四.2021.11.12

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());

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<<"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;

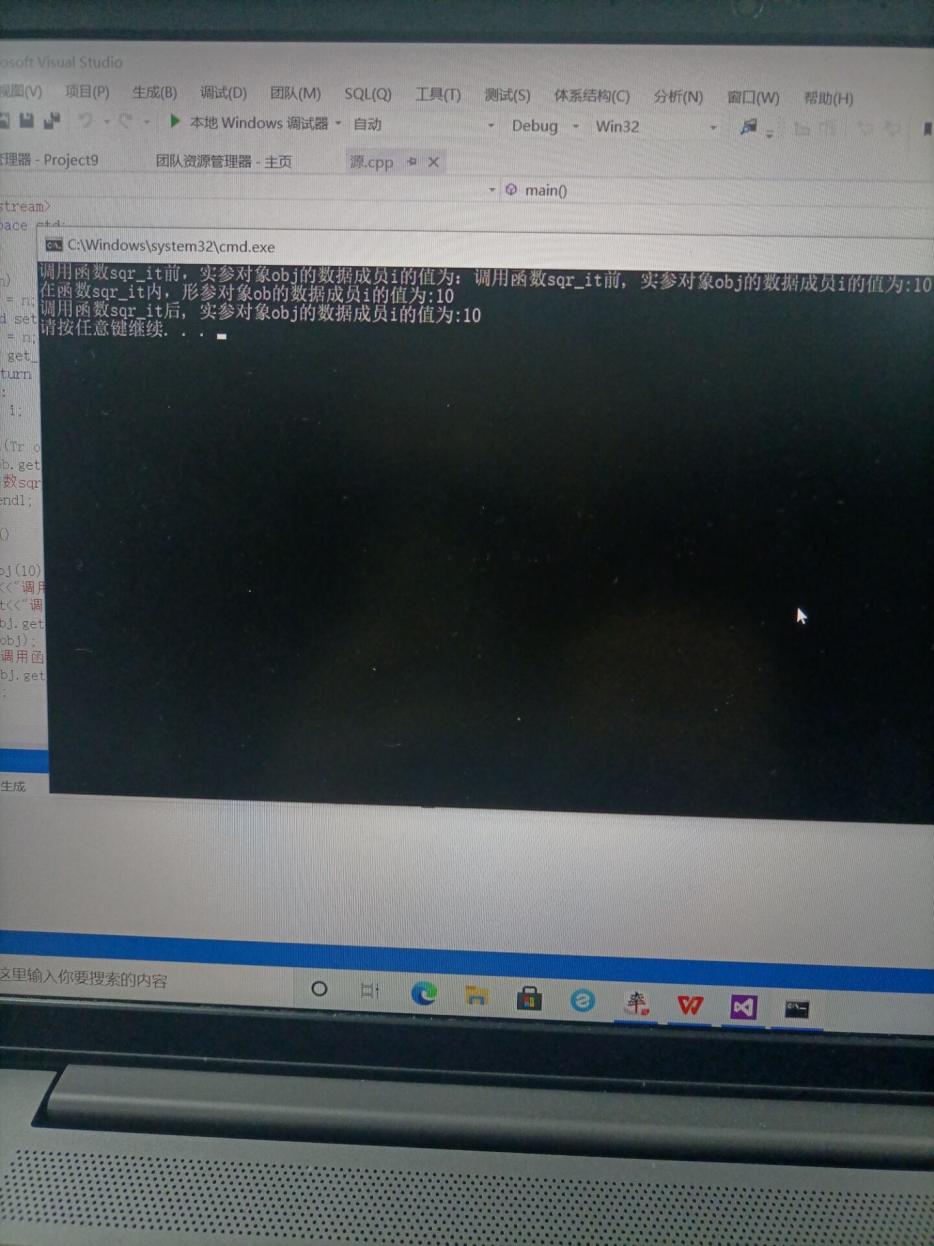
sqr\_it(obj);

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;

}



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';}

};

