【技术向】云评测基础——游戏数据的获取方法

作者 Moemod Hyakuya（杰拉德临死前）



食用本教程需要有强烈的计算机技术背景，尤其是针对C语言和C++语言的了解。另外可能需要x86汇编语言的功底。

# 配置环境

主要工具：IDA Pro

至于如何获取这个软件嘛，它并不是免费软件。建议联系Hex-Ray公司购买正版软件，并在电脑上安装然后继续接下来的操作。

参考源码：CS16ND或ReGameDLL

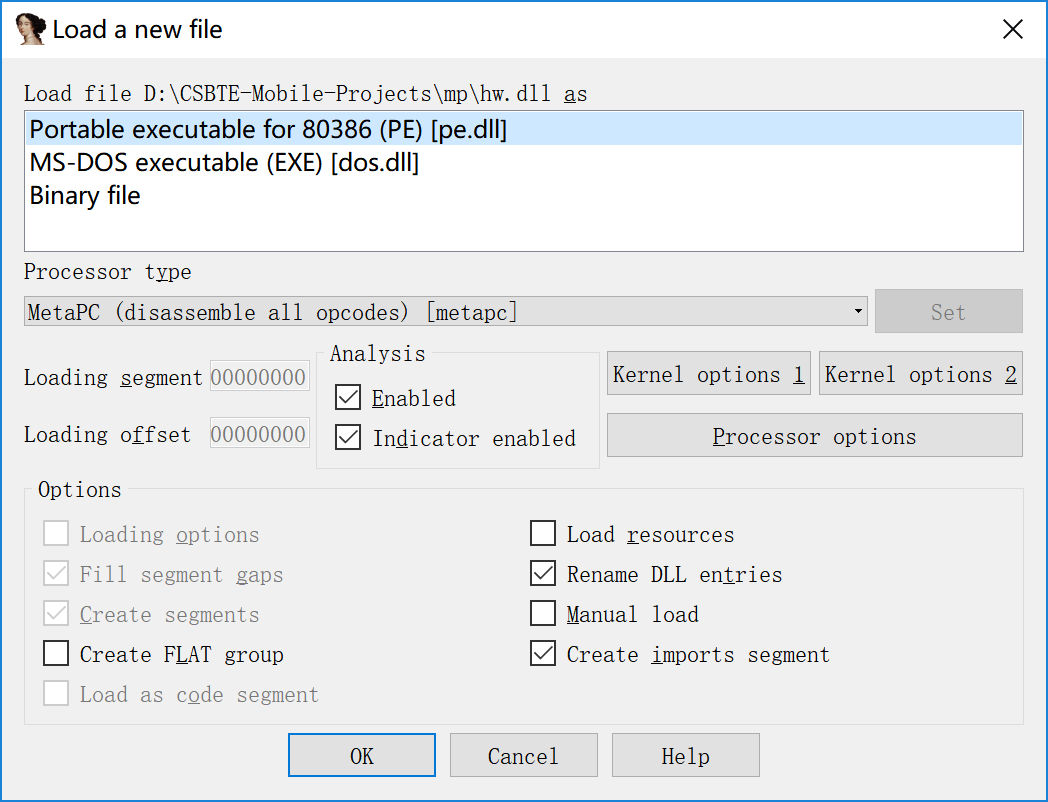
在全球最大的同性交友网站上可以找到。

偏移信息：csnz-sdk

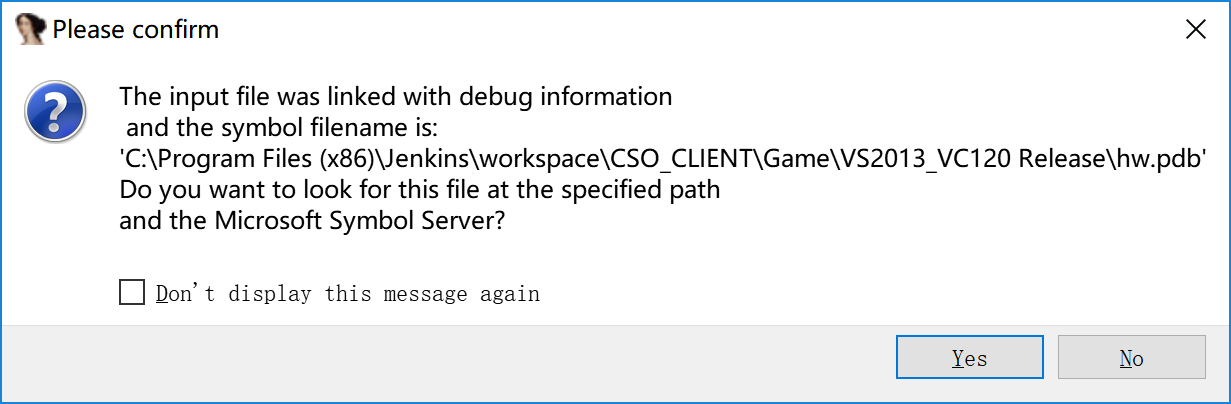
和上面的一样。

# 载入Dll文件

找到需要查看的合适的mp.dll和hw.dll文件，然后使用IDA Pro打开它们。

