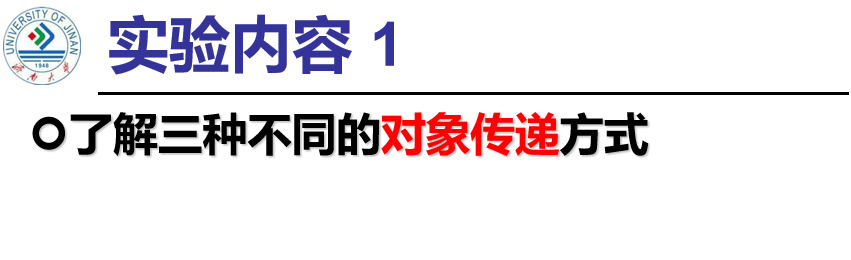
实验四 对象传递与静态成员

1：



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)

{

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;

//}

//

//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);

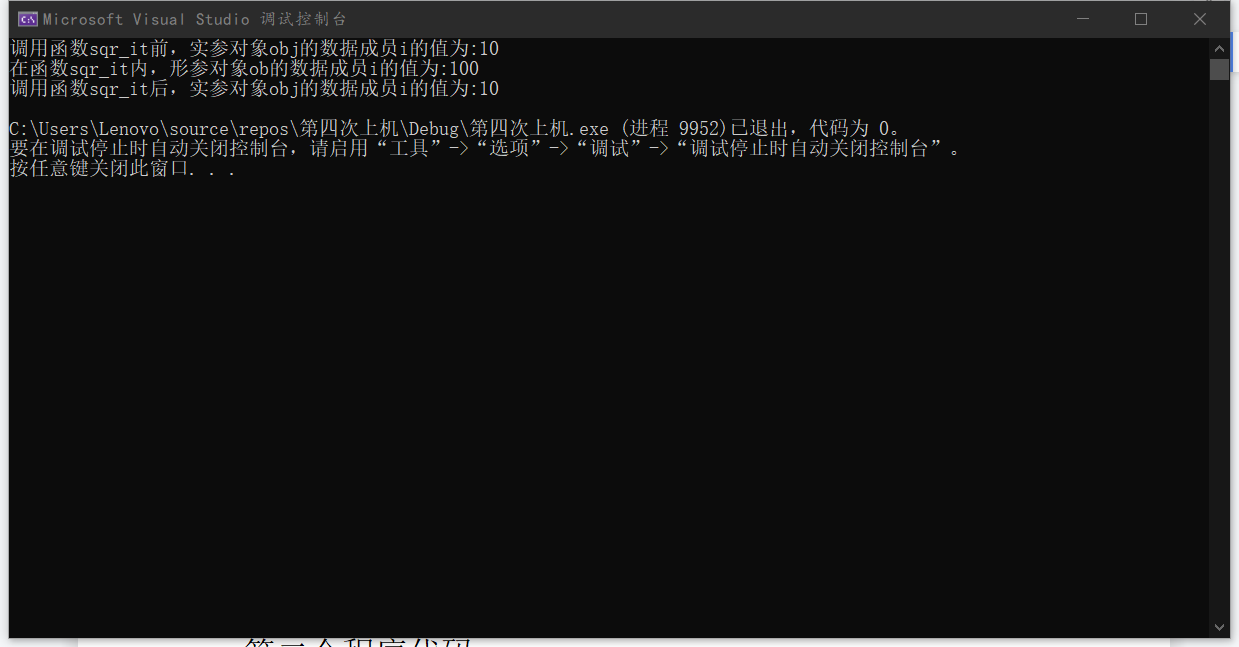
cout << "调用函数sqr\_it后，实参对象obj的数据成员i的值为:";

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

return 0;

}

结果：



2：



代码：

#include<iostream>

using namespace std;

class Tstudent {

public:

void InitStudent(char name[])

{

//strcpy(m\_name, name);

}

void ShowMoney()

{

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

}

void ExpendMoney(float money)

{

m\_ClassMoney = m\_ClassMoney - money;

}

private:

char m\_name[6];

static float m\_ClassMoney;

};

float Tstudent::m\_ClassMoney = 1000;

int main()

{

Tstudent A, B, C;

A.ExpendMoney(50);

A.ShowMoney();

B.ExpendMoney(98.5);

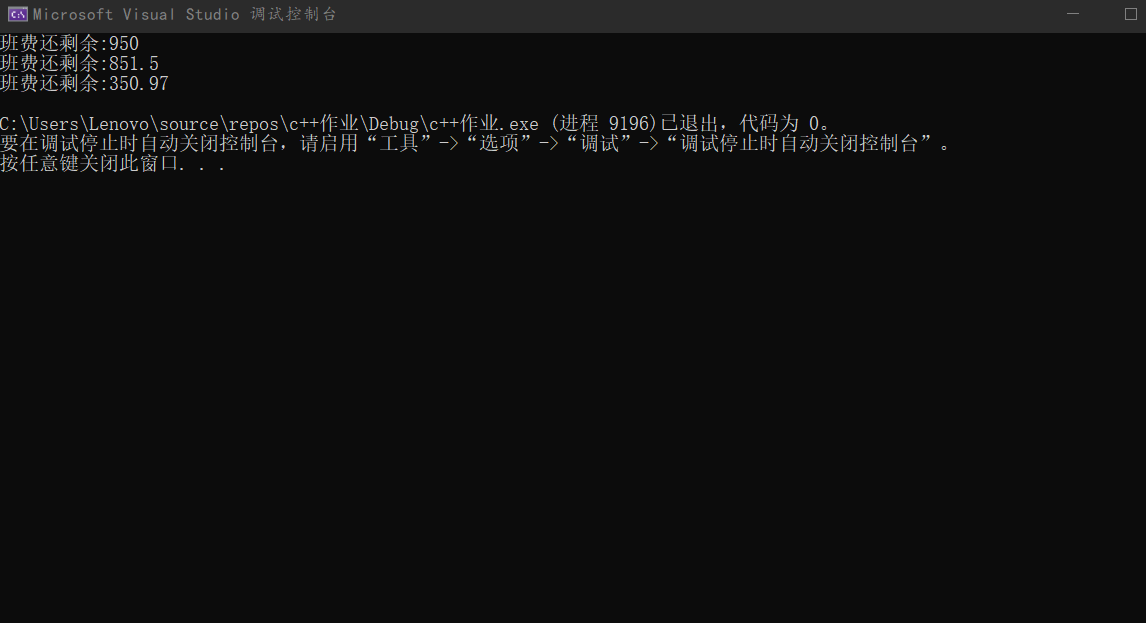
B.ShowMoney();

C.ExpendMoney(500.53);

C.ShowMoney();

}

结果：



心得：

针对不同的函数调用模式，要有相对应的函数调用模式，普通的成员函数调用、指针函数调用、对象函数调用方式。静态成员实现一个类的多个对象之间的数据共享，静态成员包括静态数据成员和静态函数成员。

Copyright ©2021-2099 MingTao Li. All rights reserved