C#调用C++类（托管C++实现）_手可摘星辰0120的博客-CSDN博客

成就一亿技术人!

往往一个项目，算法是用C++写的，而界面需要用C#的Winform去编写，这个时候C#界面需要去调用C++类里面的函数，因为C#和C++是两种不同的语言，两者之间不能直接调用，所以他们之间需要一个桥梁，而托管C++可以充当这个桥梁

项目代码链接：

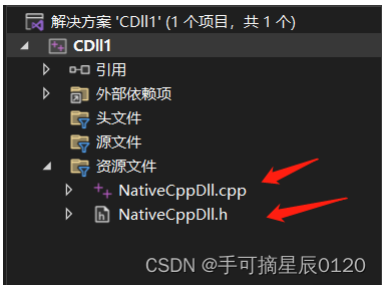
链接：https://pan.baidu.com/s/1VGkQZjygyc6K_hTR3upugg
提取码：5678

具体实现过程如下：

1.建立一个普通的C++DLL文件（这一步一般不需要做，这是第三方给的库文件）

看看第三方库文件是如何创建出来的

1.1首先建立一个空项目，添加下面两个文件

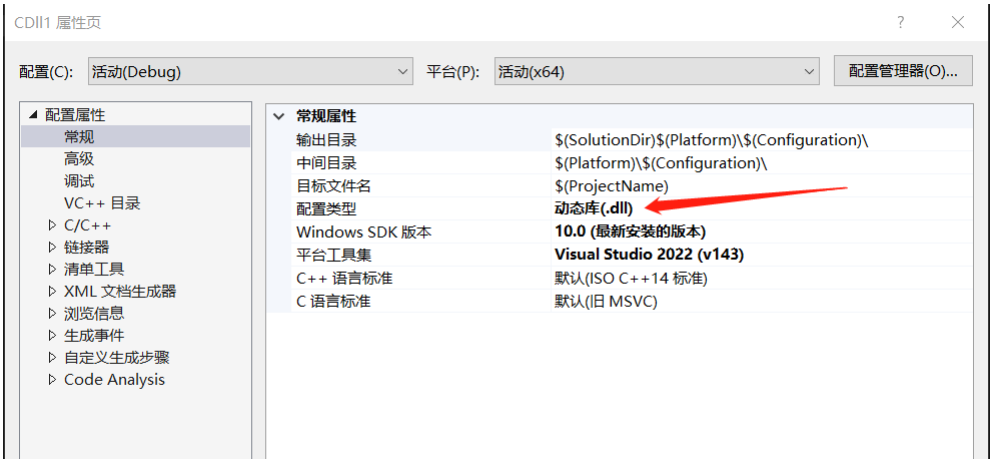


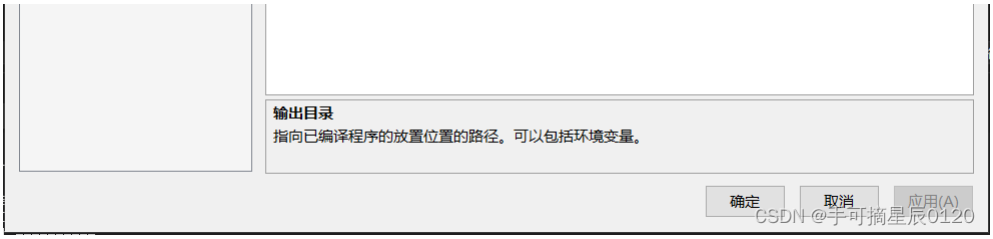

```
#pragma once

#ifndef LX_DLL_CLASS_EXPORTS
#define LX_DLL_CLASS __declspec(dllexport)
#else
#define LX_DLL_CLASS __declspec(dllimport)
#endif

class LX_DLL_CLASS CPerson
{
public:
CPerson();
CPerson(const wchar_t* pName, const wchar_t cSex, int iAge);

void SetName(const wchar_t* pName);
wchar_t* GetName();

void SetSex(const wchar_t cSex);
wchar_t GetSex();

void SetAge(int iAge);
int GetAge();

wchar_t* GetLastError();

private:
wchar_t m_szName[128];
wchar_t m_cSex;
int m_iAge;
wchar_t m_szLastError[128];

void ShowError();
};
```

```
#include "NativeCppDll.h"
#include <iostream>
#include <tchar.h>

using namespace std;

CPerson::CPerson()
{
wcscpy_s(m_szName, _T("No Name"));
    m_cSex = 'N';
    m_iAge = 0;

wcscpy_s(m_szLastError, _T("No Error"));
}

