第5章 类的继承

1．将《C++面向对象程序设计（第 4 版）》中例 5.1 的程序片段补充和改写成一个完整、正确的程序，用公用继承方式。在程序中应包括输入数据的函数，在程序运行时输入 num，name，sex，age，addr 的值，程序应输出以上 5 个数据的值。

【解】 根据题意，写出程序如下：

```
#include<iostream>
using namespace std;
class Student
  {public:
     void get_value()
       {cin>>num>>name>>sex;}          //输入基类的 3 个私有数据成员的值
     void display()
       {cout<<"num:"<<num<<endl;       //输出基类的 3 个私有数据成员的值
        cout<<"name:"<<name<<endl;
        cout<<"sex:"<<sex<<endl;}
   private:
     int num;
     char name[10];
     char sex;
  };

class Student1: public Student          //定义公用派生类 Student1
 {public:
   void get_value_1()                   //函数的作用是输入 5 个数据成员的值
    {get_value();                       //调用函数，输入基类的 3 个私有数据成员的值
     cin>>age>>addr;                    //输入派生类的两个私有数据成员的值
    }
   void display_1()
      {cout<<"age:"<<age<<endl;         //输出派生类两个私有数据成员的值
       cout<<"address:"<<addr<<endl;
      }
```

```
  private:
    int age;
    char addr[30];
 };

int main()
  {Student1 stud1;                    //定义公用派生类 Student1 的对象 stud1
   stud1.get_value_1();               //输入 5 个数据
   stud1.display();                   //输出基类的 3 个私有数据成员的值
   stud1.display_1();                 //输出派生类两个私有数据成员的值
   return 0;
  }
```

运行结果如下：

```
10101 Li M 20 Beijing↙
num:10101
name:Li
sex:M
age:20
address:Beijing
```

实际上，程序还可以改进，在派生类的 display_1 函数中调用基类的 display 函数，在主函数中只要写一行

```
stud1.display_1();
```

即可输出 5 个数据。本题程序只是为了说明派生类成员的引用方法。读者可参考本章第 2 题的程序。

2．将《C++面向对象程序设计（第 4 版）》中例 5.2 的程序片段补充和改写成一个完整、正确的程序，用私有继承方式。在程序中应包括输入数据的函数，在程序运行时输入 num，name，sex，age，addr 的值，程序应输出以上 5 个数据的值。

【解】 根据题意，写出程序如下：

```
#include<iostream>
using namespace std;
class Student
  {public:
     void get_value()
      {cin>>num>>name>>sex;}
     void display()
      {cout<<"num:"<<num<<endl;
       cout<<"name:"<<name<<endl;
       cout<<"sex:"<<sex<<endl;}
   private:
```

```
    int num;
    char name[10];
    char sex;
 };

class Student1: private Student          //定义私有派生类 Student1
 {public:
    void get_value_1()
      {get_value();
       cin>>age>>addr;}
   void display_1()
      {display();                        //调用基类中的公用成员函数
       cout<<"age:"<<age<<endl;          //引用派生类的私有成员，正确
       cout<<"address:"<<addr<<endl;}    //引用派生类的私有成员，正确
  private:
    int age;
    char addr[30];
 };

int main()
 {Student1 stud1;
  stud1.get_value_1();
  stud1.display_1();                     //只须调用一次 stud1.display_1()
  return 0;
 }
```

