派生类定义

- 在一个单继承的继承关系中，定义派生类的语法为：

```
class 派生类名：继承方式 基类名
{
    派生类成员声明;
};
```

- 其中，继承方式包括public（公有继承）、protected（保护继承）和private（私有继承）三种，其含义将在后面介绍。这里我们先使用public作为继承方式。

【例3-1】定义Person类及其派生类Student类。

```
// Person.h
#include <iostream>
using namespace std;
class Person
{
public:
    void SetName(char *name)
    { strcpy(m_name, name); }
    char* GetName() { return m_name; }
    void SetSex(bool sex) { m_sex = sex; }
    bool GetSex() { return m_sex; }
private:
    char m_name[20];        // 姓名
    bool m_sex;    // 性别（true：男，false：女）
};
```

```
// Student.h
#include "Person.h"
#include <iostream>
using namespace std;
class Student : public Person
{
public:
    void SetSNO(char *sno)
    { strcpy(m_sno, sno); }
    char* GetSNO() { return m_sno; }
    void SetMajor(char *major)
    { strcpy(m_major, major); }
    char* GetMajor() { return m_major; }
    void DisplayInfo();
private:
    char m_sno[8];              // 学号
    char m_major[20];       // 专业
};
```

```cpp
// Student.cpp
#include "Student.h"
void Student::DisplayInfo()
{
    cout<<"学生信息："<<endl
        <<"学号："<<m_sno<<endl
        <<"姓名："<<GetName()<<endl
        <<"性别：";
    if (GetSex()==true) cout<<"男"<<endl;
    else cout<<"女"<<endl;
    cout<<"专业："<<m_major<<endl;
}
```

```cpp
// testStudent.cpp
#include "Student.h"
int main()
{
    Student student;
    student.SetSNO("1210101");
    student.SetName("张三");
    student.SetSex(true);
    student.SetMajor("计算机应用");
    student.DisplayInfo();
    return 0;
}
```

基类中的私有成员在派生类的成员函数中无法直接访问。例如，如果将DisplayInfo()函数中的GetName()和GetSex()函数调用分别改为m_name和m_sex，则编译程序时会报错。

如果希望在派生类中直接访问基类的成员，则可以将基类的成员声明为public或protected。下表给出了public、private和protected三种类成员访问控制方式的含义。

访问控制方式	含　义
public	类的公有成员，在任何地方都可以直接访问
private	类的私有成员，只在该类的成员函数中可以直接访问，在其他地方均不能直接访问
protected	类的保护成员，在该类及其派生类的成员函数中可以直接访问，在其他地方不能直接访问

例如，只要将例3-1中的Person类定义改为：

```
class Person
{
public:
    void SetName(char *name)
    { strcpy(m_name, name); }
    char* GetName()
    { return m_name; }
    void SetSex(bool sex)
    { m_sex = sex; }
    bool GetSex() { return m_sex; }
protected:
    char m_name[20];        // 姓名
    bool m_sex;             // 性别（true：男，false：女）
};
```

就可以在Student类的成员函数中直接访问Person的私有数据成员m_name和m_sex.