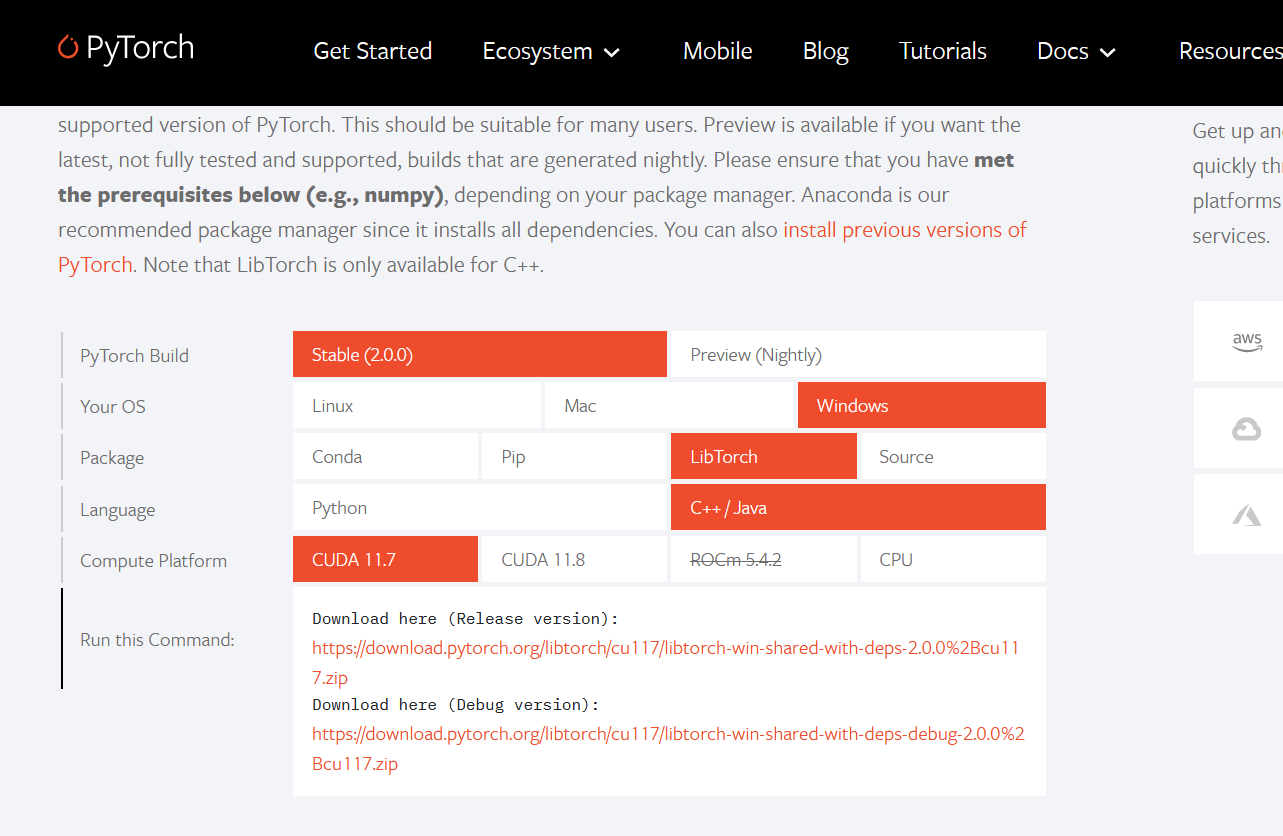
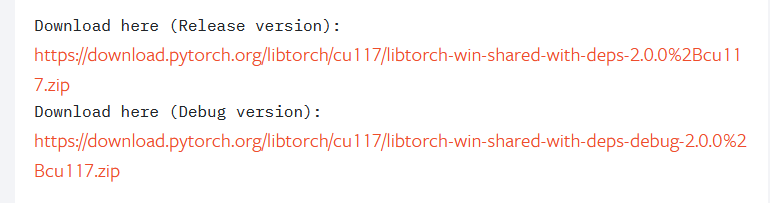
首先是检查cuda版本（默认已经安装cuda，且熟练使用visual studio），libtorch版本要和cuda版本保持一致，访问libtorch官网<https://pytorch.org/>



可以看到现在已经更新到CUDA11.7了（我下载的时候还是11.6），如果你的cuda版本和这个不兼容，有两种方法，第一种是卸载重装cuda环境，第二种是，注意下面这个截图中的网址，里面有/cu117/这就代表cuda11.7版本，假设你的cuda是11.6，那么可以尝试把里面的网址相关的部分进行替换，例如：将<https://download.pytorch.org/libtorch/cu117/libtorch-win-shared-with-deps-2.0.0%2Bcu117.zip>

改成<https://download.pytorch.org/libtorch/cu116/libtorch-win-shared-with-deps-2.0.0%2Bcu116.zip>

