# Text3 使用类定义技能

1. **设计程序的逻辑思路**

本实验采用C++的程序风格，建立头文件Hero.h,源文件Hero.cpp,main.cpp实现程序的功能。在hero.h文件里面，建立了一个类名为Hero的类，其中里面有两个权限，一个私有权限private和公有权限public，在private下面定义了三个私有变量skill1；skill2；skill3，在public下面定义了三种不同类型的构造函数和一个析构函数。在Hero.cpp下面分别对三个不同类型的构造函数和一个析构函数进行了具体定义，最后在main.cpp里面输出了每个函数具体的作用和功能显示。

1. **程序主体代码**

**hero.h**

#include <iostream>

using namespace std;

class Hero

{

public:

Hero();//默认构造函数

Hero(int x1,int x2,int x3);//重载构造函数

Hero(const Hero & name);//复制构造函数

~Hero();//析构函数

Hero &operator=(const Hero &rhs);

void showskill();

private:

int skill1;

int skill2;

int skill3;

};

**hero.cpp**

#include"hero.h"

Hero::Hero()

{

cout<<"默认构造函数"<<endl;

skill1=0;

skill2=0;

skill3=0;

}

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

{

cout<<"重载构造函数"<<endl;

skill1=x1;

skill2=x2;

skill3=x3;

}

Hero::Hero(const Hero & name)

{

cout<<"复制构造函数"<<endl;

skill1=name.skill1;

skill2=name.skill2;

skill3=name.skill3;

}

Hero::~Hero()

{

cout<<"析构函数"<<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**

#include <iostream>

#include"hero.h"

using namespace std;

int main()

{

Hero libai;

cout<<"libai"<<endl;

libai.showskill();

Hero sunwukong(10,10,10);

cout<<"sunwukong"<<endl;

sunwukong.showskill();

Hero hanxin(sunwukong);

cout<<"hanxi"<<endl;

hanxin.showskill();

libai=sunwukong;

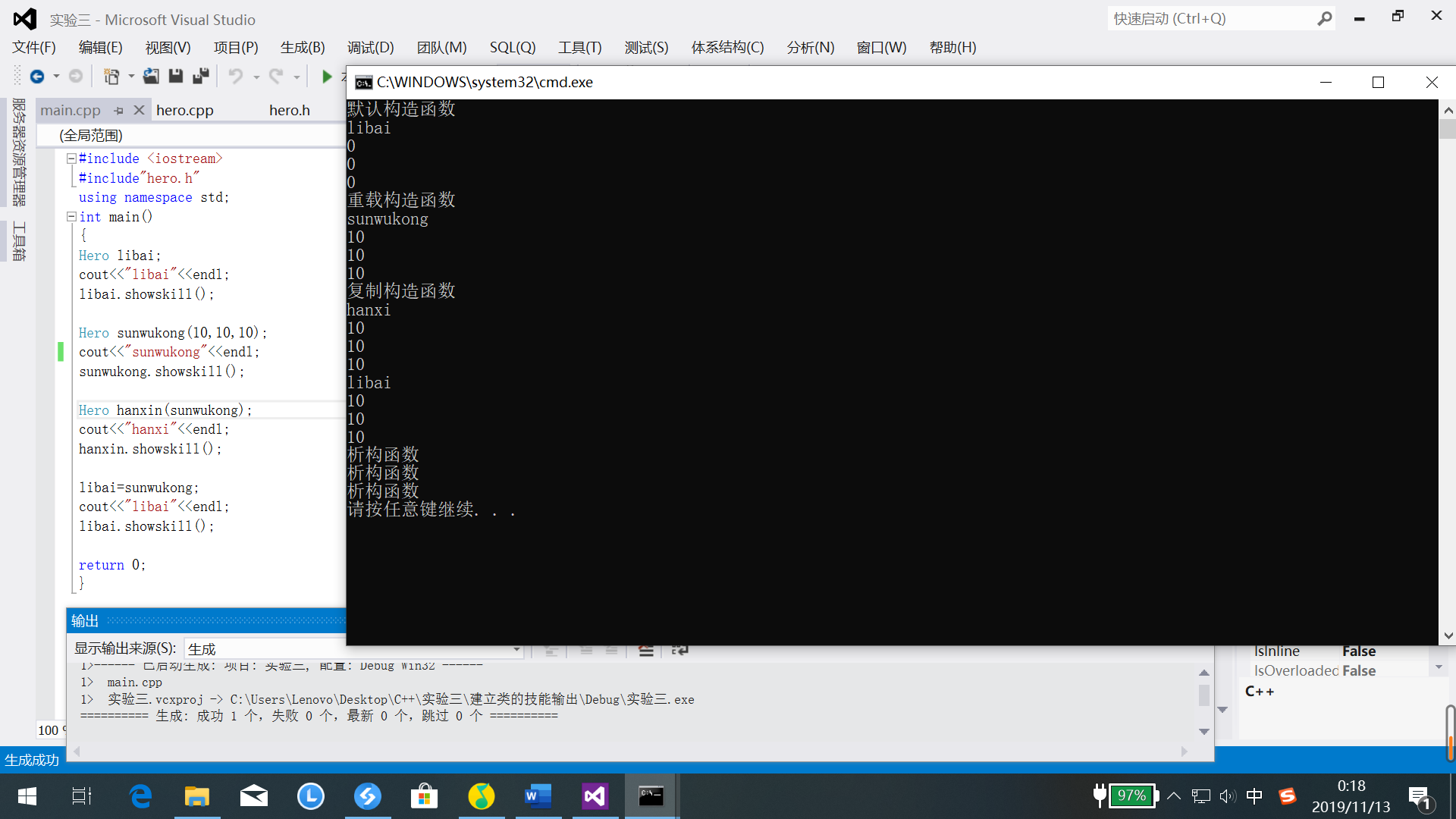
cout<<"libai"<<endl;

