一程序代码

**实验一：**

# 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;

}

当传递对象方式为使用对象作为函数参数时，函数表示为

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;

}

当传递对象方式为使用对象指针作为函数参数时，函数表示为

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;

}

同时主函数更改为

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;

}

当传递对象方式为使用对象引用作为函数参数时，函数表示为

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;

}

主函数更改回原主函数，即

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;

}

**实验二：**

#include <iostream>

using namespace std;

class TSstudent

{

public:

double money;

static double m\_ClassMoney;

void lnitstudent(char name[])

{

}

void expendmoney(double a)

{

money = a;

}

double ShowMoney()

{

m\_ClassMoney -= money;

return m\_ClassMoney;

}

};

double TSstudent::m\_ClassMoney = 1000;

int main()

{

TSstudent A;

TSstudent B;

TSstudent C;

A.expendmoney(50);

cout << "班费还剩余" << A.ShowMoney() << endl;

B.expendmoney(98.5);

cout << "班费还剩余" << B.ShowMoney() << endl;

C.expendmoney(500.53);

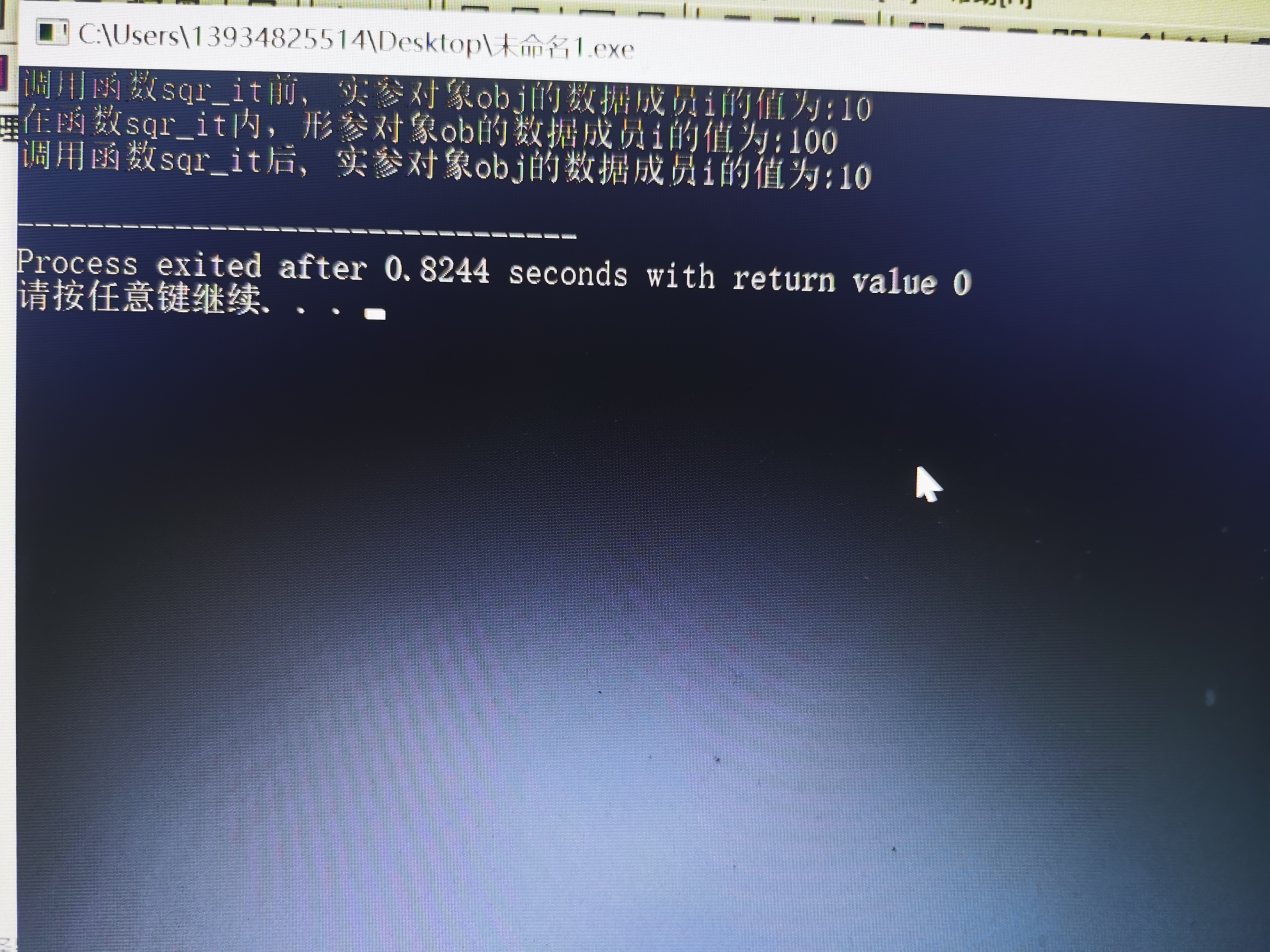
cout << "班费还剩余" << C.ShowMoney() << endl;