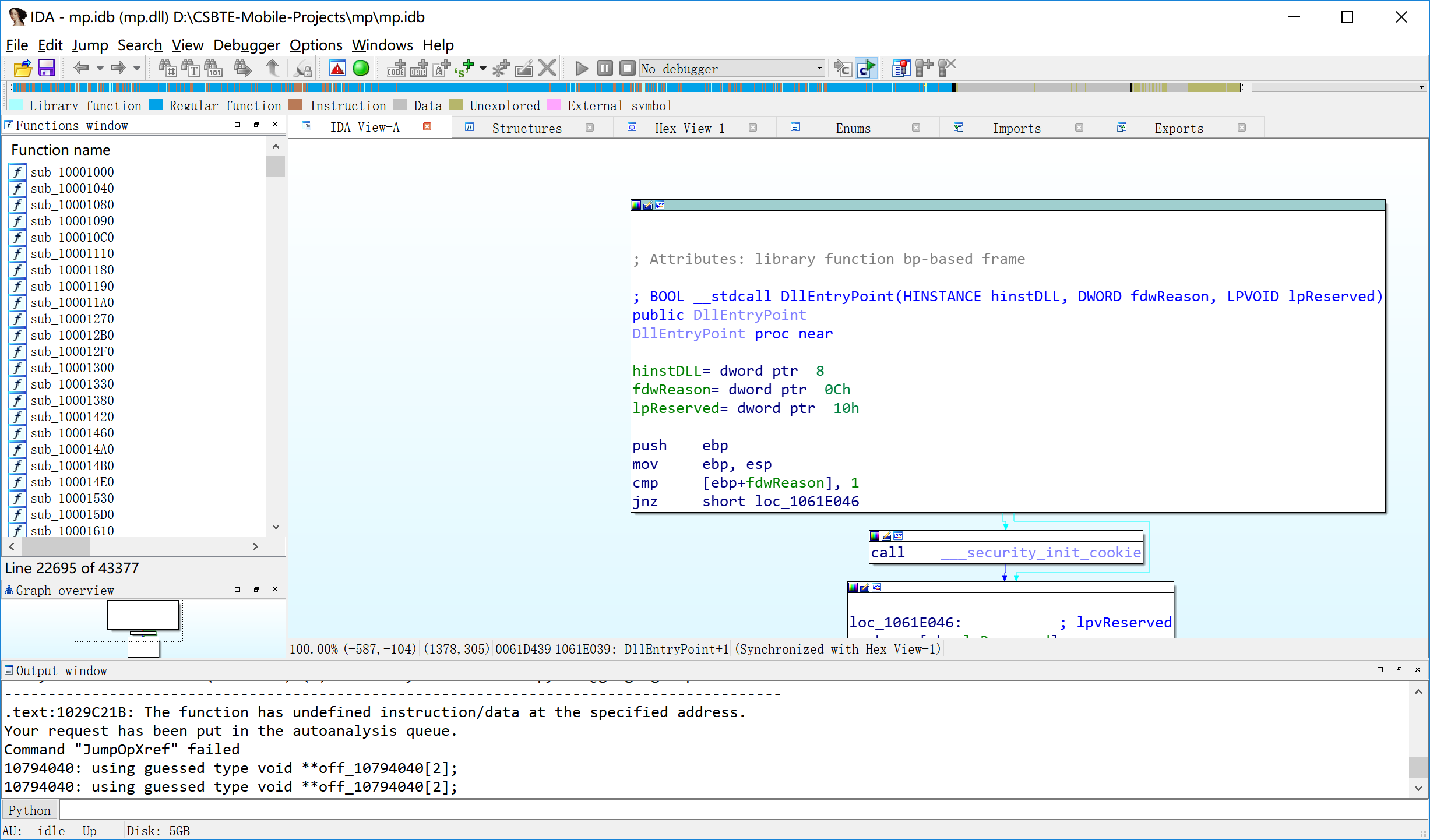


大概会出现这样子的窗口，直接点击OK。

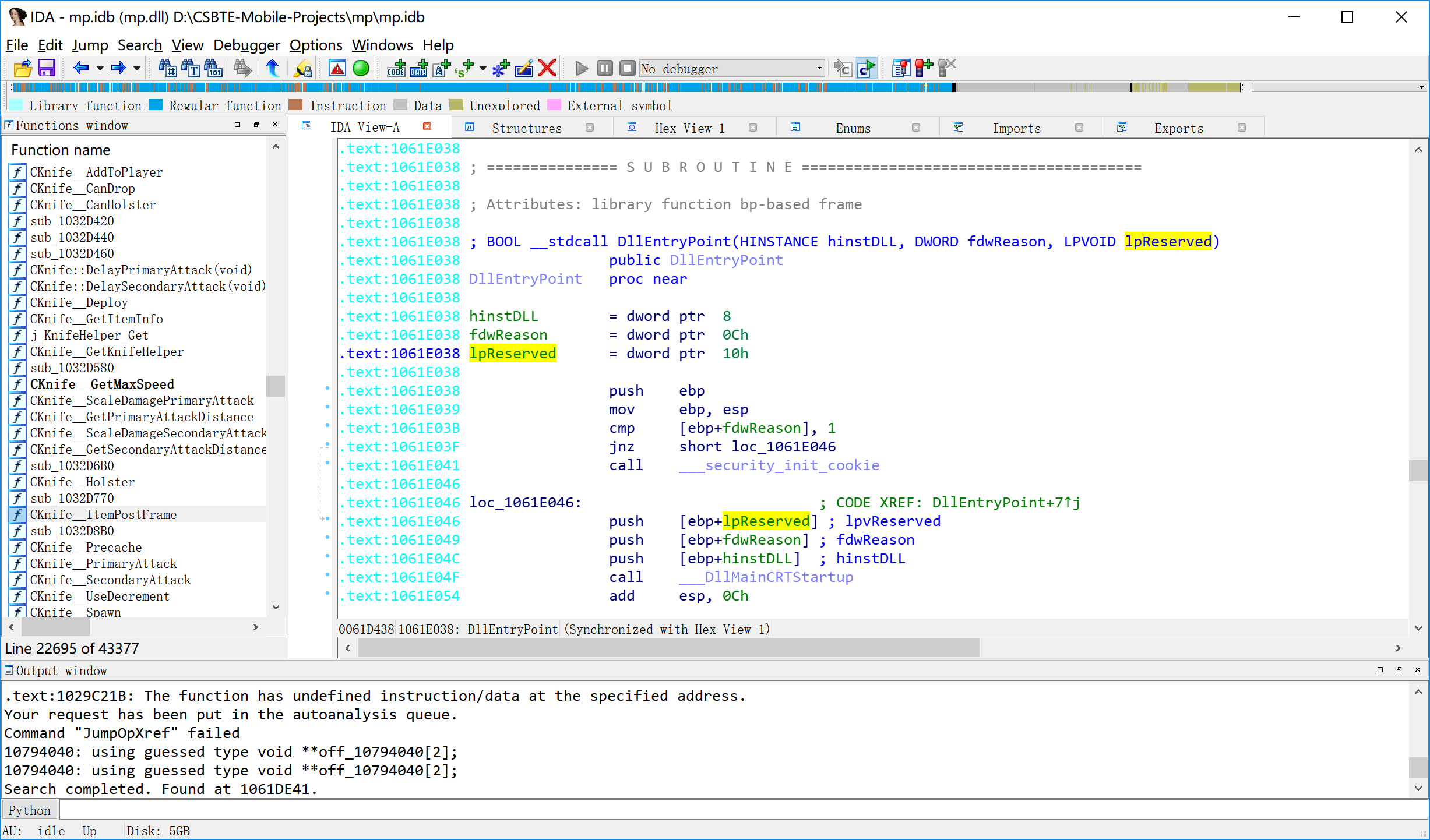


若出现了这样的窗口，选择No即可。

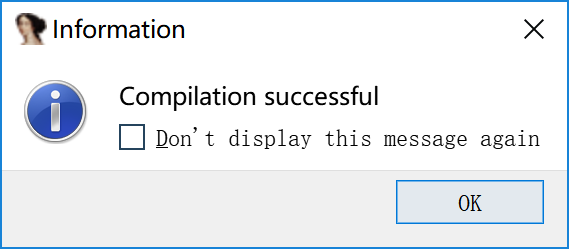
然后看着左下角的数字会不停地滚动，最后大概会看见这样子的窗口，表示dll加载完毕。



