**实验内容 1**

1.了解三种不同的对象传递方式

**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;

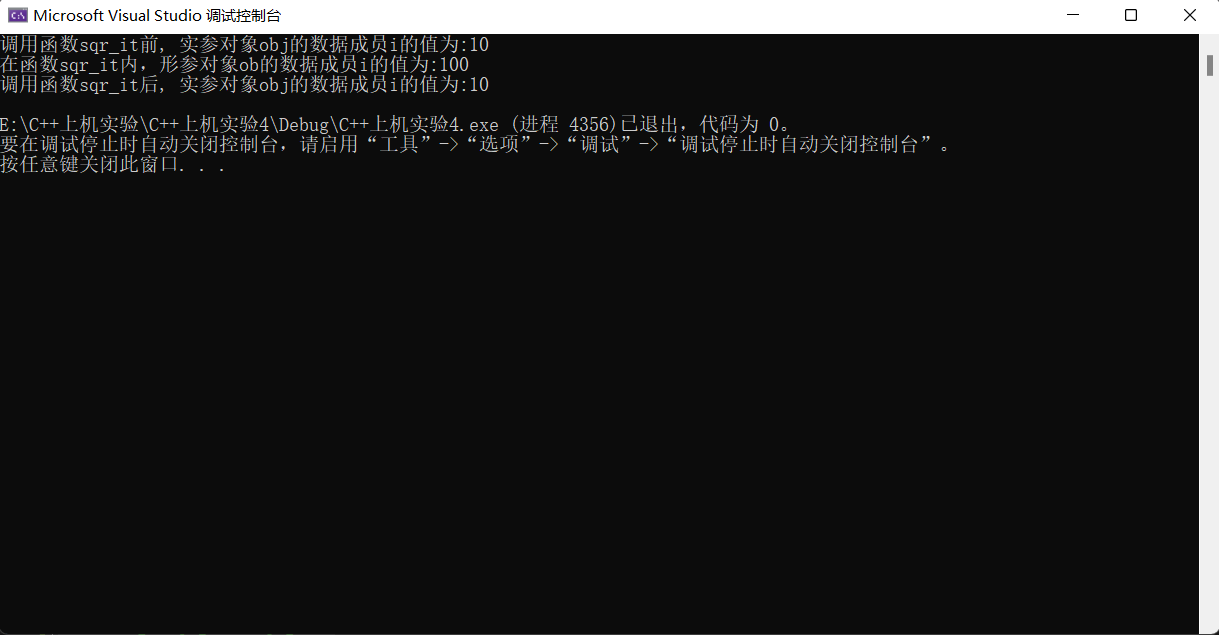
sqr\_it(obj);

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**

* **掌握静态成员的概念和使用**
* **由关键字static 修饰说明的成员，称为静态成员（static class member）。**
* **静态成员为所有对象共享，只有一份存于公用内存中。**
* **静态成员包括静态数据成员和静态函数成员。**

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");

