**一、程序代码**

1.实验1

# 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;

}

2.实验2

#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);

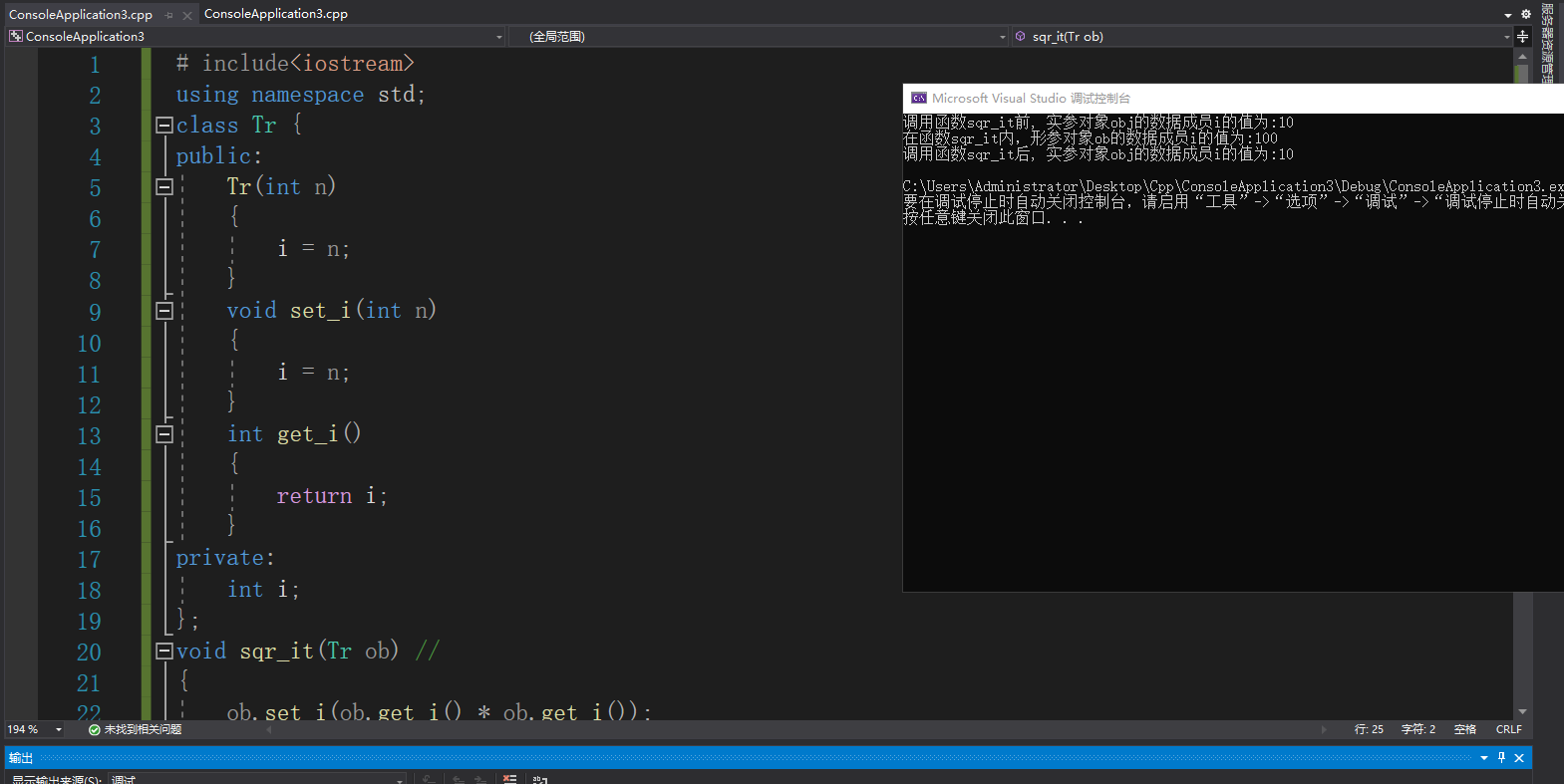
cout << "班费还剩余" << C.ShowMoney() << endl;

