**C++ 函数与类的导出**

**一 导出函数（sdk）**

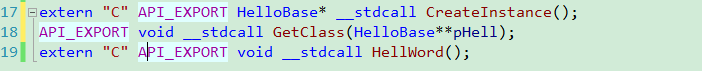
\_\_declspec(dllexport) 导出关键词

\_\_stdcall 调用约定

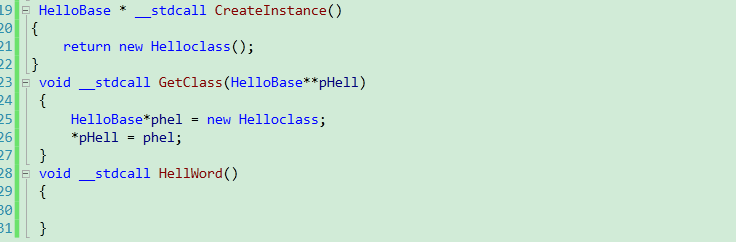
Extern “C” 按C语言进行编译。

.def文件

**.h文件**



.cpp



几点注意：

1.使用depend 观察导出接口名称，发现如果不加def 文件，导出的函数包括这个函数（名称和签名）。这时在动态加载函数的时候，会加载不到.



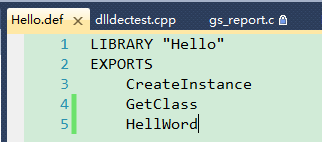
如果加入 extern “C”。会出现 \_xxx@0 这种形式：



在加载函数的时候，名称必须写这个全称才能获取到地址。



只有添加了def文件，才能得到只有名称的导出函数。



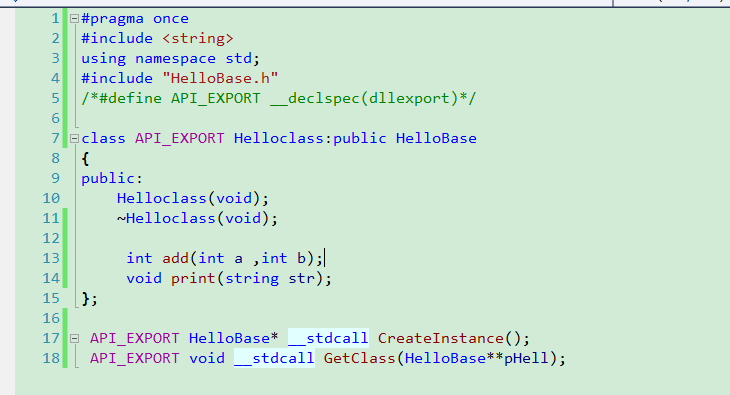


总结：通过关键字加def的方式导出函数是最合适的。

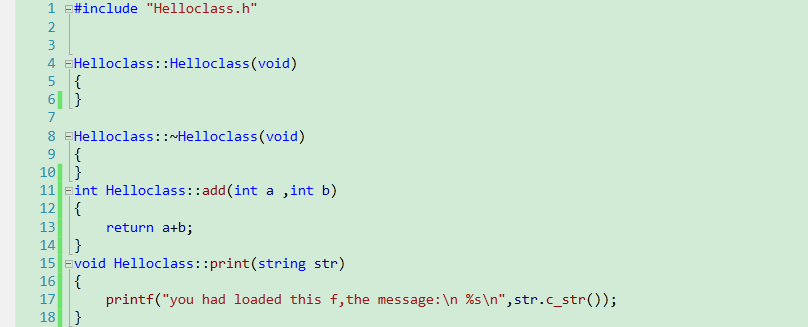
**二 导出类**

**通过关键词 \_\_declspec(dllexport) 导出类。**

**.h文件：**



.cpp文件：



**三 动态导入**

1. 关于函数的使用

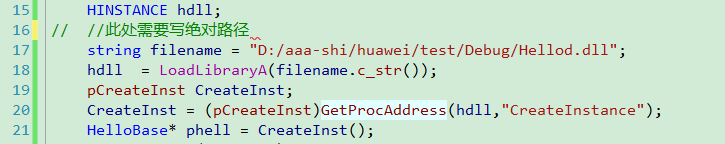
不需要头文件，只需要一个.dll 即可。

LoadLibraryA : 获取库句柄

GetProcAddress: 获取函数指针

定义函数指针： 使用typede 定义函数指针。

Y9(K6N5TW4QB)1DIBHHBXD2



注意：在运行的时候，会加载该目录下的dll。确保Hellod.dll库所需的所有库都能加载上。

1. 关于类的使用

如果确定库需要动态导入使用，一般，不直接使用类。而是通过导出实例构造函数来获取类实例。这样首先动态导入在编译阶段没法获取到成员函数（找不到实现函数，如果是纯虚的就不用去找实现），如果修改库，也可能造成一些其他问题。

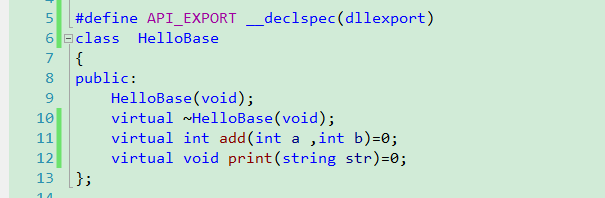
有两种方式：① 定义一个纯虚的基类，在CreateInst里创建子类（导出类）对象，用基类指针去接收，然后就可以调用成员函数。②封装dll的createInst时候，参数用\*\*,&\*。这样可以将实例对象传出来。给基类指针。

见第一节。关于CreateInst的定义。如下图，为两种获取导入类，调用成员函数的方法。

`HOJ8U3H6S}S~XS_8B87HR8



纯虚基类定义：



导出类继承基类。



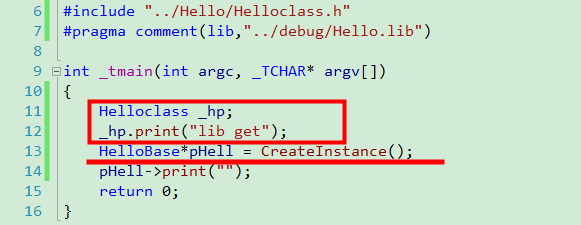
**四 静态导入**

1.关于函数的使用

2.关于类的使用

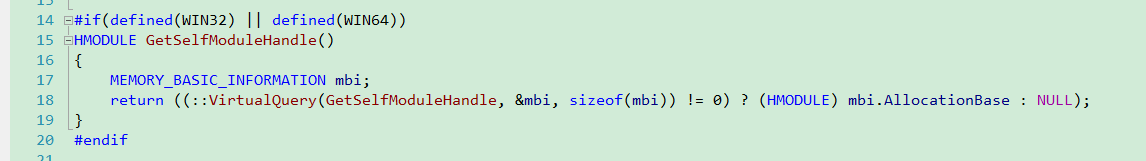
#pragma comment:

关于静态库的使用，只需包含头文件，和加载lib库就能正常使用导出的函数和类了。



五 模板类导入

六 获取当前所在模块的句柄。



调用并加载模块中的函数。



1. Linux下动态加载so动态库
2. 定义函数指针

包含头文件#include <dlfcn.h>



1. 