libai.showskill();

return 0;

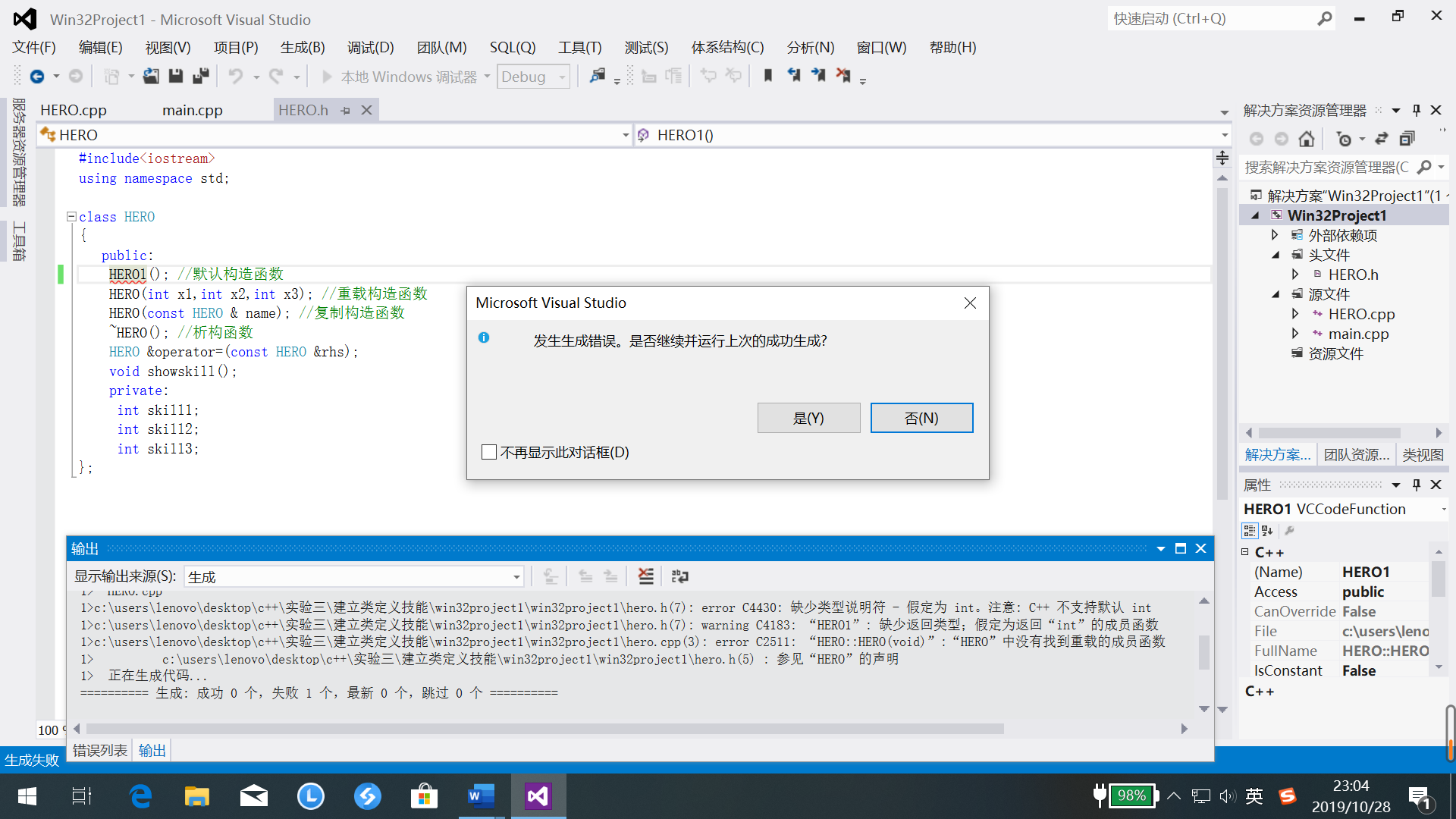
}

其输出显示如图所示：



