程序代码：

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) // 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;

system("pause");

return 0;

}

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;

}

2、

# include<iostream>

using namespace std;

class Ctest{

static int count; //私有成员

public:

Ctest() {

++count;cout<<"对象数量="<<count<<'\n';}

};

int Ctest::count=0; //A，对静态数据定义性说明

int main(void){

Ctest a[3];

return 0;

}

void main()

{

TStudent stu[3];//定义三个学生

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();

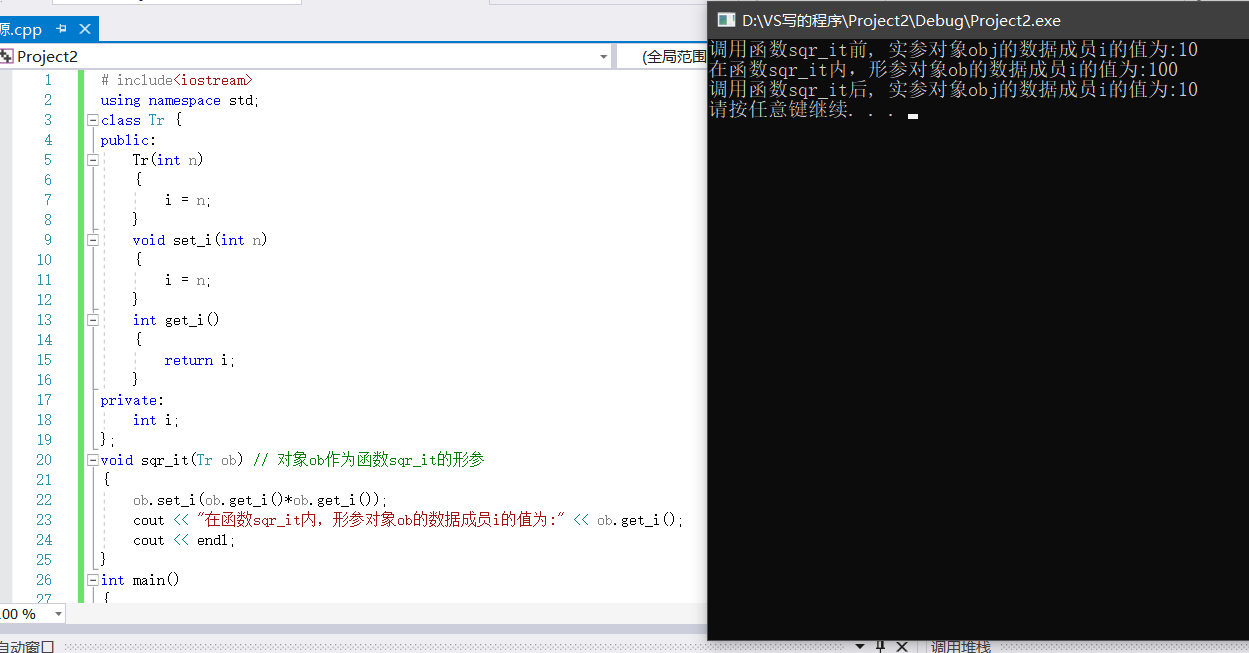
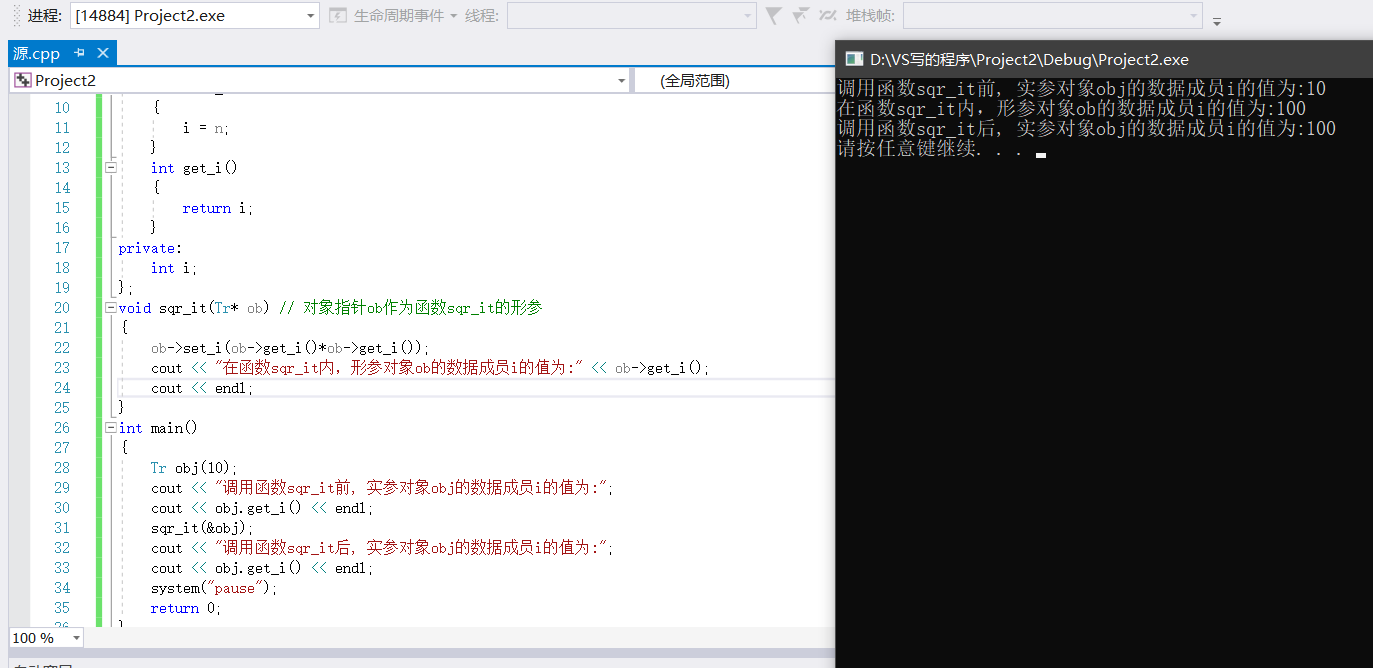
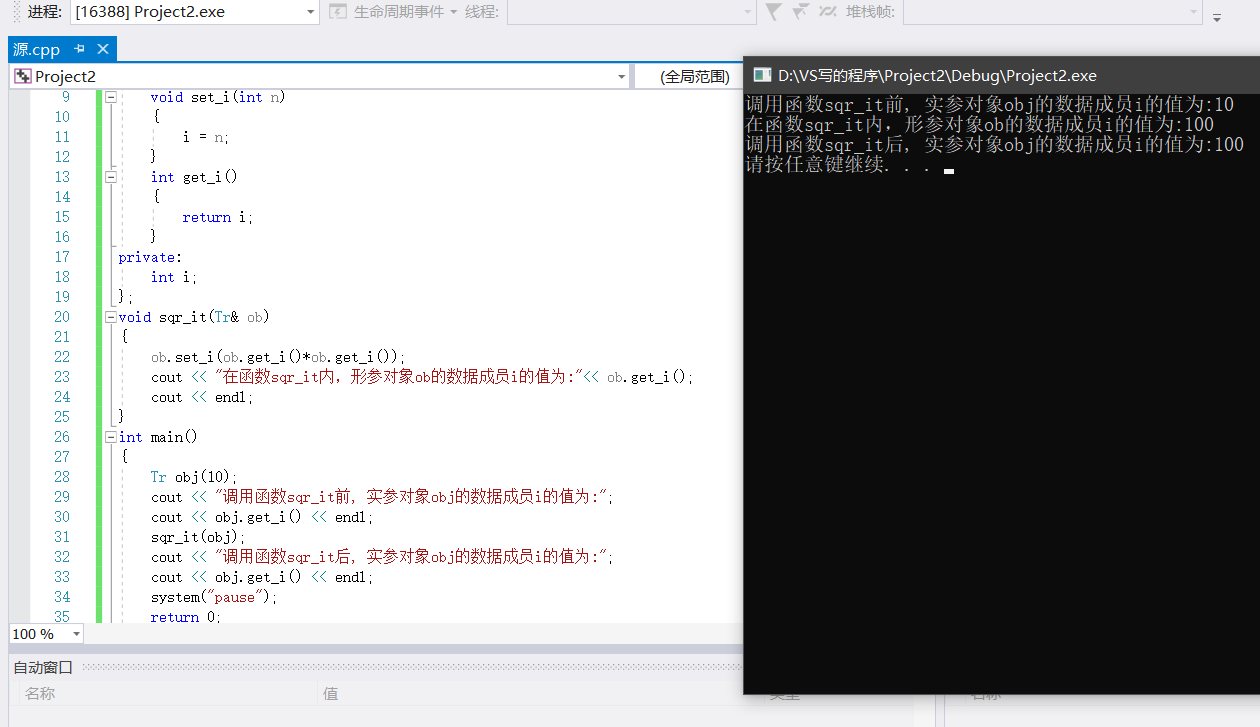
stu[2].ExpendMoney(500.53);

