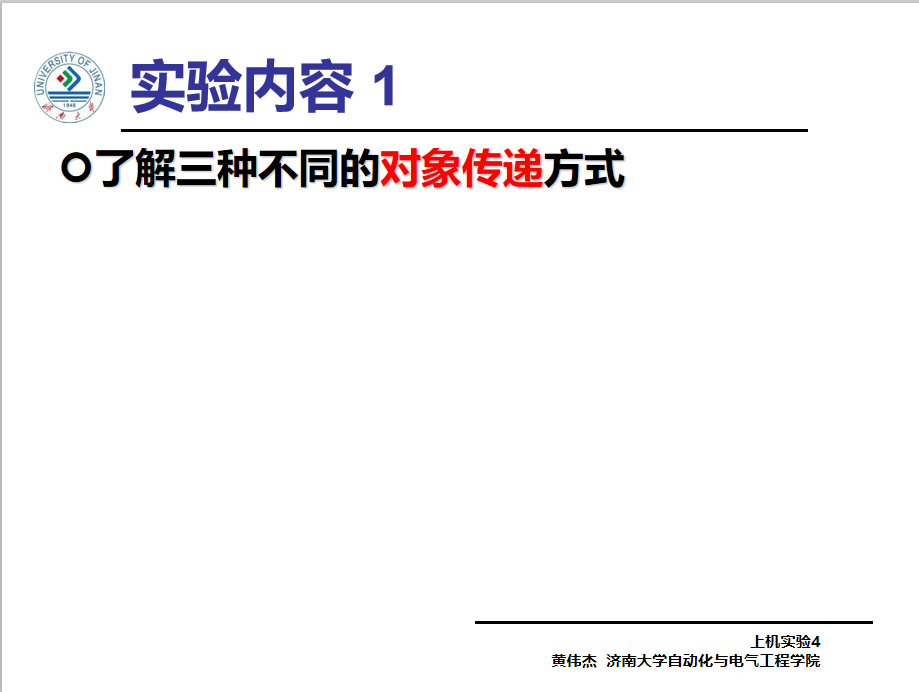
**第四次上机实验：**

**实验内容（一）：**



**代码**

#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;

}

//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;

//}

//

//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);

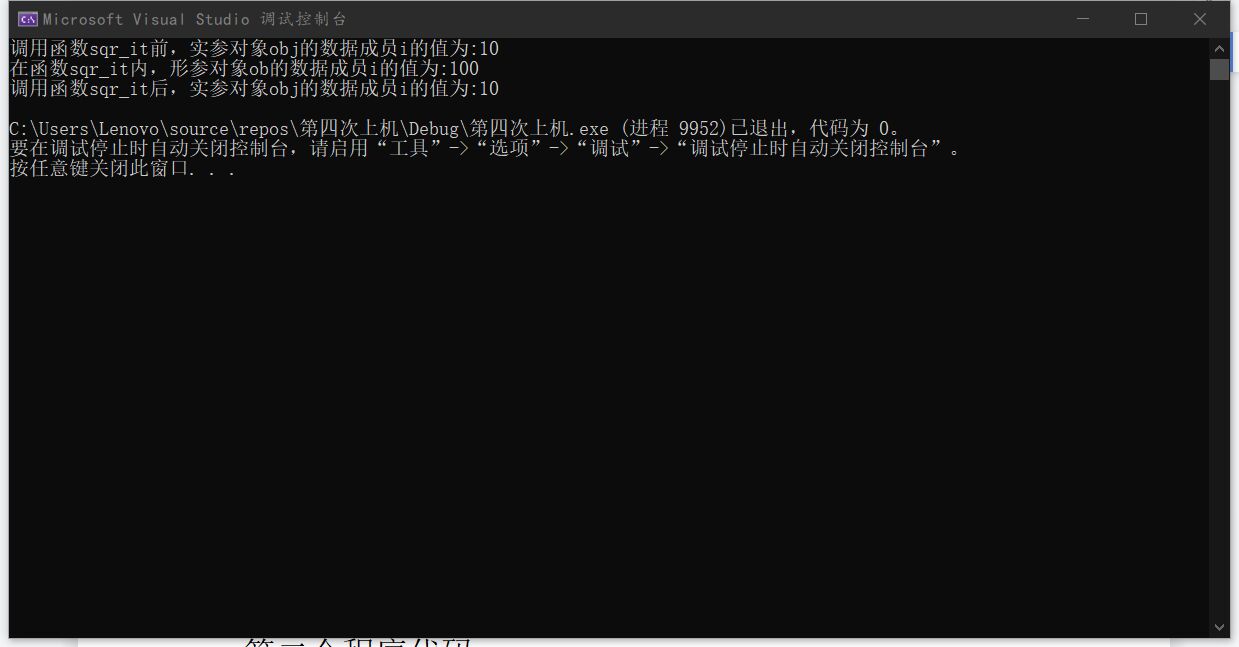
cout << "调用函数sqr\_it后，实参对象obj的数据成员i的值为:";