右键打开菜单，选择Text View，让观察的界面变得正常一点。

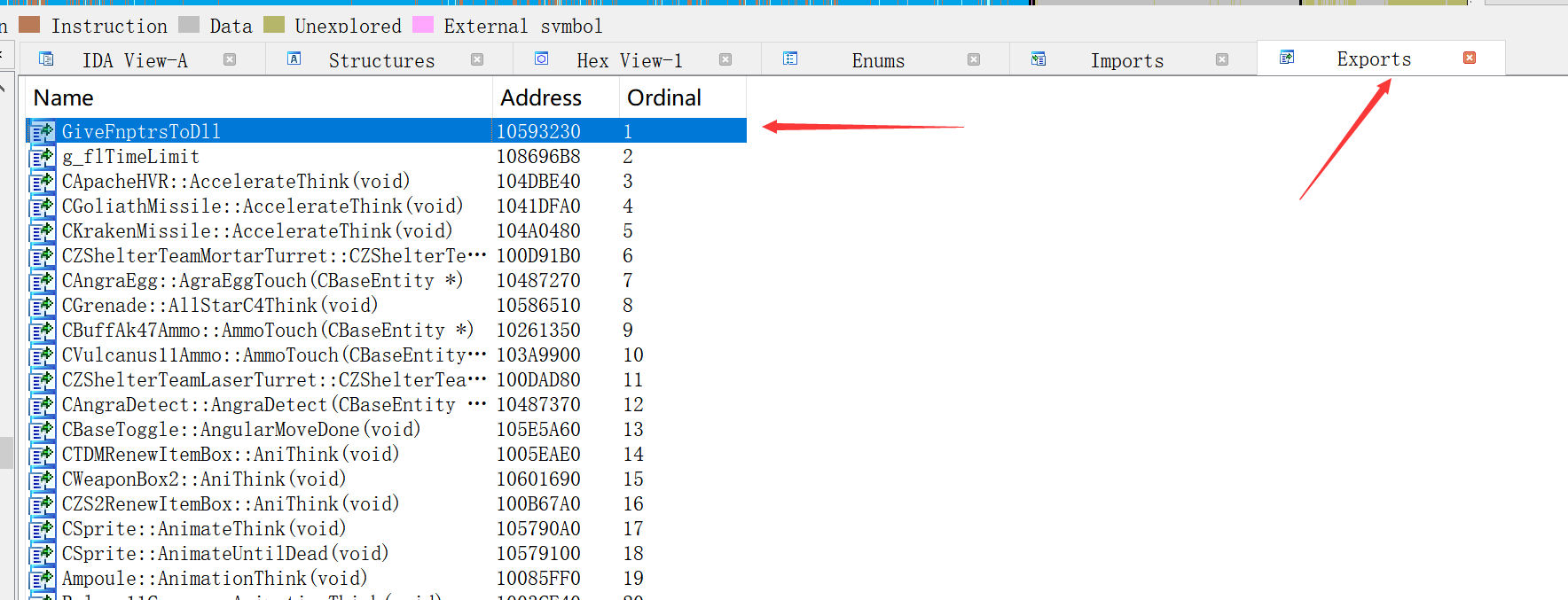


此时是使用黑魔法的时刻。按下Ctrl+F9，选择csnz-sdk.h（在csnz-sdk里面有提供），如果没有意外的话会看到这样子的小窗窗



# 全局变量

首先需要按照步骤1的方法打开mp.dll文件。（注意不是hw啊喂）

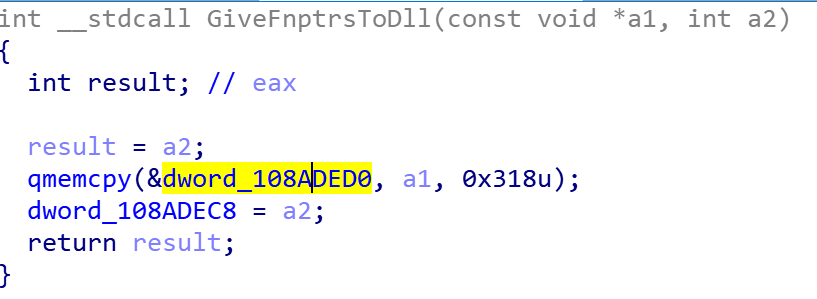


切换到导出函数表（Exports），找到GiveFnptrsToDll函数，按下回车跳转到对应的位置。

什么？你不知道什么是导出函数？嗯没关系，百度会告诉你，虽然知道了也没有什么用。不过，可以理解为，导出函数就是已经有名字不需要去自己命名的啦。

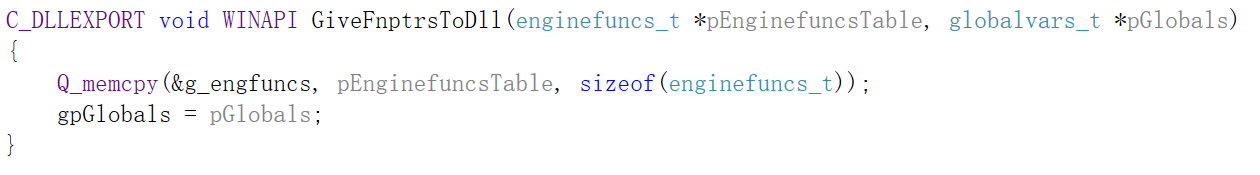


乌拉！如果你学过x86汇编语言的话应该可以明白这些是干什么的啦。咳咳……还是使用魔法比较好。请按下你的F5键。



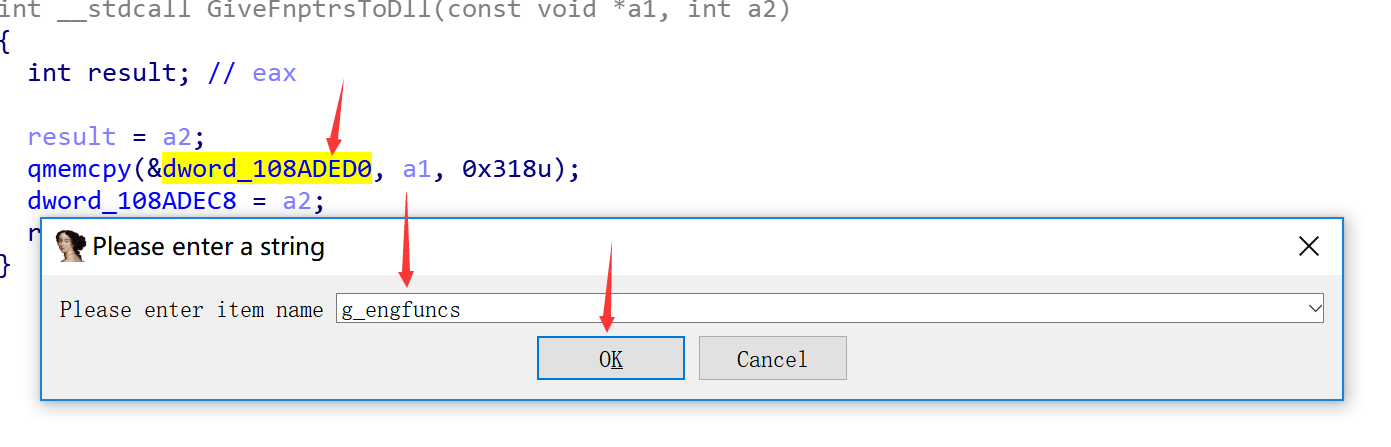
于是它被“翻译”成了C语言。

