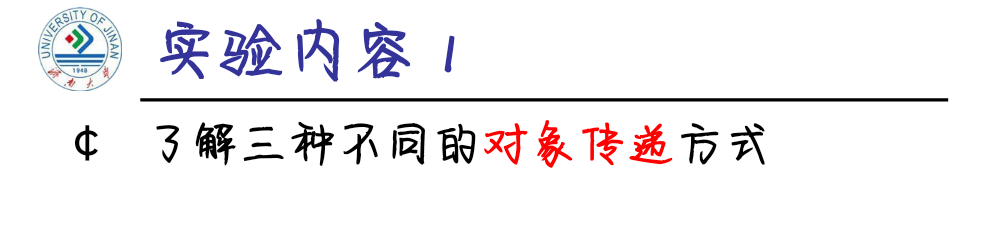
**实验内容 1**

****

**程序代码**

# 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;

}

//void sqr\_it(Tr\* ob) // 对象指针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;

//}

//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);

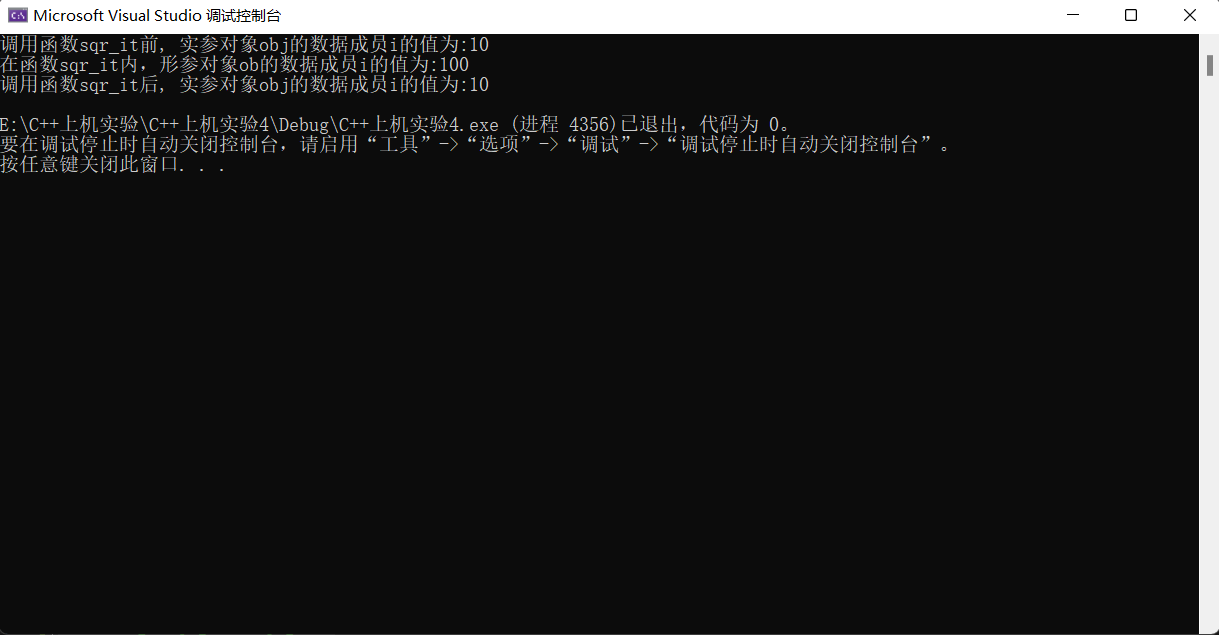
cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

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

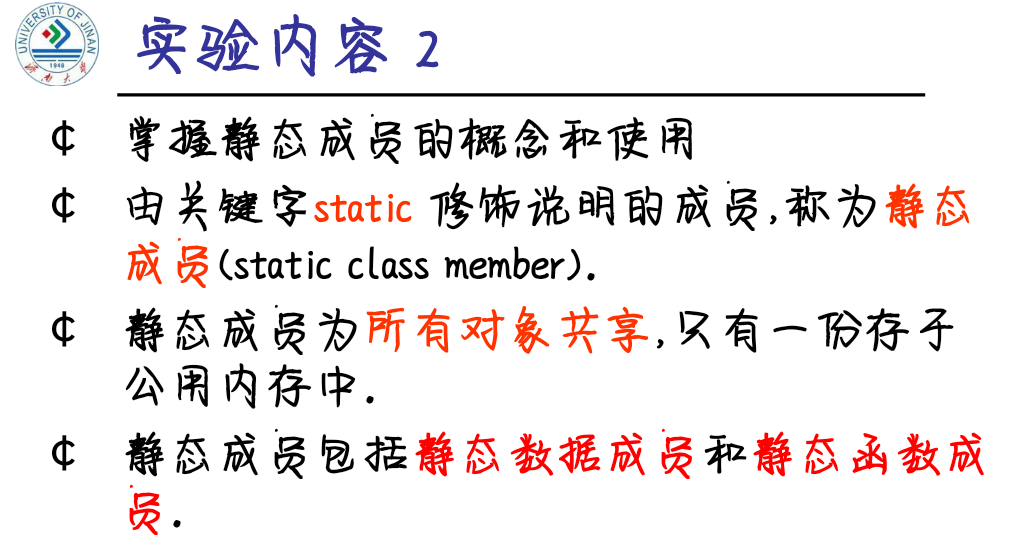
return 0;