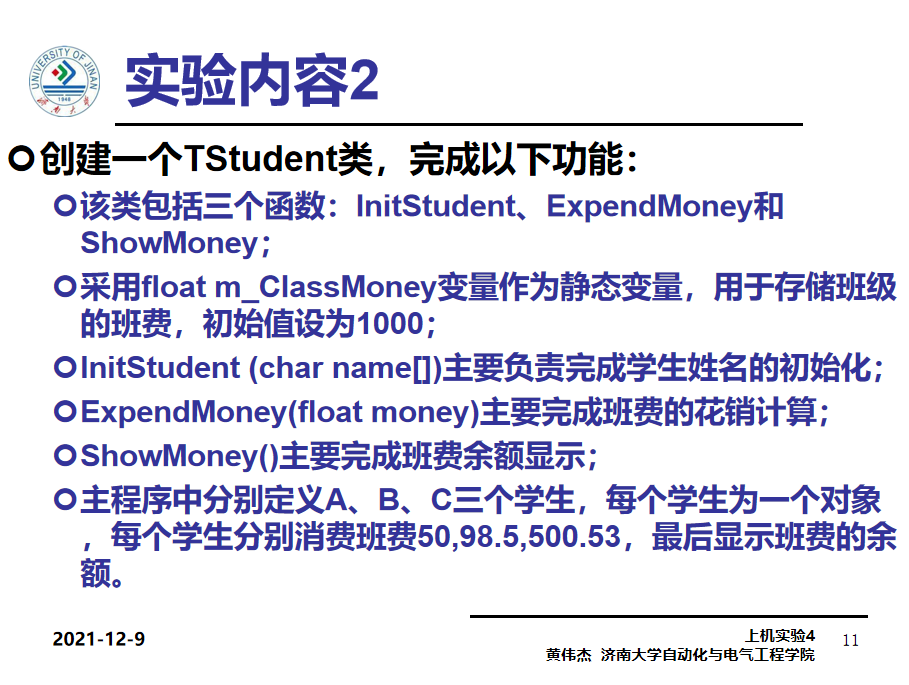
cout << obj.get\_i() << endl;

return 0;

