## 程序代码

**实验内容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;

}

当传递对象方式为使用对象作为函数参数时，函数表示为

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;

}

当传递对象方式为使用对象指针作为函数参数时，函数表示为

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;

}

同时主函数更改为

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;

}

当传递对象方式为使用对象引用作为函数参数时，函数表示为

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;

}

主函数更改回原主函数，即

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;

}

**实验内容2：**

#include <iostream>

using namespace std;

class TSstudent

{

public:

double money;

static double m\_ClassMoney;

void lnitstudent(char name[])

{

}

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);

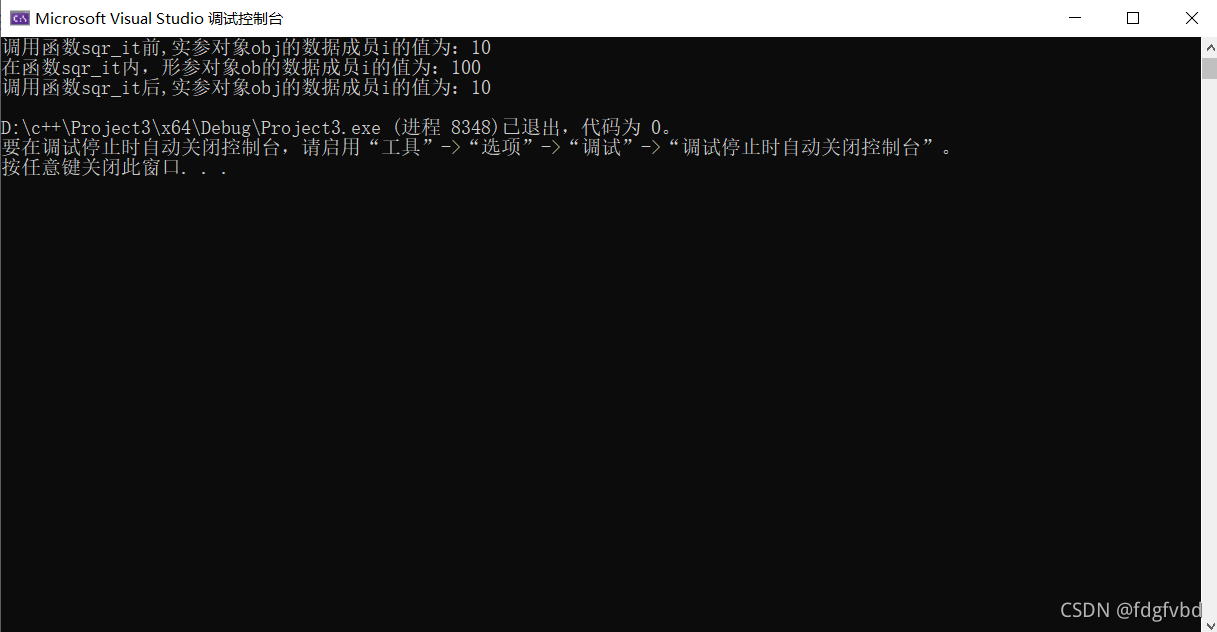
cout << "班费还剩余" << C.ShowMoney() << endl;

