第四次上机实验报告

**学号**：202030310001 **姓名**：陈金科 **班级**：自2002

**实验一程序代码**

#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);

//sqr\_it(&obj);// 对象指针ob作为函数sqr\_it的形参

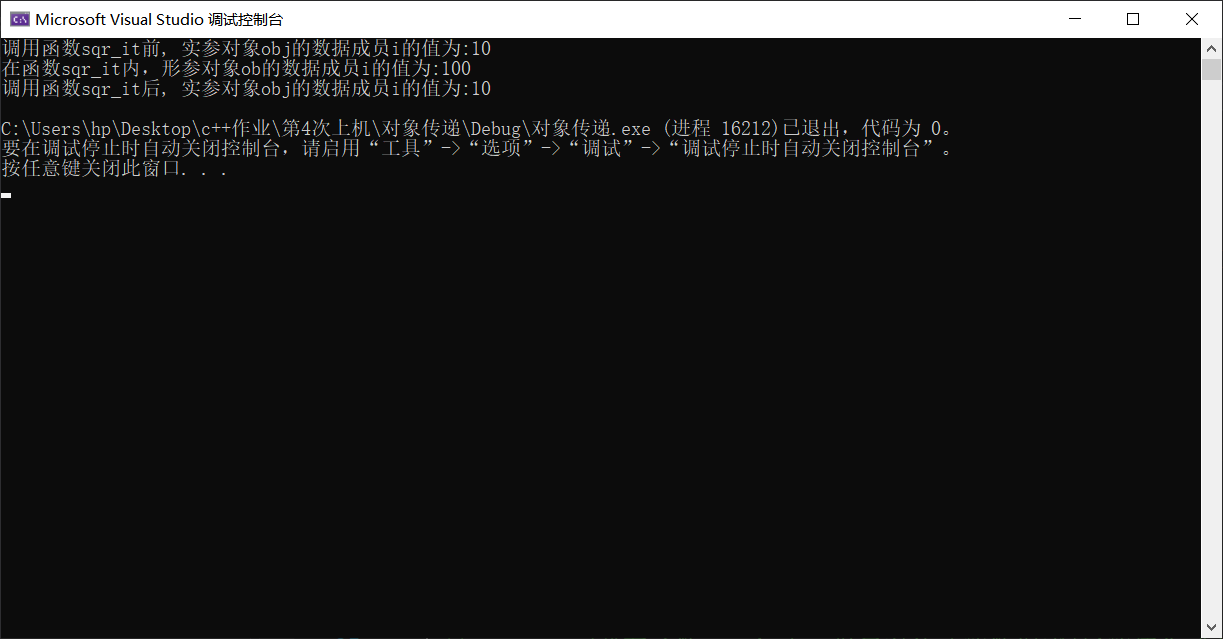
cout << "调用函数sqr\_it后, 实参对象obj的数据成员i的值为:";

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

return 0;

}

程序结果



**实验二代码**

# include<iostream>

using namespace std;

class Ctest {

static int count;

public:

Ctest() {

++count; cout << "对象数量=" << count << '\n';

}

};

int Ctest::count = 0;