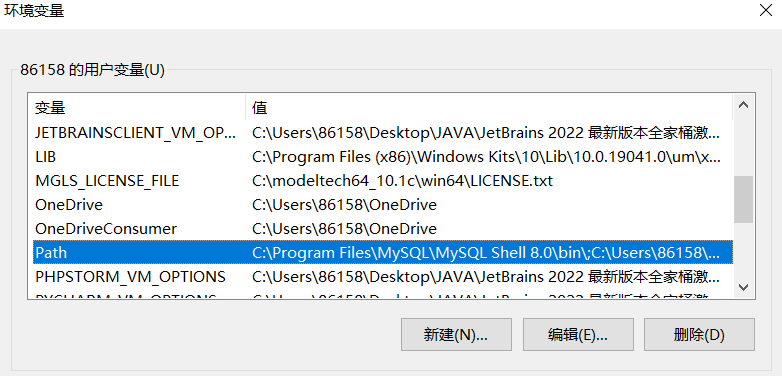
还有一种方法，我这里有11.6的安装包，如果你的cuda环境恰好也是11.6的，可以找我要。



安装libtorch之后（release和debug都建议安装，即上面两个都要下载），新建一个visual studio项目来做测试，目的是为了测试libtorch好不好用，在此之前，需要将libtorch添加到环境变量，







在用户变量里的PATH中新增以下几条

D:\libtorch-win-shared-with-deps-1.13.1+cu116\libtorch\lib

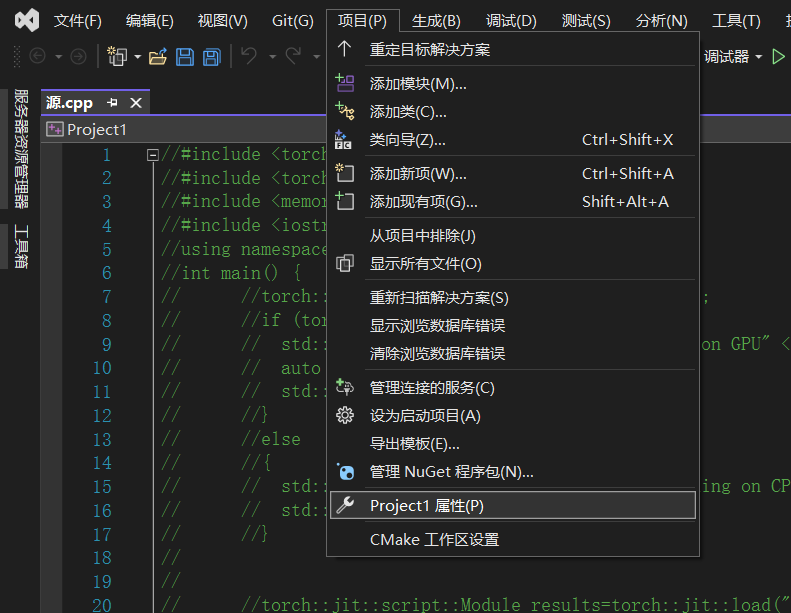
D:\libtorch-win-shared-with-deps-1.13.1+cu116\libtorch\include

D:\libtorch-win-shared-with-deps-debug-1.13.1+cu116\libtorch\lib

D:\libtorch-win-shared-with-deps-debug-1.13.1+cu116\libtorch\include

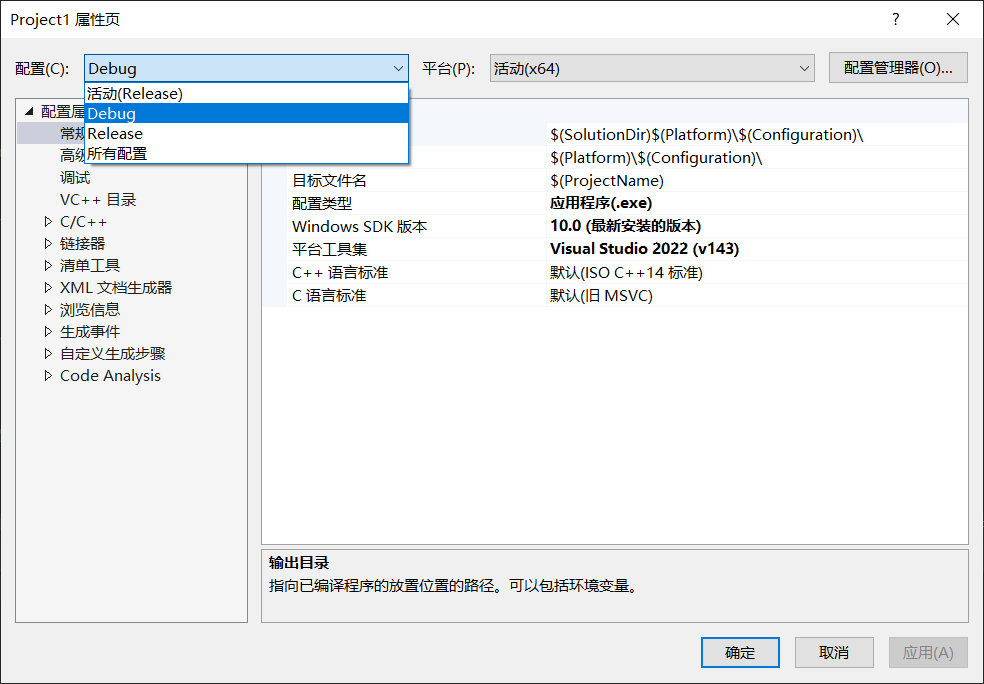
**注意：这里所展示的是我的路径，你的路径可能不同，其实重点是我标黄的位置，你要在你的libtorch文件夹中找到对应的标黄的文件夹，将其导入。另外，上面两条其实就是release版本的，下面两条是debug版本的。**