本程序能通过编译，可正常运行，运行结果同第 2 题。通过此题，可以看到怎样正确引用基类中的私有成员。

3. 将《C++ 面向对象程序设计（第 4 版）》中例 5.3 的程序修改、补充，写成一个完整、正确的程序，用保护继承方式。在程序中应包括输入数据的函数。

【解】 根据题意，写出程序如下：

```
#include<iostream>
using namespace std;
class Student                            //声明基类
{public:                                 //基类公用成员
   void get_value();
   void display();
 protected :                             //基类保护成员
  int num;
  char name[10];
  char sex;
};

void Student::get_value()
```

```
 {cin>>num>>name>>sex;}

void Student::display()
 {cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl;
 }

class Student1: protected Student   //声明一个保护派生类
  {public:
     void get_value_1();
     void display1();
   private:
     int age;
     char addr[30];
  };

void Student1::get_value_1()
 {get_value();
  cin>>age>>addr;
 }
void Student1::display1()
  {cout<<"num:"<<num<<endl;          //引用基类的保护成员
   cout<<"name:"<<name<<endl;        //引用基类的保护成员
   cout<<"sex:"<<sex<<endl;          //引用基类的保护成员
   cout<<"age:"<<age<<endl;          //引用派生类的私有成员
   cout<<"address:"<<addr<<endl;     //引用派生类的私有成员
  }

int main()
 {Student1 stud1;                    //stud1 是派生类 student1 类的对象
  stud1.get_value_1();               //调用派生类对象 stud1 的公用成员函数
  stud1.display1();                  //调用派生类对象 stud1 的公用成员函数
  return 0;
 }
```

本程序能通过编译，可正常运行，运行结果同第 2 题。

4. 修改《C++ 面向对象程序设计（第 4 版）》中例 5.3 的程序，改为用公用继承方式。上机调试程序，使之能正确运行并得到正确的结果。对这两种继承方式作比较分析，考虑在什么情况下二者不能互相代替。

【解】 根据题意，写出程序如下：

```
#include<iostream>
using namespace std;
class Student                        //声明基类
```

```
 {public:                              //基类公用成员
    void get_value();
    void display();
  protected:                           //基类保护成员
    int num;
    char name[10];
    char sex;
 };

void Student::get_value()
 {cin>>num>>name>>sex;}

void Student::display()
 {cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl;
 }

class Student1:public Student         //声明一个公用派生类
 {public:
    void get_value_1();
    void display1();
  private:
    int age;
    char addr[30];
 };

void Student1::get_value_1()
 {get_value();
  cin>>age>>addr;
 }
void Student1::display1()
 {cout<<"num:"<<num<<endl;            //引用基类的保护成员，合法
  cout<<"name:"<<name<<endl;          //引用基类的保护成员，合法
  cout<<"sex:"<<sex<<endl;            //引用基类的保护成员，合法
  cout<<"age:"<<age<<endl;            //引用派生类的私有成员，合法
  cout<<"address:"<<addr<<endl;       //引用派生类的私有成员，合法
 }

int main()
 {Student1 stud1;            //stud1 是派生类 student1 类的对象
  stud1.get_value_1();       //调用派生类对象 stud1 的公用成员函数 get_value_1
  stud1.display1();          //调用派生类对象 stud1 的公用成员函数 display1
  return 0;
 }
```

本程序能通过编译，可正常运行，运行结果同第 2 题。

将此程序与第 3 题比较，只有在定义派生类时采用继承方式不同（在本题中用公用继承方式代替了第 3 题的保护继承方式），其他部分完全相同，运行结果也完全相同。但千万不要得出错误的结论，以为在任何情况下二者可以互相替换。要作具体分析。

对两个程序的执行过程作比较，见表 5.1。

表 5.1

内　　容	第3题的程序(保护继承)	本题（公用继承）
（1）stud2.get_value_2( );	调用派生类公用成员函数	调用派生类公用成员函数
（2）在 stud2.get_value_2 函数中调用基类 get_value 函数	get_value 函数在派生类中是保护成员函数	get_value 函数在派生类中是公用成员函数
（3）get_value 函数引用 num 等	num 等为保护成员，可以引用	num 等为保护成员，可以引用
（4）stud2.display( );	调用派生类公用成员函数	调用派生类公用成员函数
（5）在 stud2.display 函数中引用 num,name,sex	num,name,sex 是保护成员	num,name,sex 是保护成员
（6）在 stud2.display 函数中引用 age,addr	age,addr 是保护成员	age,addr 是保护成员