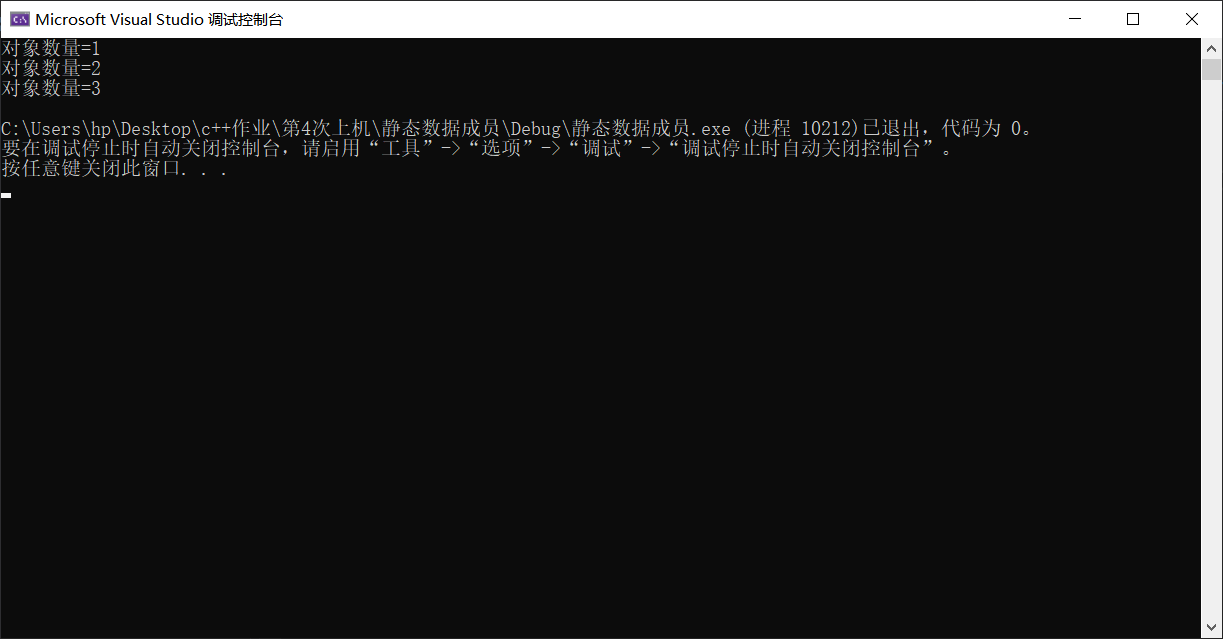
int main(void) {

Ctest a[3];

return 0;

}

程序结果



**实验三代码**

#include <iostream>

using namespace std;

class TSstudent

{

public:

float money;

static double m\_ClassMoney;

void InitStudent()

{

char name[100];

}

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;

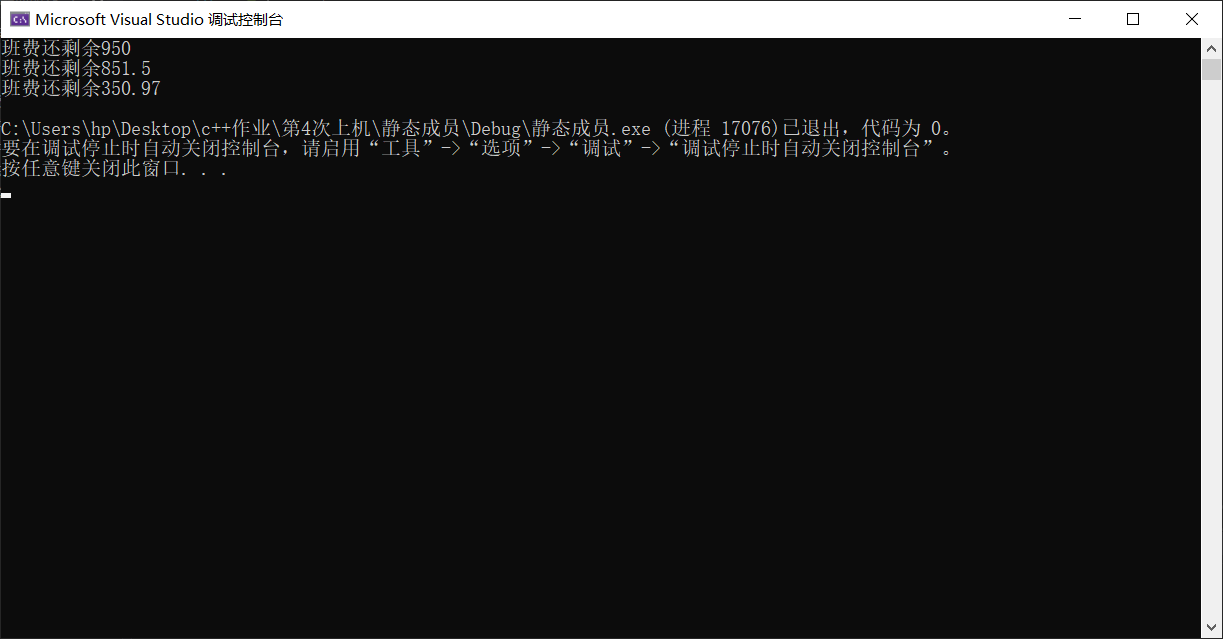
C.ExpendMoney(500.53);

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

return 0;

}

程序结果



**心得感悟**

在这次编程中，我学到了很多类和对象的知识，还有了解到了值传递，地址传递，别名传递以及函数调用的方法，类是抽象的，不占用内存。对象是具体的，占用内存。类体中的数据成员的声明前加上static关键字，该数据成员就成为了该类的静态数据成员。类似于全局变量，和其他数据成员一样，静态数据成员也遵守public/protected/private访问规则。

Copyright ©2021-2099 JinkeChen. All rights reserved