接下来，新建visual studio项目，



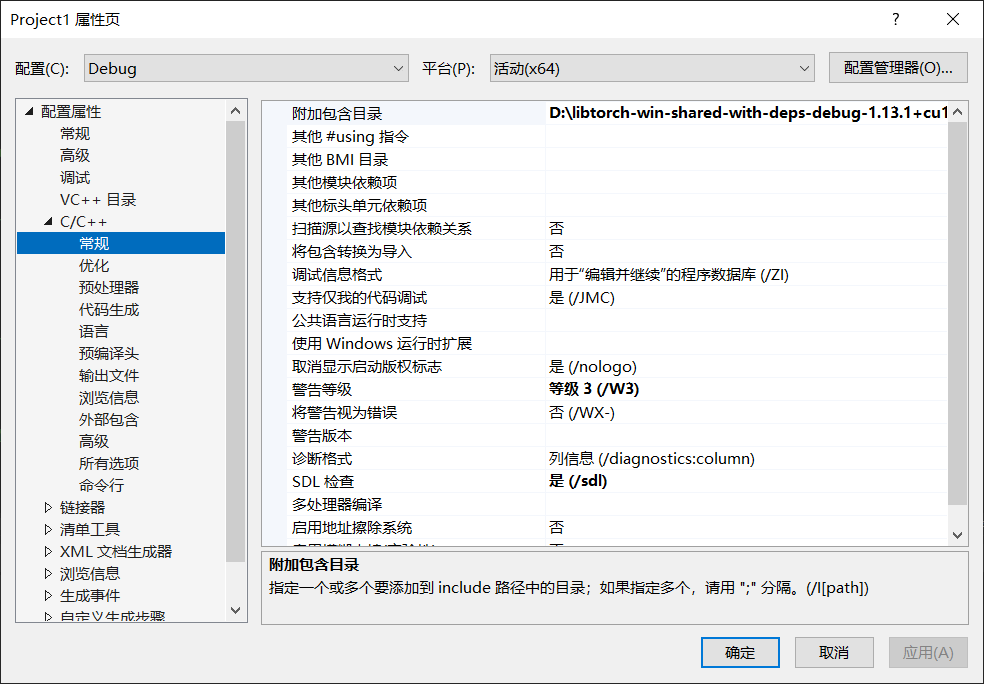


找到如图所示位置，项目属性。可以看到属性页里有debug和release，这两个都要配置。





先配置debug

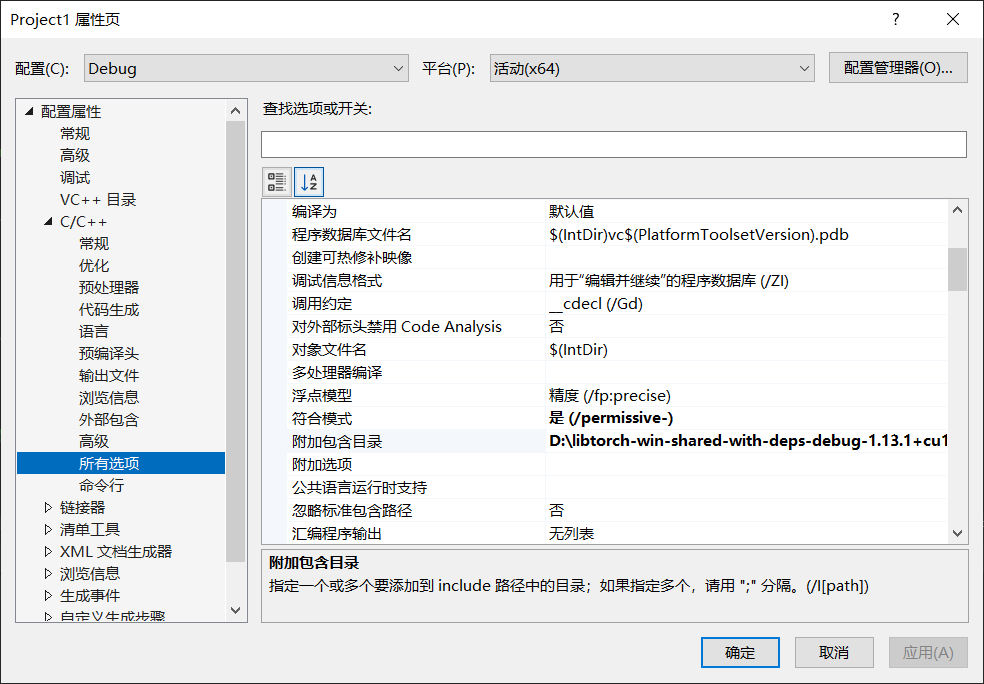




在附加包含目录里添加这两条（**注意：这里的路径和上面一样，也是因人而异的，重点也是标黄部分，还有，记得这里是libtorch的debug版本目录，我给加粗了，记得别弄成release版本的，下文中出现路径也以这种方式提示，不在赘述**）

D:\libtorch-win-shared-with-deps-**debug**-1.13.1+cu116\libtorch\include