1. **在设计程序时产生的错误及解决方法**

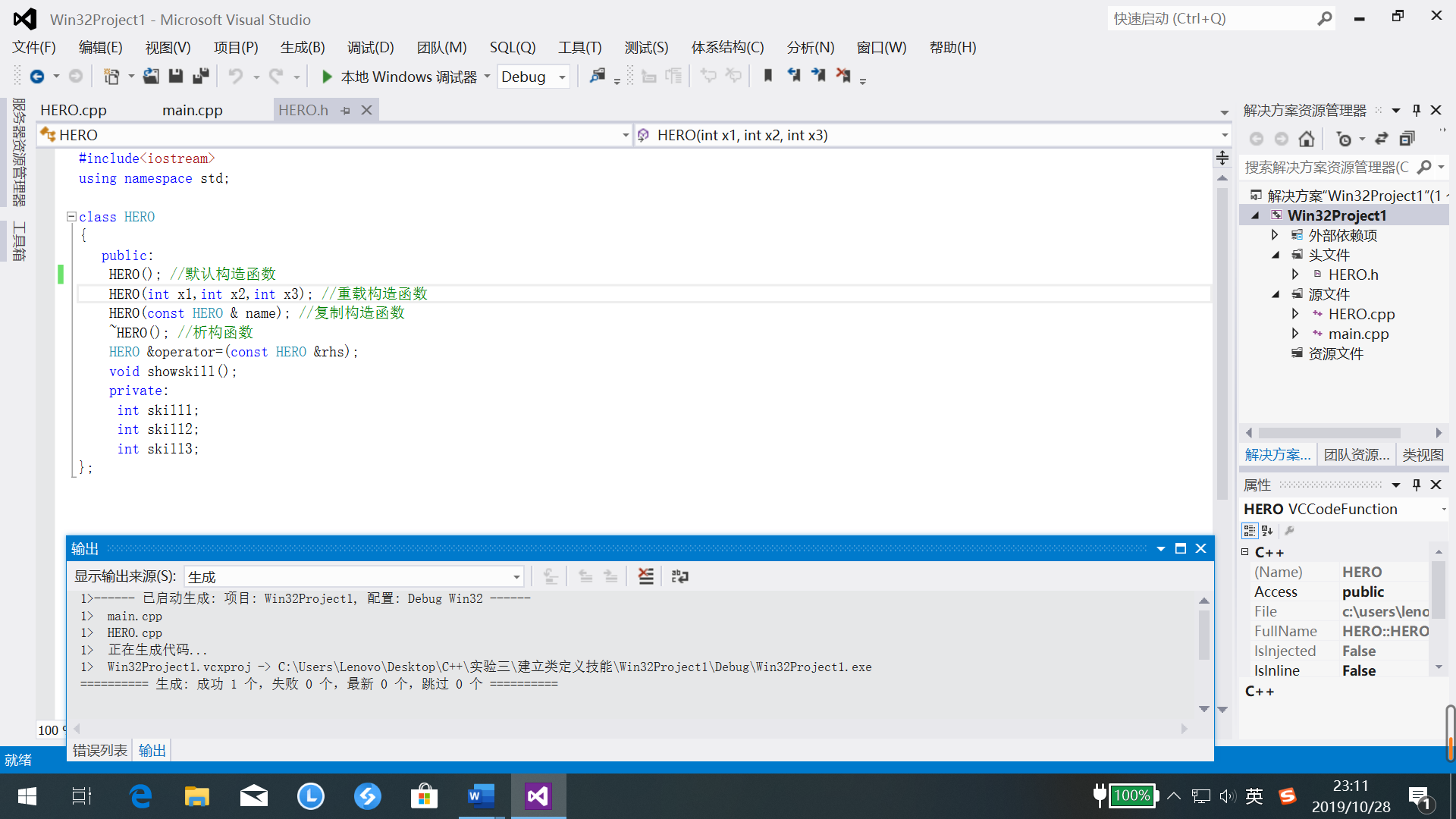
在函数的类中，其默认的构造函数