类的友元

- 类成员访问控制方式使得普通函数无法直接访问类的私有成员或保护成员，一个类中的函数也无法直接访问另一个类的私有成员或保护成员。在程序中，如果普通函数或另一个类中的函数需要经常通过类提供的公有接口来访问类的私有成员或保护成员，为了提高程序运行的效率，可以将他们声明为类的朋友——友元，它们就可以直接访问类的任何成员了。友元提供了一个一般函数与类的成员之间、不同类的成员之间进行数据共享的机制。

友元分为3类，用friend关键字来声明。

（1）友元函数

将普通函数声明为类的友元函数的形式为：

friend <数据类型><友元函数名>(参数表);

例如：下面是将普通函数“int fun(int x);”声明为A类的友元函数。声明后，普通函数“int fun(int x);”有权访问A类中的任何成员，包括私有成员和保护成员。

```
class A
{
    friend int fun(int x);              //声明普通函数为友元函数
}
```

（2）友元成员

将一个类的成员函数声明为另一个类的友元函数，就称这个成员函数就是友元成员。声明友元成员的形式为：

```
friend <类型> <含有友元成员的类名>::<友元成员名>(参数表);
```

例如：下面是将A类中的成员函数声明为B类的友元函数。声明后，A类的成员函数“int fun(int x);”有权访问B类中的任何成员，包括私有成员和保护成员。

```
class A
{
    ......
    int fun(int x);
    ......
}
class B
{
    ......
    friend int A::fun(int x); //声明友元成员
    ......
}
```

（3）友类将一个类声明为另一个类的友类，它的语法形式为：

friend <友类名>;

或 friend class <友类名>;

例如：下面是将A类声明为B类的友类。声明后，A类的任何成员函数都有权访问B类中的任何成员，包括私有成员和保护成员。

```
class B
{
    friend class A;
}
```

- 【例2-15】友元使用的示例。下面的代码中，普通函数getStudentInfo()声明为学生类的友元函数。教师可以修改学生的成绩，于是将教师类的成员函数SetScore()声明为学生类的友元。管理员类对学生类具有更多的操作权限，所以，将管理员类声明为学生类的友类。

```
// DefineClass.h
class Student;
void getStudentInfo(Student& s);
class Teacher
{
public:
    void SetScore(Student& s,double sc);
private:
    long m_number;
    char m_name[10];
};
class Manager
{
public:
    void ModifyStudentInfo(Student& s, long , char *, double);
private:
    long m_number;
    char m_name[10];
};
```

```
class Student
{
public:
    friend void getStudentInfo(Student& s); //声明友元函数
    friend void Teacher::SetScore(Student& s,double sc); //声明友元成员
    friend class Manager; //声明友类
    double GetScore()
    {
        return m_score;
    }
private:
    long m_number;
    char m_name[10];
    double m_score;
};
```

```
// DefineClass.cpp
#include "DefineClass.h"
#include <string>
#include <iostream>
using namespace std;
void Teacher::SetScore(Student& s,double sc)
{    s.m_score=sc;    } //直接访问学生对象s的私有成员m_score
void Manager::ModifyStudentInfo(Student& s, long number, char * name, double sc)
{
    s.m_number=number; //直接访问学生对象s的私有成员m_number
    strcpy(s.m_name,name); //直接访问学生对象s的私有成员m_name
    s.m_score=sc; //直接访问学生对象s的私有成员m_score
}
void getStudentInfo(Student& s)
{
    cout<<"学号:"<<s.m_number<<"  姓名:"<<s.m_name<<"  成绩:"<<s.m_score<<endl;
}
```

```cpp
// testFriendMember.cpp
#include <iostream>
#include "DefineClass.h"
using namespace std;
int main()
{
    Teacher t;
    Manager m;
    Student s;
    t.SetScore(s,85.5);
    m.ModifyStudentInfo(s,1201201,"周海洋",95);
    getStudentInfo(s);
    return 0;
}
```