}

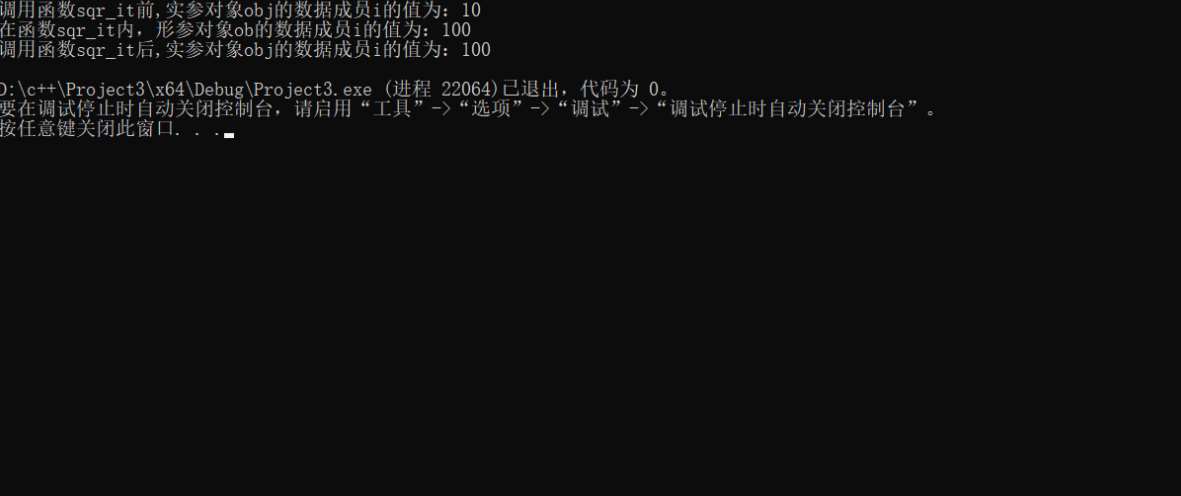
## 程序结果

**实验内容1：**

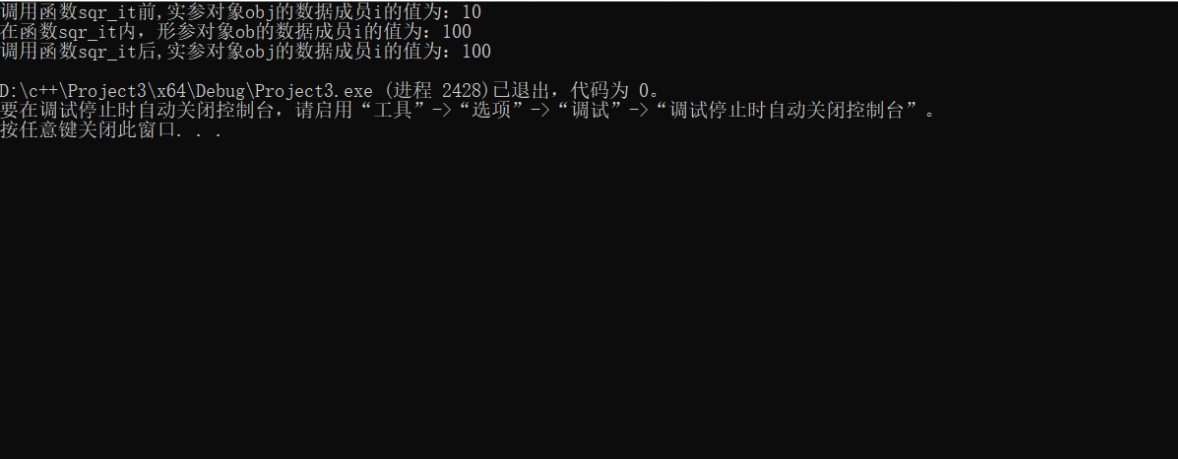
**（1）对象作参数**

****

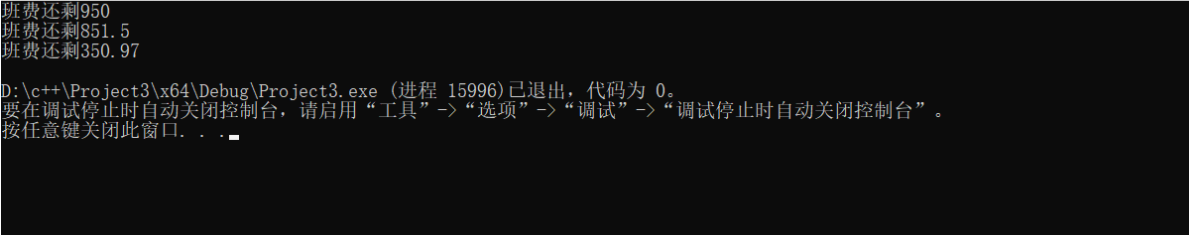
1. **对象指针作参数**



1. **对象引用作参数**



**实验内容2：**



## 程序心得

1. C++中涉及函数传递对象的问题，与C的处理过程类似，需要注意的是使用对象作为参数实现的是实参与形参之间的值传递，只能由实参传给形参，而使用对象指针、使用对象引用差别较小，可类比C。

2、一个类中可以有一个或多个静态成员变量，所有的对象都共享这些静态成员变量，都可以引用它。static 成员变量和普通 static 变量一样，都在内存分区中的全局数据区分配内存，到程序结束时才释放。这就意味着，static 成员变量不随对象的创建而分配内存，也不随对象的销毁而释放内存。而普通成员变量在对象创建时分配内存，在对象销毁时释放内存。静态成员变量必须初始化，而且只能在类体外进行。静态成员变量既可以通过对象名访问，也可以通过类名访问，但要遵循 private、protected 和 public 关键字的访问权限限制。当通过对象名访问时，对于不同的对象，访问的是同一份内存。

Copyright ©2021-2099 BinLu. All rights reserved