}

二结果

实验一：

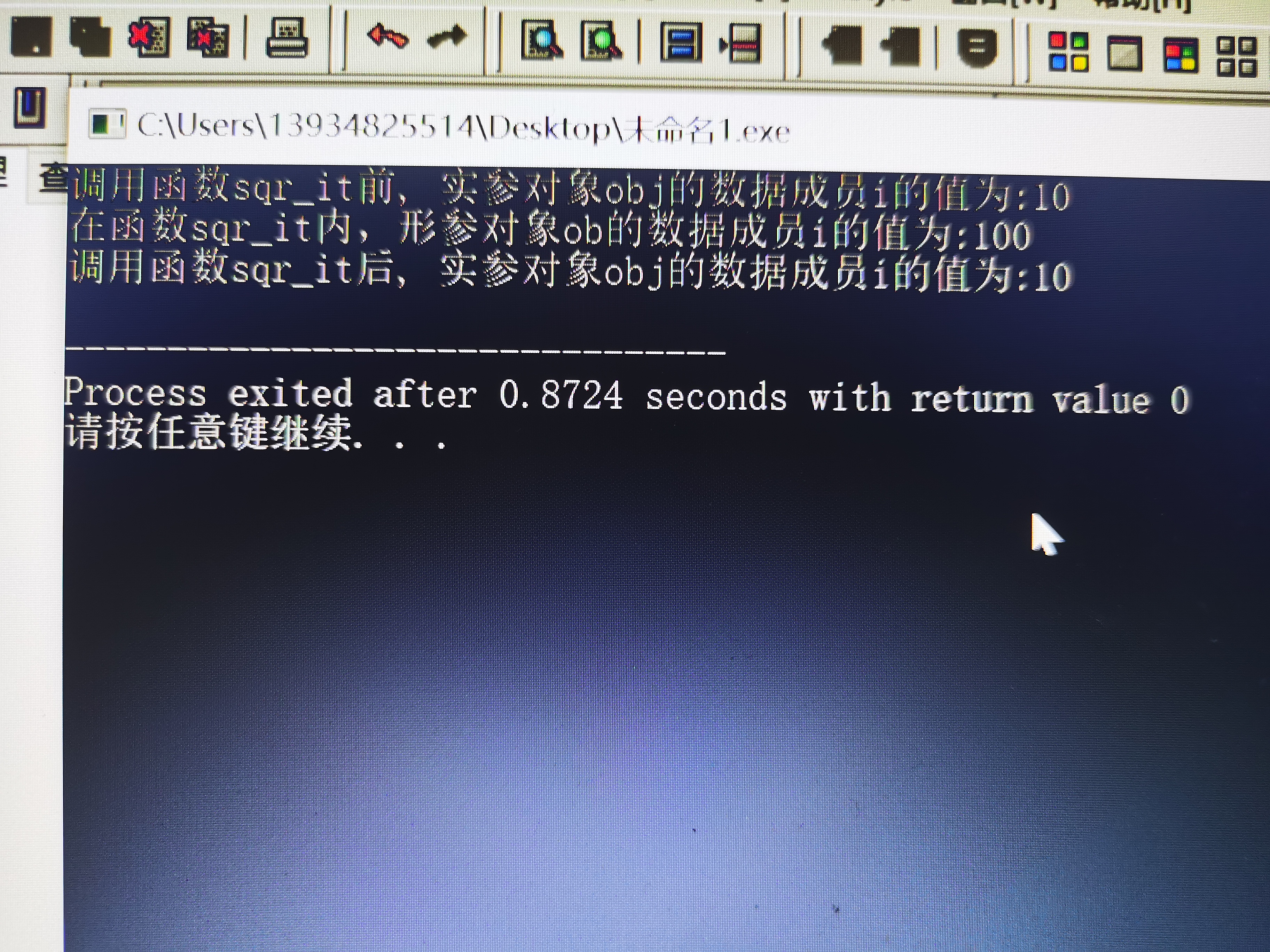
（1）对象做参数



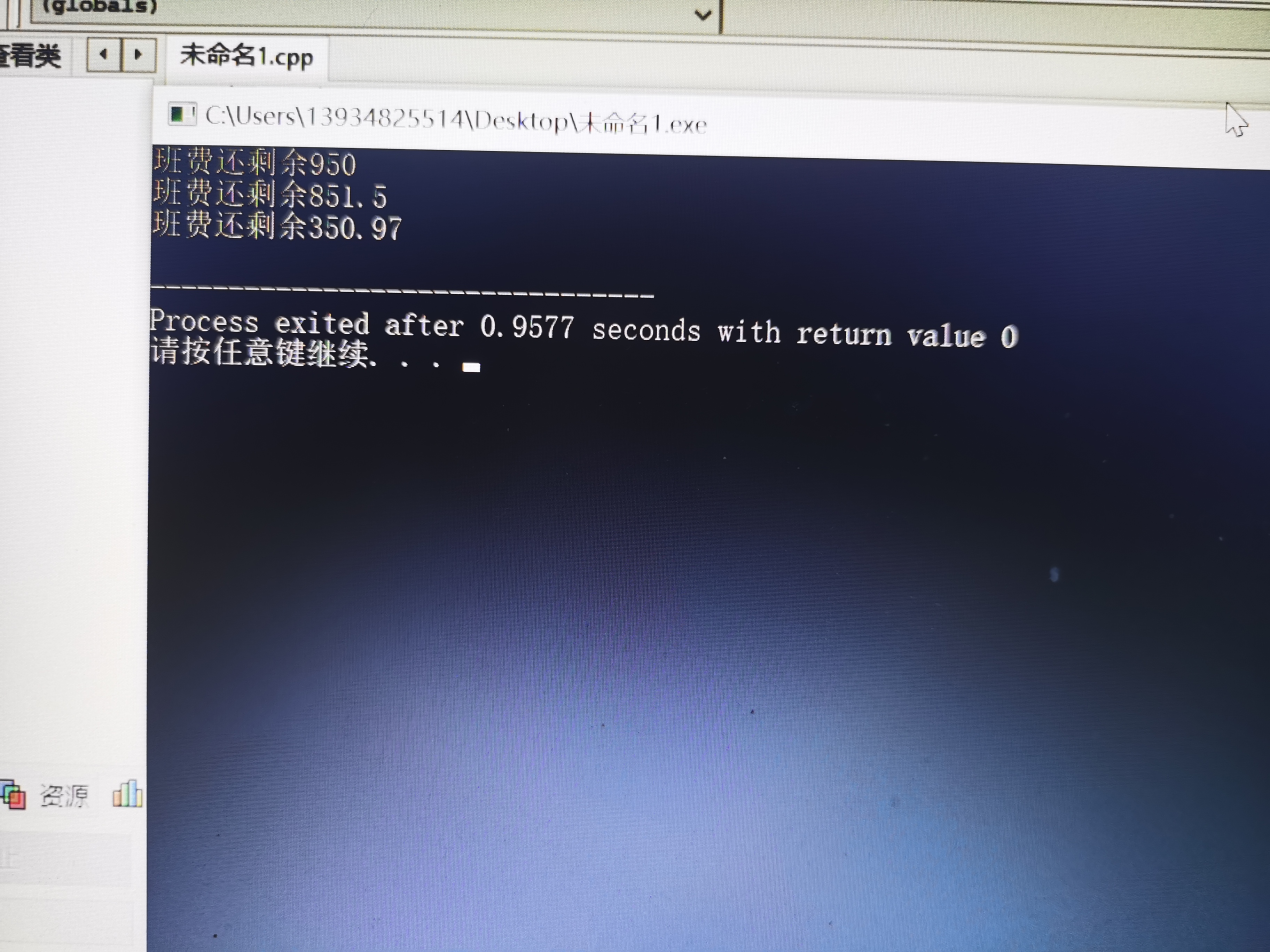
（2）对象指针做参数



（3）对象引用作参数



实验二：



三心得体会

1.向函数传递对象时，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，在传递过程中，只能由形参传递给实参，而不能有实参传递给形参，即单向传递

2.对形参对象指针所指对象值的改变也同样影响着实参对象的值，所以在实验中要多加注意

3.为了实现一个类的多个对象之间的数据共享，所以用到了静态成员，静态成员定义和普通成员一样，但要在前面加上关键字static，静态成员的初始化应该在类外单独进行，这是我在实验中没有注意到而出现的一个很大的我问题

Copyright ©2021-2099 lvlei. All rights reserved