工程地址：E:\proj

生成：

重新生成：

生成解决方案：有些库依赖以及顺序之类的

编译：只对cpp或者

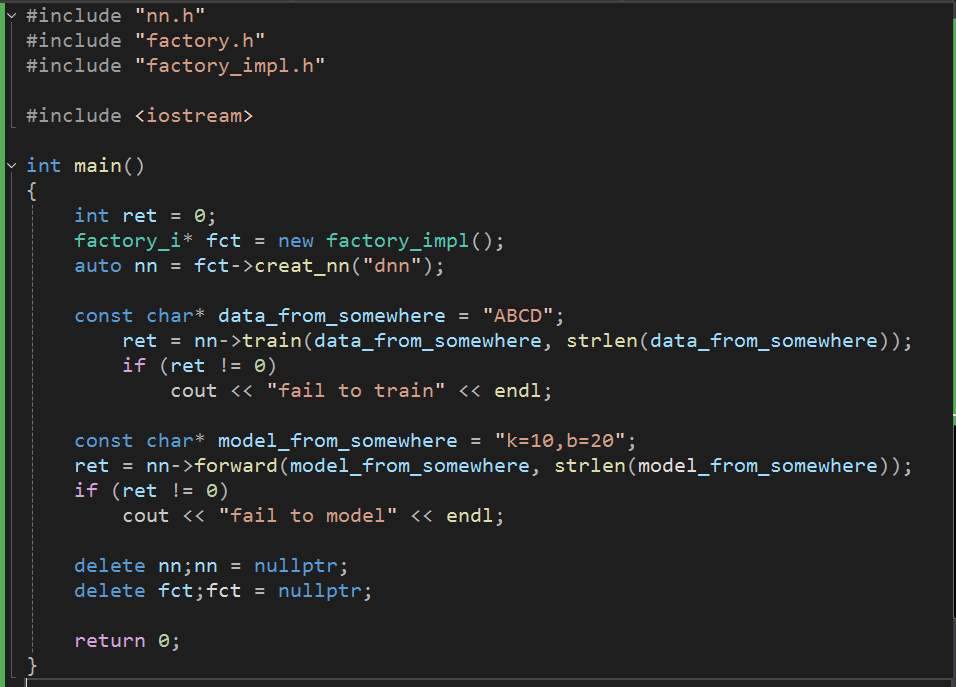
Debug vs release

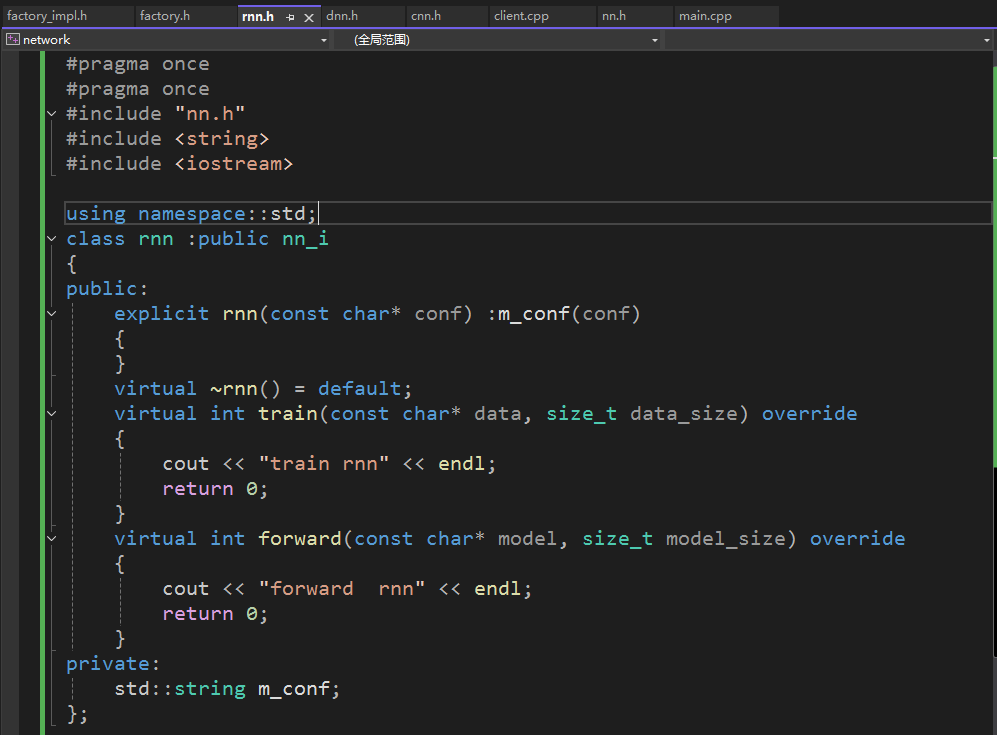
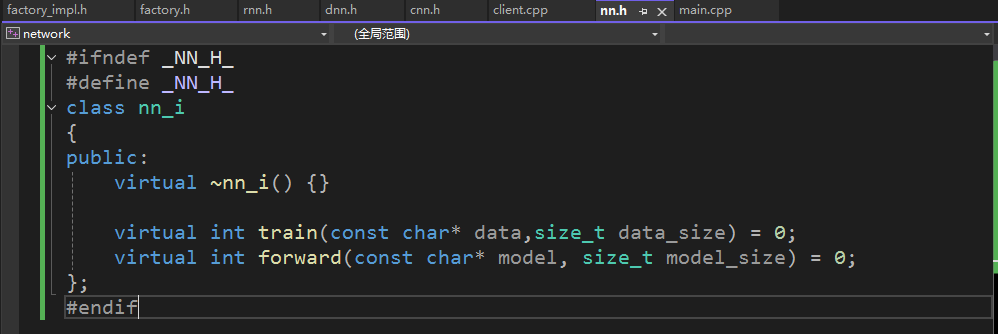
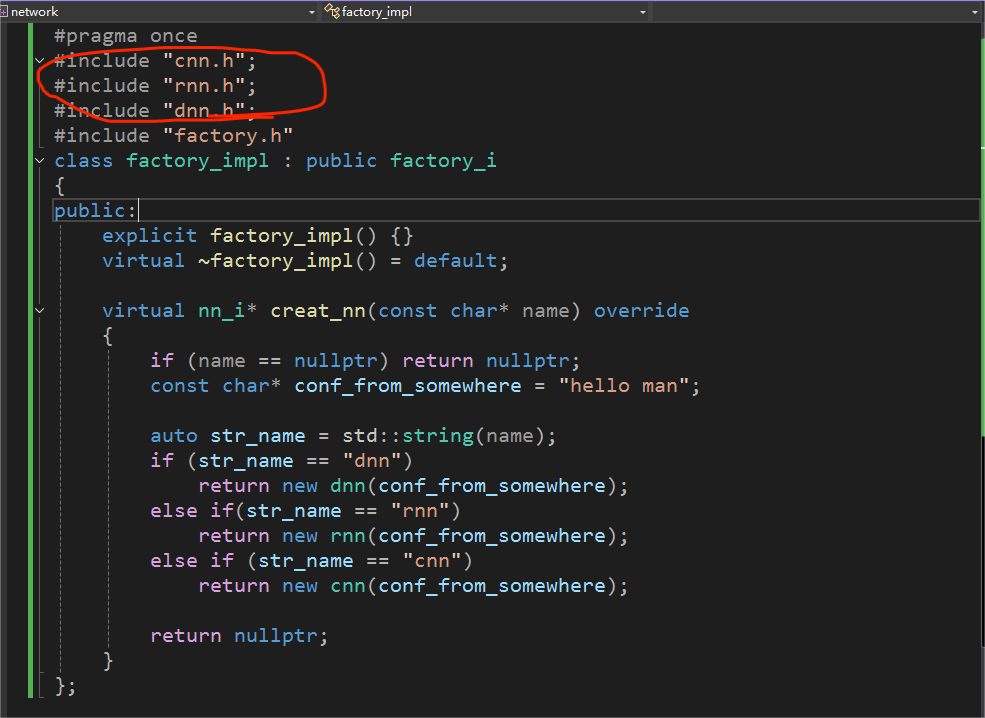
Debug vs release x86 x64什么的，比较繁琐，使用cmake会比较方便