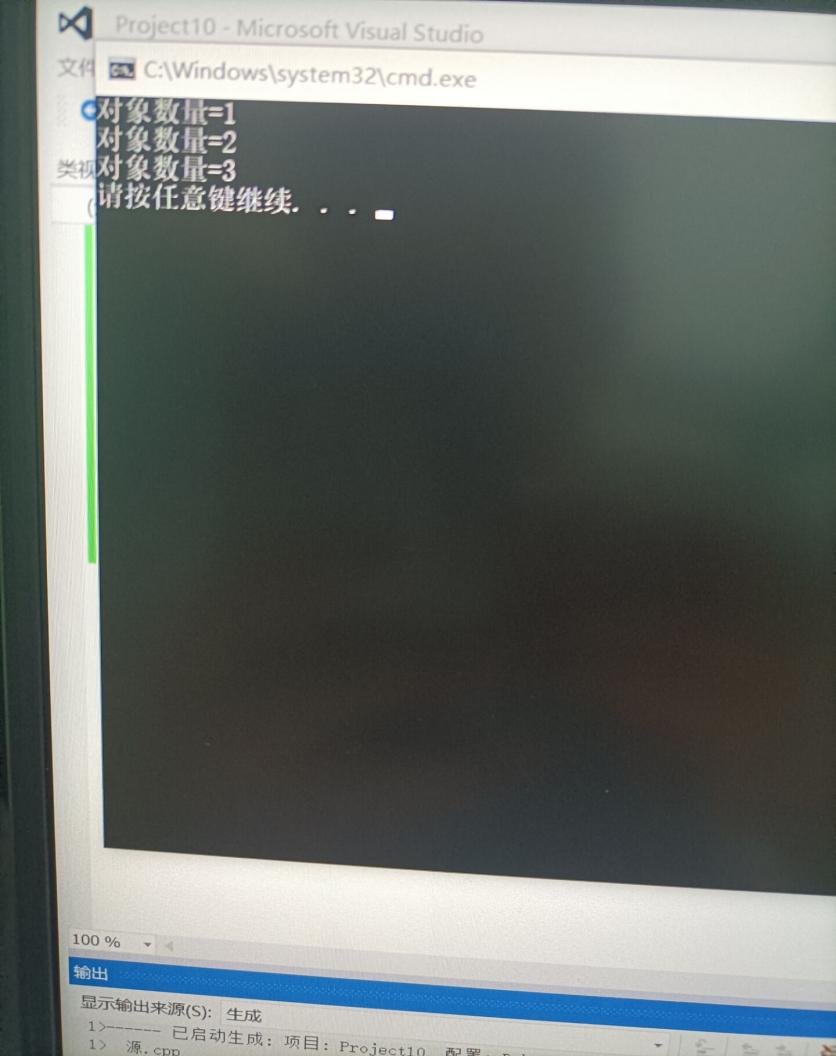
int Ctest::count=0; //A，对静态数据定义性说明

int main(void){

Ctest a[3];

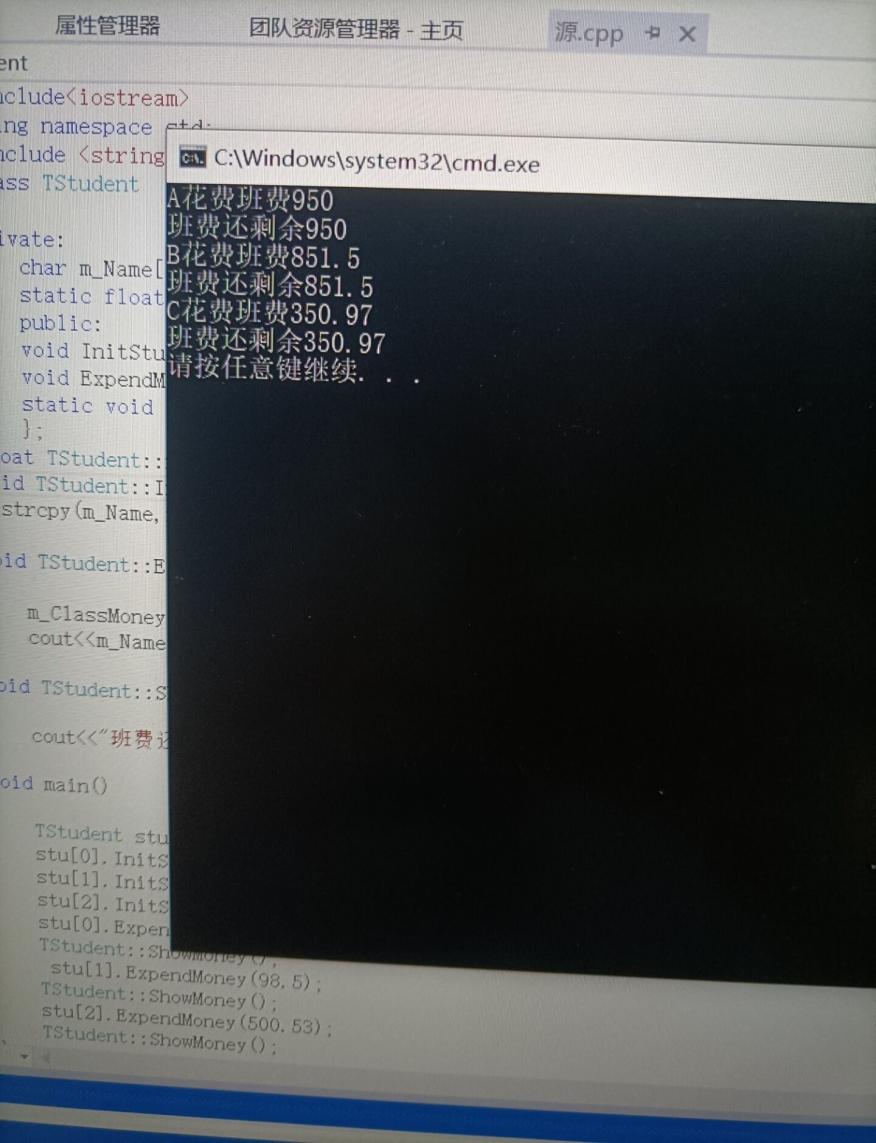
return 0;

}



3.

#include<iostream>  
using namespace std;  
#include <string.h>  
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();  
}



4.实验感想

实验主要考验对数据参数传递的掌握，注意三种传递

（1）值传递：有一个形参向函数所属的栈拷贝数据的过程，如果值传递的对象是类对象或是大的结构体对象，将耗费一定的时间和空间。  
  
（2）.指针传递：同样有一个形参向函数所属的栈拷贝数据的过程，但拷贝的数据是一个固定为4字节的地址。  
  
（3）.引用传递：同样有上述的数据拷贝过程，但其是针对地址的，相当于为该数据所在的地址起了一个别名。  
  
效率上讲，指针传递和引用传递比值传递效率高。一般主张使用引用传递，代码逻辑上更加紧凑、清晰。