面向对象方法程序实例
– 友元和运算符重载

【例2-22】设计一个Student类，每名学生包含学号、姓名和总评成绩3个属性，学生的学号、姓名和总评成绩可以通过初始化得到，学生的总评成绩可以通过赋值运算符“=”更改。能够通过普通函数display函数输出学生的学号、姓名和总评成绩等信息。要求：

①合理地设计属性和方法。

②合理地设计类成员的访问控制方式和友元。

③考虑如何初始化学生对象的“学号”和“姓名”属性。

④考虑如何通过“对象名=总评成绩”实现给对象的“总评成绩”属性赋值。

⑤用主函数测试类。

⑥要求用多文件结构实现程序。

类名	Student	
	含义	C++描述
属性	学号 姓名 总评成绩	private: char *m_pcSno; private: char *m_pcSname; private: int m_nScore;

方法	构造函数
	
	析构函数
	=运算符重载
		public: Student(char *pcSno, char *pcSname, int nScore);
		public: ~Student();
		public: Student& operator=(int nScore);

友元	显示学生信息	friend void display(const Student &stu);

```
// Student.h
class Student
{
private:
    char *m_pcSno;        // 学号
    char *m_pcSname;      // 姓名
    int m_nScore;         // 总评成绩
public:
    Student(char *pcSno, char *pcSname, int nScore); // 构造函数
    ~Student();           // 析构函数
    Student& operator=(int nScore); // =运算符重载
    friend void display(const Student &stu);
};
```

```cpp
// Student.cpp
#include "Student.h"
#include <iostream>
using namespace std;

Student::Student(char *pcSno, char *pcSname,
int nScore)
{
    m_pcSno = new char[strlen(pcSno)+1];
    strcpy(m_pcSno, pcSno);
    m_pcSname = new char[strlen(pcSname)+1];
    strcpy(m_pcSname, pcSname);
    m_nScore = nScore;
}
Student::~Student()
{
    delete []m_pcSno;
    delete []m_pcSname;
}
```

```cpp
Student& Student::operator=(int nScore)
{
    m_nScore = nScore;
    return *this;
}
void display(const Student &stu)
{
    cout<<"学号："<<stu.m_pcSno<<endl
        <<"姓名："<<stu.m_pcSname<<endl
        <<"总评成绩："<<stu.m_nScore<<endl;
}
```

```cpp
// testStudent.cpp
#include "Student.h"
#include <iostream>
using namespace std;
int main()
{
    Student s("YC1710011", "张明", 90);
    display(s);
    s = 95;
    display(s);
    return 0;
}
```

运行结果：

```
学号：YC1710011
姓名：张明
总评成绩：90
学号：YC1710011
姓名：张明
总评成绩：95
```