实验报告（第四次上机）

自2001 杨洺宇

实验一：

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) // 对象ob作为函数sqr\_it的形参

{

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);

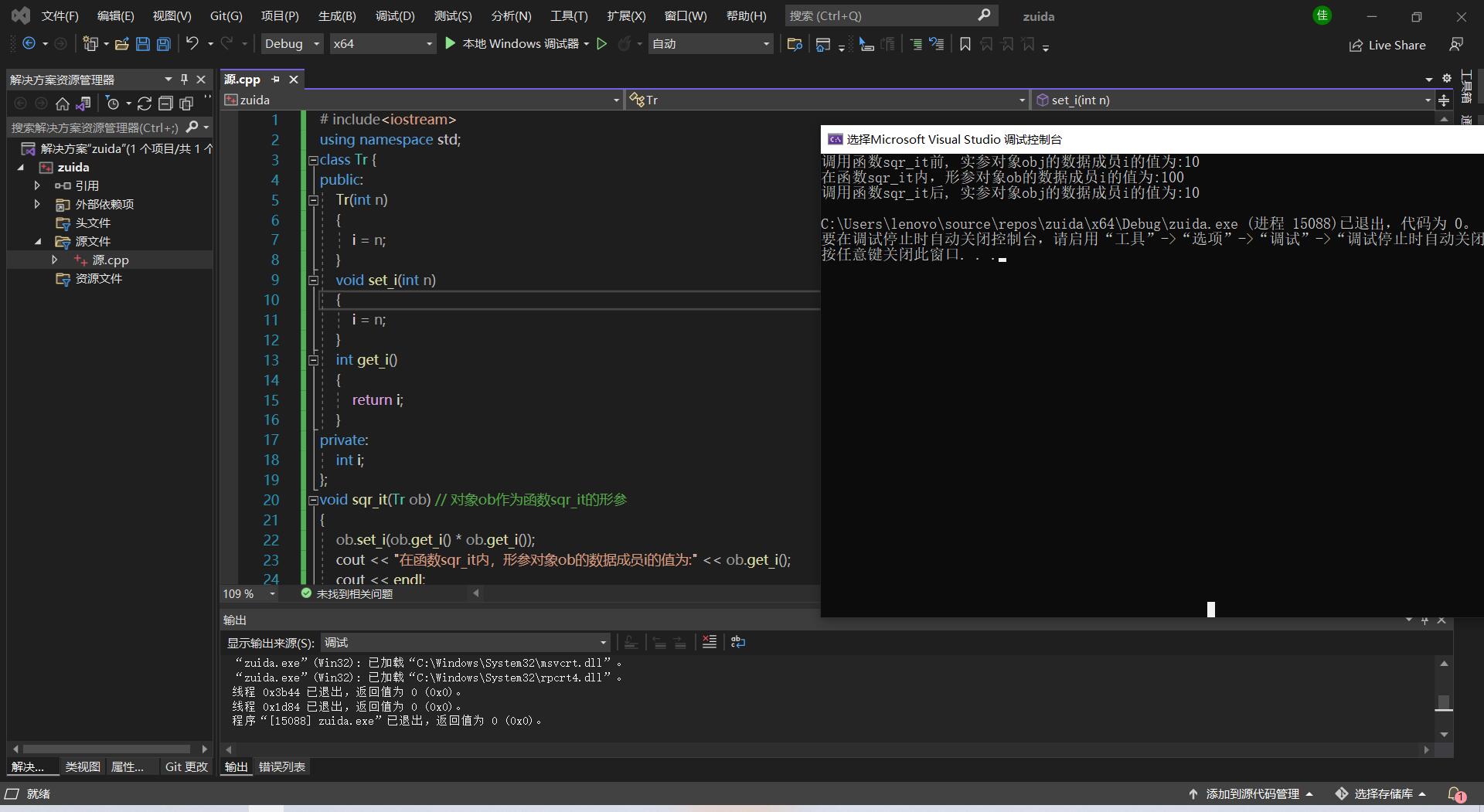
cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

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

return 0;

}

图片：



实验二：

代码：

#include<iostream>

#include<cmath>

#include<string>

using namespace std;

class tstudent {

private:

static float m\_classmoney;

string name;

public:

static void showmoney();

void initstudent(string name1);

void expendmoney(float money);

};

float tstudent::m\_classmoney = 1000;

void tstudent::initstudent(string name1)

{

name = name1;

}

void tstudent::expendmoney(float money)

{

m\_classmoney = m\_classmoney - money;

}

void tstudent::showmoney()