可以看到，两者只有第 2 点是不同的，在保护继承时， get_value 函数在派生类中是保护成员函数，而在公用继承时，它在派生类中是公用成员函数，但都可以被派生类的成员函数调用，效果相同，因此，两者的执行过程是相同的，运行结果也当然相同。

但是如果把程序修改如下（程序 2）：

```
#include<iostream>
using namespace std;
class Student                                   //声明基类
  {public:                                      //基类公用成员
    void get_value();
    void display();
   protected:                                   //基类保护成员
    int num;
    char name[10];
    char sex;
  };

void Student::get_value()                       //函数的作用是输入 3 个数据
 {cin>>num>>name>>sex;}

void Student::display()                         //函数的作用是输出 3 个数据
 {cout<<"num:"<<num<<endl;
  cout<<"name:"<<name<<endl;
  cout<<"sex:"<<sex<<endl;
 }
 class Student1:protected Student               //声明一个保护派生类
```

```
 {public:
    void get_value_1();
    void display1();
  private:
    int age;
    char addr[30];
 };

void Student1::get_value_1()    //函数的作用是输入两个数据
 {cin>>age>>addr;}

void Student1::display1()       //函数的作用是输出两个数据
  {cout<<"age:"<<age<<endl;
   cout<<"address:"<<addr<<endl;
  }

int main()
  {Student1 stud1;              //stud1 是派生类 student1 类的对象
   Stud1.get_value();           //出错！调用派生类对象 stud1 的保护函数
   Stud1.get_value_1();         //正确！调用派生类对象 stud1 的公用成员函数
   Stud1.display();             //出错！调用派生类对象 stud1 的保护函数
   Stud1.display1();            //正确！调用派生类对象 stud1 的公用成员函数
   return 0;
  }
```

在定义派生类 Student1 时声明为公用继承，程序能通过编译，正常运行。但如果改为保护继承，则编译时出错。对程序 1 和程序 2 的分析见表 5.2。

表　5.2

内　　容	程序 1（公用继承）	程序 2（保护继承）
stud1.get_value( );	调用基类的公用函数 get_value，它在派生类中仍为公用函数	调用基类的公用函数 get_value，它在派生类中为保护函数
stud1.display( );	调用基类的公用函数 display，它在派生类中仍为公用函数	调用基类的公用函数 display，它在派生类中为保护函数

get_value 和 display 函数是基类的公用函数，在公用继承时，它在派生类中仍为公用函数，可以在类外通过对象名来调用它，但在保护继承时，它在派生类中为保护函数，可以被派生类的成员函数调用，但不能被派生类外通过对象名调用。因此编译出错。

不同的继承方式使派生类的成员具有不同的特性，会对程序的执行产生不同的影响。在一般情况下，用 public 和用 protected 声明继承方式是不等价的。应当仔细分析程序，作出判断。

5．有以下程序结构，请分析访问属性。

```
class A                         //A 为基类
  {public:
```

```
    void f1();
    int i;
  protected:
    void f2();
    int j;
  private:
    int k;
};

class B: public A                    //B 为 A 的公用派生类
{public:
   void f3();
  protected:
   int m;
  private:
   int n;
};

class C: public B                    //C 为 B 的公用派生类
{public:
   void f4();
  private:
   int p;
};

int main()
{A a1;                               //a1 是基类 A 的对象
 B b1;                               //b1 是派生类 B 的对象
 C c1;                               //c1 是派生类 C 的对象
 ⋮
 return 0;
}
```

请问：

（1）在 main 函数中能否用 b1.i，b1.j 和 b1.k 引用派生类 B 对象 b1 中基类 A 的成员？

（2）派生类 B 中的成员函数能否调用基类 A 中的成员函数 f 1 和 f 2？

（3）派生类 B 中的成员函数能否引用基类 A 中的数据成员 i，j，k？

（4）能否在 main 函数中用 c1.i，c1.j，c1.k，c1.m，c1.n，c1.p 引用基类 A 的成员 i，j，k，派生类 B 的成员 m，n，以及派生类 C 的成员 p？

