第5章　继　　承

在C程序设计中，一般要为每个应用单独地开发，这是由于每种应用的要求、程序结构和编码都不同。由于C语言缺乏软件重用的机制，所以这种方法重复工作很多。面向对象技术强调软件的可重用性，C++的模板和类的继承机制较好地解决了软件重用问题，模板已在第3章详细讨论，本章着重介绍类的继承方法。

5.1　继承与派生

5.1.1　继承与派生的概念

类的继承是指新类从已有类中得到已有类的特性。从已有类产生新类的过程就是类的派生。类的继承使程序员无须修改已有类，只要在已有类的基础上增加少量代码或修改少量代码得到新类，较好地解决了代码重用问题。由已有类产生新类时，已有类称为基类或父类，新类称为派生类或子类，新类在包含已有类的特征的同时也可以加入新特性。派生类也可以作为基类再次派生出新类，从而形成类的层次结构。

下面通过实例说明为什么要使用继承。现有一个Person(人)类，包含有name(姓名)、age(年龄)、sex(性别)等数据成员与相关成员函数，具体声明如下：

```
//声明 Person(人)类
class Person
{
protected:
//数据成员
    char name[18];                                          //姓名
    int age;                                                //年龄
    char sex[3];                                            //性别

public:
//公有函数
    Person(const char nm[],int ag,const char sx[]):age(ag){ strcpy(name,nm);
        strcpy(sex,sx); }                                   //构造函数
    void SetName(const char nm[]) { strcpy(name,nm); }      //设置姓名
    void SetAge(int ag) { age=ag;; }                        //设置年龄
    void SetSex(const char sx[]) { strcpy(sex,sx); }        //设置性别
    const char * GetName() const { return name; }           //返回姓名
    int GetAge() const { return age; }                      //返回年龄
    const char * GetSex() const { return sex; }             //返回性别
    void Show() const;                                      //显示相关信息
};
```

现在要声明 Student(学生)类,包含有 num(学号)、name(姓名)、age(年龄)、sex(性别)数据成员与相关成员函数,具体声明如下:

```
//声明 Student(学生)类
class Student
{
protected:
//数据成员
    int num;                                                    //学号
    char name[18];                                              //姓名
    int age;                                                    //年龄
    char sex[3];                                                //性别

public:
//公有函数
    Student(int n,const char nm[],int ag,const char sx[]): num(n),age(ag)
    { strcpy(name,nm); strcpy(sex,sx); }                        //构造函数
    void SetNum(int n) { num=n; }                               //设置学号
    void SetName(const char nm[]) { strcpy(name,nm); }          //设置姓名
    void SetAge(int ag) { age=ag; }                             //设置年龄
    void SetSex(const char sx[]) { strcpy(sex,sx); }            //设置性别
    int GetNum() const { return num; }                          //返回学号
    const char * GetName() const { return name; }               //返回姓名
    int GetAge() const { return age; }                          //返回年龄
    const char * GetSex() const { return sex; }                 //返回性别
    void Show() const;                                          //显示相关信息
};
```

从以上两个类的声明中可以看出,这两个类中的数据成员和成员函数大部分是相同的。只要在类 Person(人)的基础上再增加数据成员 num(学号)、成员函数 SetNum()和 GetNum(),然后再对成员函数 Show()做适当修改,就可以声明 Student(学生)类。这样声明的两个类的代码严重重复。为提高代码的重用性,引入继承机制,将 Student 类说明成类 Person 的派生类,这样相同的成员在 Student 类中就不需要再次进行声明。

说明:在 Person 类和 Student 类中,使用了关键字 protected 将相关数据成员说明成保护成员。保护成员不但可以被本类的成员函数和友元函数所访问,还可以被本类的派生类的成员函数和友元函数访问,但类外的非友元函数的访问都是非法的。

5.1.2 派生类的声明

为了理解从类派生出另一个类,观察如下从 Person 类派生出 Student 类的方法。

```
//声明 Person(人)类
class Person
{
protected:
//数据成员
```

```
    char name[18];                                  //姓名
    int age;                                        //年龄
    char sex[3];                                    //性别

public:
//公有函数
    …
};

//声明 Student(学生)类
class Student: public Person                        //声明为 Person 类的派生类
{
protected:
//数据成员
    int num;                                        //学号

public:
//公有函数
    void SetNum(int n) { num=n; }                   //设置学号
    int GetNum() const { return num; }              //返回学号
    …
};
```

