上机实验4：了解三种不同的对象传递方式；

1. 将对象ob作为函数的形参

# 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;}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;}

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ob的数据成员i的值为：";

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

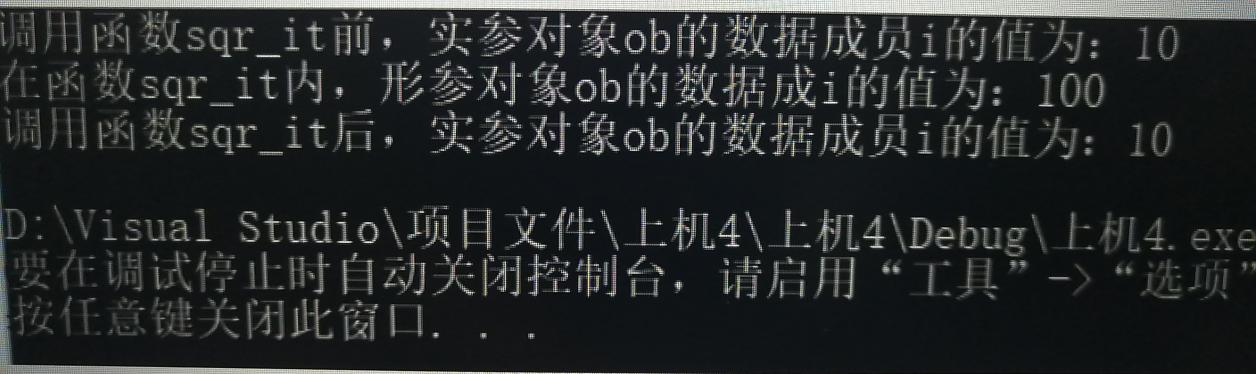
sqr\_it(obj);

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

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

return 0;

}



1. 将对象指针ob作为函数的形参

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ob的数据成员i的值为：";

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

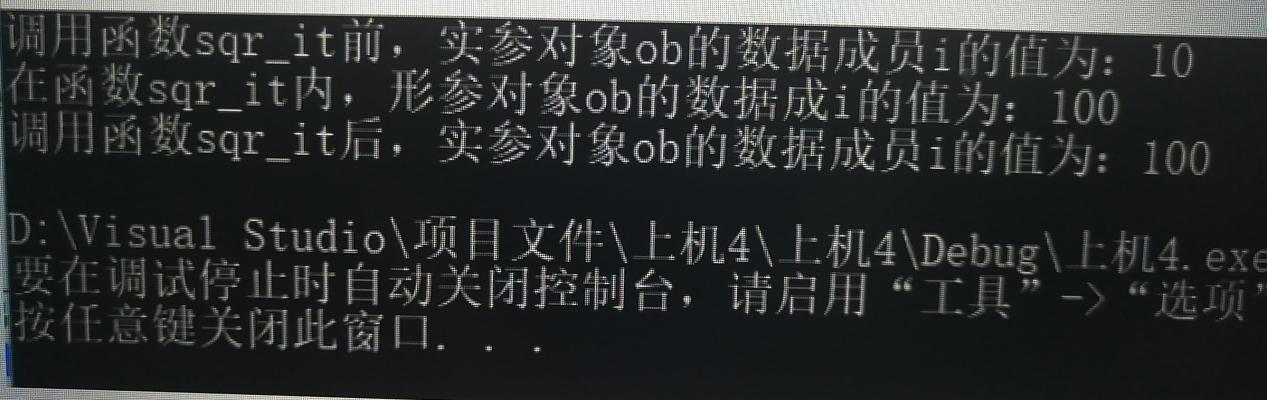
sqr\_it(&obj);

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

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

return 0;

}



1. 将对象地址作为形参

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ob的数据成员i的值为：";

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

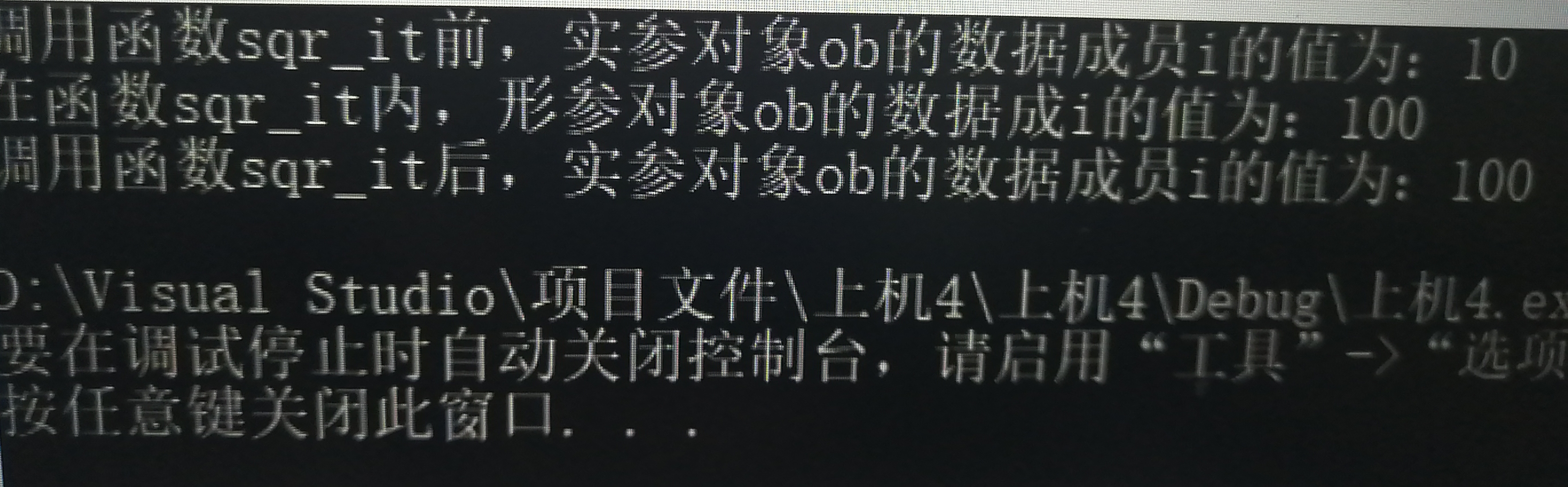
sqr\_it(obj);

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

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

return 0;

}



实验2

#include <iostream>

#include <string.h>

using namespace std;

class TStudent

{

private:

char m\_Name[6];

static float m\_ClassMoney;

public:

void InitStudent(char \*);

void ExpendMoney(float);

static void ShowMoney();

};

float TStudent::m\_ClassMoney = 1000;

void TStudent::InitStudent(char name[])

{

strcpy(m\_Name, name);

}

void TStudent::ExpendMoney(float money)

{

m\_ClassMoney -= money;

cout << m\_Name << "花费班费" << m\_ClassMoney << endl;

}

void TStudent::ShowMoney()

{

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

}

void main()

{

TStudent stu[3];

stu[0].InitStudent("A");

stu[1].InitStudent("B");

stu[2].InitStudent("C");

stu[0].ExpendMoney(50);

TStudent::ShowMoney();

stu[1].ExpendMoney(98.5);

TStudent::ShowMoney();

stu[2].ExpendMoney(500.53);

TStudent::ShowMoney();

}

通过这次上机实验 我深刻学习到了对象传递与静态成员 我学习到了静态成员的声明为static 对静态成员十分透彻 尤其是实验2 明白可静态成员的数值不会和动态一样 用完就会改变数值