D:\libtorch-win-shared-with-deps-**debug**-1.13.1+cu116\libtorch\include\torch\csrc\api\include

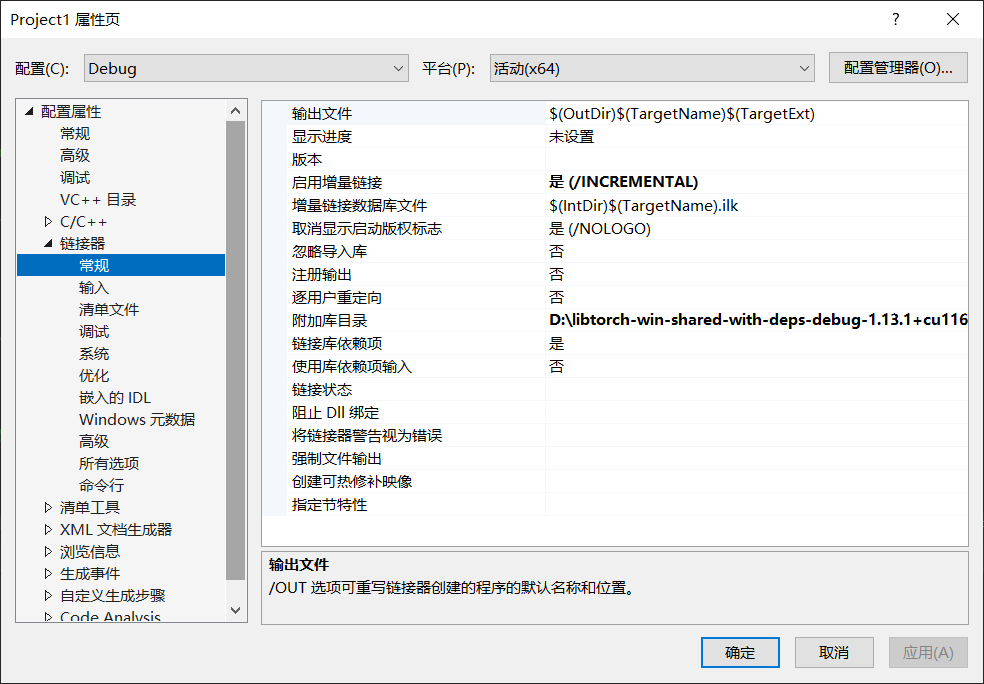




在附加包含目录里添加

D:\libtorch-win-shared-with-deps-debug-1.13.1+cu116\libtorch\include

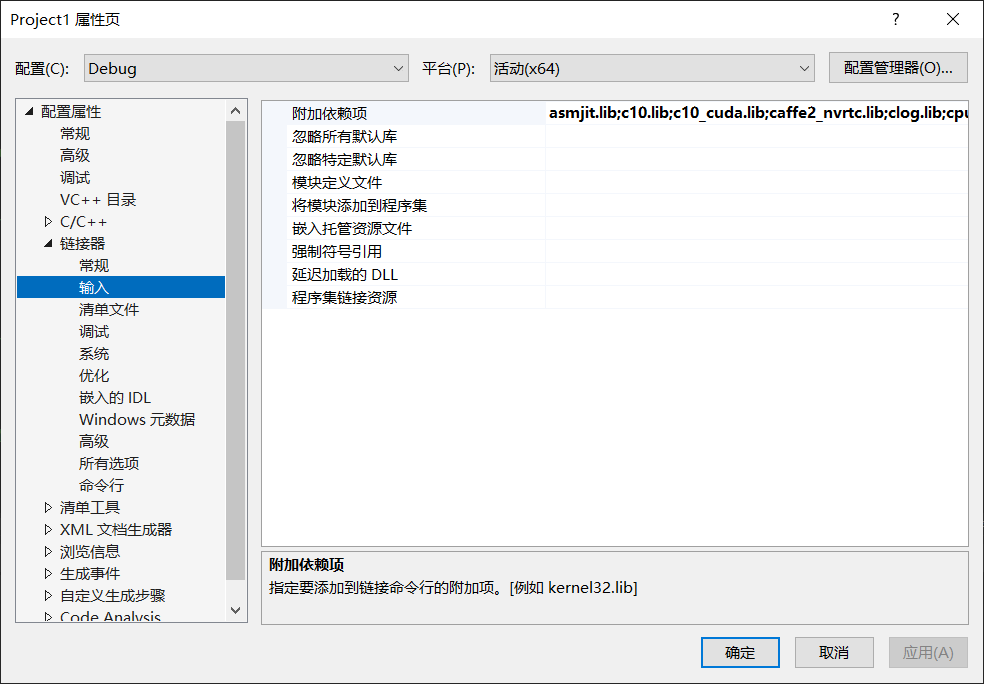
D:\libtorch-win-shared-with-deps-debug-1.13.1+cu116\libtorch\include\torch\csrc\api\include