CPerson::CPerson(const wchar_t* pName, const wchar_t cSex, int iAge)
{
wcscpy_s(m_szLastError, _T("No Error"));

SetName(pName);
SetSex(cSex);
SetAge(iAge);
}

void CPerson::SetName(const wchar_t* pName)
{
if ((pName == NULL) || (wcslen(pName) == 0) || (wcslen(pName) > 127))
    {
wcscpy_s(m_szName, _T("No Name"));

wcscpy_s(m_szLastError, _T("The length of the input name is out of range."));

ShowError();

return;
    }

wcscpy_s(m_szName, pName);
}

wchar_t* CPerson::GetName()
{
return m_szName;
}

void CPerson::SetSex(const wchar_t cSex)
{
if ((cSex != 'F') && (cSex != 'M') && (cSex != 'm') && (cSex != 'f'))
    {
        m_cSex = 'N';

wcscpy_s(m_szLastError, _T("The input sex is out of [F/M]."));

ShowError();
```

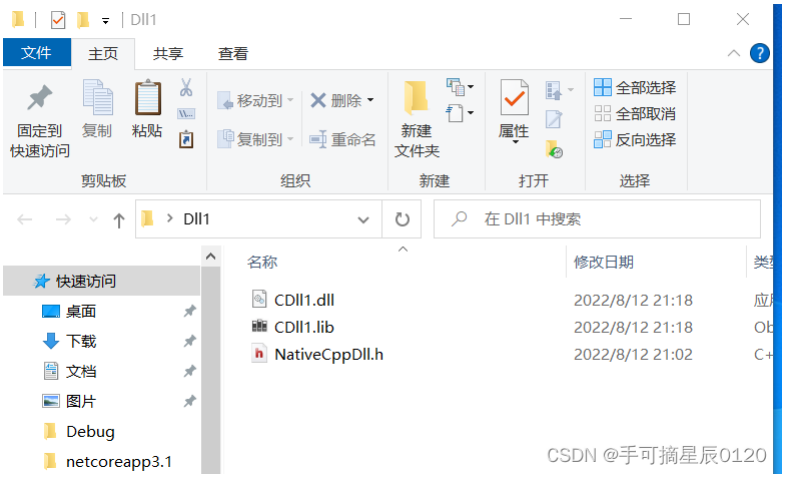
```
return;
    }

    m_cSex = cSex;
}

wchar_t CPerson::GetSex()
{
return m_cSex;
}

void CPerson::SetAge(int iAge)
{
if ((iAge < 0) || (iAge > 150))
    {
        m_iAge = 0;

wcscpy_s(m_szLastError, _T("The input age is out of range."));

ShowError();

return;
    }

    m_iAge = iAge;
}

int CPerson::GetAge()
{
return m_iAge;
}

wchar_t* CPerson::GetLastError()
{
return m_szLastError;
}

void CPerson::ShowError()
{
    cerr << m_szLastError << endl;
}
```