如果熟悉C++的同学应该可以在h\_export.cpp里面找到对应的函数（在cs16nd里面）

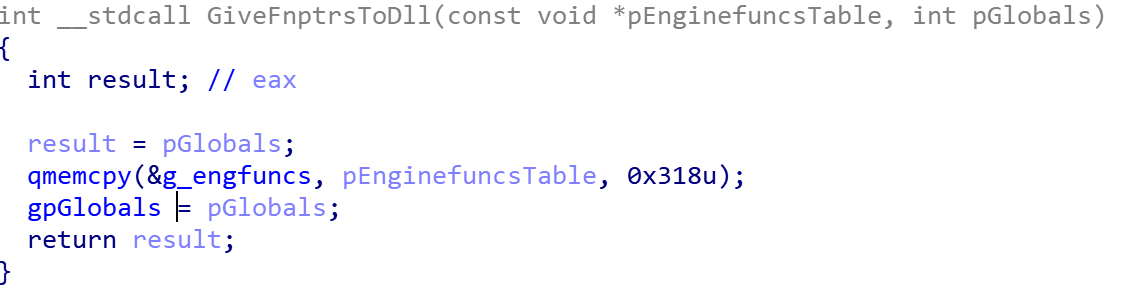


这时候就要展现充分的抽象思维能力了。嗯嗯，脑补一下图片上指向的这个东西应该叫做g\_engfuncs，那么就用鼠标点击它并且按n，改名为g\_engfuncs

这里的g\_engfuncs和gpGlobals我们把它称为全局变量，至于是干什么的任何一本C/C++语言的书籍应该有谈到。（除了Java和C#等邪教没有！）



同样的，你可以把你看得懂的所有变量都改名。结果大概是会变成这样子。

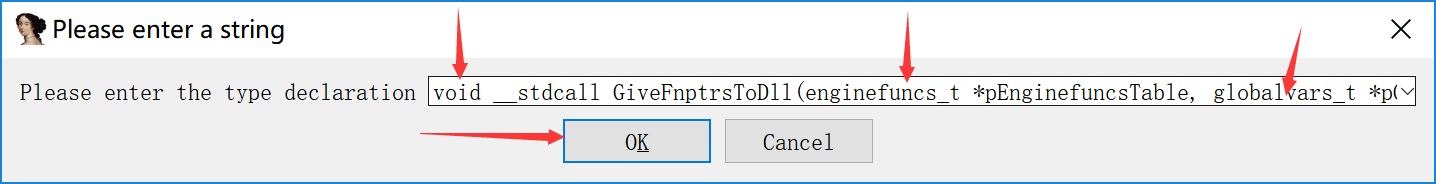


接下来，又是利用魔法的时刻。刚才我们载入的头文件包括了这些结构体的声明，我们可以用y键更改变量的类型。如

const void \*pEnginefuncsTable

改为

enginefuncs\_t \*pEnginefuncsTable



另外千万记得，以下很重要：

把g\_engfuncs改为enginefuncs\_t类型（结构体变量）

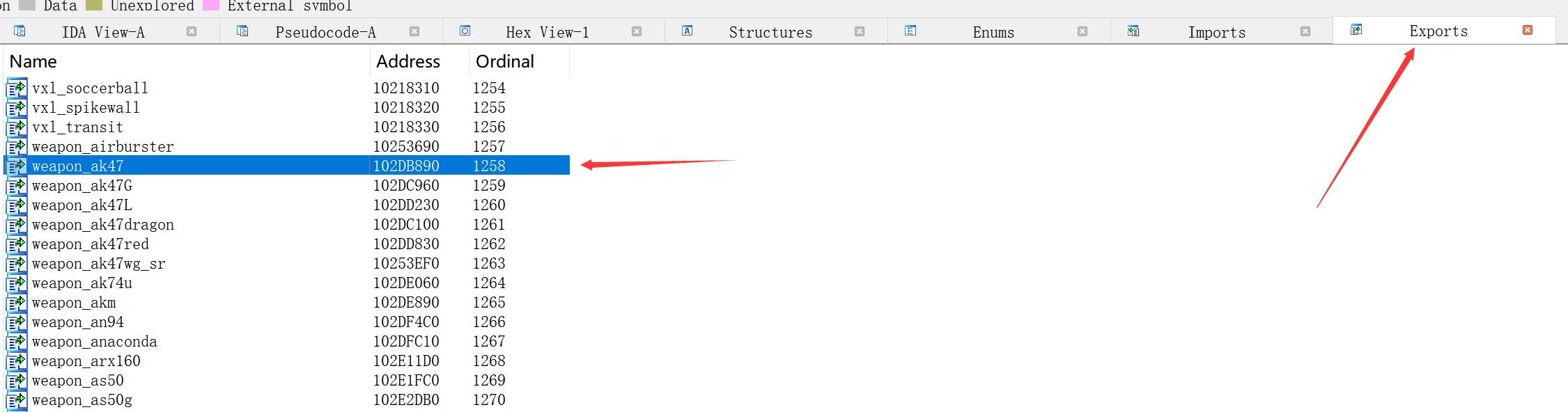