Person 类和 Student 类之间的继承关系可用图 5.1 表示,图中的箭头表示“继承于”,图 5.1 表示 Student 类继承于 Person 类。

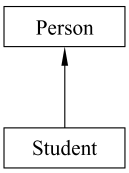

图 5.1 Person 类和 Student 类的继承关系

不难发现,在类名 Student 的“:”后面跟着关键字 public 与类名 Person,表示 Student 类继承 Person 类的特性。其中 Person 类为直接基类,简称为基类,Student 类是直接派生类,简称为派生类。关键字 public 指出派生的方式,告诉编译程序,派生类 Student 是基类 Person 的公有派生类。

一个派生类只有一个直接基类的情况,称为单继承。一个派生类同时有多个直接基类的情况称为多继承。本节先介绍单继承,在 5.4 节会讨论多继承。

对于单继承,声明派生类的一般格式如下:

```
class 派生类名: 继承方式 基类名
{
    //派生类新增的数据成员和成员函数
    …
};
```

其中,“基类名”是一个已经定义的类的名称;“派生类名”是继承原有类的特性而生成的新类的名称;“继承方式”表示如何访问从基类继承的成员,可以是关键字 private、protected 和 public,分别表示私有继承、保护继承和公有继承。因此由 Person 类派生出 Student 类可以

采用如下 3 种格式。

(1) 格式 1：公有继承。

```
class Student: public Person                    //声明为 Person 类的公有派生类
{
    …
};
```

(2) 格式 2：私有继承。

```
class Student: private Person                   //声明为 Person 类的私有派生类
{
    …
};
```

(3) 格式 3：保护继承。

```
class Student: protected Person                 //声明为 Person 类的保护派生类
{
    …
};
```

如果不明确给出继承方式关键字，系统会默认为私有继承(private)，但在一般情况下会采用公有继承方式。

【例 5.1】 声明基类 Person 和派生类 Student 说明由基类派生出派生类的方法示例，程序演示见 5011.mp4～5013.mp4。


```
//文件路径名:e5_1\main.cpp
#pragma warning(disable:4996)           //编译预处理命令,禁止对代号为 4996 的警告
#include <iostream>                     //编译预处理命令
#include <string.h>                     //编译预处理命令
using namespace std;                    //使用命名空间 std

//声明 Person(人)类
class Person
{
protected:
//数据成员
    char name[18];                      //姓名
    int age;                            //年龄
    char sex[3];                        //性别
```

```
public:
//公有函数
    Person(const char nm[],int ag,const char sx[]): age(ag)     //构造函数
    { strcpy(name,nm); strcpy(sex,sx); }
    void SetName(const char nm[]) { strcpy(name,nm); }          //设置姓名
    void SetAge(int ag) { age=ag;; }                            //设置年龄
    void SetSex(const char sx[]) { strcpy(sex,sx); }            //设置性别
    const char * GetName() const { return name; }               //返回姓名
    int GetAge() const { return age; }                          //返回年龄
    const char * GetSex() const { return sex; }                 //返回性别
    void Show() const;                                          //显示相关信息
};

void Person::Show() const                                       //显示相关信息
{
    cout<<"姓名:"<<name<<endl;                                  //显示姓名
    cout<<"年龄:"<<age<<endl;                                   //显示年龄
    cout<<"性别:"<<sex<<endl;                                   //显示性别
}

//声明 Student(学生)类
class Student: public Person                                    //声明为 Person 类的派生类
{
protected:
//数据成员
    int num;                                                    //学号

public:
//公有函数
    Student(int n,const char nm[],int ag,const char sx[]):Person(nm,ag,sx),
    num(n) {}                                                   //构造函数
    void SetNum(int n) { num=n; }                               //设置学号
    int GetNum() const { return num; }                          //返回学号
    void Show() const;                                          //显示相关信息
};

void Student::Show() const                                      //显示相关信息
{
    cout<<"学号:"<<num<<endl;                                   //显示学号
    cout<<"姓名:"<<name<<endl;                                  //显示姓名
    cout<<"年龄:"<<age<<endl;                                   //显示年龄
    cout<<"性别:"<<sex<<endl;                                   //显示性别
}

int main()                                                      //主函数 main()
```

```
{
    Student s(2008101,"张倩",28,"男");                    //定义对象
    s.Show();                                              //显示相关信息

    return 0;                                              //返回值 0, 返回操作系统
}
```