属性-配置属性-配置类型{1.exe 2.dll 3.lib}

简单工程组织： 生成 - 生成解决方案 - 调试 - 开始执行不调试

可以发现main函数不直接依赖于cnn.h等三个文件，但是间接依赖





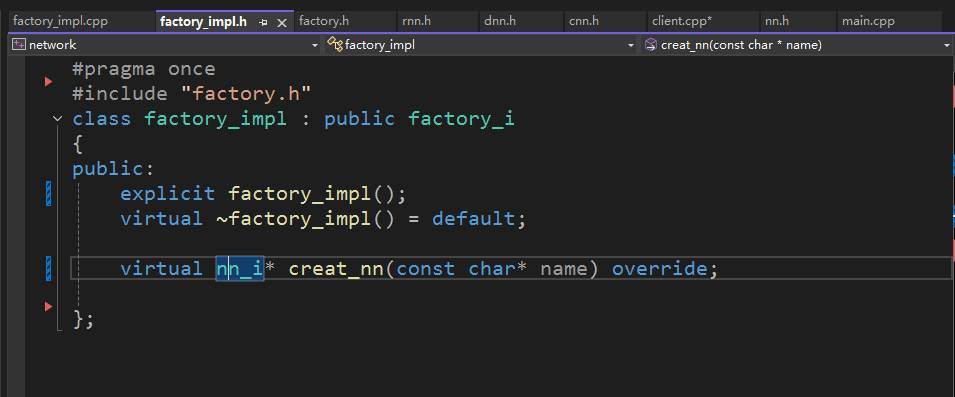
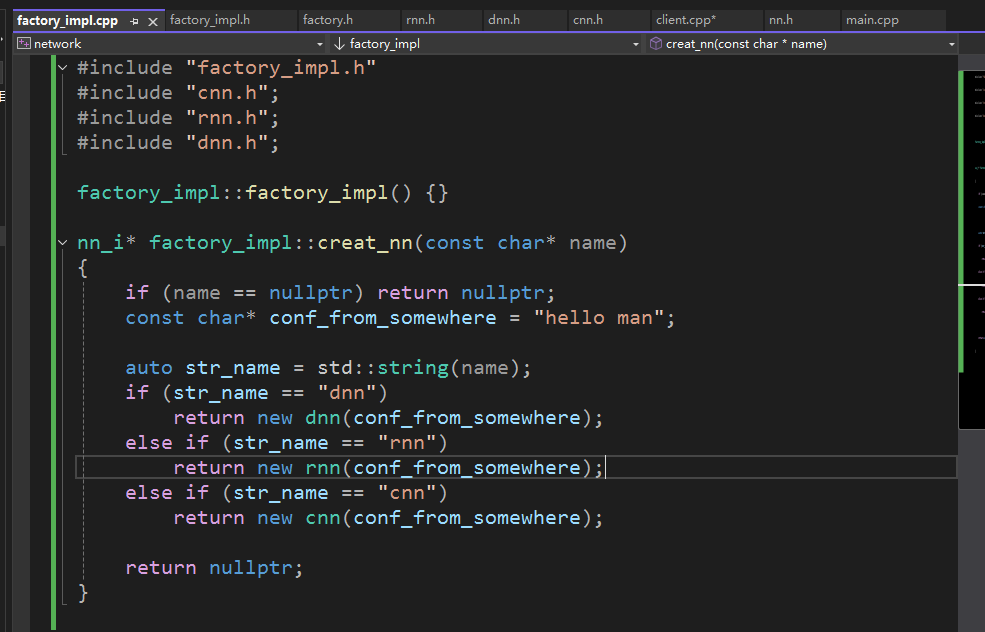
把factory\_impl.h里的代码实现放到factory\_inpl.cpp中去，把

#include "cnn.h";

#include "rnn.h";

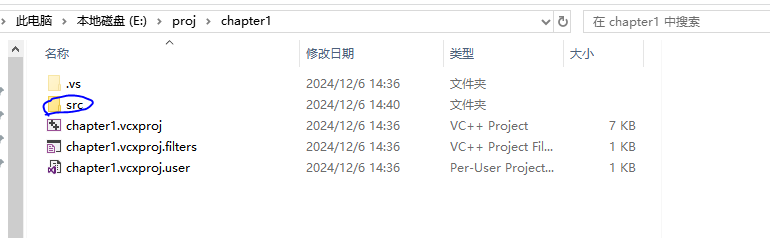
#include "dnn.h";

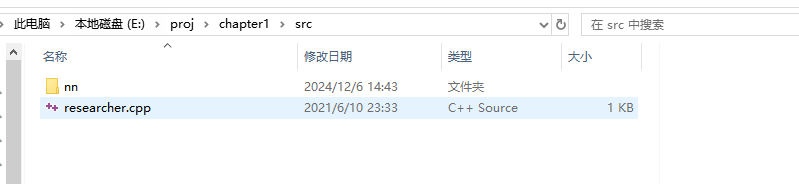
也放到factory\_inpl.cpp中去，client将不在依赖于这三个头文件

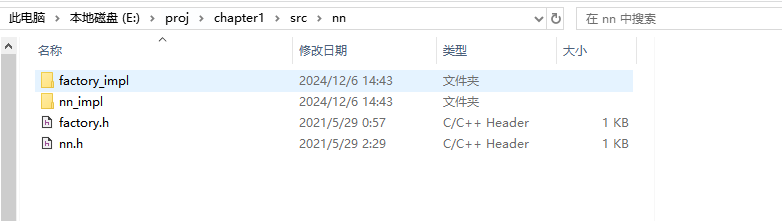


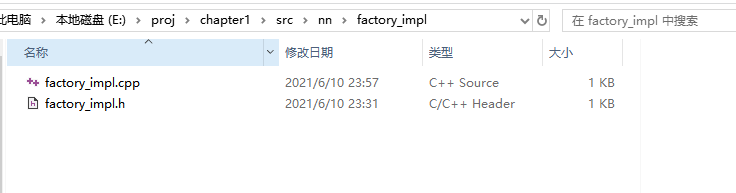
此时Client只依赖于factory\_impl抽象类和nn\_i接口，但不依赖于具体的神经网络，因为神经网络是在factory\_impl.cpp里面。只要factory\_impl.h的逻辑和内容不发生变换，则影响不到client。