必须与类名相同，否则会发生如下图所示的错误：



解决方法就是将默认构造函数HERO1改成HERO即可。

下面是对程序进行解释：

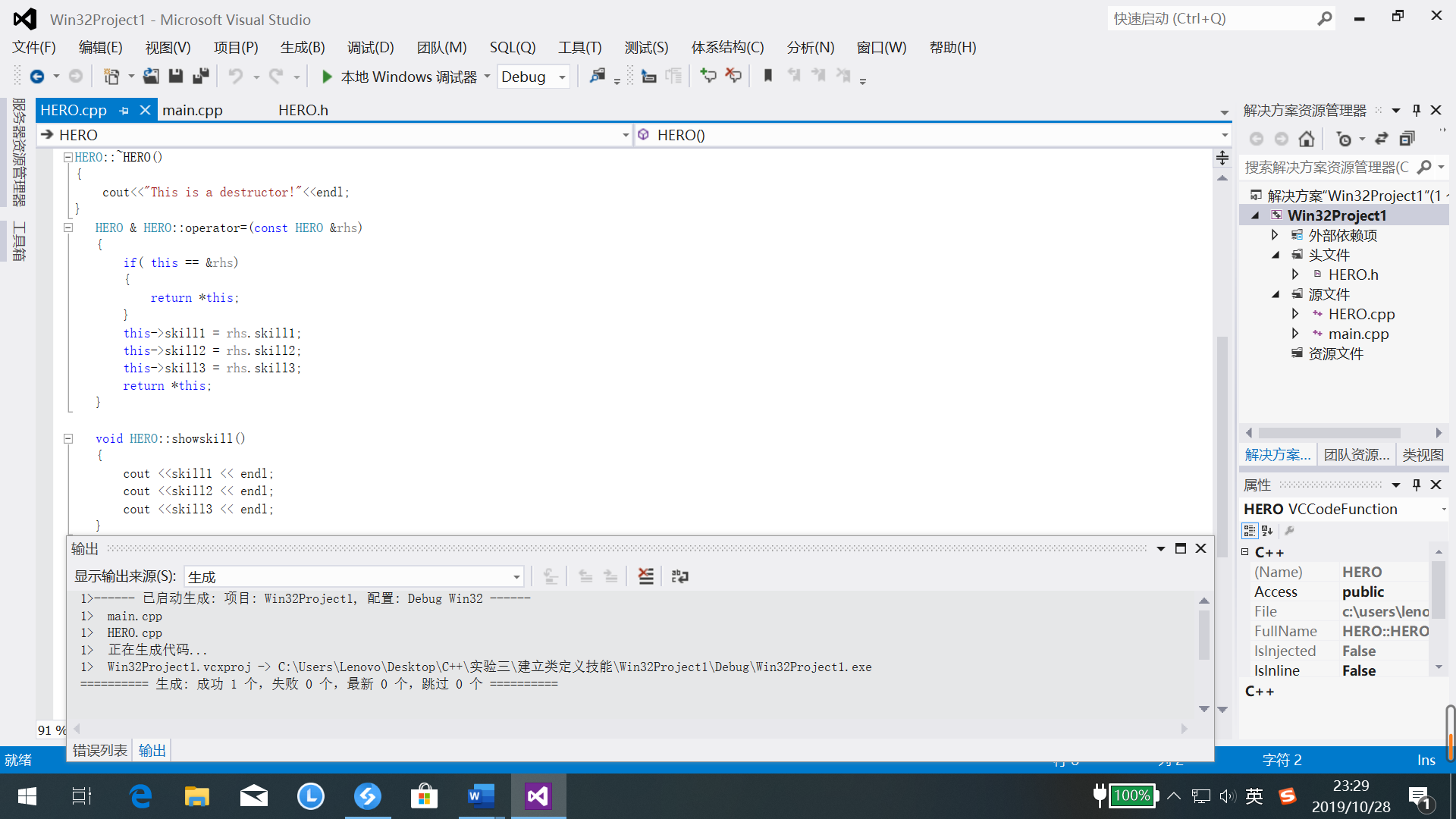
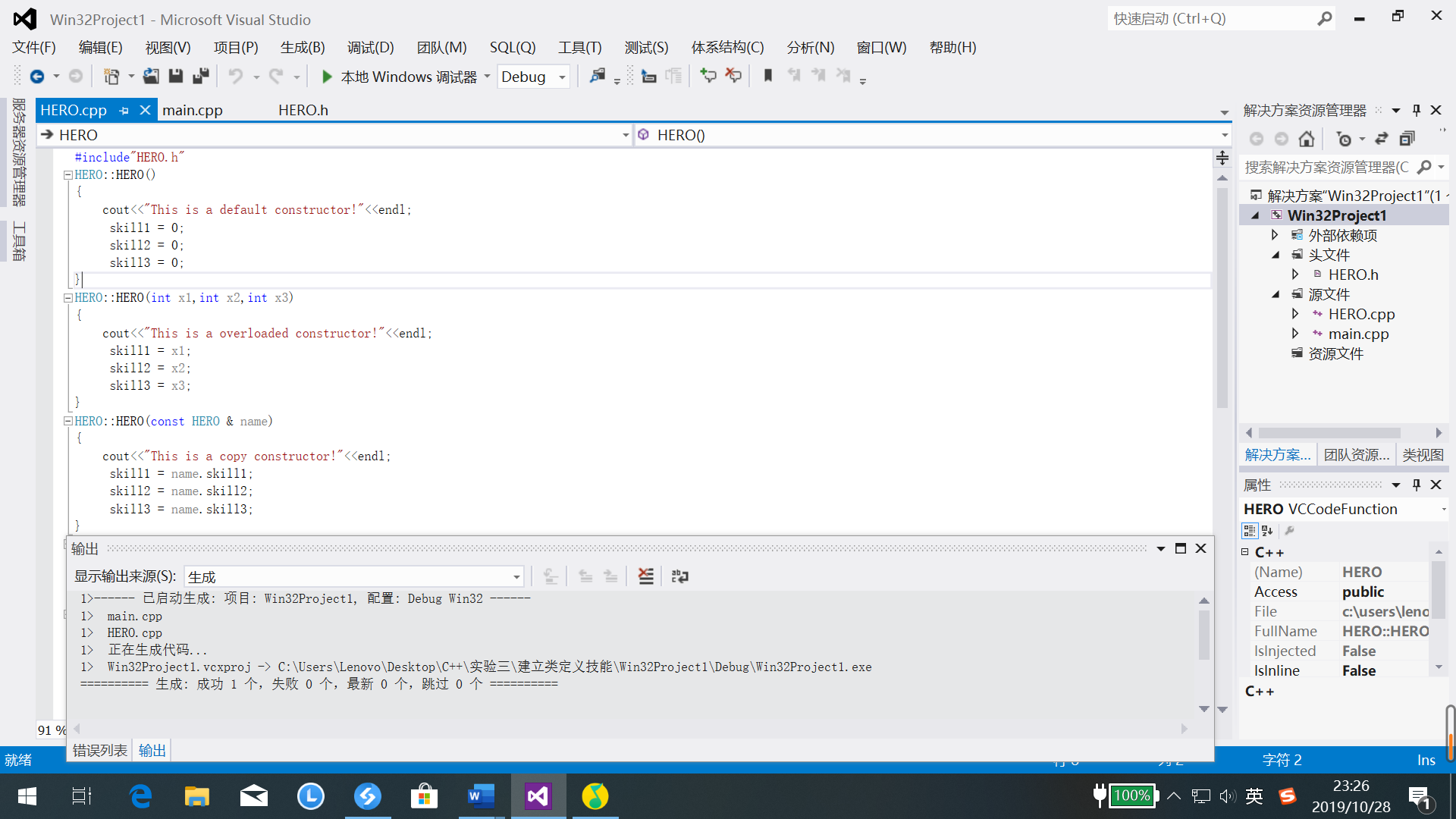
**首先在HERO.h中：**