（5）能否在 main 函数中用 c1.f 1( )，c1.f 2( )，c1.f 3( )和 c1.f 4( )调用 f 1，f 2，f 3，f 4 成员函数？

（6）派生类 C 的成员函数 f 4 能否调用基类 A 中的成员函数 f 1，f 2 和派生类中的成员函数 f 3？

【解】

（1）可以用 b1.i 引用对象 b1 中的基类 A 的成员 i，因为它是公用数据成员。

不能用 b1.j 引用对象 b1 中的基类 A 的成员 j，因为它是保护数据成员，在类外不能访问。

不能用 b1.k 引用对象 b1 中的基类 A 的成员 k，因为它是私有数据成员，在类外不能访问。

（2）可以调用基类 A 中的成员函数 f1 和 f 2，因为 f1 是公用成员函数，f 2 是保护成员函数，B 对 A 是公有继承方式，因此它们在派生类中仍然保持原有的访问权限，可以被派生类的成员函数访问。

（3）可以引用基类 A 中的数据成员 i 和 j，因为它们在派生类中是公用成员和保护成员，可以被派生类的成员函数访问。不可以引用基类 A 中的数据成员 k，它在派生类中是不可访问的成员。

（4）可以用 c1.i 引用对象 c1 中基类 A 的成员 i，不能用 c1.j，c1.k 引用基类 A 的成员 j 和 k，因为它们是保护成员和私有成员，不能被类外访问。也不能访问 c1 中派生类 B 的成员 m，n，它们也是保护成员和私有成员，不能被类外访问。也不能访问派生类对象 c1 中的私有成员 p。

（5）可以调用成员函数 f1，f 3，f4，它们是公用成员函数。不能调用成员函数 f 2，因为它是保护成员函数。

（6）可以，f1，f 3 是公用成员函数，f 2 是保护成员函数，都可以被派生类 C 的成员函数调用。

6．有以下程序结构，请分析所有成员在各类的范围内的访问权限。

```
class A                                    //基类
  {public:
     void f1();
   protected:
     void f2();
   private:
     int i;
  };

class B:public A                           //B 为 A 的公用派生类
  {public:
     void f3();
     int k;
   private:
     int m;
  };

class C:protected B                        //C 为 B 的保护派生类
  {public:
     void f4();
```

```
  protected:
    int n;
  private:
    int p;
 };

class D:private C                          //D 为 C 的私有派生类
 {public:
    void f5();
  protected:
    int q;
  private:
    int r;
 };
void main()
 { A a1;                                   //a1 是基类 A 的对象
   B b1;                                   //b1 是派生类 B 的对象
   C c1;                                   //c1 是派生类 C 的对象
   D d1;                                   //d1 是派生类 D 的对象
     ⋮
 }
```

【解】 各成员在各类的范围内的访问权限如表 5.3 所示。

表 5.3

类的范围	f1	f2	i	f3	k	m	f4	n	p	f5	q	r
基类 A	公用	保护	私有									
公用派生类 B	公用	保护	不可访问	公用	公用	私有						
保护派生类 C	保护	保护	不可访问	保护	保护	不可访问	公用	保护	私有			
私有派生类 D	私有	私有	不可访问	私有	私有	不可访问	私有	私有	不可访问	公用	保护	私有

根据以上的分析，可以知道：

（1）在派生类外，可以通过对象调用 f5 函数，如 d1.f5( )。其他成员均不能访问。

（2）派生类 D 的成员函数 f5 可以访问基类 A 的成员 f1 和 f2，派生类 B 的成员 f3 和 k，派生类 C 的成员 f4 和 n，派生类 D 的成员 q 和 r。

（3）派生类 C 的成员函数 f4 可以访问基类 A 的成员 f1 和 f2，派生类 B 的成员 f3 和 k，派生类 C 的成员 n 和 p。

（4）派生类 B 的成员函数 f3 可以访问基类 A 的成员 f1 和 f2，派生类 B 的成员 k 和 m。

（5）基类 A 的成员函数 f1 可以访问基类 A 的成员 f2 和 i。