# 文件夹与工程组织

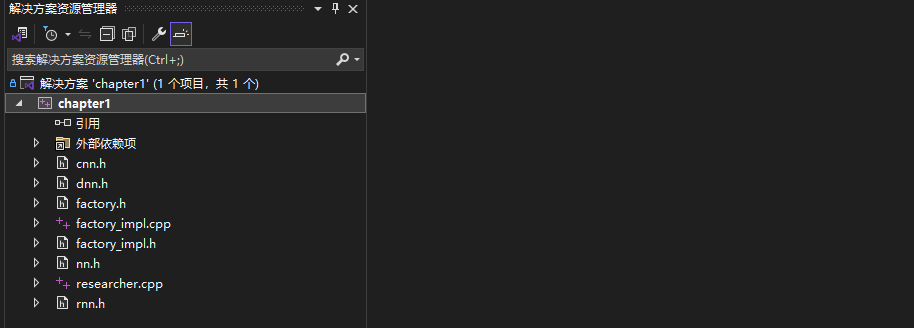




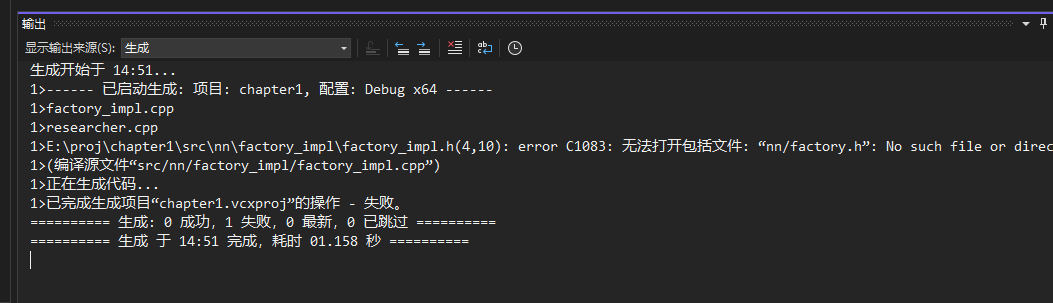




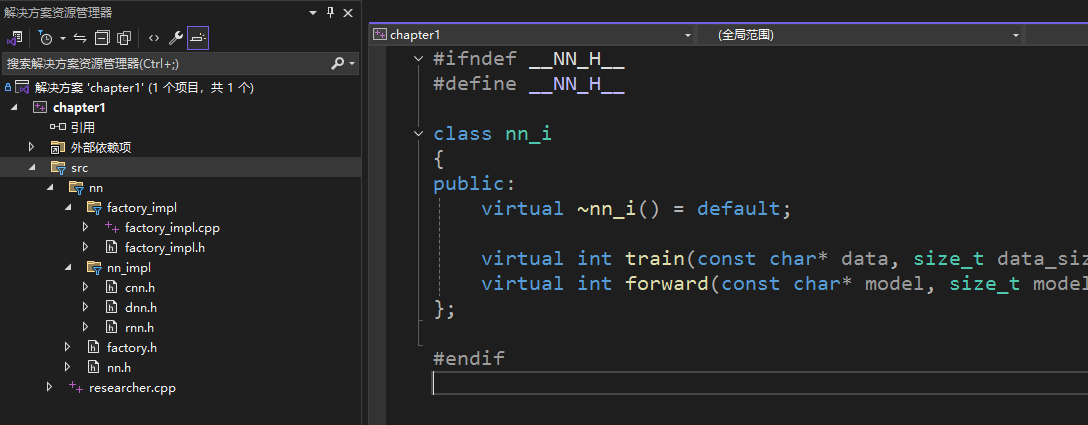
Vs中添加-现有项



生成-生成解决方案会失败

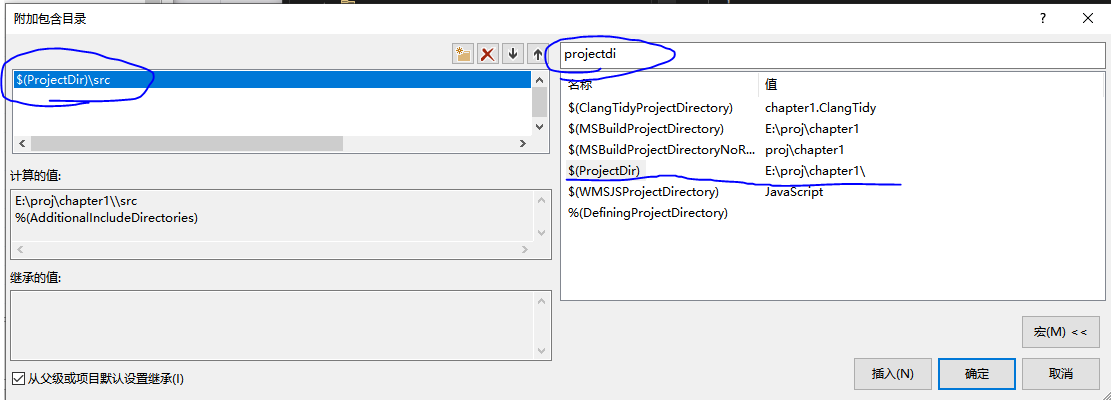


添加过滤器，保持和文件夹一致

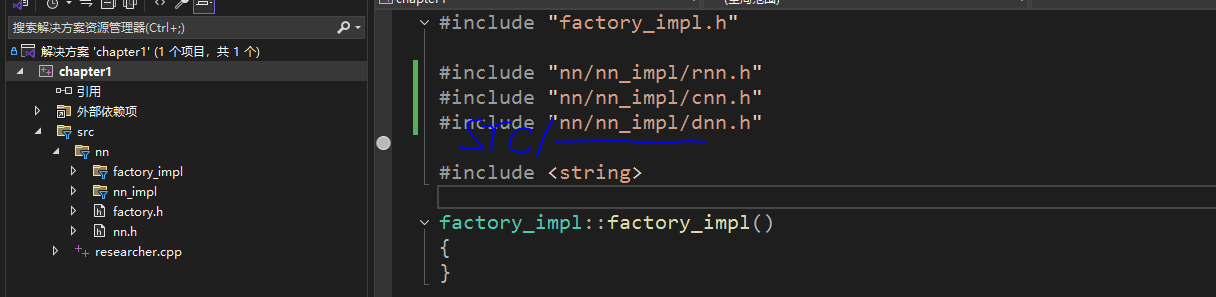


生成-报错-双击报错（一般是头文件路径的问题）

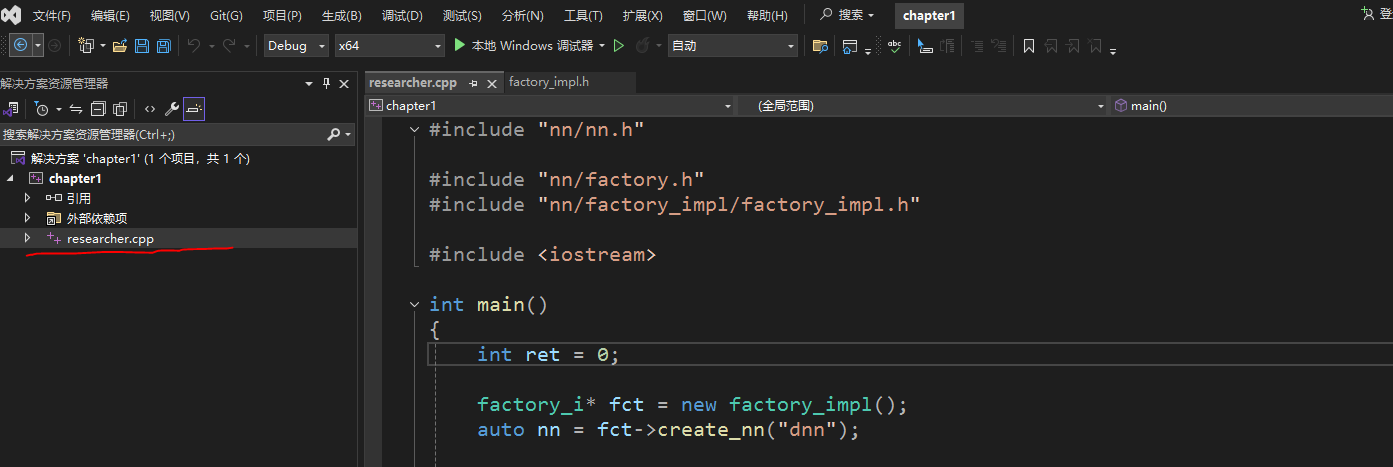
右键工程-属性-C/C++ - 常规 - 附加包含目录 - 可以使用宏来定位位置（设定基础位置，要和debug，x64对应）