1.2修改项目属性





1.3将生成的三个文件拿出来备用（dll和lib在项目输出目录下）

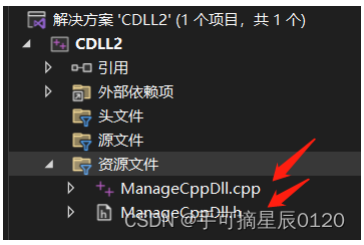


2.建立一个CLR空项目，将第一步生成的dll重新封装

在对第一步的C++类进行暴露的时候，只需要将共有的类成员暴露就行

2.1建立一个CRL空项目

2.2添加下面两个文件




```
#pragma once
#define LX_DLL_CLASS_EXPORTS

#include "NativeCppDll.h"

#pragma comment(lib,"CDll1.lib")

using namespace System;

namespace ManageCppDll
{
public ref class Person
    {
```

```
public:
Person();
Person(String^ strName, Char cSex, int iAge);

        ~Person();

        property String^ Name
        {
void set(String^ strName);
            String^ get();
        }

        property Char Sex
        {
void set(Char cSex);
Char get();
        }

        property int Age
        {
void set(int iAge);
int get();
        }

        String^ GetLastError();

private:

        CPerson* m_pImp;
    };
};
```

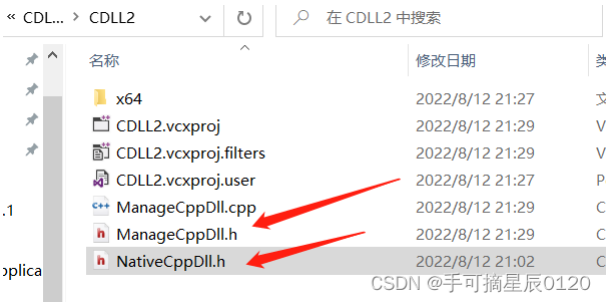


```
#include "ManageCppDll.h"
#include <vcclr.h>

namespace ManageCppDll
{



    Person::Person()
    {
        m_pImp = new CPerson();
    }

    Person::Person(String^ strName, Char cSex, int iAge)
    {

        pin_ptr<const wchar_t> wcName = PtrToStringChars(strName);

        m_pImp = new CPerson(wcName, cSex, iAge);
    }

    Person::~Person()
    {
```

```
delete m_pImp;
    }

void Person::Name::set(String^ strName)
    {
        pin_ptr<const wchar_t> wcName = PtrToStringChars(strName);

        m_pImp->SetName(wcName);
    }

    String^ Person::Name::get()
    {
return gcnew String(m_pImp->GetName());
    }

void Person::Sex::set(Char cSex)
    {
        m_pImp->SetSex(cSex);
    }

    Char Person::Sex::get()
    {
return m_pImp->GetSex();
    }

void Person::Age::set(int iAge)
    {
        m_pImp->SetAge(iAge);
    }

int  Person::Age::get()
    {
return m_pImp->GetAge();
    }

    String^ Person::GetLastError()
    {
return gcnew String(m_pImp->GetLastError());
    }
};
```