}

**运行结果**



**实验内容 2**

****

1. **程序代码**

#include<iostream>

#include<string>

using namespace std;

class TStudent {

private:

static float m\_ClassMoney;

string Name;

public:

void InitStudent(const char name[]) {

Name = name;

}

void ExpendMoney(float money) {

m\_ClassMoney = m\_ClassMoney - money;

}

void ShowMoney() {

cout<<Name<< "：班费还剩余" << m\_ClassMoney << endl;

}

};

float TStudent::m\_ClassMoney = 1000;

int main()

{

TStudent A, B, C;

A.InitStudent("A");

A.ExpendMoney(50);

A.ShowMoney();

B.InitStudent("B");

B.ExpendMoney(98.5);

B.ShowMoney();

C.InitStudent("C");

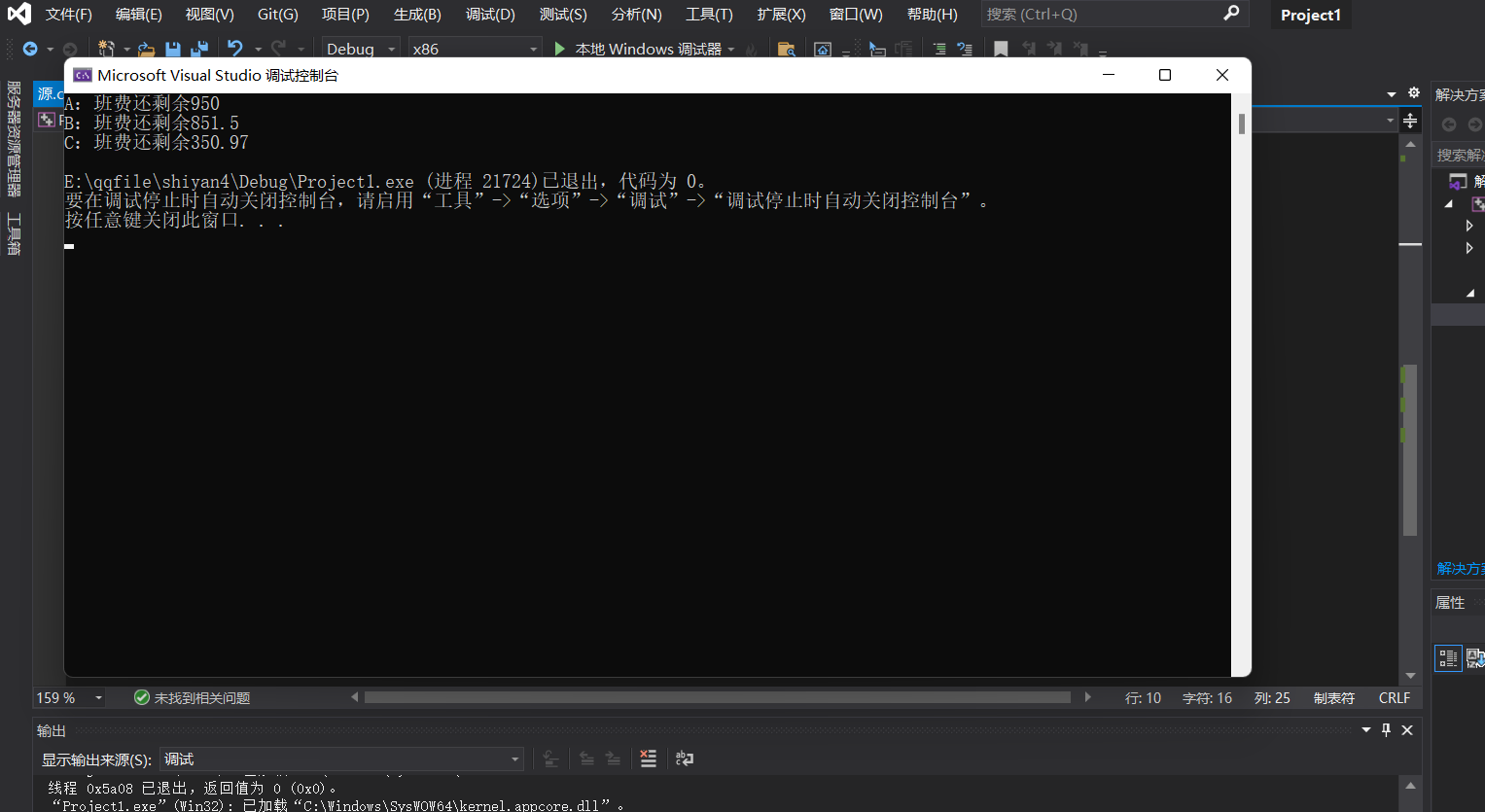
C.ExpendMoney(500.53);

C.ShowMoney();

return 0;

}

1. **运行结果**



1. **心得感想**

1.值传递:有一个形参向函数所属的栈拷贝数据的过程,如果值传递的对象是类对象或是大的结构体对象,将耗费一定的时间和空间。

2.指针传递:同样有一个形参向函数所属的栈拷贝数据的过程,但拷贝的数据是一个固定为4字节的地址。

3.引用传递:同样有上述的数据拷贝过程,但其是针对地址的,相当于为该数据所在的地址起了一个别名。

在效率上讲，指针传递和引用传递比值传递效率高。一般主张使用引用传递，代码逻辑上更加紧凑、清晰。引用传递做函数参数”是C++的特性。

Copyright ©2021-2099 BonanWu. All rights reserved