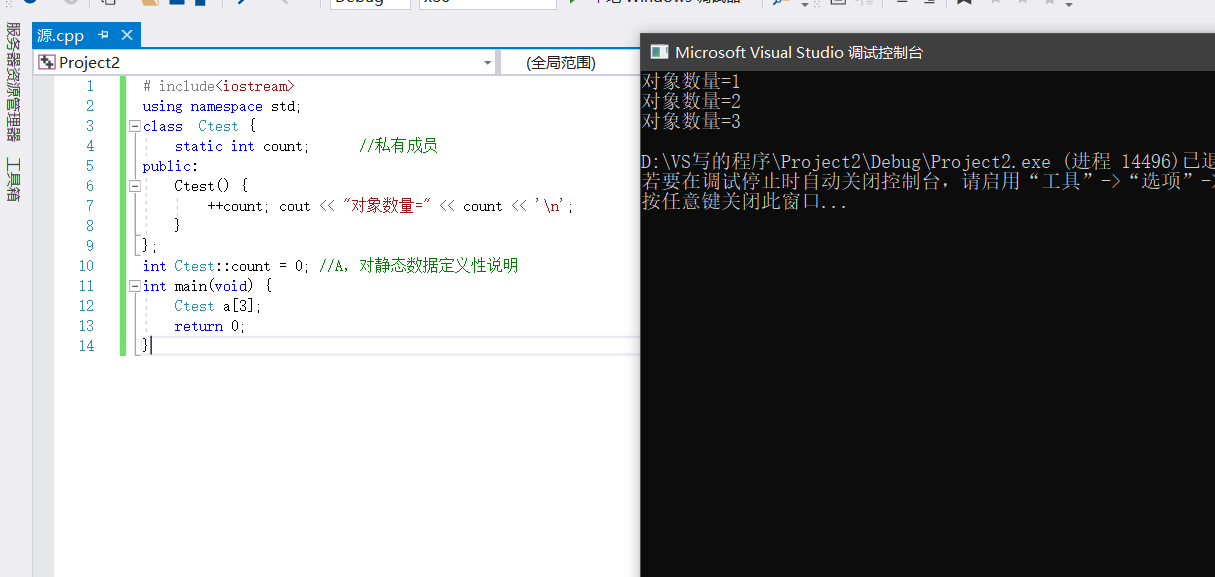
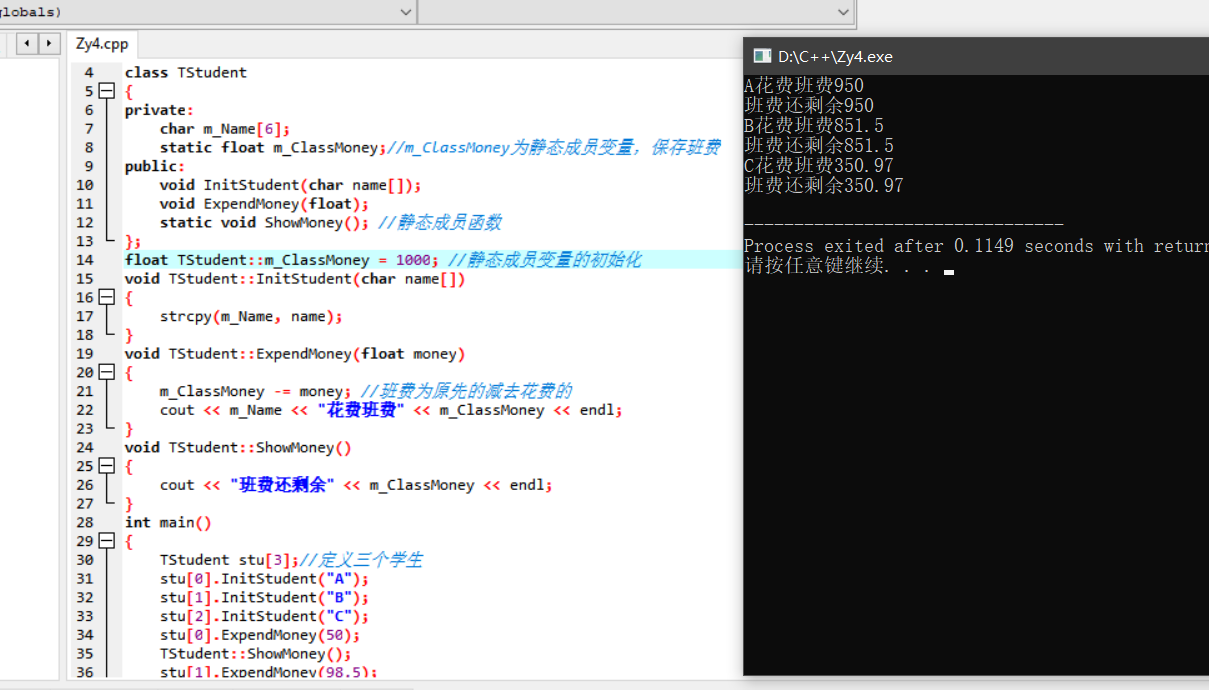
TStudent::ShowMoney();

}

运行结果:

1、

2、

感想心得：

有时对象引用的使用比使用指针更方便，简洁。静态成员函数一般不访问普通数据成员，它的作用主要是访问和操作同类中的静态数据成员或全局变量。

一个类的静态成员函数与非静态成员函数不同，它不需要创建任何该类的对象就可以被调用。