格等
  - 构造函数
  - 拨打电话、接听电话、挂断电话
  - 析构函数
- 再设计智能手机类
  - 电话号码、型号、价格、OS、内存等
  - 构造函数
  - 拨打电话、接听电话、挂断电话、
  - 发送短信、发送微信、浏览网络、闹钟等
  - 析构函数

普通手机类的定义

```
class mobile                                    //普通手机类，作为基类
{
private:
        char mynumber[12];          //机主的电话号码
        char m_type[40];                        //手机型号
        float price;                //手机价格
public:
        mobile()            //构造函数
        {
                init("00000000000","Non_type",0);
        }
        void init(char *number,char *pt,float pri); //初始化
        void dial();                                                    //拨打电话
        void answer(char othernumber[]);            //接听电话
        void hangup();                              //挂断电话
        void show();                                //显示普通手机信息
};
```

普通手机类成员函数的定义

```
void mobile::init(char *number,char *pt,float pri) // 赋值
{           strcpy(mynumber,number);
            strcpy(m_type,pt);
            price=pri;
}
void mobile::dial()
{           cout<<"Dialing number is"<<mynumber<<endl;
            cout<<"Dialing on..."<<endl;
}
void mobile::answer(char *othernumber)
{           cout<<"Answering number is"<<othernumber<<endl;
            cout<<"Answering in..."<<endl;
}
void mobile::hangup()
{           cout<<"Hanging up..."<<endl;}
void mobile::show()
{           cout<<mynumber<<'\t'<<m_type<<'\t'<<price<<endl;}
```

智能手机类的定义

```
class smartphone:public mobile//派生类，public是继承修饰符
{
private:
        char OS[20];        //交互式操作系统，派生类新增数据成员
        int memory;         //存储卡容量，派生类新增数据成员
public:
        smartphone()
        {
                init("00000000000","Non_type",0,"Non_OS",0);
        }
        void init(char *number,char *pt,float pri,char *os,int mem);
        //派生类初始化
        void send(char othernumber[],char message[]);                    //发送短信
        void showmemory();          //显示内存大小
        void show();                //显示智能手机信息
};
```

智能手机类成员函数的定义

```
void smartphone::init(char *number,char *pt,float pri,char *os,int mem)//函数覆盖
{       mobile::init(number,pt,pri);  //调用基类成员函数
        strcpy(OS,os);      //操作系统初始化
        memory=mem;         //内存初始化
}
void smartphone::send(char othernumber[],char message[])
{       cout<<"Sending message is "<<message<<"to"<<othernumber<<endl;
        cout<<"Sending on..."<<endl;
}
void smartphone::show()
{
        mobile::show();
        cout<<OS<<'\t'<<memory<<endl;
}
void smartphone::showmemory()
{       cout<<"Memory is:"<<memory<<endl;}
```

测试主函数

```
int main()
{       mobile m;       //声明手机对象
        smartphone m1;  //声明智能手机对象
        m.init("1111111111","motorola",3000);       //调用基类的init
        m.dial();
        m.answer("2222222222");
        m.hangup();
        m.show();
        //调用派生类的init
        m1.init("3333333333","sungxing",5000,"windows8",2048);
        m1.send("2222222222","hello!");
        m1.dial();
        m1.answer("2222222222");
        m1.hangup();
        m1.showmemory();
        m1.show();
        return 0;
}
```

感谢收看！