把gpGlobals改为globalvars\_t \*类型（指向结构体变量的指针）

# 武器相关函数

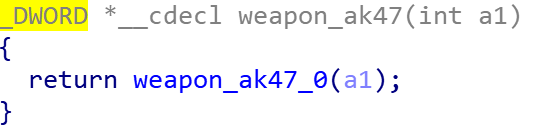
进入本部分前，请确保你已经掌握了按f5生成魔法代码、按n为变量或函数命名、按y更改变量或函数的类型的高科技。

那么你应该可以了解了，我们所做的是由已知（cs16nd）推理未知（某mp.dll）的过程。

这部分我将会带你从基本的武器（ak47）开始了解武器代码的构造。



首先在导出函数表（Exports）找到weapon\_ak47跳转并F5



然后大致会出现这样子的场景。没关系，继续跳转。



玄学科普时间。

这个函数有必要了解一下。如果你学习过C++可能知道，虚函数是体现多态性的重要方式，而这种方式大部分需要用虚函数表来实现。

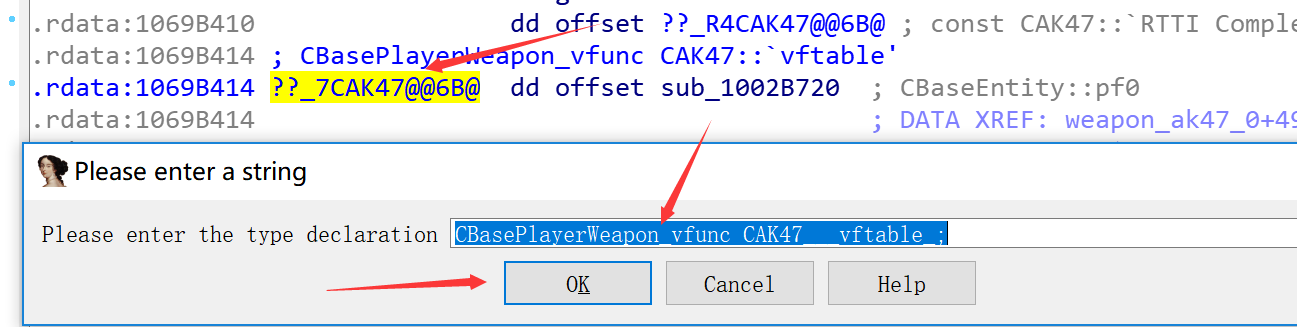
即对于同样的武器实体，他们有着共同的基类CBasePlayerWeapon，而各自对于基类的虚函数做出了重写（override）的操作，使得使用基类指针或引用调用这些虚函数（方法）时得到的是不同的实现。

