**第四次上机报告**

**上机内容：**

**实验目的：**

一、对象传递使用方法：

（1）普通对象传递（2）引用对象传递（3）指针对象传递

二、指针和引用的区别

三、静态成员概念和使用

**上机程序：**

**hero.h程序**

#include<iostream>

using namespace std;

class Hero

{

public://公有

Hero(); //构造函数

Hero(int x1,int x2,int x3);

Hero(const Hero & name);

~Hero(); //析构函数

Hero &operator=(const Hero&rhs);//等号运算符重载 右等

void showskill();

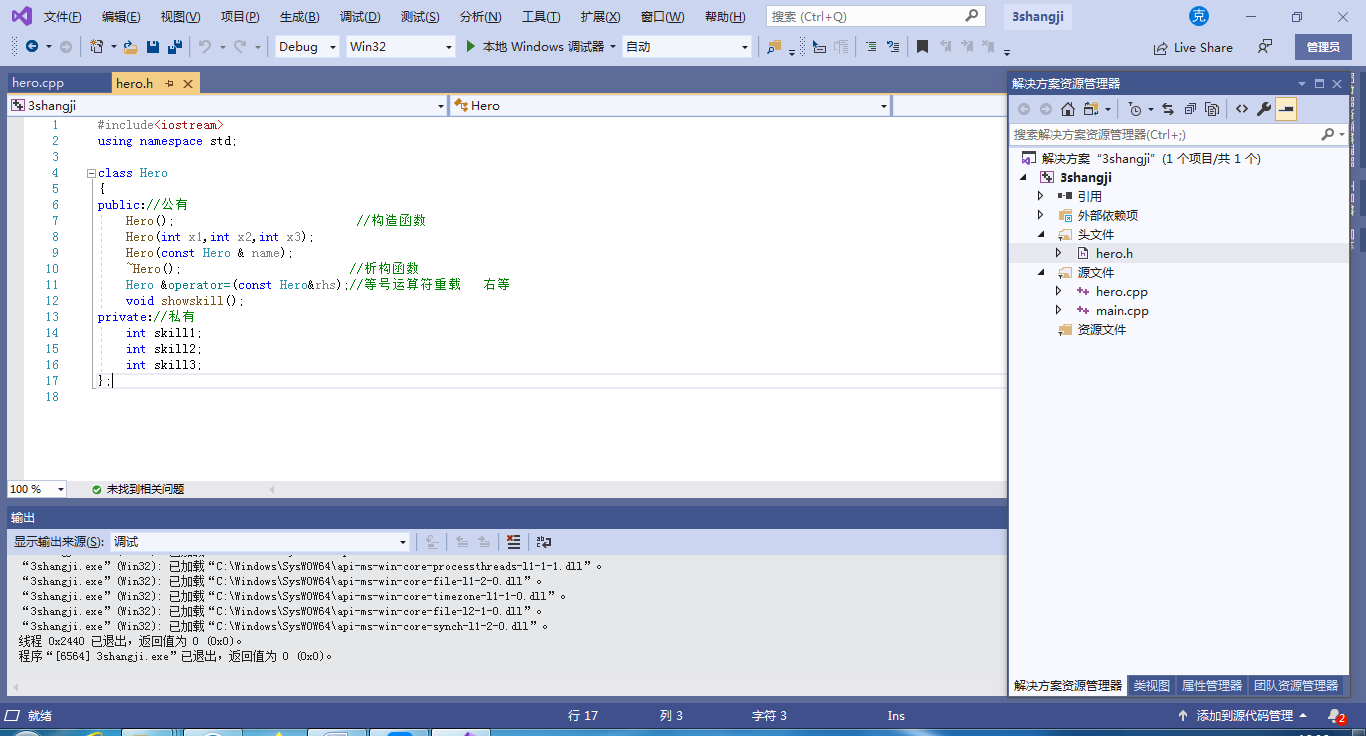
private://私有

int skill1;

int skill2;

int skill3;

};



**hero.cpp程序：**

#include"hero.h"

Hero::Hero()

{

cout<<"This is a defult constructor!"<<endl;

skill1 =0;

skill2 =0;

skill3 =0;

}

Hero::Hero(int x1,int x2,int x3)

{

cout<<"This is a overloaded constructor!"<<endl;

skill1 =x1;

skill2 =x2;

skill3 =x3;