在链接器-常规-附加库目录里添加

D:\libtorch-win-shared-with-deps-debug-1.13.1+cu116\libtorch\lib





在输入-附加依赖项里添加（直接复制添加）

asmjit.lib

c10.lib

c10\_cuda.lib

caffe2\_nvrtc.lib

clog.lib

cpuinfo.lib

dnnl.lib

fbgemm.lib

kineto.lib

libprotobuf.lib

libprotobuf-lite.lib

libprotoc.lib

pthreadpool.lib

torch.lib

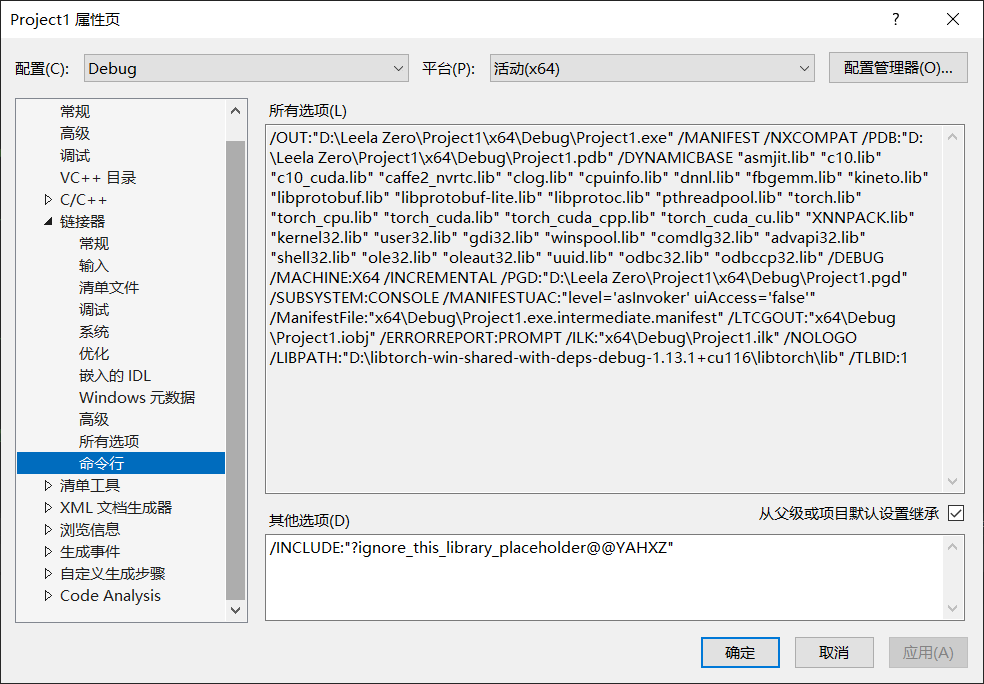
torch\_cpu.lib

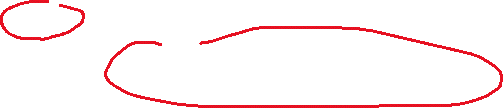
torch\_cuda.lib

torch\_cuda\_cpp.lib

torch\_cuda\_cu.lib