生成 - 根据报错修改头文件的路径

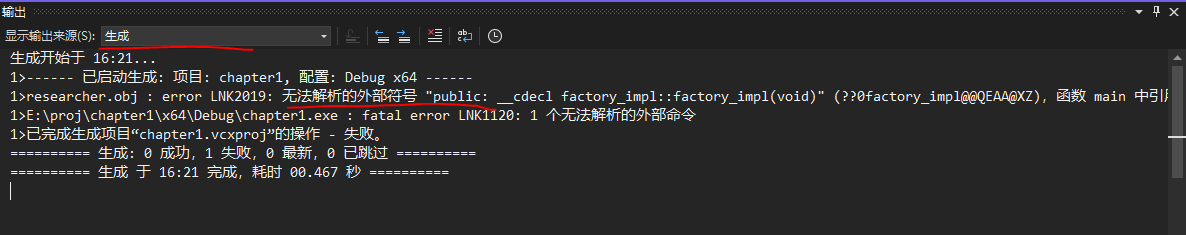


# 静态库工程组织

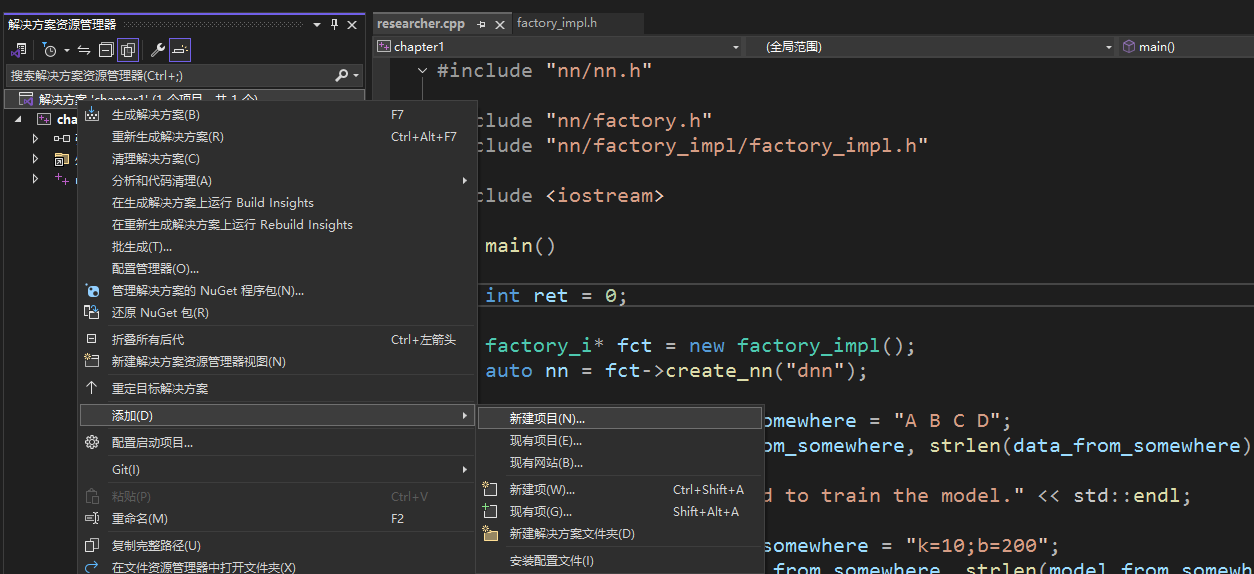


此时ctrl+f7 编译不会有问题（从cpp到obj）

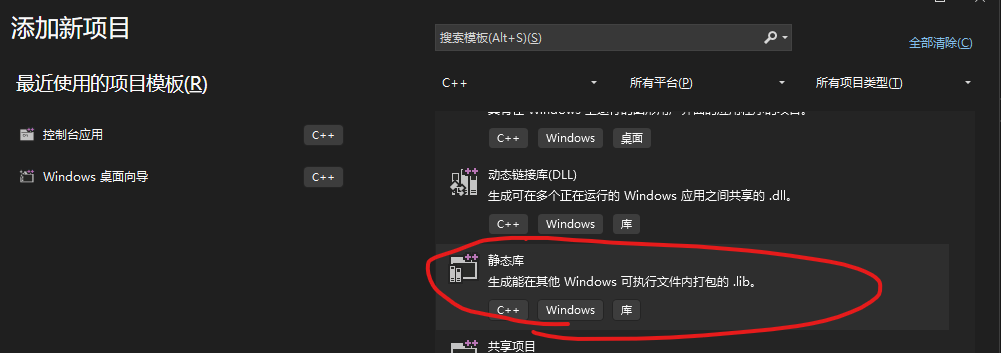
生成会报错（生成的过程会有链接）



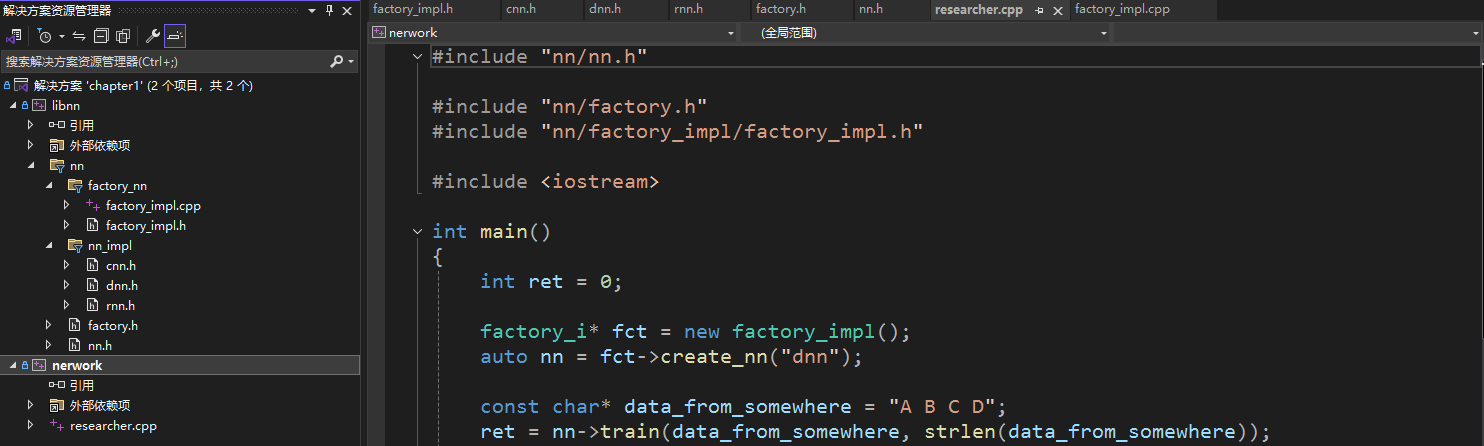
下面开始把nn相关的建成一个lib库：



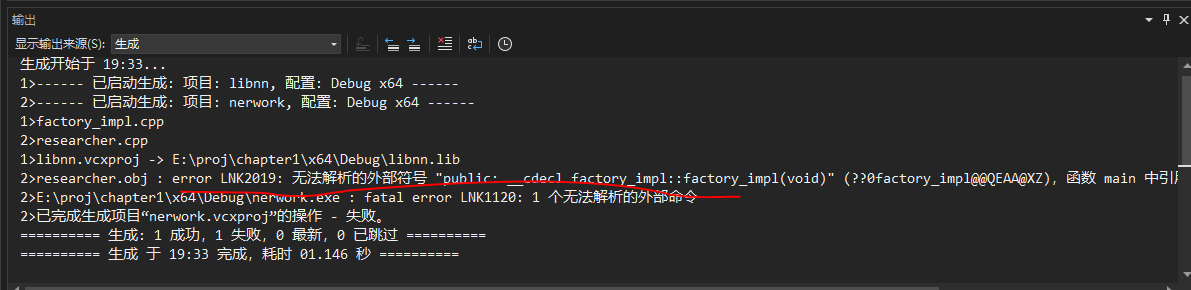
添加一个lib



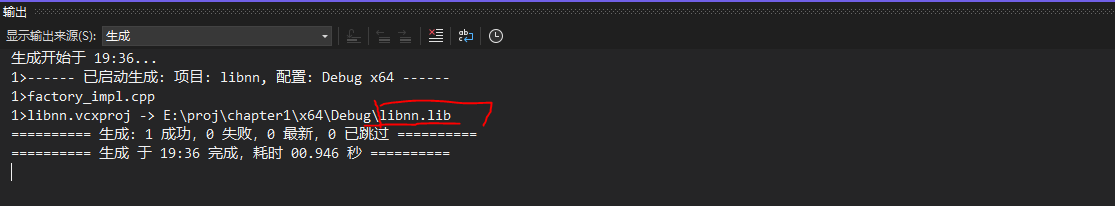
删掉无用的，改成这样



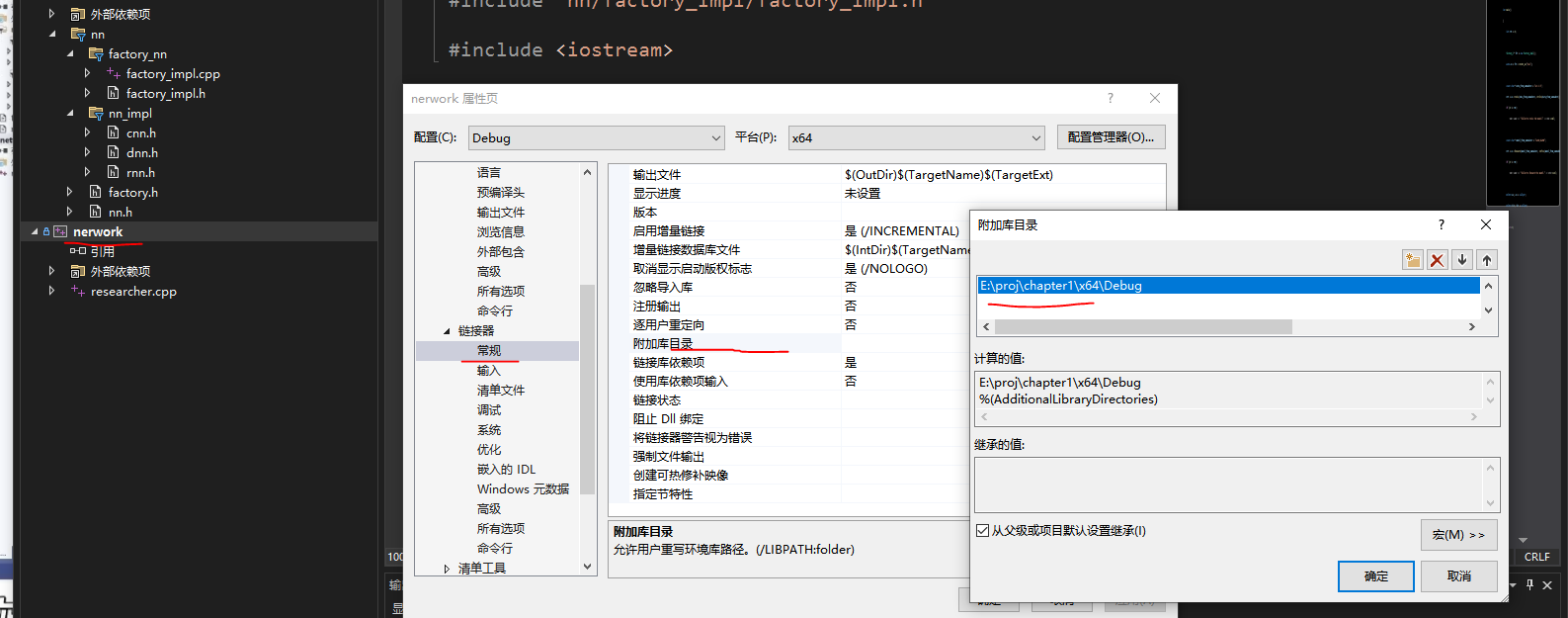