{

cout << "There is " << m\_classmoney << "yuan left" << endl;

}

int main()

{

int i;

float b[3];

string a[3];

tstudent s1, s2, s3;

for (i = 0; i < 3; i++)

{

cout << "Please input student's name: " << endl;

cin >> a[i];

cout << "Please input how much money did this student spend: " << endl;

cin >> b[i];

}

s1.initstudent(a[0]);

s1.expendmoney(b[0]);

s2.initstudent(a[1]);

s2.expendmoney(b[1]);

s3.initstudent(a[2]);

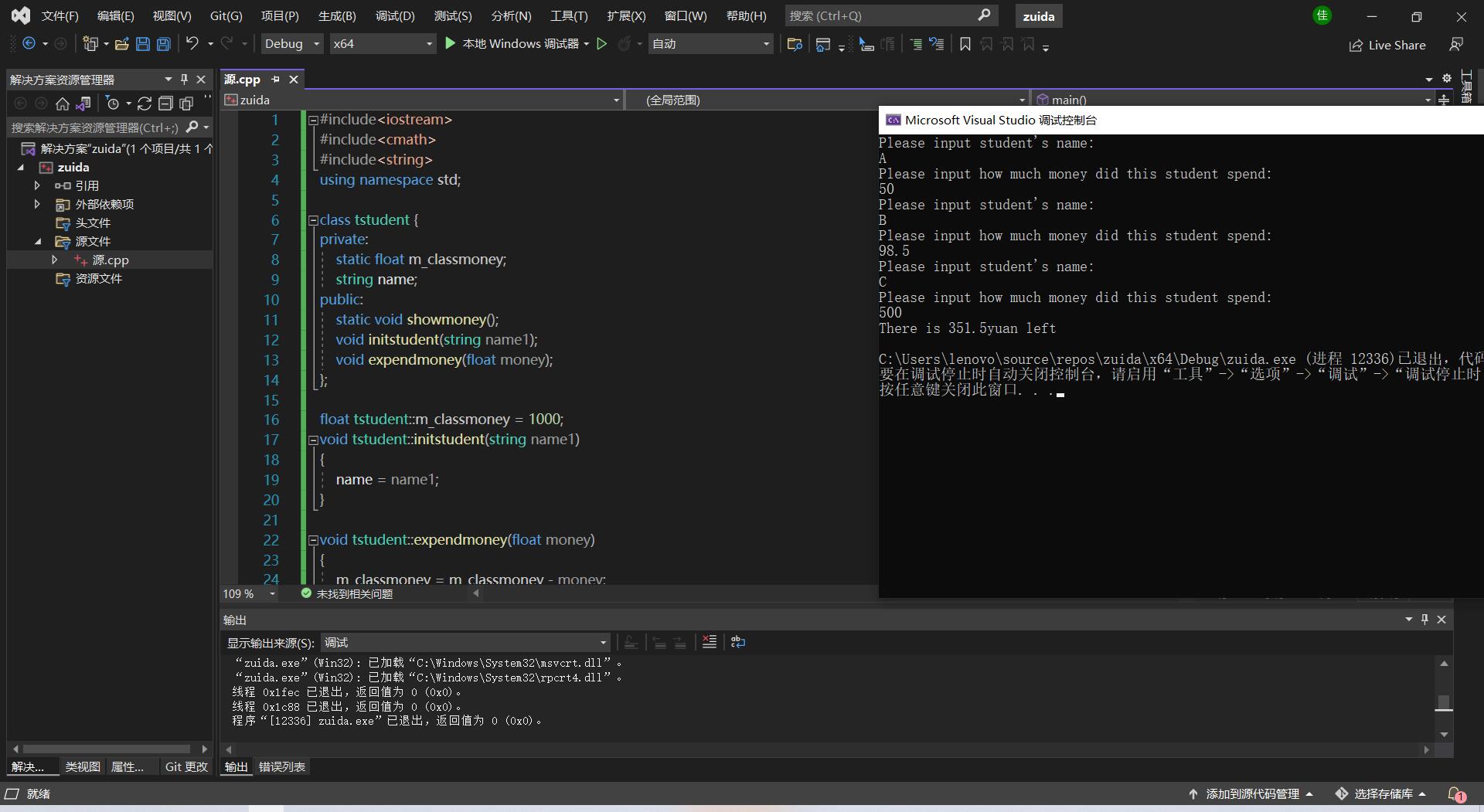
s3.expendmoney(b[2]);

s1.showmoney();

return 0;

}

图片：



总结：

本次实验我了解到

（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。