在这个头函数内，定义了一个名为class的类，其中里面包括两个权限，一个私有权限（private），一个公有权限（public），在公有权限内，分别定义了默认构造函数，重载构造函数，复制构造函数，析构函数，成员函数。在私有权限下面，定义了三个整形变量。在类的内部（定义类的代码内部），无论成员被声明为 public、protected 还是 private，都是可以互相访问的，没有访问权限的限制。在类的外部（定义类的代码之外），只能通过对象访问成员，并且通过对象只能访问 public 属性的成员，不能访问 private、protected 属性的成员。

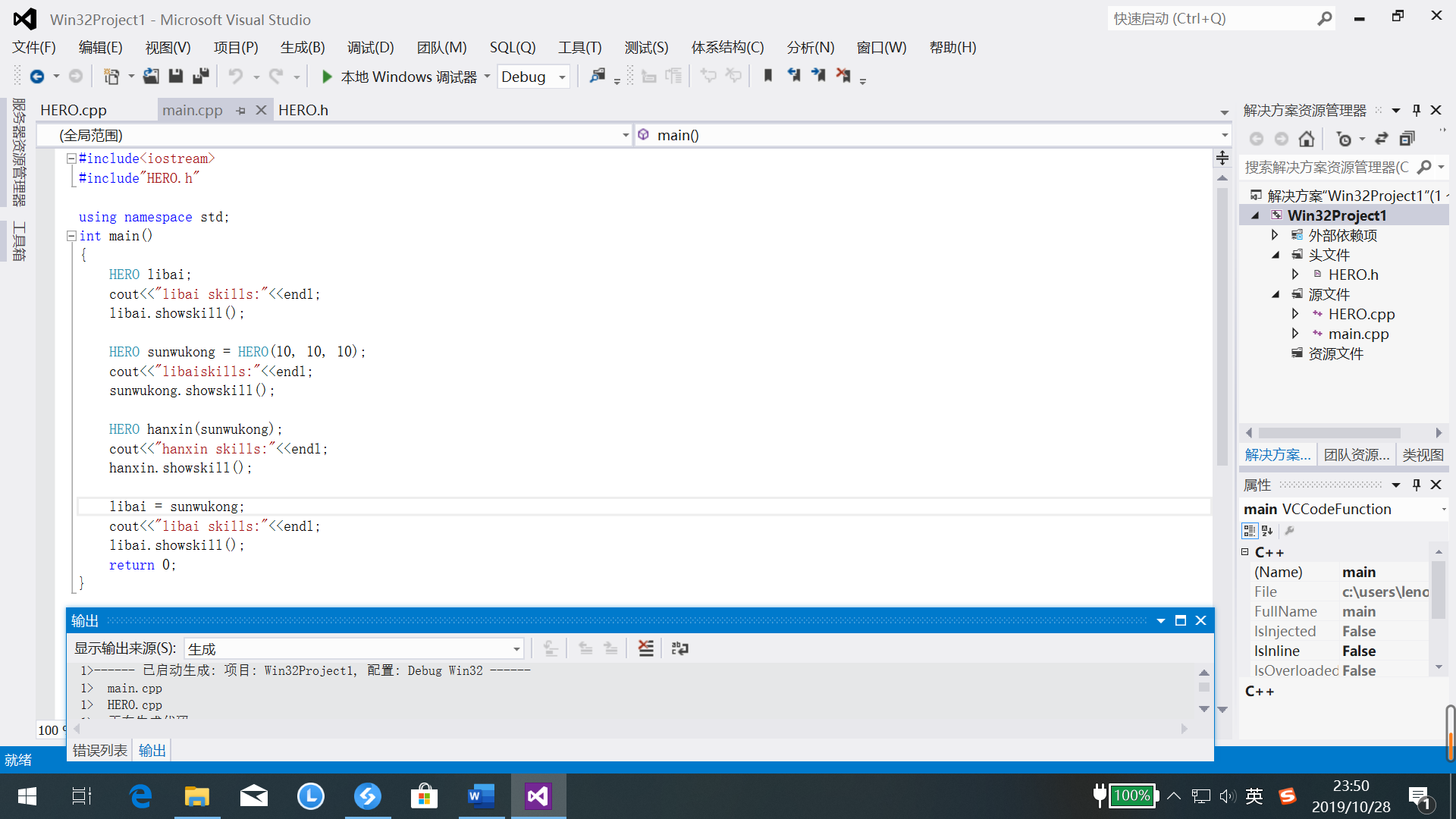
**在HERO.cpp中**

对于HERO.h头文件里的HERO类中的每个公有权限下面的函数进行具体定义，具体定义如下图所示：



**在main.cpp中**

主要是对于HERO.cpp中的函数及HERO.h中的函数进行调用，具体调用如下图所示：



1. **思想感悟**