不用管



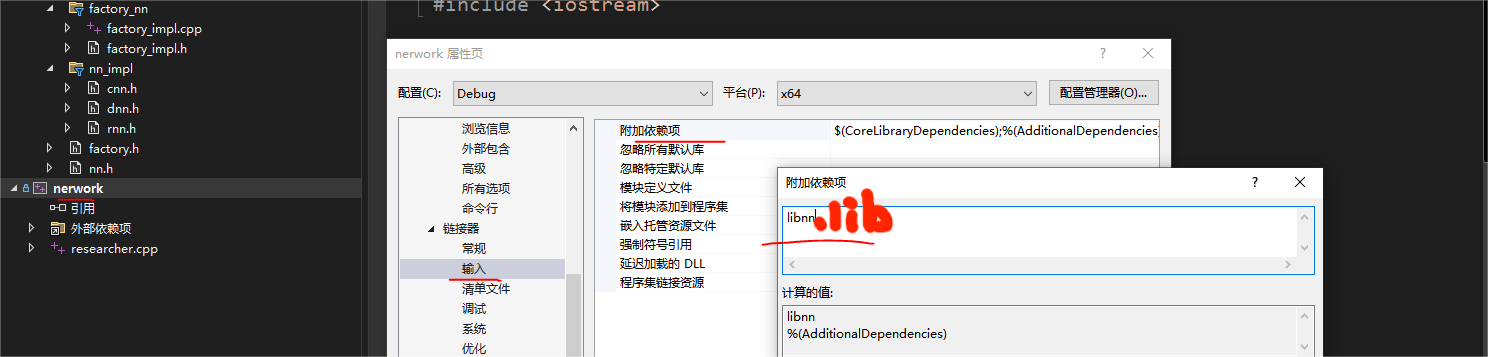
不要选择整个项目生成，而是邮件libnn生成



添加库

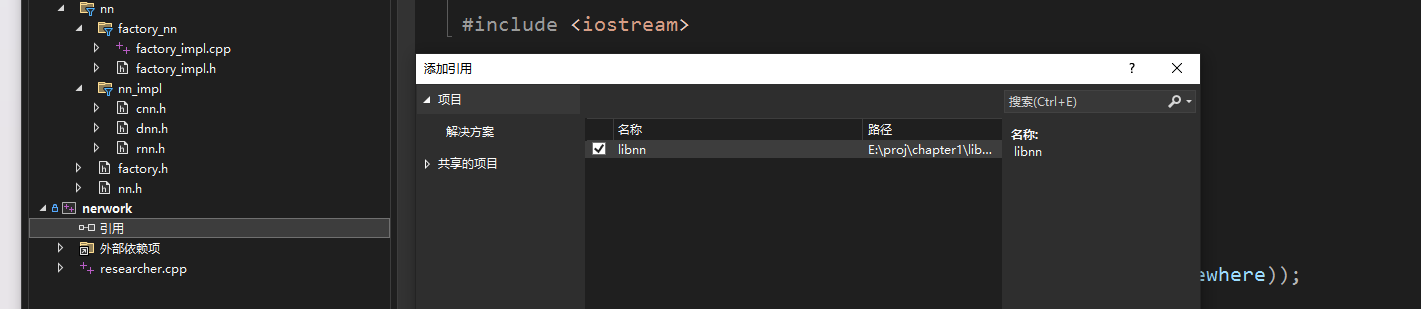


指明libnn

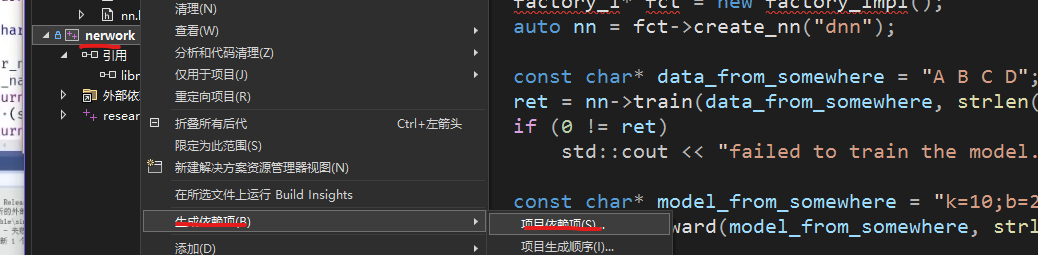


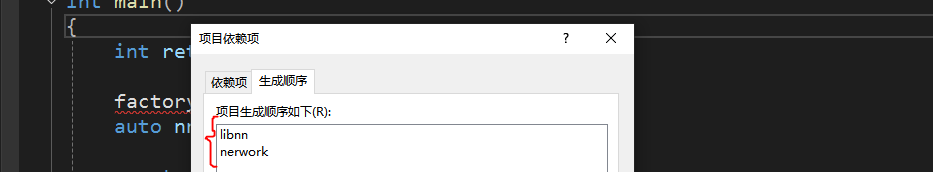
Network - 生成（结束！）

或者对于vs，不去设置附加依赖项等，可以这样



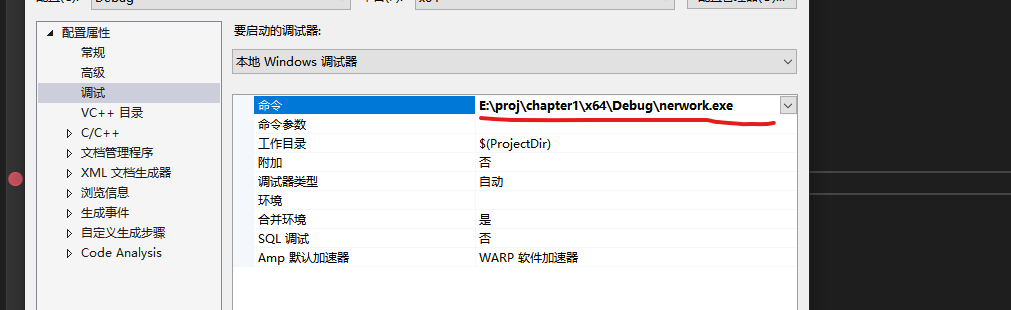
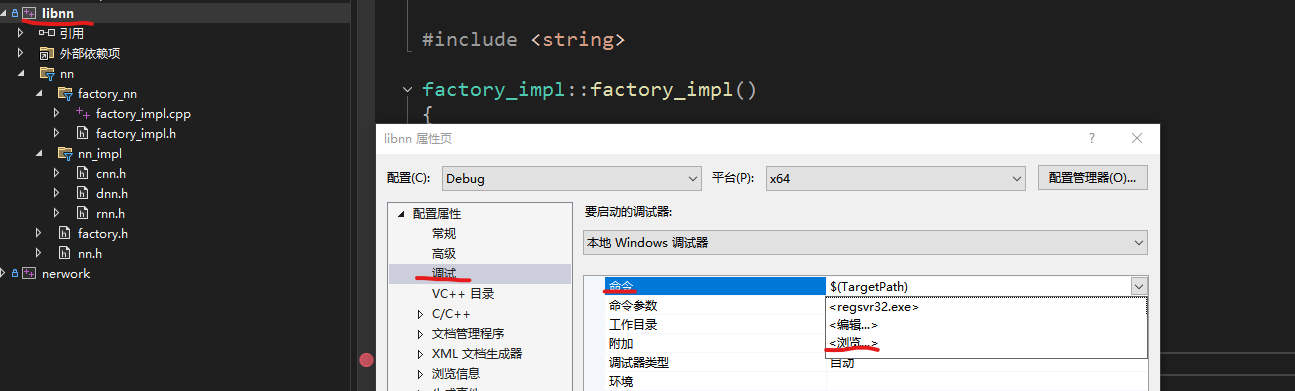
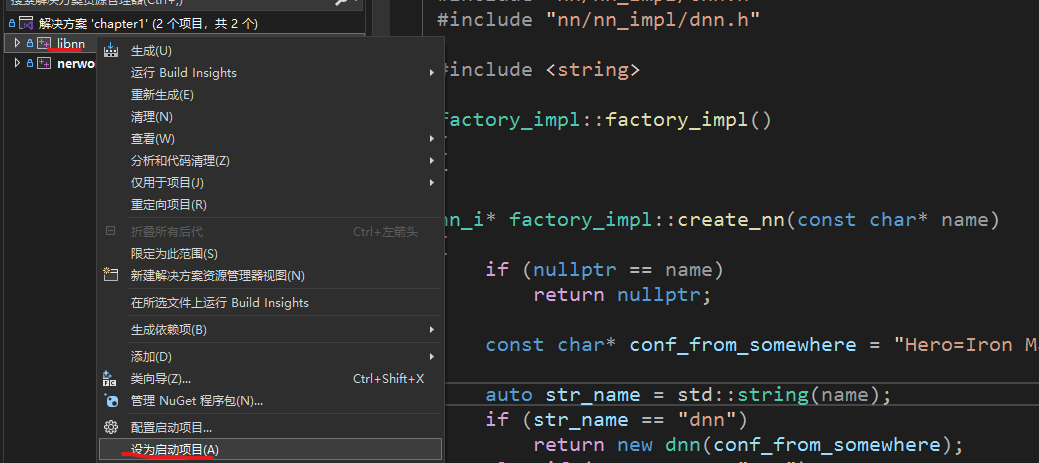
查看