而使用虚函数表的方式将函数调用变为索引查询，即C++代码中的pWeapon->Reload()会被翻译为C语言中的pWeapon->vfptr[142](pWeapon)。

那么问题来了，为什么刚才的函数调用明明是没有参数的，却凭空多出来一个呢？答案是需要向成员函数（方法）传递this指针指向当前对象。具体请参阅C++入门教程blabla……

扯远了，这个函数我指向的部分是CAK47的构造函数，编译器会在生成代码时对构造函数隐式添加虚函数表指针（vfptr）赋值的代码，于是我们可以在这种构造函数中找到虚函数表的位置。基于CAK47由CBasePlayerWeapon派生而来，我们将对虚函数表里面的所有函数进行命名。

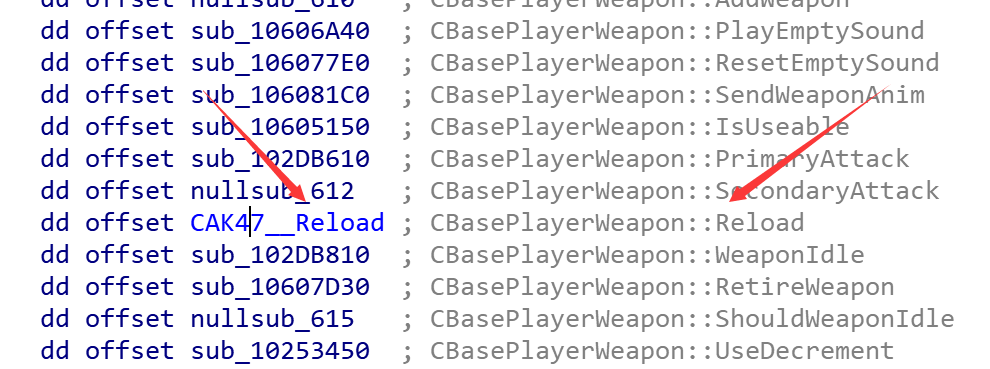
首先跳转到我指向的CAK47::`vftable'，然后都对它指定类型CBasePlayerWeapon\_vfunc。