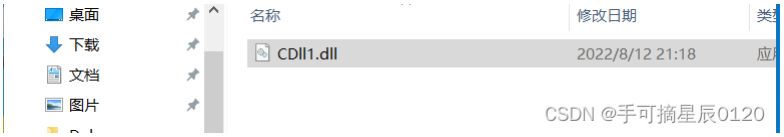
2.3引入第一步建立的库文件

2.3.1第一步的.h文件



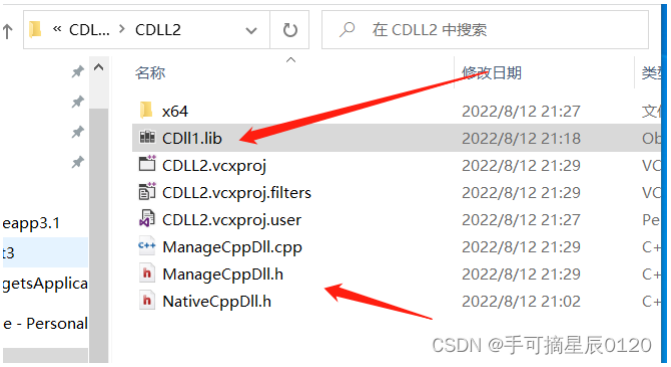
2.3.2 dll放在项目的输出目录下，也就是exe同级目录下

（例如：E:\vs2022Code\Dlltest\CDLL2\x64\Debug）

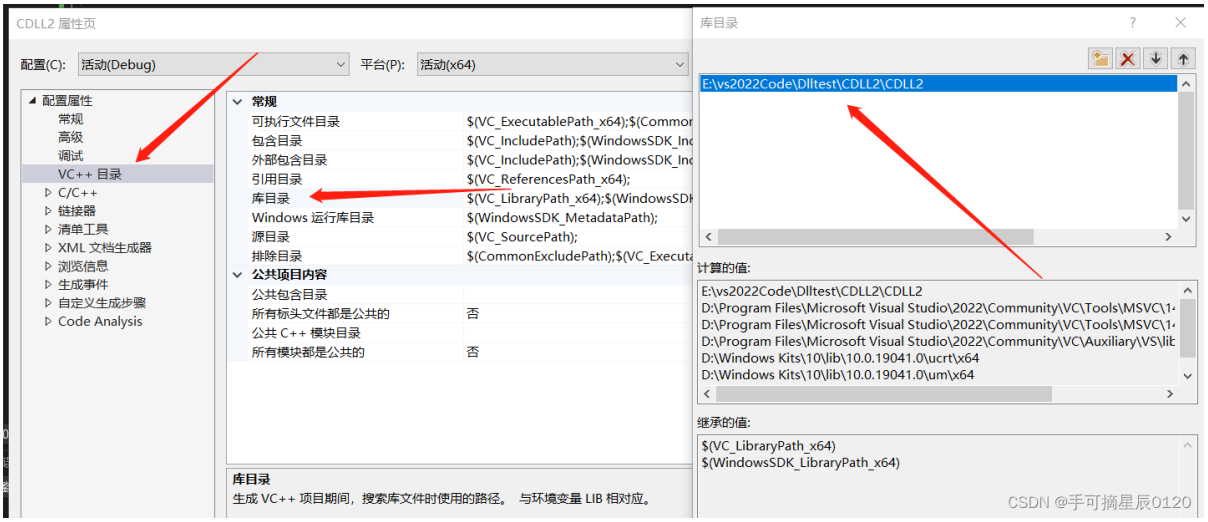


2.3.3 lib放在项目哪里都行，但必须得告诉编译器地址

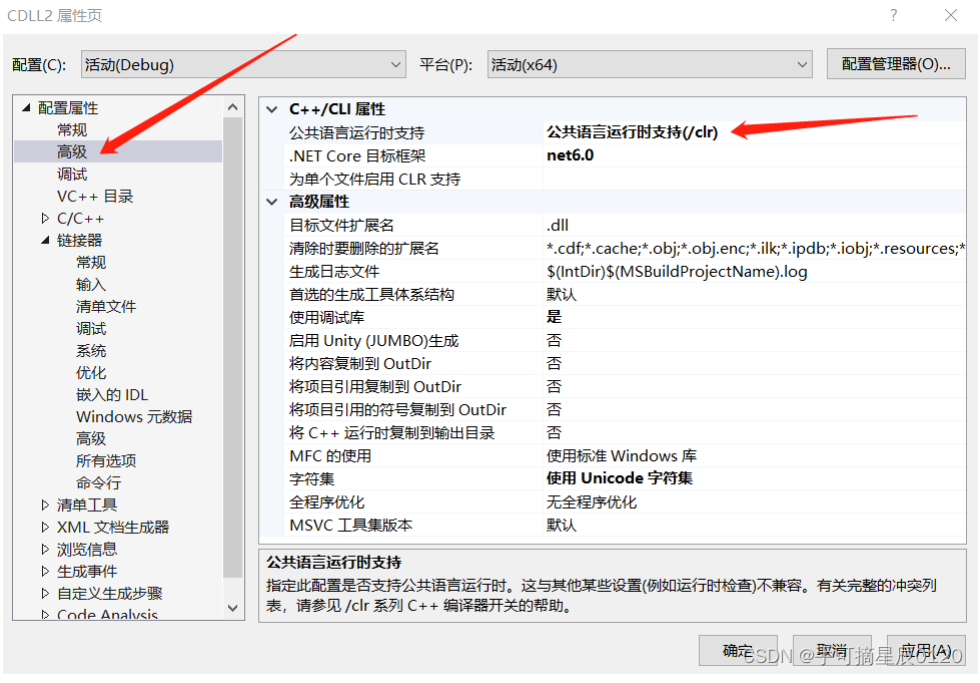
我就暂且放在这里了

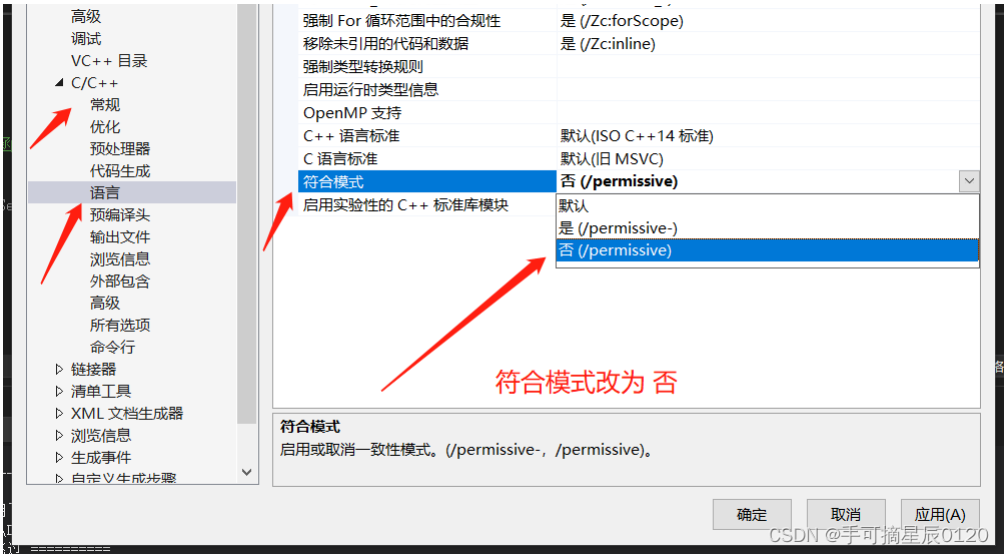