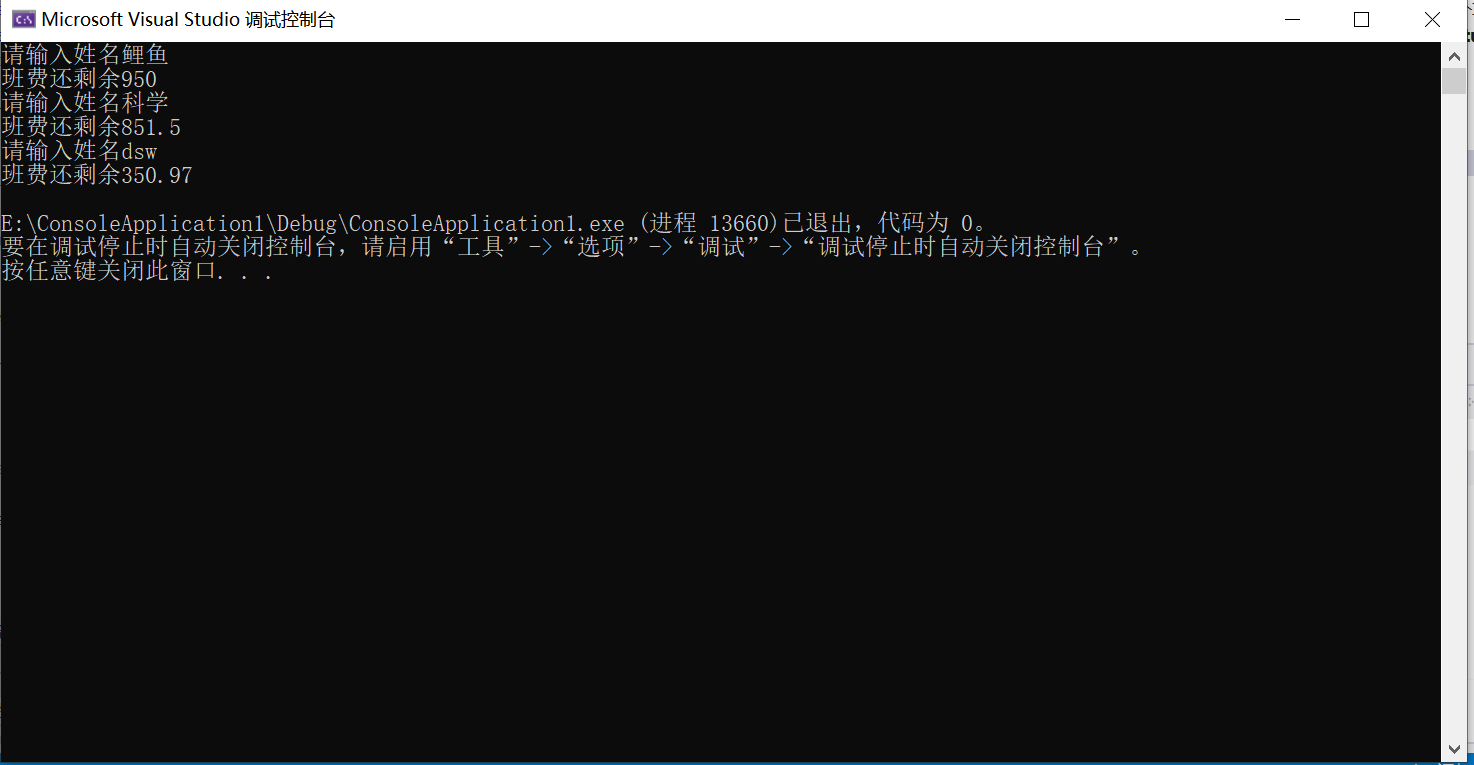
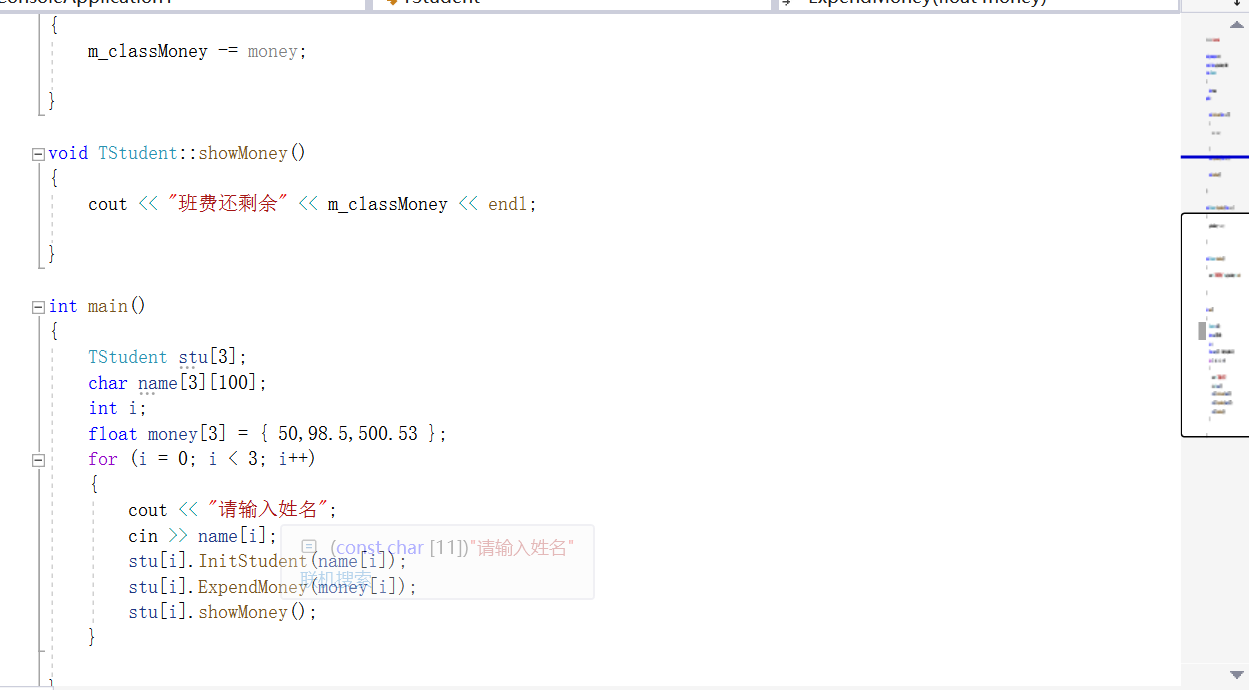
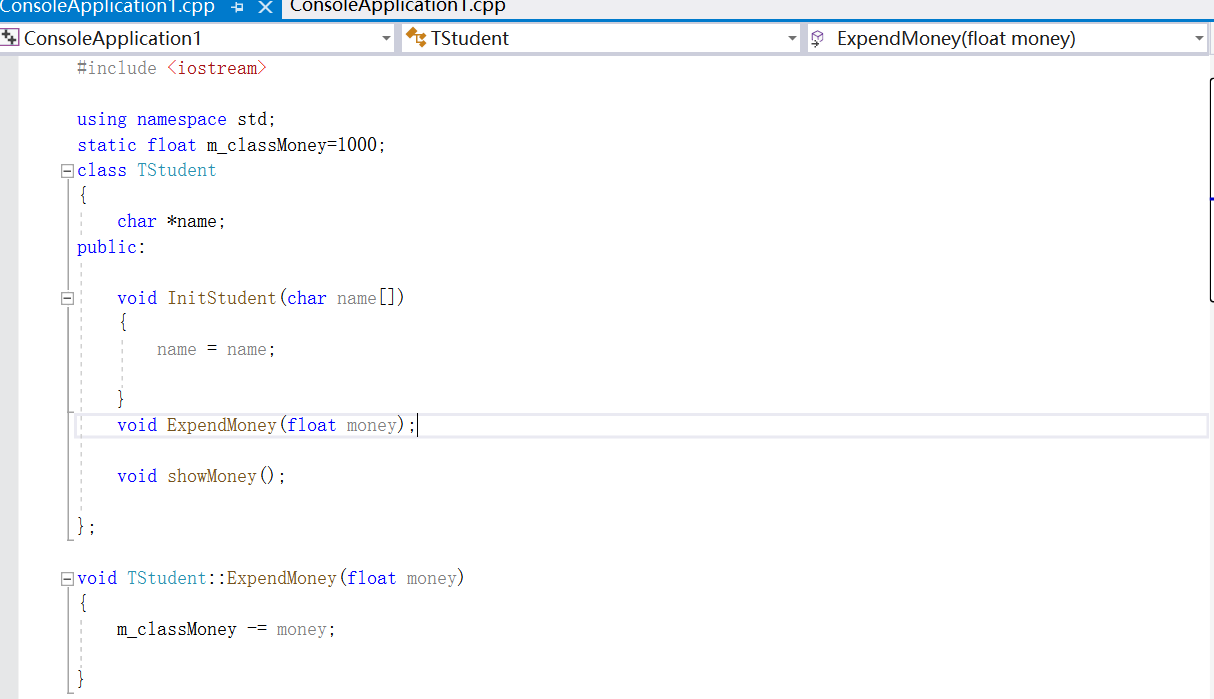
}

**结果**

**实验内容（二）：**



**代码及运行结果：**



感想心得：

（1）在进行对象传递时，可以使用对象作为函数参数，使用对象指针作为函数参数，使用对象引用作为函数参数，这三种方法，它们各有不同的特点。其中，使用对象作为函数参数时，对象的值在调用函数后不改变，使用对象指针和对象引用作为函数参数时，可以修改对象本身的值，使用时可根据不同的程序功能来选用。  
（2）在一个类中，将一个数据成员说明为 static，这种成员称为静态数据成员。无论建立多少个类的对象，都只有一个静态数据成员的拷贝。从而实现了同一个类的不同对象之间的数据共享。每次对对象变量 操作，其值都会在原有的基础上发生改变。所以能够达到跟随性的目的。  
（3）当程序运行出错后，可采用多种方法进行修改，使用时，明白其错误原因，应用最合适的方法。Copyright 2021-2099 Kongyouhao. All rights reserved