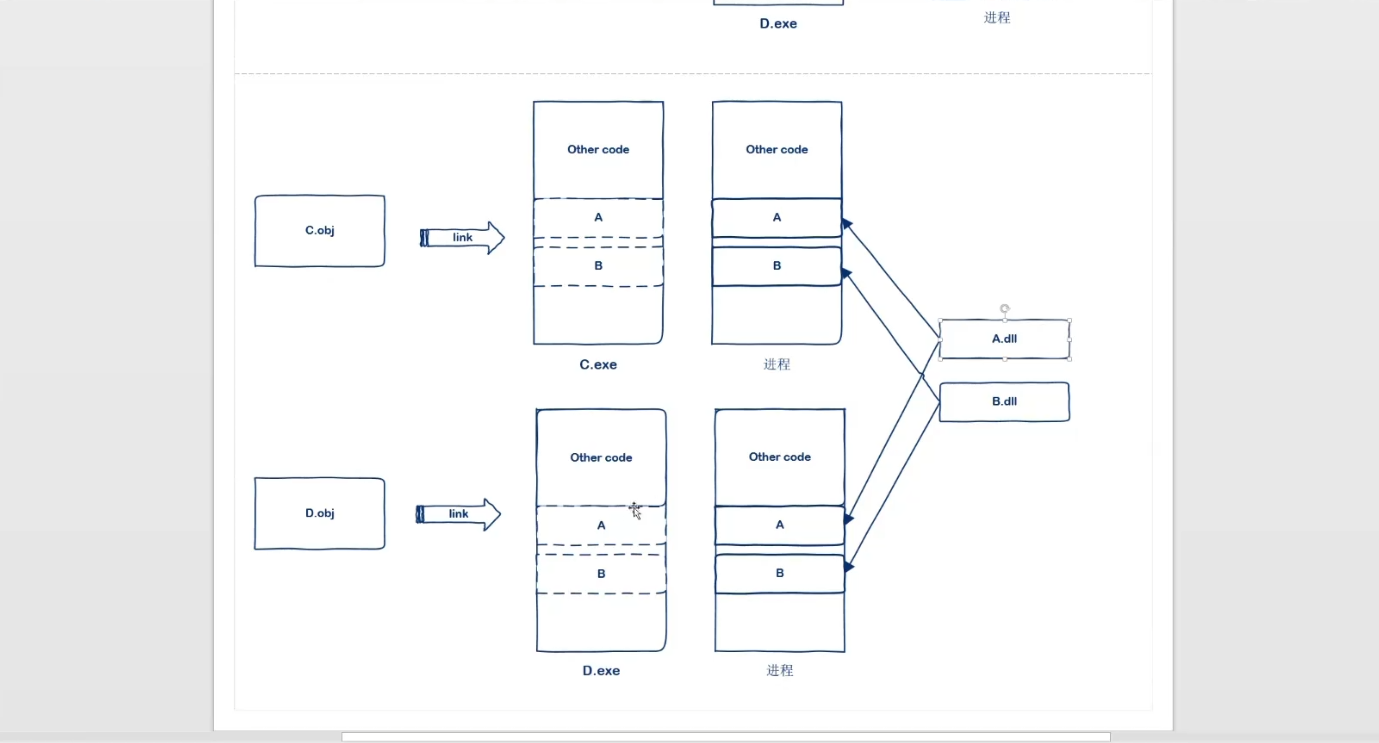
调试就是f5

Libnn是lib，不像network是exe，但是他也可以设置为启动项。



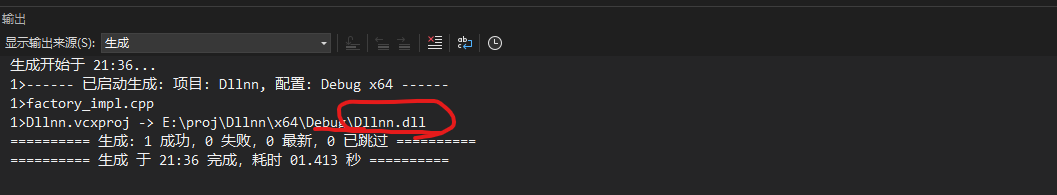
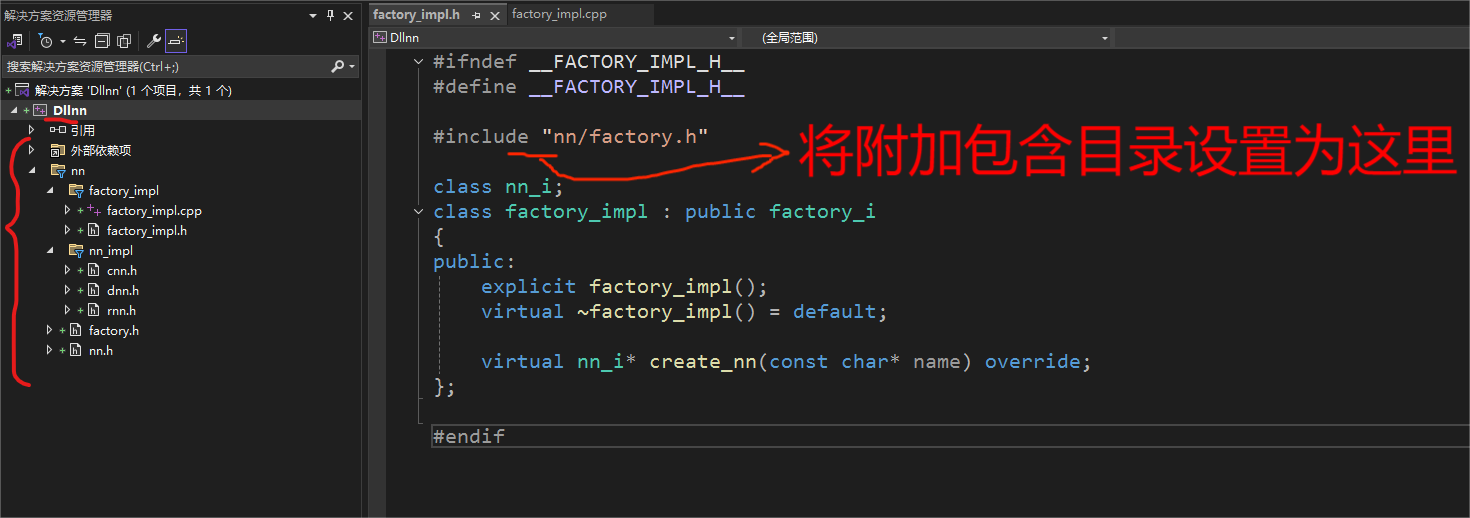
将其挂靠到选中的exe文件上，一样可以调试