}

Hero::Hero(const Hero & name)

{

cout<<"This is a copy constructor!"<<endl;

skill1 =name.skill1;

skill2 =name.skill2;

skill3 =name.skill3;

}

Hero::~Hero() //析构函数

{

cout<<"This is a constructor!"<<endl;

}

Hero & Hero::operator=(const Hero &rhs)

{

if (this ==&rhs)

{

return\*this;

}

this->skill1=rhs.skill1;

this->skill2=rhs.skill2;

this->skill3=rhs.skill3;

return\*this;

}

void Hero::showskill()

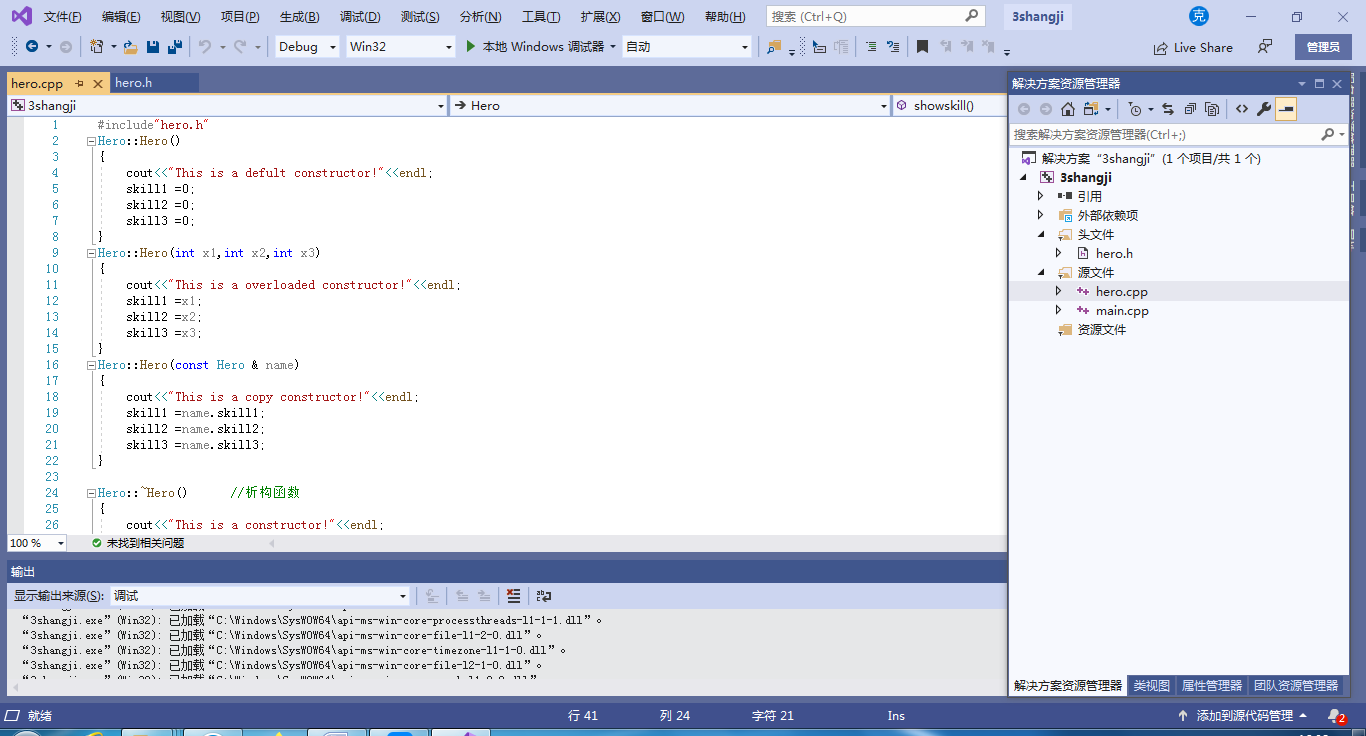
{

cout<<skill1<<endl;

cout<<skill2<<endl;

cout<<skill3<<endl;

}



**main.cpp程序：**

#include<iostream>

#include"hero.h"

using namespace std;

int main()

{

Hero Libai;

