源代码：

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qr\_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qr\_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qr\_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qr\_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qr\_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qr\_it后，形参对象obj的数据成员i的值为:";

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

return 0;

return 0;

}

2、

#include<iostream>

using namespace std;

class TStudent

{

private:

static float m\_ClassMoney;

char name[100];

float money;

public:

void InitStudent()

{

cout << "输入学生姓名：";

cin >> name;

cout << "输入学生花费：";

cin >> money;

}

void ExpendMoney()

{

m\_ClassMoney = m\_ClassMoney - money;

}

void ShowMoney()

{

cout << "剩余的钱：" << m\_ClassMoney << endl;

}

};

float TStudent::m\_ClassMoney = 1000;

int main()

{

TStudent A, B, C;

A.InitStudent();

A.ExpendMoney();

A.ShowMoney();

B.InitStudent();

B.ExpendMoney();

B.ShowMoney();

C.InitStudent();

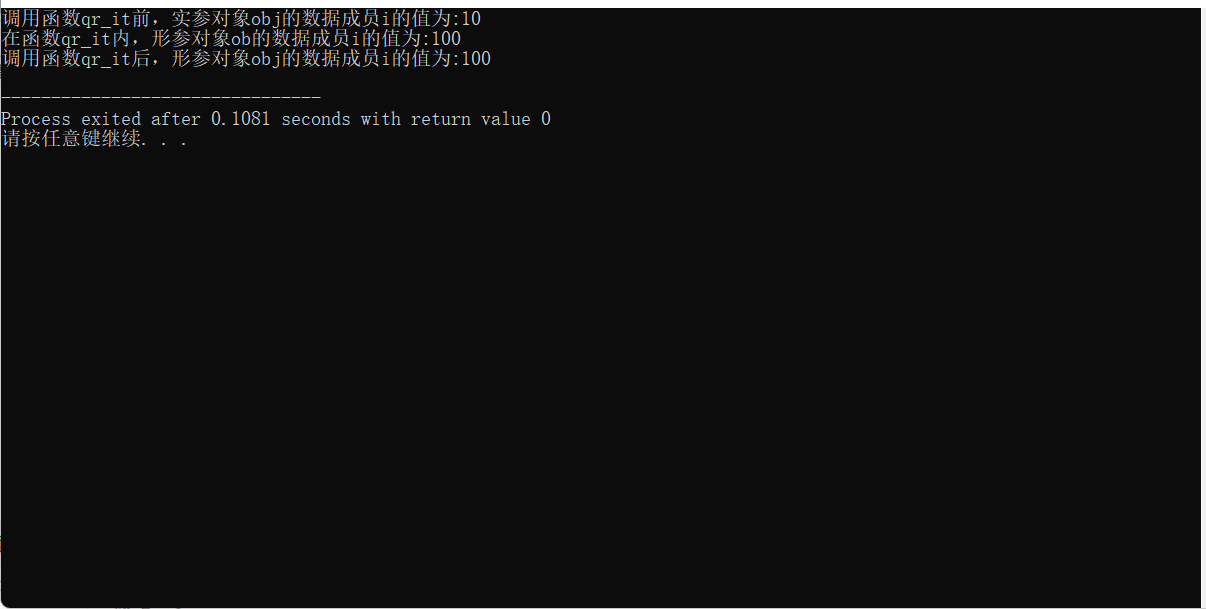
C.ExpendMoney();

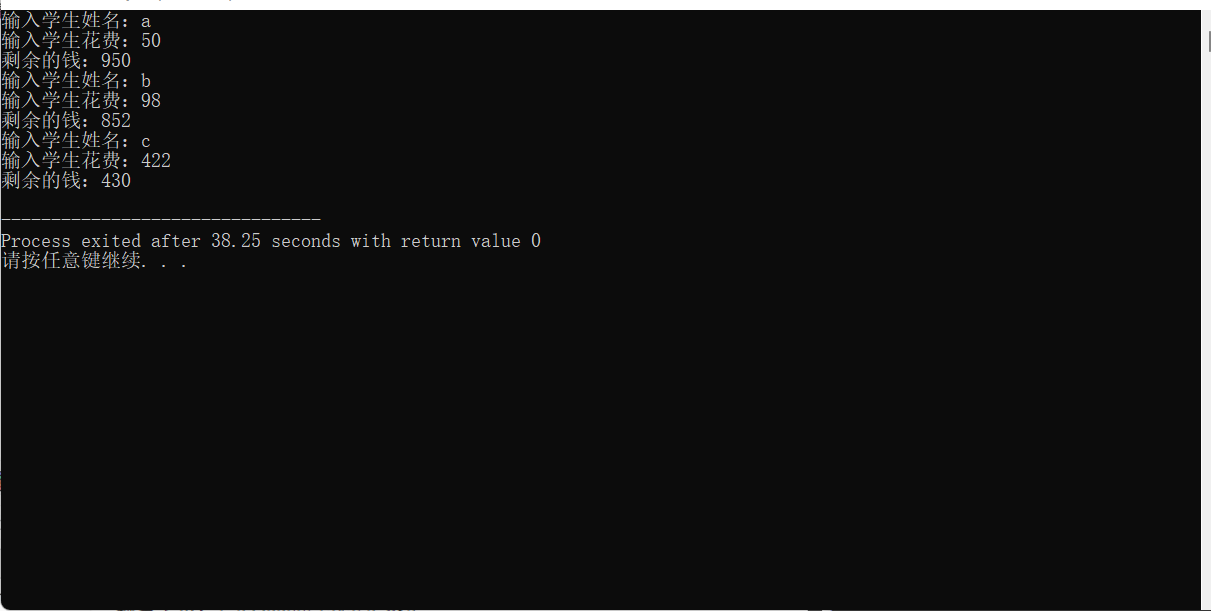
C.ShowMoney();

return 0;

}

运行结果：





心得体会：通过本次实验，深入学习了静态成员的相关概念和使用方式，增强了我的编程能力。值传递，就是把你的变量的值传递给函数的形式参数，实际就是用变量的值来新生成一个形式参数，因而在函数里对形参的改变不会影响到函数外的变量的值。地址传递，就是把变量的地址赋给函数里形式参数的指针，使指针指向真实的变量的地址，因为对指针所指地址的内容的改变能反映到函数外，能改变函数外的变量的值。引用传递(C++中特有，C语言没有)，实际是通过指针来实现的，能达到使用的效果如传址，可是使用方式如传值。

Copyright ©2021-2099 Wanghaohan.All rights reserved