告诉编译器



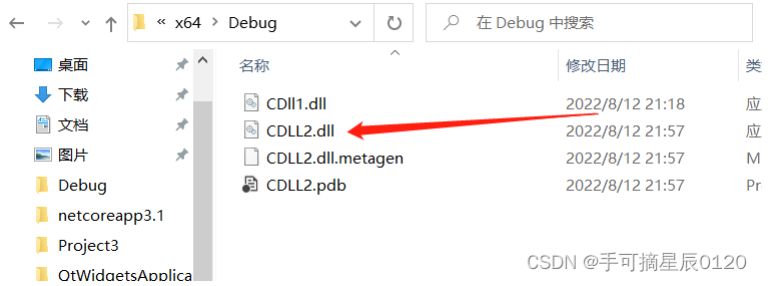
2.4项目属性调整





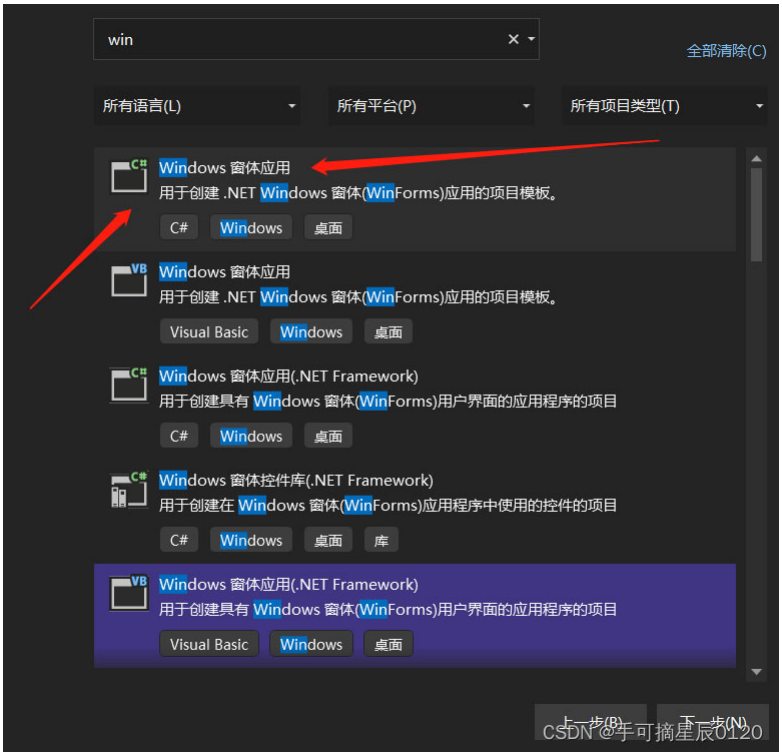
2.5生成托管C++的dll文件

点击生成后，会在类似这个文件夹下 E:\vs2022Code\Dlltest\CDLL2\x64\Debug



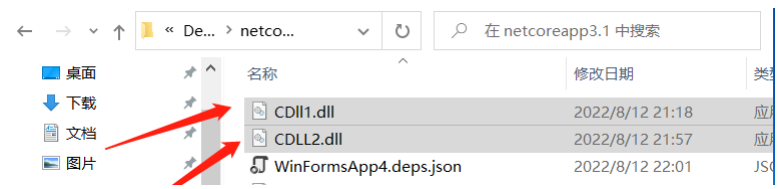
3.在C#项目中调用

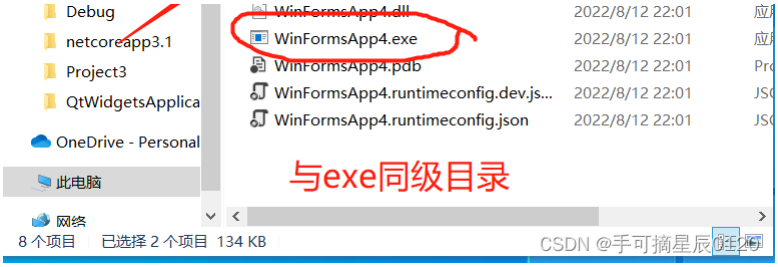
3.1建立窗体项目，并运行一下项目



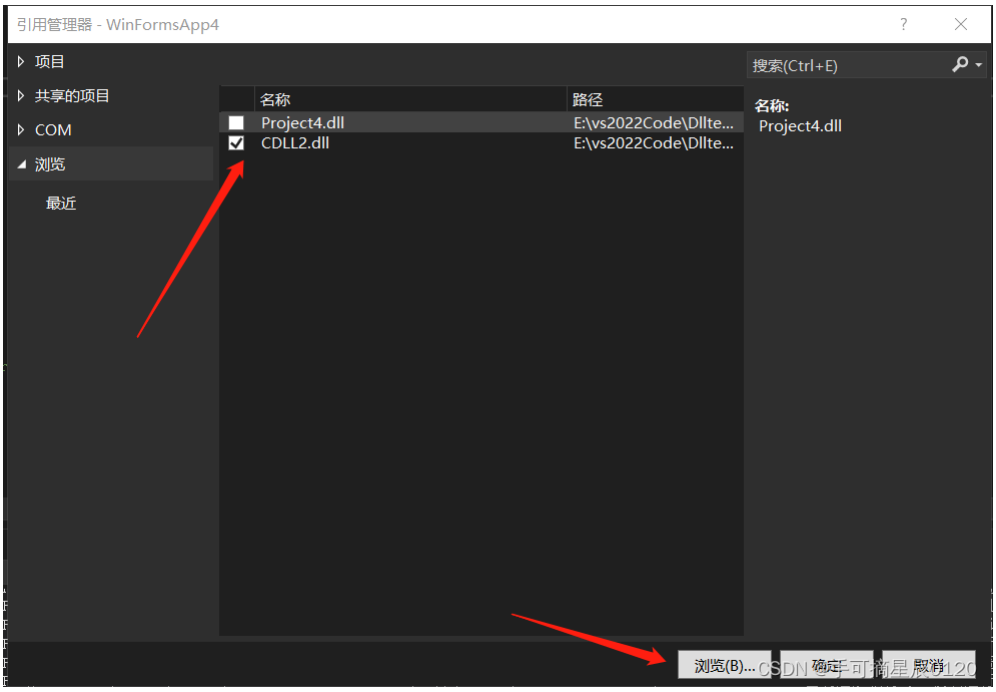
3.2引入前两步生成的dll