程序运行时屏幕输出如下：

```
学号：2008101
姓名：张倩
年龄：28
性别：男
```

从本例可以看出，派生类的构造函数初始化表要包含“基类构造函数名(参数表)”，表示调用基类的构造函数初始化基类的数据成员，一般派生类的构造函数参数初始化表格式如下：

```
基类构造函数名(参数表), 数据成员 1(参数表 1), 数据成员 2(参数表 2), …
```

5.1.3 派生类与基类中的同名成员

在例 5.1 中，在派生类与基类中都定义了成员函数 Show()。实际上，在声明派生类时 C++ 允许在派生类中定义的成员与基类中的成员名字相同。也就是说，派生类可以重新定义与基类成员同名的成员。在派生类中使用这样的成员意味着访问在派生类中重新定义的成员。为了在派生类中使用基类的同名成员，必须在该成员名之前加上基类名和作用域运算符“::”，访问基类的同名成员，具体使用下面格式：

```
基类名::成员名
```

【例 5.2】 通过“基类名：：成员名”方试调用基类成员来改写例 5.1，程序演示见 5021.mp4～5023.mp4。


```
//文件路径名:e5_2\main.cpp
#pragma warning(disable:4996)                  //编译预处理命令,禁止对代号为 4996 的警告
#include <iostream>                            //编译预处理命令
#include <string.h>                            //编译预处理命令
using namespace std;                           //使用命名空间 std

//声明 Person(人)类
class Person
{
protected:
```

```
//数据成员
    char name[18];                                          //姓名
    int age;                                                //年龄
    char sex[3];                                            //性别

public:
//公有函数
    Person(const char nm[],int ag,const char sx[]): age(ag)   //构造函数
    { strcpy(name,nm); strcpy(sex,sx); }
    void SetName(const char nm[]) { strcpy(name,nm); }      //设置姓名
    void SetAge(int ag) { age=ag;; }                        //设置年龄
    void SetSex(const char sx[]) { strcpy(sex,sx); }        //设置性别
    const char * GetName() const { return name; }           //返回姓名
    int GetAge() const { return age; }                      //返回年龄
    const char * GetSex() const { return sex; }             //返回性别
    void Show() const;                                      //显示相关信息
};

void Person::Show() const                                   //显示相关信息
{
    cout<<"姓名:"<<name<<endl;                              //显示姓名
    cout<<"年龄:"<<age<<endl;                               //显示年龄
    cout<<"性别:"<<sex<<endl;                               //显示性别
}

//声明 Student(学生)类
class Student: public Person                              //声明为 Person 类的派生类
{
protected:
//数据成员
    int num;                                                //学号

public:
//公有函数
    Student(int n,const char nm[],int ag,const char sx[]):Person(nm,ag,sx),
        num(n) {}                                           //构造函数
    void SetNum(int n) { num=n; }                           //设置学号
    int GetNum() const { return num; }                      //返回学号
    void Show() const;                                      //显示相关信息
};

void Student::Show() const                                  //显示相关信息
{
    cout<<"学号:"<<num<<endl;                               //显示学号
    Person::Show();                                 //调用基类 Person 的成员函数 Show()
```

```
}

int main()                                                    //主函数 main()
{
    Student s(2008101,"张倩",28,"男");                        //定义对象
    s.Show();                                                 //显示相关信息

    return 0;                                                 //返回值 0, 返回操作系统
}
```

程序运行时屏幕输出如下：

```
学号: 2008101
姓名: 张倩
年龄: 28
性别: 男
```

在例 5.1 和本例中，基类与派生类中的成员函数 Show()有大部代码是相同的，本例采用

```
Person::Show();
```

通过作用域运算符“::”指定调用基类 Person 的成员函数 Show()，这样避免了代码的重复，使用程序更短小。

5.2 继 承 方 式

在 5.1.2 节中关于派生类的声明格式中已讨论过继承方式，派生类对基类的继承方式包括公有继承(public)、私有继承(private)和保护继承(protected)3 种。继承方式用于规定基类成员在派生类中的访问权限。