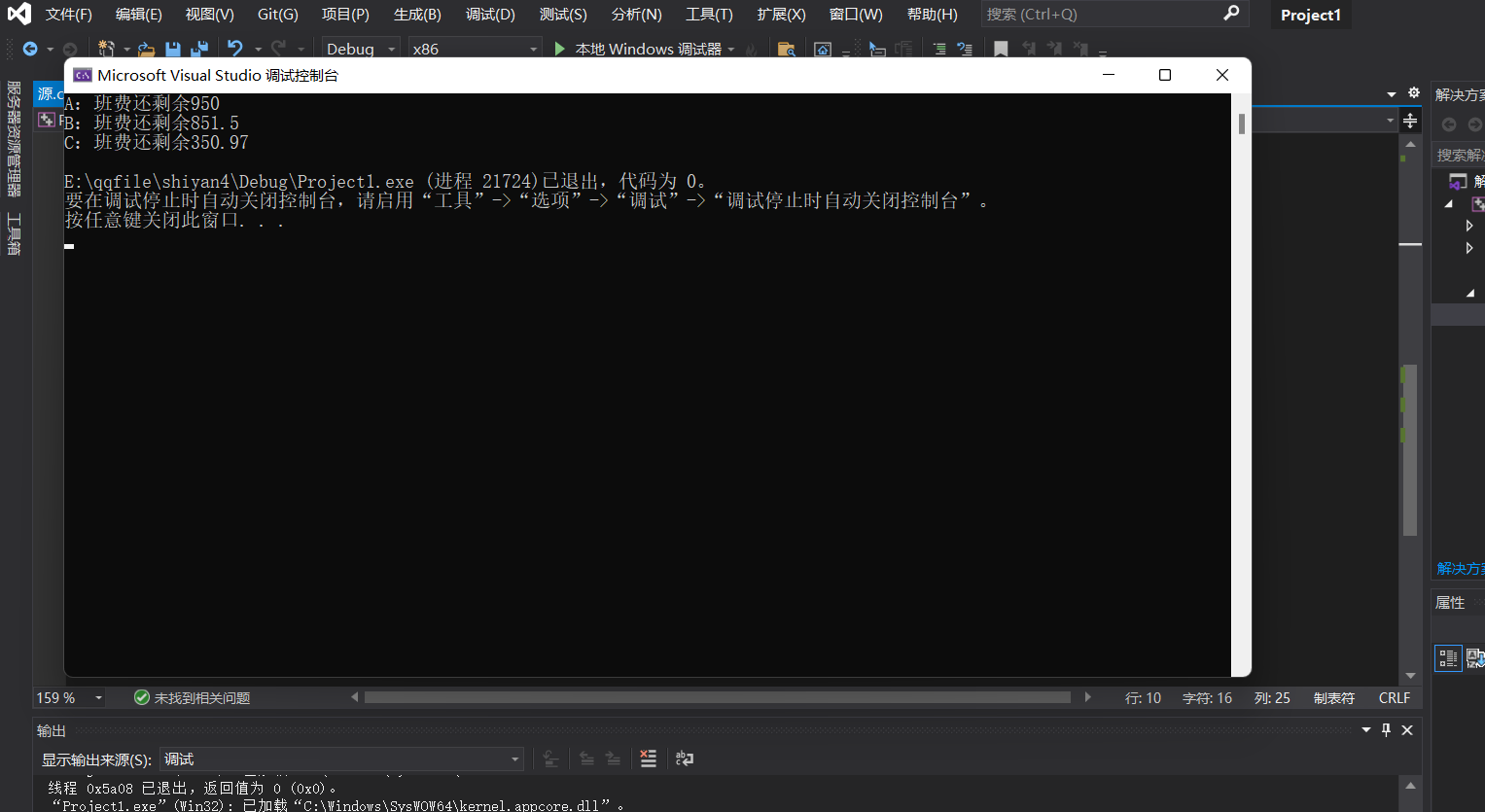
C.ExpendMoney(500.53);

C.ShowMoney();

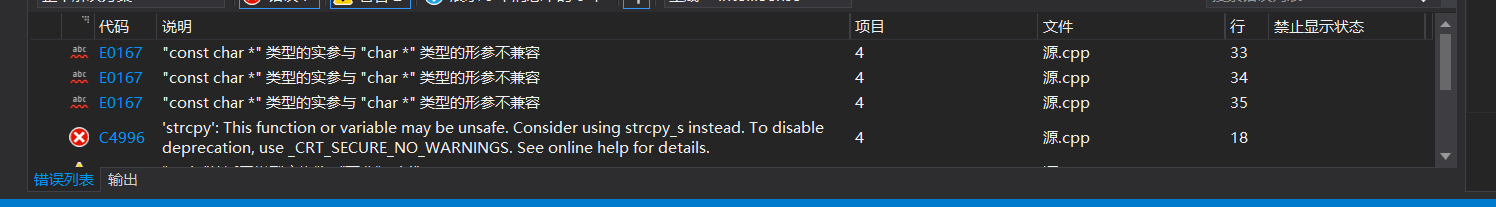
return 0;

}

1. **运行结果**



1. **心得感想**



**对名字的初始化无法完成，传参过程遇到问题，用const 解决了。++i和i++这两种形式，在不需要得到i改变前的值的情况下，尽量不要采用后置情况，会给编译器增加不必要的负担，需要分出空间来存储这个改变前的值。**