# 动态库介绍&静态导入和动态加载

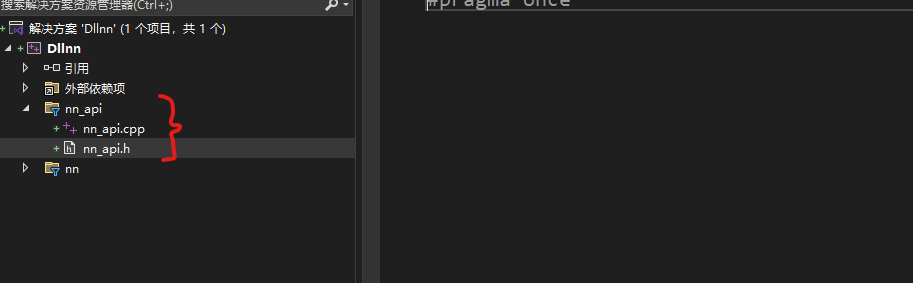


单独建一个动态库解决方案

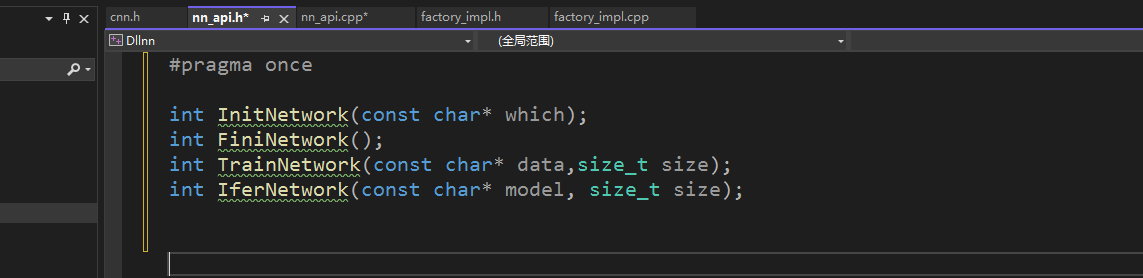
取消预编译头，设置附加包含目录，生成



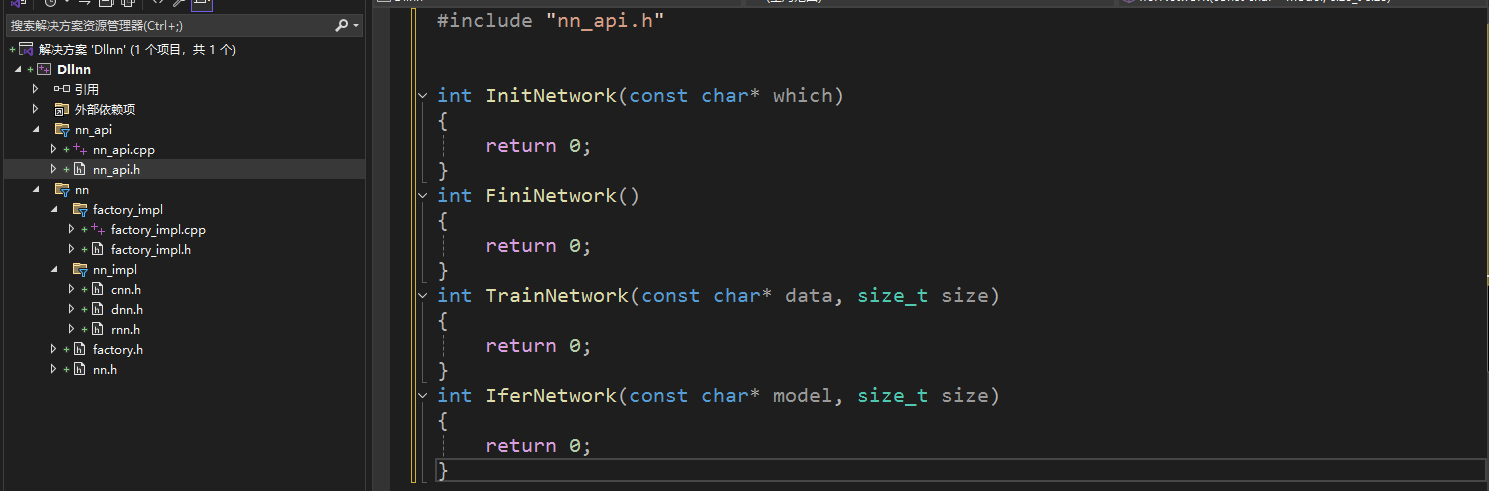
添加api过滤器和文件，用于暴露核心功能，初始化函数，销毁函数。



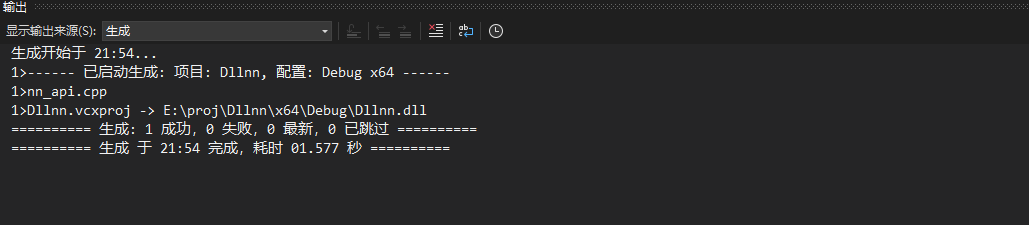
声明几个函数



实现它们

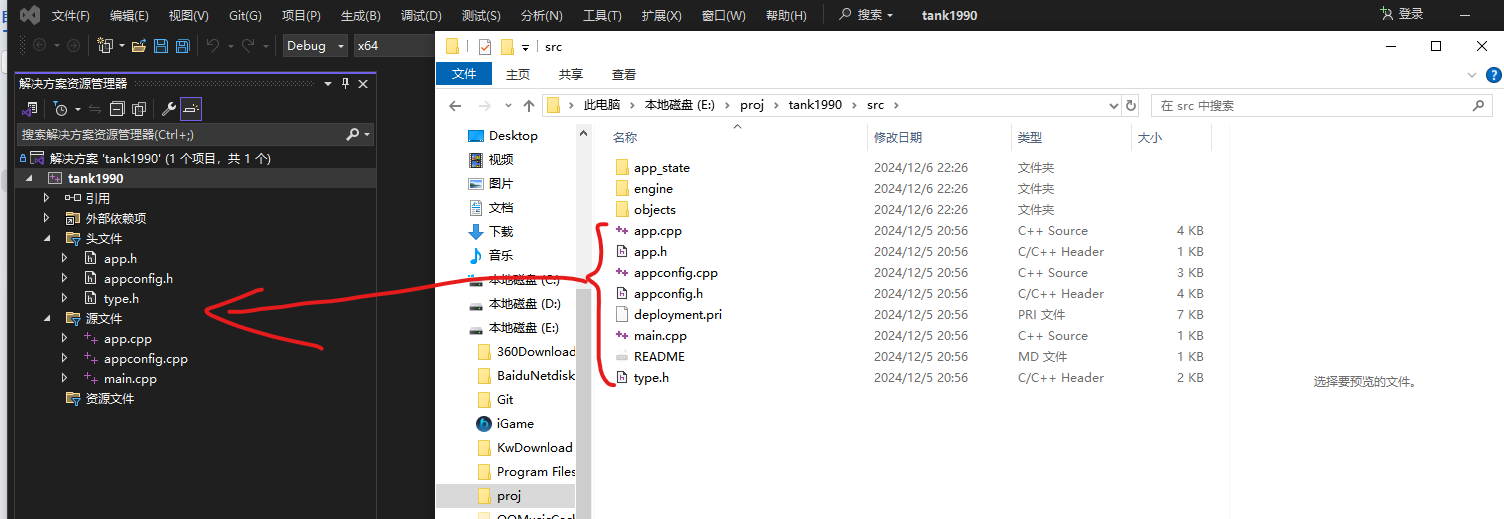


生成

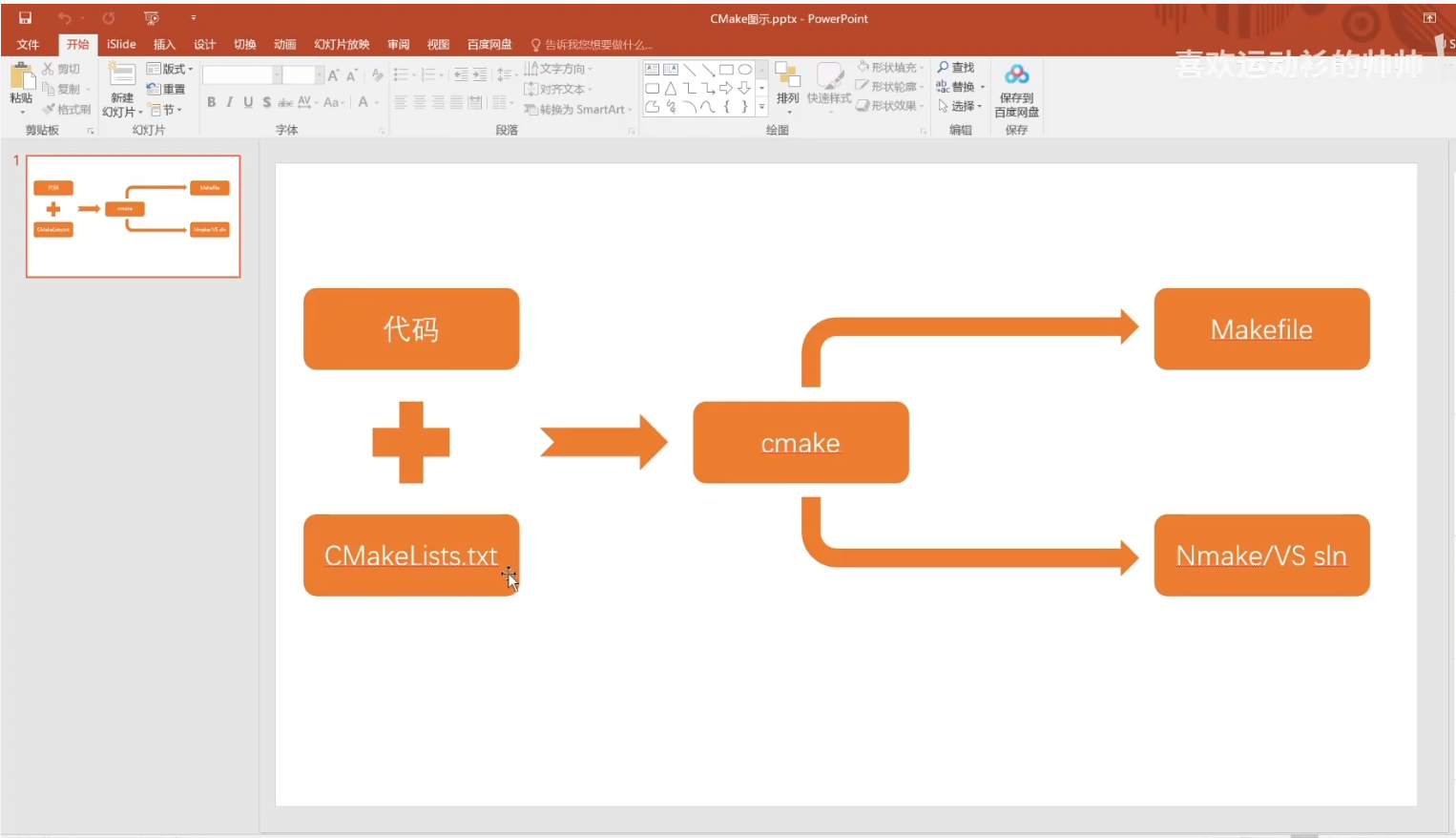


# 构建坦克工程

新建一个工程，把src文件拷贝过来，添加这几个主要的，剩下的搞成lib库



# Cmake



Cmake:

在linux上取代写makefile的过程

在windows上 ： 手动建一个工程，代码编译，链接

Cmakelist一套文件可以在不同平台上构建工程（使用宏来区分不同的平台）

构建：全自动完成代码编译链接打包的过程，管理组件，链接第三方库

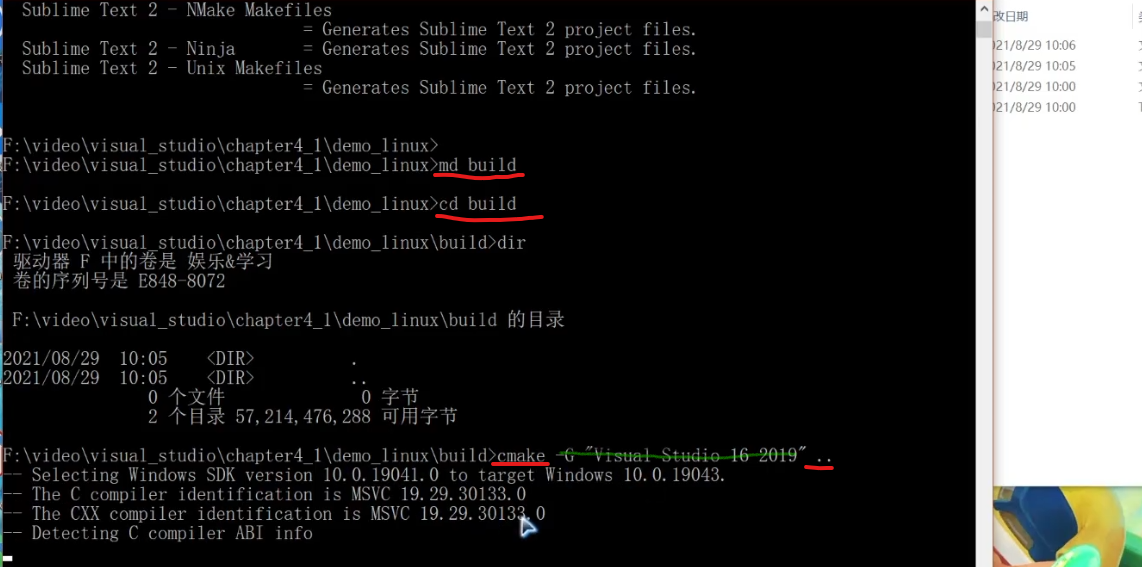
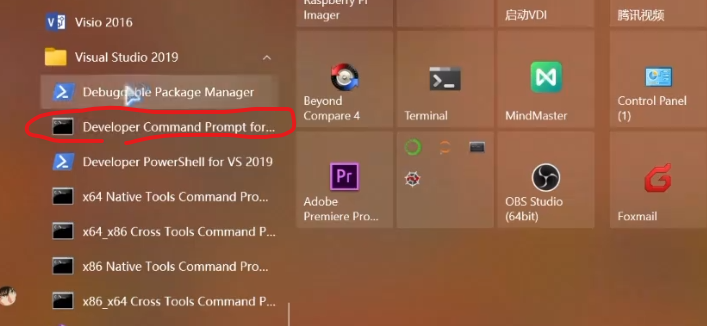
不用再手工配置makefile，vs或者xcode工程

