创建动态库

动态库是在程序运行时加载的库文件，并不占用程序本身大小。

选择动态库项目：

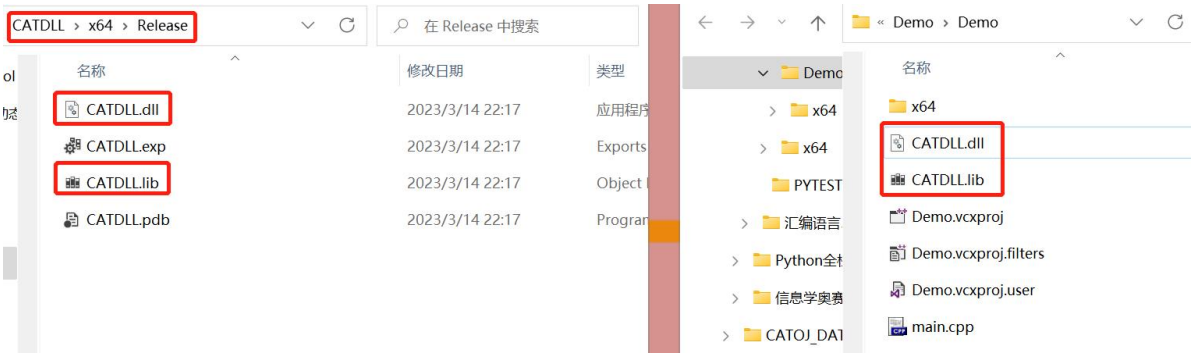
新建.h和.cpp文件：

```
# cat.h
#pragma once

extern "C" _declspec(dllexport) int sum(int a, int b);
# cat.cpp

#include "pch.h"
#include "cat.h"

extern "C" _declspec(dllexport) int sum(int a, int b) {
    return a + b;
}
```