}

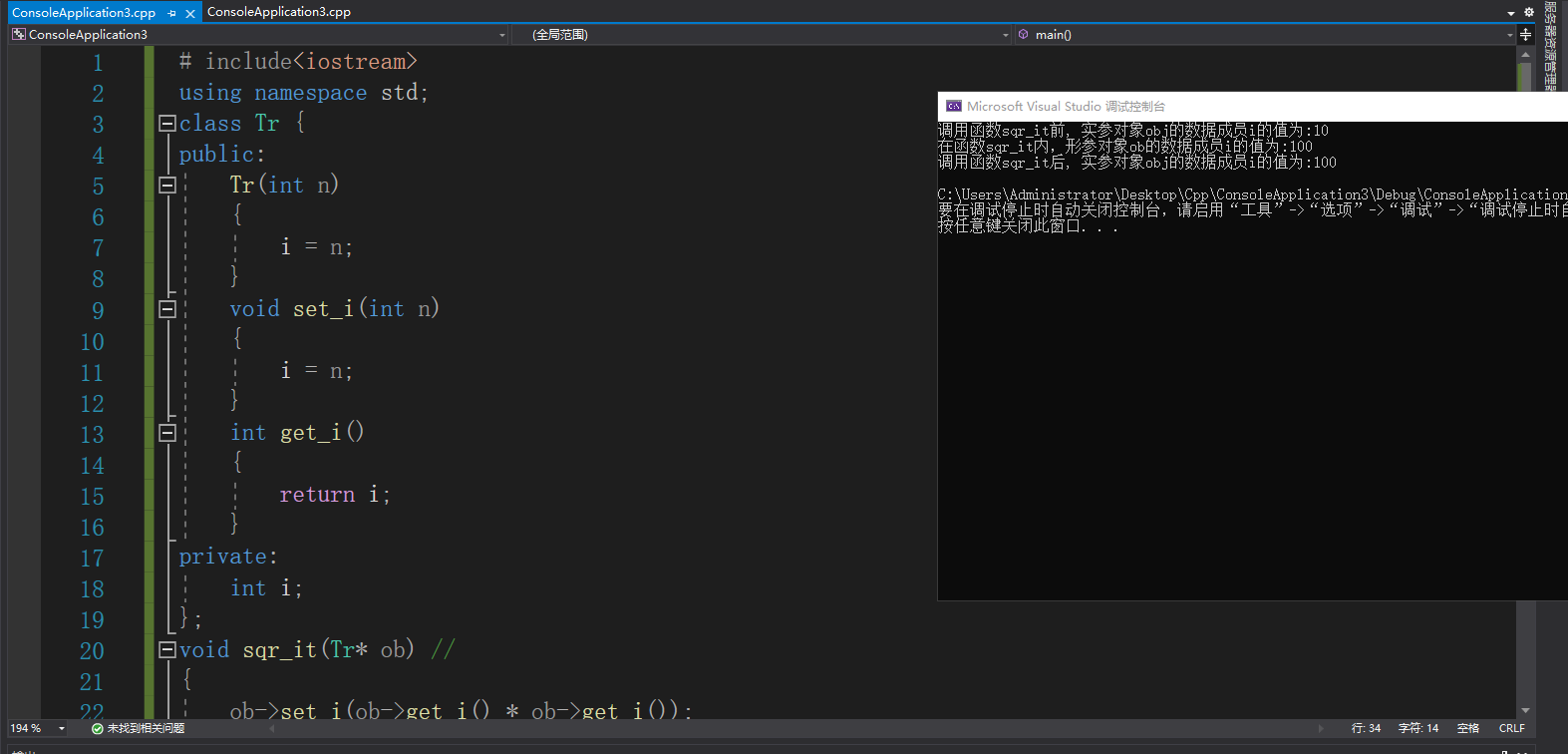
**二、实验结果**

1.实验1：

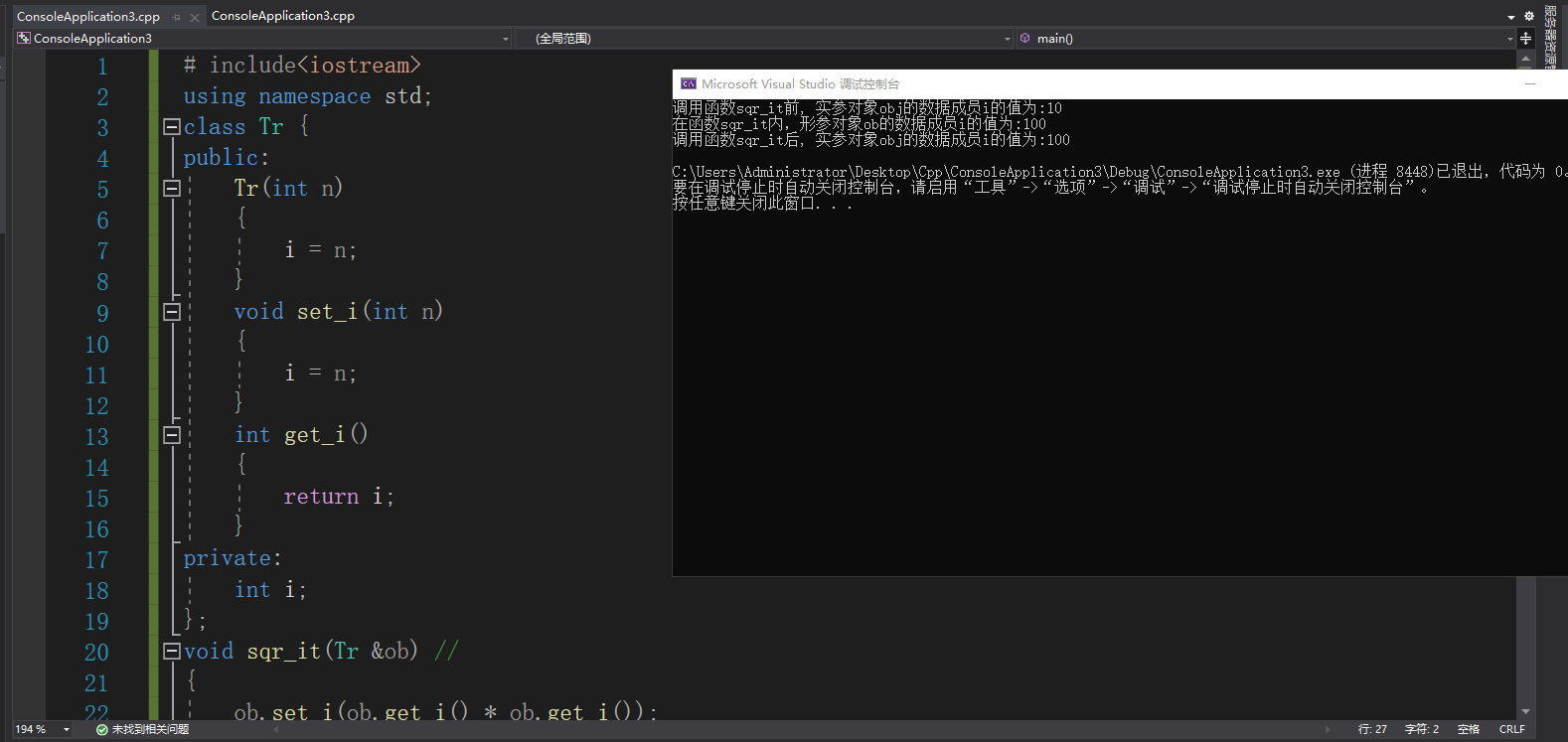
（1）对象作参数

****

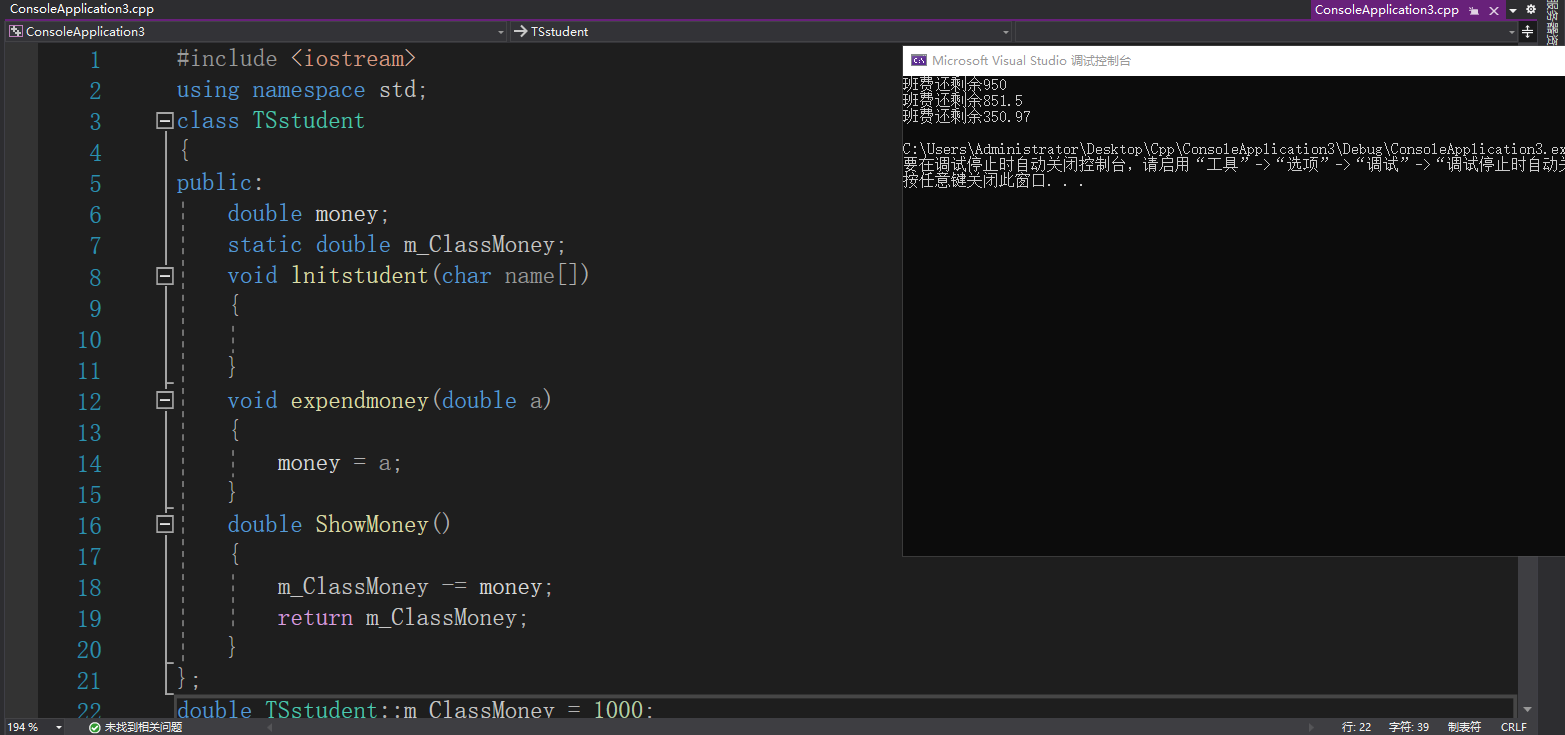
1. 对象指针作参数



1. 对象引用作参数

****

2.实验2：



**三、实验心得**

1. C++中涉及函数传递对象的问题，与C的处理过程类似，需要注意的是使用对象作为参数实现的是实参与形参之间的值传递，只能由实参传给形参，而使用对象指针、使用对象引用这两种操作差异不大，可以类比C再对应起来理解。
2. 要熟练理解->、.、&等通配符的应用，明确其含义，避免因为简单的符号错误复杂化debug过程。
3. 我认为静态变量是同一个类中各个对象的共享数据项。无论被哪一个对象访问，所得的结果都是一样的。它与全局变量仅仅作用于同一个类的多个对象之间不同，对外不可见，

它体现了类的封装性。在编写代码的过程中，我认为关键的逻辑在于明确哪一变量是要被作为静态变量进行使用的，哪一个变量要同时作用于多个对象，同时被多个变量影响。

Copyright ©2021-2099 TongmingZhang. All rights reserved