XNNPACK.lib

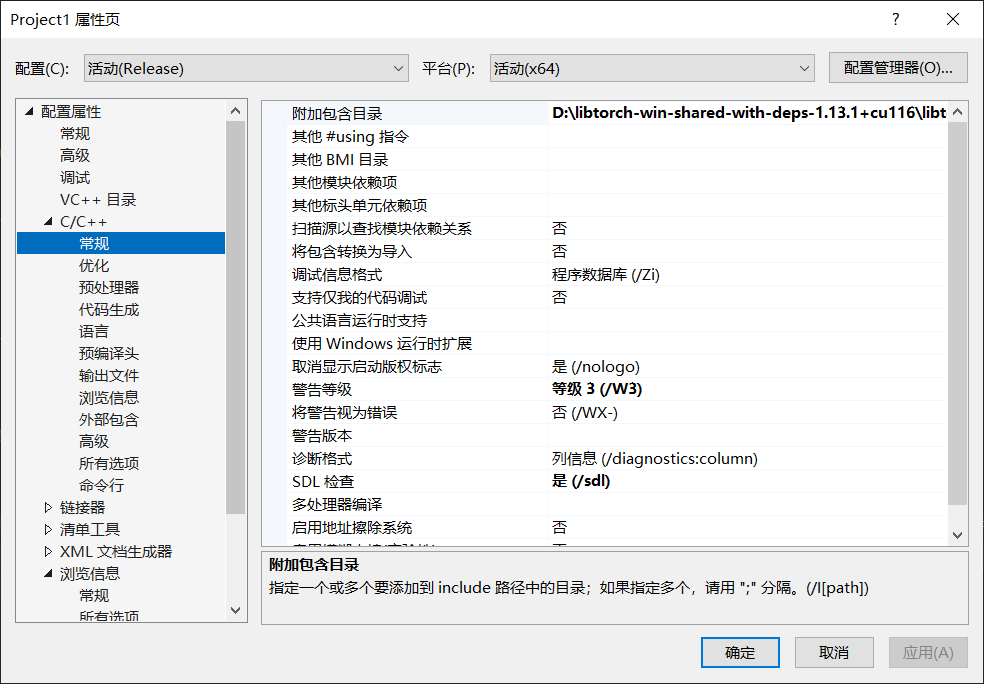




添加

/INCLUDE:"?ignore\_this\_library\_placeholder@@YAHXZ"

接下来配置release（事实上和debug相同，只不过导入的路径不一样）

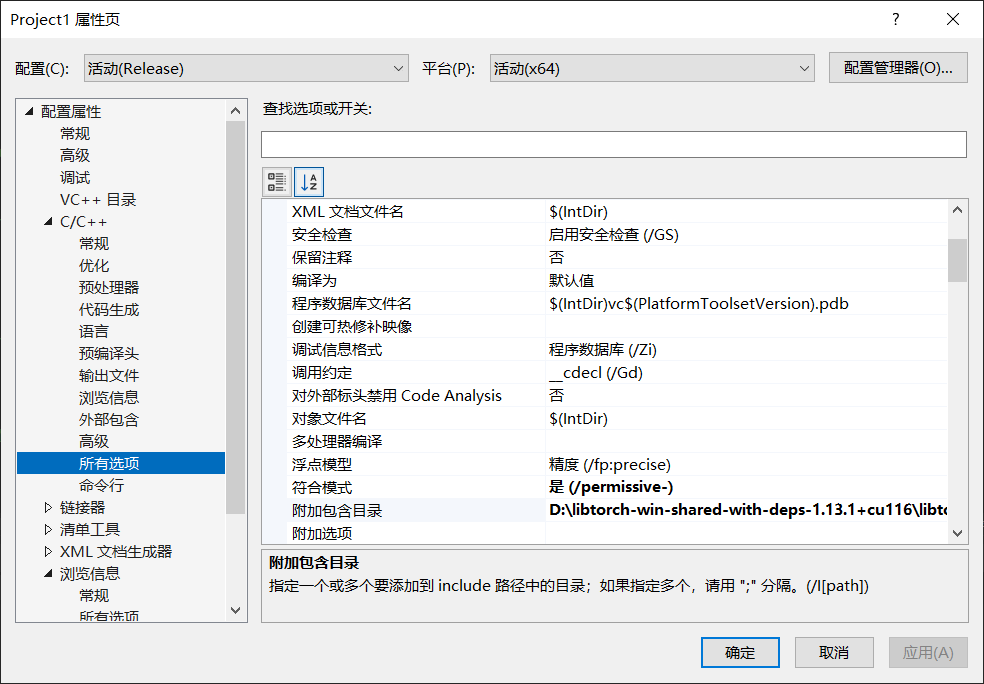


添加下面两行，其实可以注意到，只是标黄部分前面的路径不一样而已，你下载debug版本和release版本的libtorch，就会有两个文件夹，标黄前面部分的路径其实就是这个文件夹的路径，一个有“debug”，一个没有。



D:\libtorch-win-shared-with-deps-1.13.1+cu116\libtorch\include

D:\libtorch-win-shared-with-deps-1.13.1+cu116\libtorch\include\torch\csrc\api\include