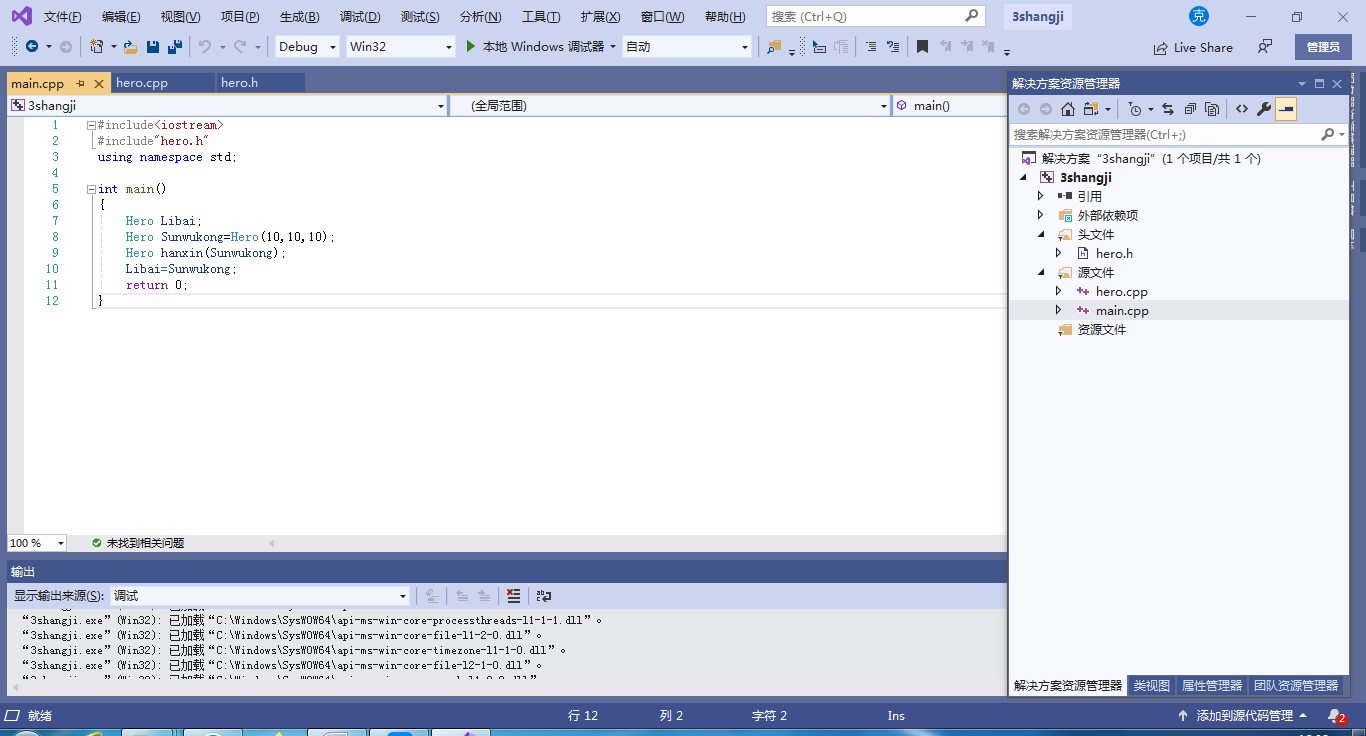
Hero Sunwukong=Hero(10,10,10);

Hero hanxin(Sunwukong);

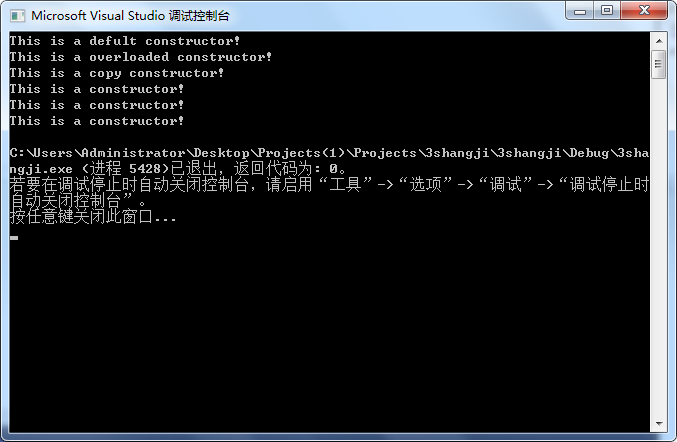
Libai=Sunwukong;

return 0;

}



**运行结果：**

****

**问题：**

注意：静态数据和函数都是属于类的不属于对象