(1) 公有继承。基类的公有成员和保护成员在派生类中仍然保持为公有成员和保护成员的访问权限，基类的私有成员在派生类中不可访问。

(2) 私有继承。基类的公用成员和保护成员在派生类中成了私有成员，基类的私有成员在派生类中不可访问。

(3) 保护继承。基类的公用成员和保护成员在派生类中成了保护成员，基类的私有成员仍在派生类中不可访问。

5.2.1 公有继承

采用公有继承方式建立的派生类称为公有派生类，其基类称为公有基类。采用公有继承方式时，基类的公有成员和保护成员在派生类中仍然保持公有成员和保护成员的访问权限，基类的私有成员在派生类中并没有成为派生类的私有成员，只有基类的成员函数或友元函数可以引用它，而不能被派生类的成员函数或友元函数引用，因此成为派生类中的不可访问的成员。公有继承方式的访问控制机制如图 5.2 所示，图中箭头表示“可以访问”。

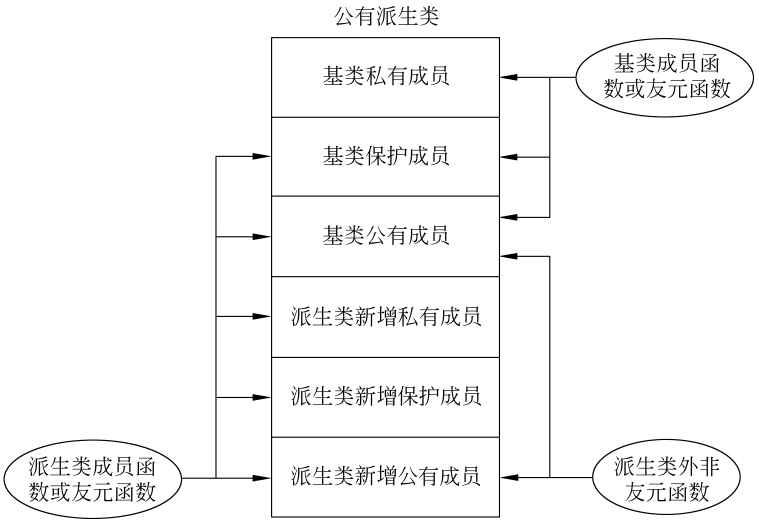

图 5.2　公有继承方式访问控制机制示意图

【例 5.3】 公有继承方式使用实例，程序演示见 5031.mp4～5033.mp4。


```
//文件路径名:e5_3\main.cpp
#pragma warning(disable:4996)                    //编译预处理命令,禁止对代号为 4996 的警告
#include <iostream>                              //编译预处理命令
#include <string.h>                              //编译预处理命令
using namespace std;                             //使用命名空间 std

//声明 Person(人)类
class Person
{
private:
//私有成员
    int age;                                     //年龄

protected:
//保护成员
    char sex[3];                                 //性别

public:
//公有成员
    char name[18];                               //姓名
    Person(const char nm[],int ag,const char sx[]):age(ag)     //构造函数
```

```
    {   strcpy(name,nm); strcpy(sex,sx); }
    int GetAge() const { return age; }                          //返回年龄
};

//声明 Student(学生)类
class Student: public Person                                    //声明为 Person 类的公有派生类
{
protected:
//数据成员
    int num;                                                    //学号

public:
//公有函数
    Student(int n,const char nm[],int ag,const char sx[]):Person(nm,ag,sx),
        num(n) {}                                               //构造函数
    void Show() const;                                          //显示相关信息
};

void Student::Show() const                                      //显示相关信息
{
    cout<<"学号:"<<num<<endl;                                   //显示学号,num 为派生类成员
    cout<<"姓名:"<<name<<endl;
                                //显示姓名,name 为基类公有成员,在派生类中保持为公有成员
//  cout<<"年龄:"<<age<<endl;
                                //显示年龄,错,age 为基类私有成员,在派生类中不可访问
    cout<<"年龄:"<<GetAge()<<endl;  //显示年龄,GetAge()为基类公有成员
    cout<<"性别:"<<sex<<endl;
                                //显示性别,sex 为基类保护成员,在派生类中保持为保护成员
}
int main()                      //主函数 main()
{
    Student s(2008101,"张倩",28,"男");          //定义对象
    cout<<s.name<<"的信息:"<<endl;     //name 为基类公有成员,可由类外非友元函数访问
    s.Show();                           //显示相关信息

    return 0;                           //返回值 0, 返回操作系统
}
```

程序运行时屏幕输出如下：

```
张倩的信息:
学号: 2008101
姓名: 张倩
年龄: 28
性别: 男
```