#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;

}

#include <iostream>

using namespace std;

class ctest

{

static int count;

public:

ctest()

{

++count; cout << "对象数量=" << count << '\n';

}

};

int ctest::count = 0;

int main(void)

{

ctest a[3];

return 0;

}

#include<iostream>

using namespace std;

class student

{

static float m\_classmoney;

public:

void initstudent(char nametemp[])

{

for (int count = 0; count < 10; count++)

{

name[count] = nametemp[count];

}

}

void expendmoney(float moneytemp)

{

money = moneytemp;

m\_classmoney -= money;

}

void showmoney()

{

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

}

private:

char name[10];

float money;

};

float student::m\_classmoney = 1000;

void main()

{

char one[10] = { 'A' };

char two[10] = { 'B' };

char three[10] = { 'C' };

student A;

student B;

student C;

A.initstudent(one);

A.expendmoney(50);

A.showmoney();

B.initstudent(two);

B.expendmoney(98.5);

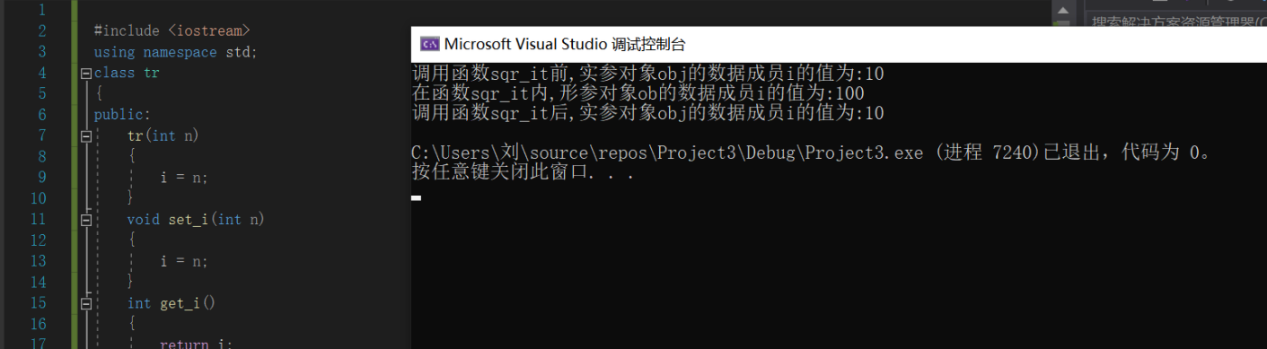
B.showmoney();

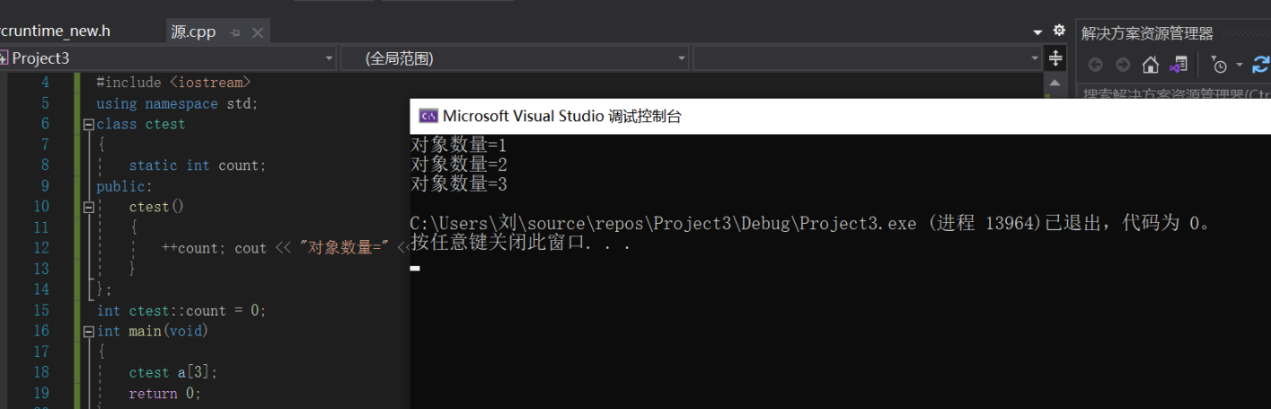
C.initstudent(three);

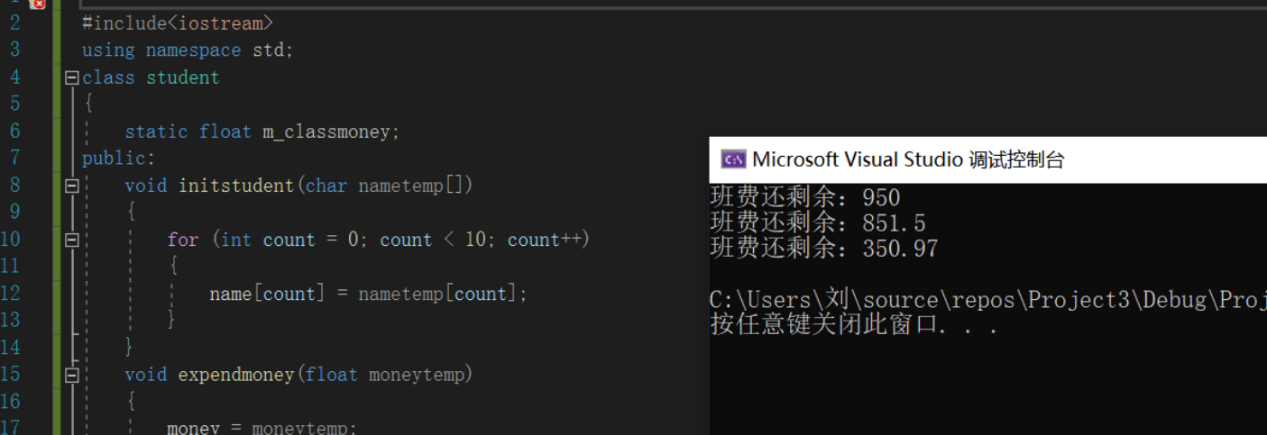
C.expendmoney(500.53);

C.showmoney();

}







**总结**

静态数据成员的初始化与普通数据成员不同。静态数据成员初始化应在类外单独进行，而且应定义对象之前进行。一般在主函数main之前，类声明之后的特殊地带为它提供定义和初始化。初始化格式如下：

数据类型 类名::静态数据成员=初始值

int Ctest::count = 0

这时在数据成员的前面不要加static

静态数据成员属于类的对象的集合，而不像普通函数那样属于某一对象，因此可以使用"::"访问静态的数据成员。

静态数据成员与静态变量一样，是在编译时创建并初始化。它在该类的任何对象被建立之前就存在。因此，共有的数据成员可以在对象被定义之前被访问。对象建立后，共有的静态数据成员也可以被对象进行访问。

使用对象作为函数参数，形参和实参是不同的对象，它们所占地址空间不同，因此，形参

的改变，并不影响实参的值。

使用对象指针作为函数参数中，所谓"传址调用",就是在函数调用时使实参对象和形参对象

的地址传递给函数，形参和实参都指向同一个地址值，此时在函数中对形参对象的修改将影

响调用该函数的实参对象本身。

使用对象引用作为函数的参数，所谓"对象引用"，就是对实参对象换了个别名，实际上它们

仍是同一个对象，所以，所谓的形参(别名对象)值的的改变，直接就是实参对象值的改变。