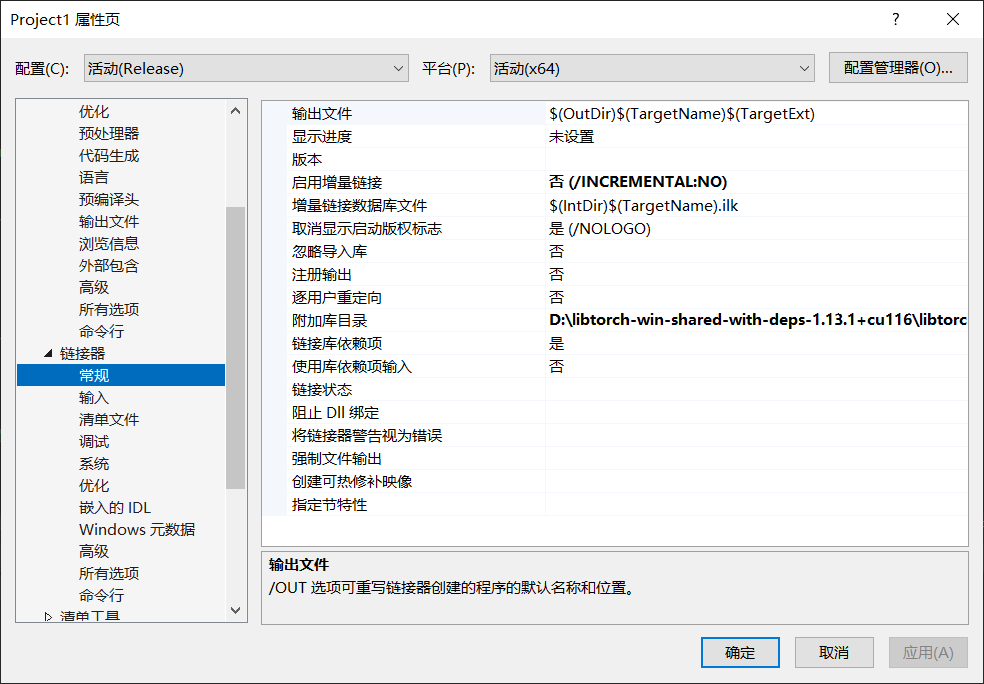




添加

D:\libtorch-win-shared-with-deps-1.13.1+cu116\libtorch\include

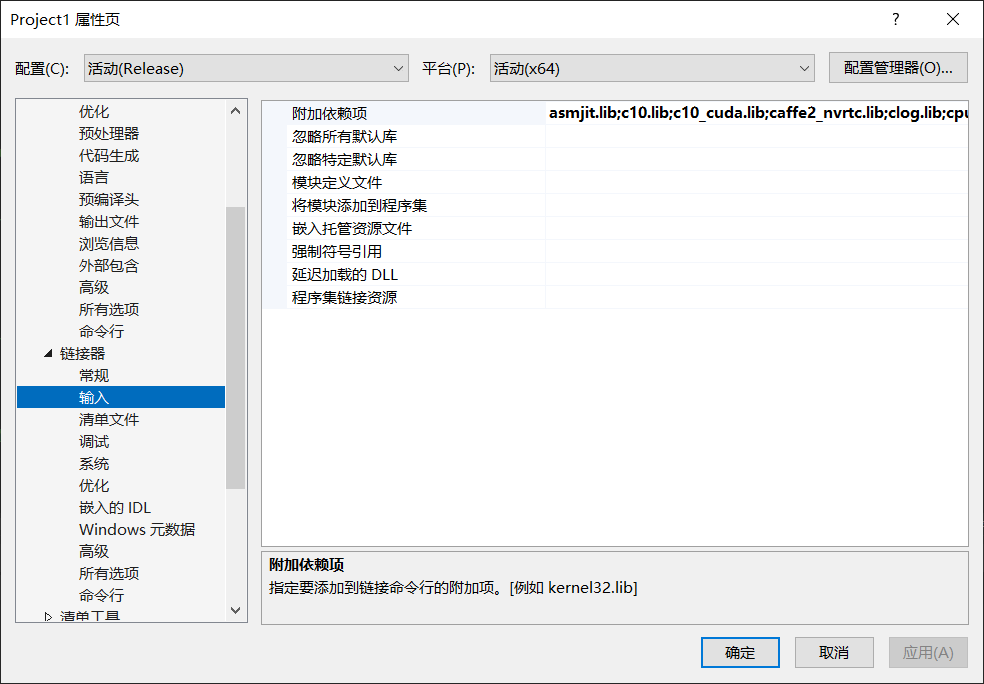
D:\libtorch-win-shared-with-deps-1.13.1+cu116\libtorch\include\torch\csrc\api\include