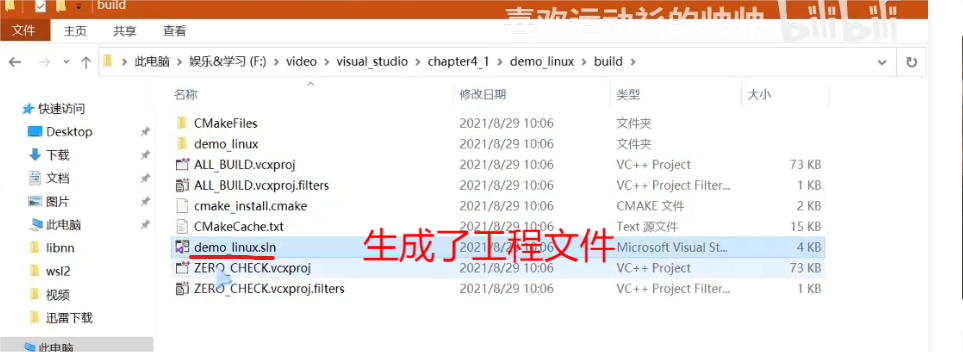
Cmake自身不带编译工具

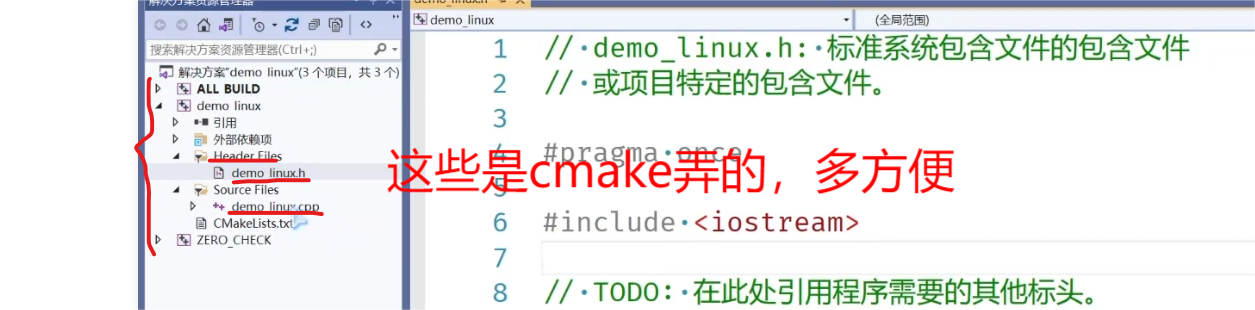
利用cmakelist可以自动生成目标平台下的原生工程文件，比如 \*.sln, makefile,\*.xcodeproj等

Cmake的本质是定义各个目标之间的关联，比如项目中由哪些可执行文件，动态库，资源文件组成，以及它们通过哪些源文件编译，用到了哪些外部关联等

可以使用这个运行cmake命令行







# 调试简介

F9 添加断点

F11 进入

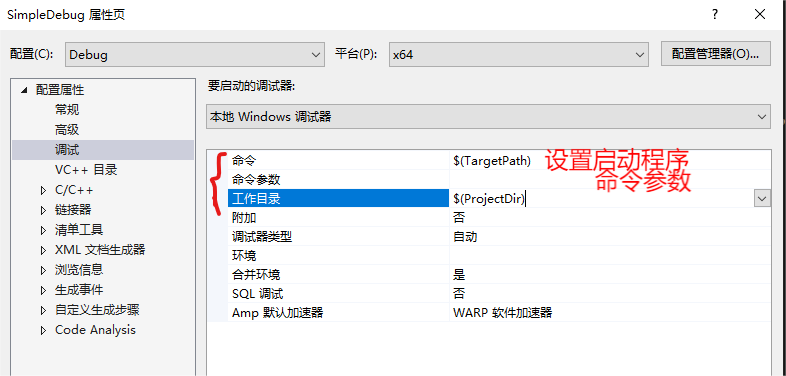
Shift+F11 跳出

F10 执行一行

方式一：

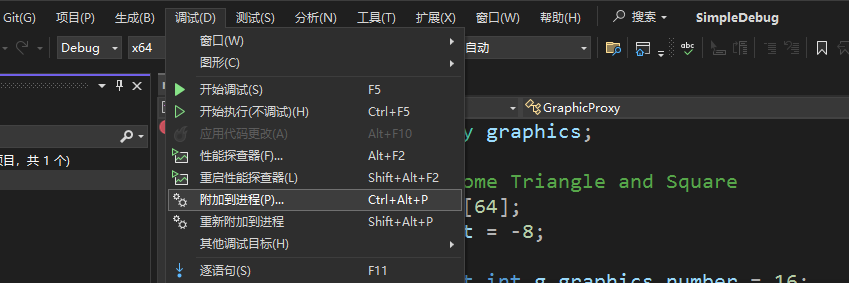
F5 开始调试（没有命令行参数）跟点击本地windows调试效果一样

如果需要命令行参数，还需要设置一下内容



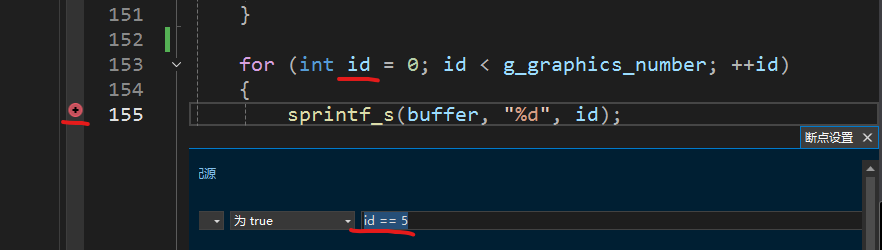
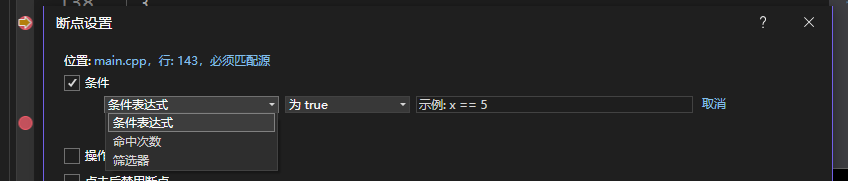
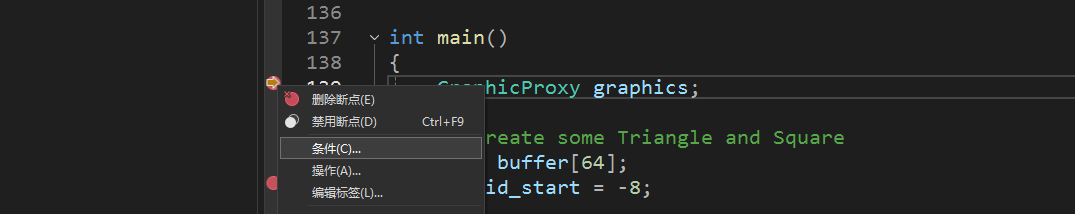
方式二：

程序运行中调试，点击附加到进程，可以附加到运行中的程序

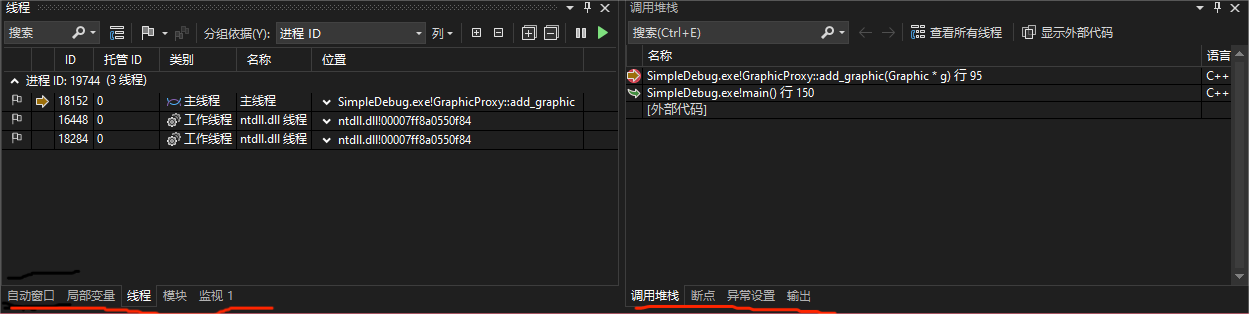


如果出现白点，一般是代码和exe文件不匹配，可以重新生成exe文件。

可以设置条件断点



Ctrl+F10 直接运行到当前行



监视：可以添加一些需要持续关注的量