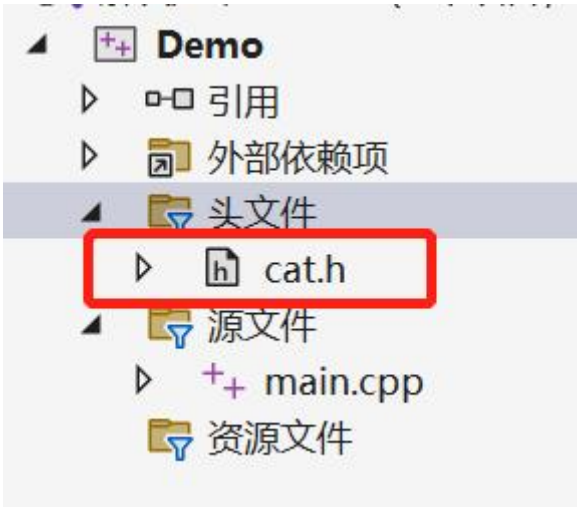
动态库发布选择Release版本，本样例使用×64位。

C++导入动态库方法一

新建一个空的C++项目，将动态库项目中.lib和.dll文件拷贝到当前项目下：



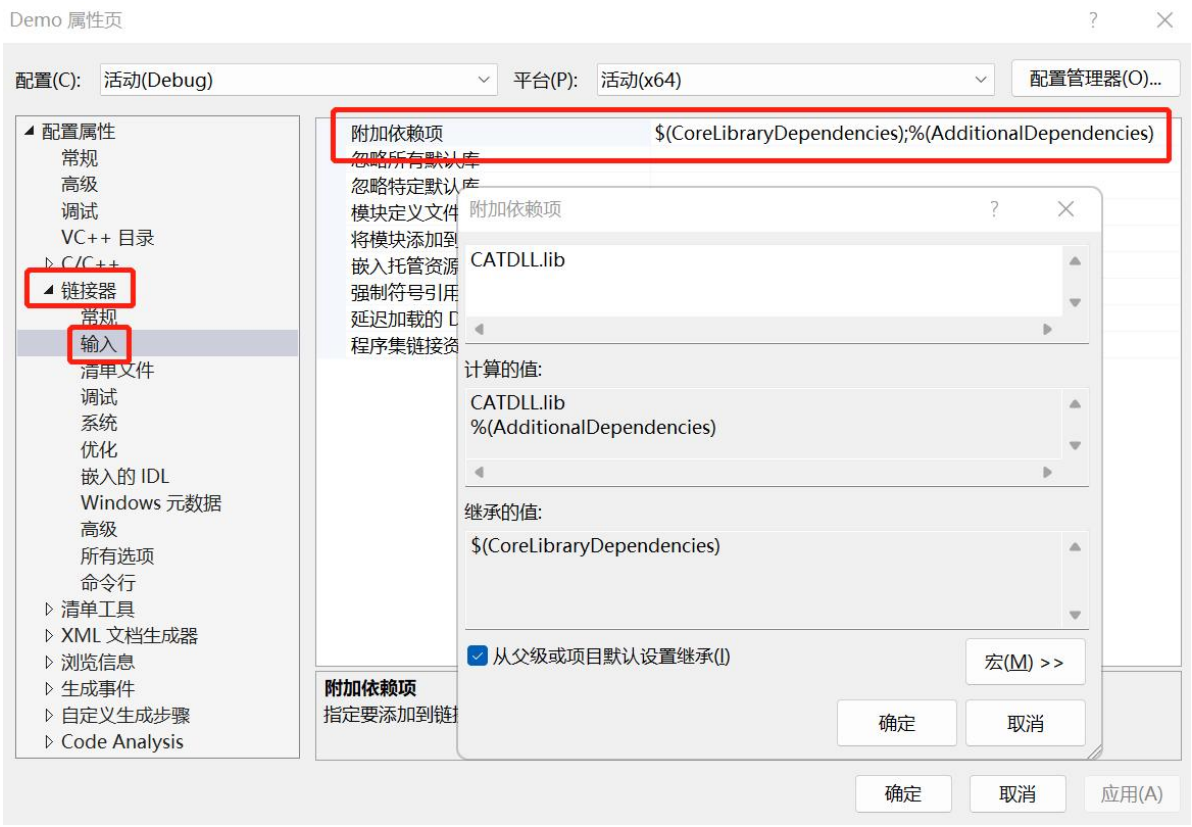
在C++项目中，添加动态库头文件，可以不复制到当前项目下，添加现有项就可以，这里只需要逻辑上引入，但是在#include时要使用.h文件的路径，绝对路径和相对路径都可以。




```cpp
#include <iostream>
#include "../../CATDLL/CATDLL/cat.h"

using namespace std;

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

int main() {

    cout << sum(1, 2) << endl;

    return 0;
}
```

- .h头文件：包含dll中声明输出的数据结构、类、函数等信息。
- .lib库文件：包含被DLL导出的项目的名称和位置，在调用dll的应用程序可执行文件中，存放的并不是被调用的函数代码，而是DLL中所要调用的项目的内存地址。
- .dll动态库：包含实际的内容，发布时只需要.exe文件和.dll文件，在同一个目录下。

也可以在项目-属性-链接器-输入-附加依赖项中进行配置：



可以省略： #pragma comment(lib, "CATDLL.lib")

C++导入动态库方法二

```cpp
#include <iostream>
#include <Windows.h>
//#include "../../CATDLL/CATDLL/cat.h"
using namespace std;

//#pragma comment(lib, "CATDLL.lib")

typedef int (*PSUM)(int, int);

int main() {

    HMODULE hModule = LoadLibrary(TEXT("CATDLL.dll"));

    PSUM psum = (PSUM)GetProcAddress(hModule, "sum");

    cout << psum(4, 5) << endl;

    FreeLibrary(hModule);

    return 0;
}
```

Python导入C++动态库

由于C++ dll是64位，Python也要使用64位。

```python
import os
from ctypes import *

# os.chdir("D:\PYTEST")

dll = cdll.LoadLibrary("./CATDLL.dll")

ret = dll.sum(3, 4)

print(ret)
```

这样，很多常用的功能都可以用 C++ 制作成动态库，供 C++ 或者Python等其他语言进行调用。