**实验1.**

程序代码

1. 对象ob作为函数sqr\_it的形参(Tr ob)

#include<iostream>  
using namespace std;  
class Tr {  
public:  
    Tr(int n)  
    {  
        i = n;  
    }  
    void set\_i(int n)  
    {  
        i = n;  
    }  
    int get\_i()  
    {  
        return i;  
    }  
private:  
    int i;  
};  
void sqr\_it(Tr ob)   
{  
    ob.set\_i(ob.get\_i() \* ob.get\_i());  
    cout << "在函数sqr\_it内，形参对象ob的数据成员i的值为:" << ob.get\_i();  
    cout << endl;  
}  
int main()  
{  
    Tr obj(10);  
    cout << "调用函数sqr\_it前, 实参对象obj的数据成员i的值为:";  
    cout << obj.get\_i() << endl;  
    sqr\_it(obj);  
    cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";  
    cout << obj.get\_i() << endl;  
    return 0;  
}

2.对象指针ob作为函数sqr\_it的形参（Tr\*ob）

#include<iostream>

using namespace std;

class Tr {

public:

Tr(int n)

{

i = n;

}

void set\_i(int n)

{

i = n;

}

int get\_i()

{

return i;

}

private:

int i;

};

void sqr\_it(Tr\*ob)

{

ob->set\_i(ob->get\_i() \* ob->get\_i());

cout << "在函数sqr\_it内，形参对象ob的数据成员i的值为:" << ob->get\_i();

cout << endl;

}

int main()

{

Tr obj(10);

cout << "调用函数sqr\_it前, 实参对象obj的数据成员i的值为:";

cout << obj.get\_i() << endl;

sqr\_it(&obj);

cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

cout << obj.get\_i() << endl;

return 0;

}

3．进行引用（Tr&ob）

#include<iostream>

using namespace std;

class Tr {

public:

Tr(int n)

{

i = n;

}

void set\_i(int n)

{

i = n;

}

int get\_i()

{

return i;

}

private:

int i;

};

void sqr\_it(Tr&ob)

{

ob.set\_i(ob.get\_i() \* ob.get\_i());

cout << "在函数sqr\_it内，形参对象ob的数据成员i的值为:" << ob.get\_i();

cout << endl;

}

int main()

{

Tr obj(10);

cout << "调用函数sqr\_it前, 实参对象obj的数据成员i的值为:";

cout << obj.get\_i() << endl;

sqr\_it(obj);

cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

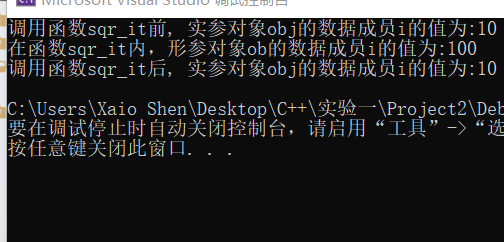
cout << obj.get\_i() << endl;

return 0;

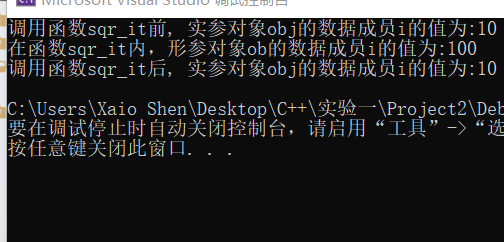
}

运行结果

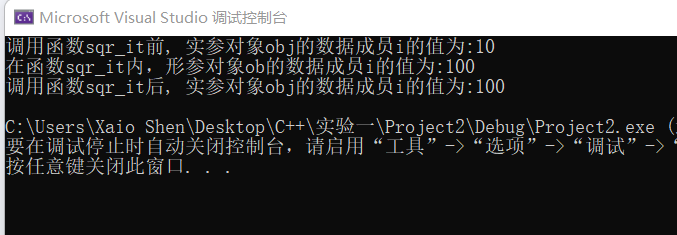
结果1.



结果2.



结果3.



**实验2.**

程序代码

#include <iostream>  
#include <string>  
using namespace std;  
class TStudent  
{  
private:  
    char m\_Name[6];  
    static float m\_ClassMoney;  
public:  
    void InitStudent(const char\*);  
    void ExpendMoney(const float);  
    static void ShowMoney();   
};  
float TStudent::m\_ClassMoney = 1000;  
void TStudent::InitStudent(const char name[])  
{  
    strcpy\_s(m\_Name, name);  
}  
void TStudent::ExpendMoney(const float money)  
{  
    m\_ClassMoney -= money;   
    cout << m\_Name << "花费班费" << m\_ClassMoney << endl;  
}  
void TStudent::ShowMoney()  
{  
    cout << "班费还剩余" << m\_ClassMoney << endl;  
}  
int main()  
{  
    TStudent stu[3];  
    stu[0].InitStudent("A");  
    stu[1].InitStudent("B");  
    stu[2].InitStudent("C");  
    stu[0].ExpendMoney(50);  
    TStudent::ShowMoney();  
    stu[1].ExpendMoney(98.5);  
    TStudent::ShowMoney();  
    stu[2].ExpendMoney(500.53);  
    TStudent::ShowMoney();  
}

运行结果



感想心得

对于就对象传递来说，若要想通过修改函数的形参来改变实参，则需要运用对象指针和对象引用来作为函数参数。

在定义静态变量时，要在类外及主函数之前进行初始化，静态成员属于私有成员，无法在主函数内被访问。

主程序调用函数的实际参数时系统会将实际参数的数值传递并复制给函数中相对应的形式参数，函数内的形式参数执行完毕后不会修改主程序中调用时变量自身的内容。值传递的作用更多是让函数内部了解外部参数的值。值传递是单向的，只能由实参传向形参。