将其粘贴到输出目录下，与exe同级





添加依赖项，右击项目依赖项，添加对CDLL2.dll的引用



3.3添加C#代码

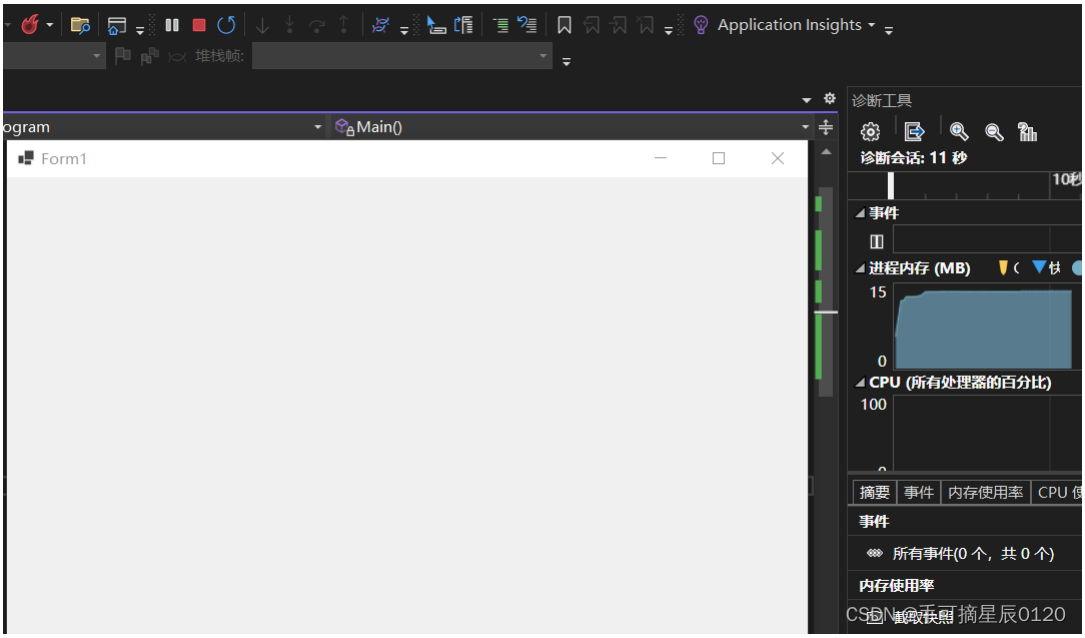
```
using ManageCppDll;


Person person = new Person();


person.Name = "StarLee";
person.Sex = 'M';
person.Age = 28;
```

项目中添加，如下结构

```


using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using ManageCppDll;
namespace WinFormsApp4
{
internal static class Program
    {
```

```
        [STAThread]
static void Main()
        {
            Person person = new Person();


            person.Name = "StarLee";
            person.Sex = 'M';
            person.Age = 28;


            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

3.4运行结果正常



总结：从上面的演示中，我们可以看到C#成功调用了C++的类，这样就C#与C++就可以互相沟通了。