在本次实验中，学会了对于类的构建，对于类中的默认构造函数，重载构造函数，复制构造函数，析构函数，成员函数有了初步的了解，对于指针地址符和绝对引用运算符有了更加深入的区分，在习惯了C语言的程序员眼里，&这个符号的意思就是取地址和对位进行与运算，但在C++中，& 被重载多了一种意思 ，就是引用。在这里我就不介绍位与运算了，我先介绍一下取地址运算，在学习指针的时候我们就已经用过了取地址符了，使用方法为在变量前面加上&即可。

权限其实就是对类成员的可访问性。

1.公有成员

类可以访问，

类对象可以访问，

通过公有继承的派生类可以访问，

通过私有继承的派生类可以访问，

通过保护继承的派生类可以访问，

通过公有继承的派生类的对象可以访问，

通过私有继承的派生类的对象不可以访问，

通过保护继承的派生类的对象不可以访问。

2.私有成员

类可以访问，

类对象不可以访问，

通过公有继承的派生类不可以访问，

通过私有继承的派生类不可以访问，

通过保护继承的派生类不可以访问，

通过公有继承的派生类的对象不可以访问，

通过私有继承的派生类的对象不可以访问，

通过保护继承的派生类的对象不可以访问。

1. **上机作业**

**程序附源代码**

程序的运行结果如下图所示：