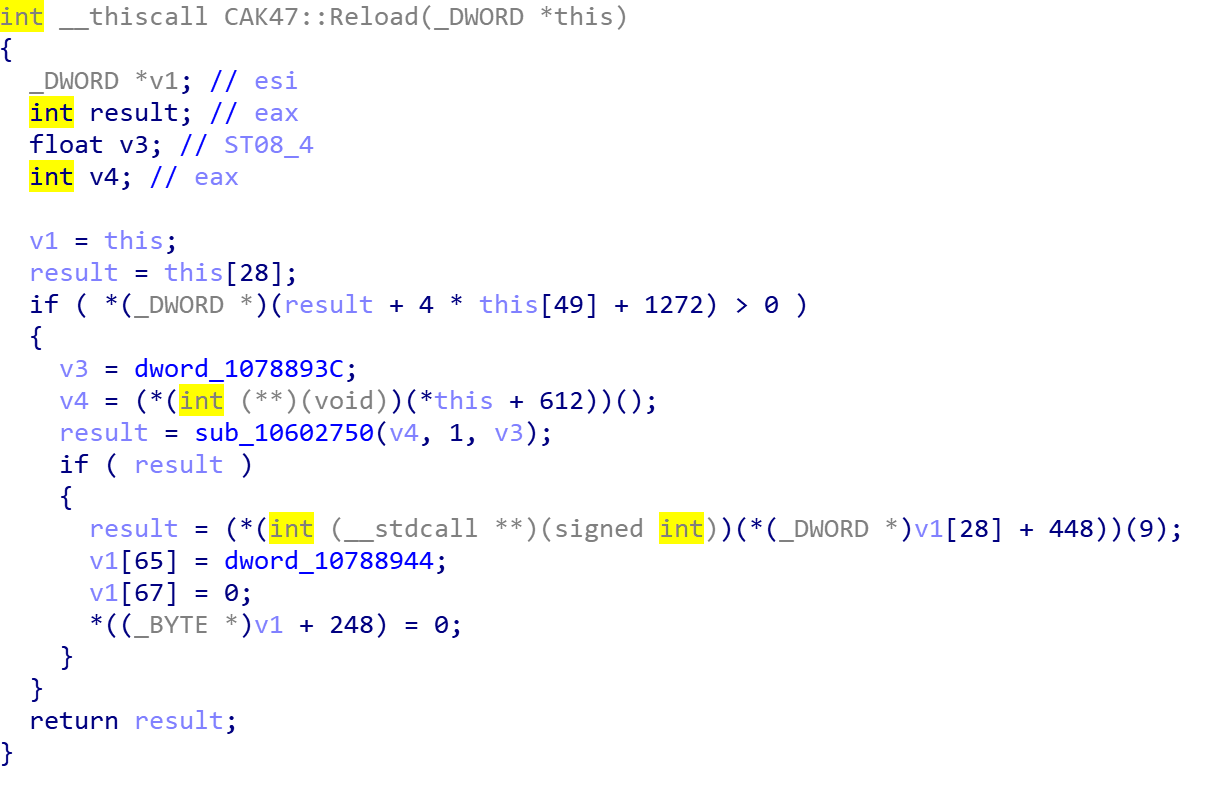
于是可以看到所有的函数都被标了出来。



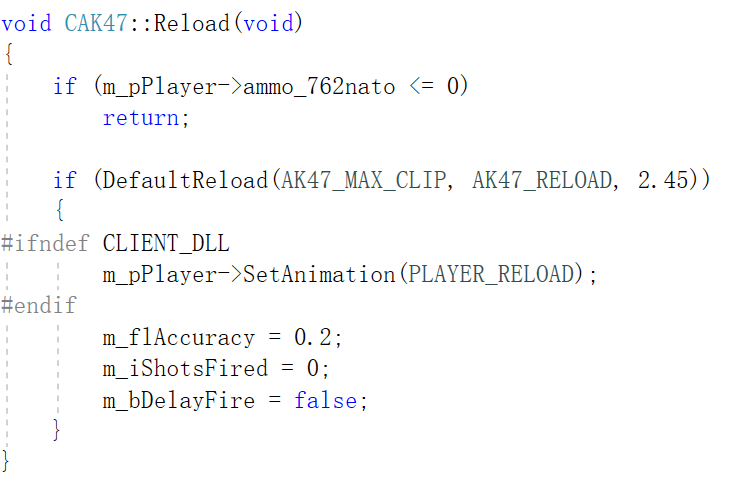
往下拖动，找到Reload函数，然后将它命名为CAK47::Reload。



跳转进去并且F5

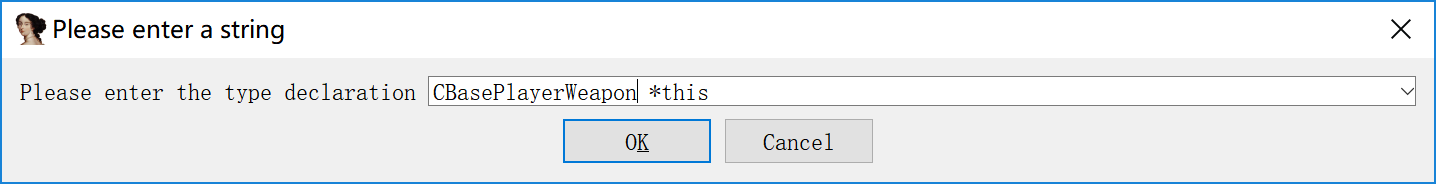


这是什么东西QAQ 没事我们可以用cs16nd的内容作为对比（wpn\_ak47.cpp）



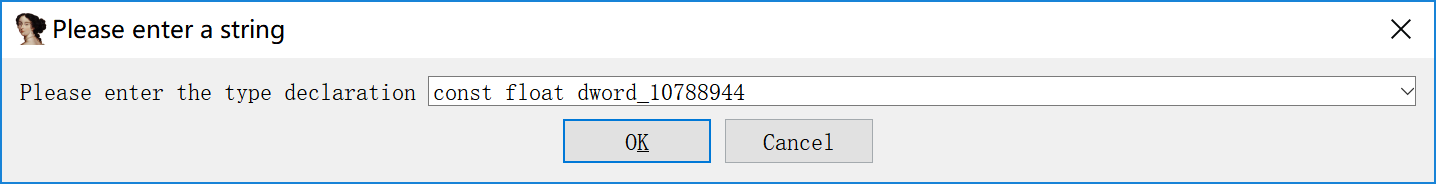
教练！看起来这个函数和刚才的GiveFnptrToDlls不一样啊喂……没事，我们需要使用魔法。

因为函数没有正确的得知其类型，所以请按y把this的类型改为CBasePlayerWeapon \*





咦好像……有点一样嘛。我们需要的关键数据在这里，我们可以知道它们都是浮点数，所以照例，按y更改类型为const float。注意必须是const float，IDA才会把它显示出来。





可以看见结果已经很相似了。我们可以推测在CSOL中AK47的换弹时间是2.45秒，换弹后的精度调节值（m\_flAccuracy）为0.2。

接下来请你根据已有的知识验证AK47的射速、命中公式、后坐力等参数。