添加

D:\libtorch-win-shared-with-deps-1.13.1+cu116\libtorch\lib





添加

asmjit.lib

c10.lib

c10\_cuda.lib

caffe2\_nvrtc.lib

clog.lib

cpuinfo.lib

dnnl.lib

fbgemm.lib

kineto.lib

libprotobuf.lib

libprotobuf-lite.lib

libprotoc.lib

pthreadpool.lib

torch.lib

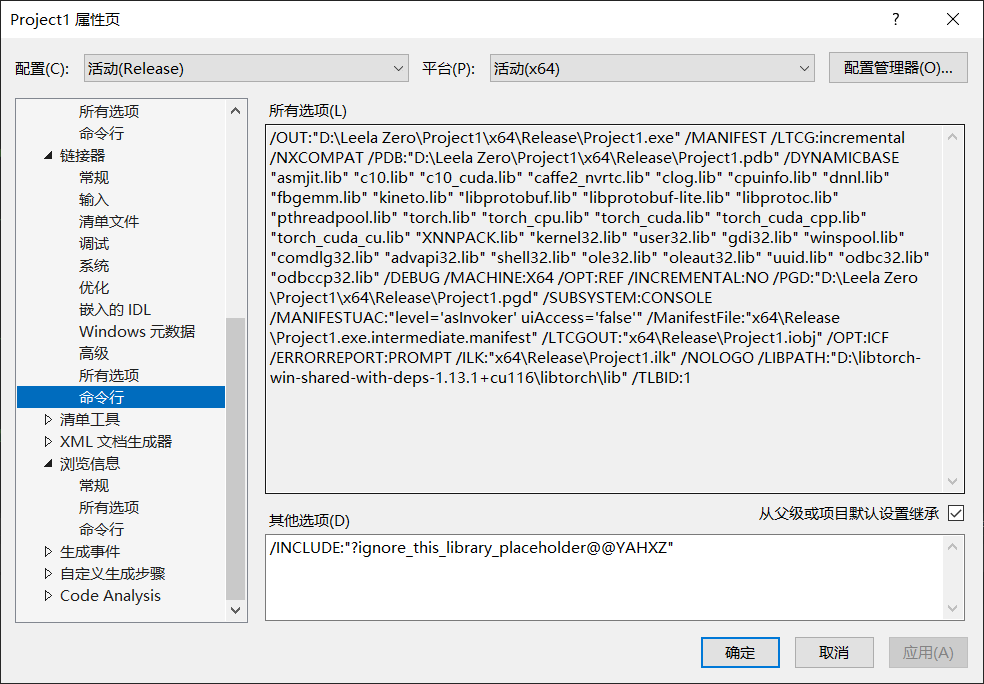
torch\_cpu.lib

torch\_cuda.lib

torch\_cuda\_cpp.lib

torch\_cuda\_cu.lib

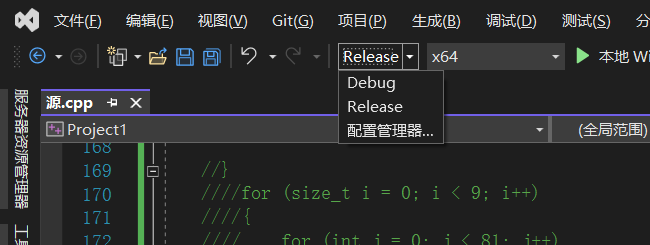
XNNPACK.lib

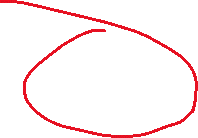


添加

/INCLUDE:"?ignore\_this\_library\_placeholder@@YAHXZ"

配置完毕，接下来进行测试。这里debug和release建议都跑一遍，debug按理来说会出问题报错，所以最后应该是用release来跑。





在你新建的项目中输入以下代码，其中FileName里的文件我会提供，只需要修改相应路径即可，如果能运行，则说明导入成功，接下来只要在你的leela项目中按照我上面的教程也导入一遍就可以了。这其中可能会遇到各种奇奇怪怪的异常和报错，不过一般经过谷歌或百度搜博客的教程或者问chatGPT应该也能得到答案，这个教程以及代码就是我看了很多博客之后写出来的。

#include <torch/script.h>

#include <iostream>

#include <memory>

#include <torch/torch.h>

int main() {

std::cout << "CUDA: " << torch::cuda::is\_available() << std::endl;

std::cout << "CUDNN: " << torch::cuda::cudnn\_is\_available() << std::endl;

std::cout << "GPU(s): " << torch::cuda::device\_count() << std::endl;

auto FileName = "D:\\Leela Zero\\kylin\\kylinnogo\\best\_121.pt";

auto device = torch::kCUDA;

torch::jit::script::Module module;

try {

module = torch::jit::load(FileName, device);

}

catch (c10::Error k)

{

std::cout << k.msg() << std::endl;

}

// 创建一个Tensor

std::vector<torch::jit::IValue> inputs;

// 构造全1的9x9张量

torch::Tensor zeros = torch::zeros({1, 8, 9, 9 });

torch::Tensor ones = torch::ones({ 1,1, 9, 9 });

torch::Tensor a = torch::cat({ zeros, ones }, /\*dim=\*/1).cuda();

//torch::Tensor a = torch::zeros({1,9,9,9}).cuda();

std::cout << a << std::endl;

//a.to(torch::kCUDA);

inputs.push\_back(a);

std::cout << a.device() << std::endl;

torch::NoGradGuard no\_grad;

c10::ivalue::TupleElements output;

//测试前向

try {

output = module.forward(inputs).toTuple()->elements();// [1] .toTensor();

}

catch (c10::Error k)

{

std::cout << k.msg() << std::endl;

}

std::cout << output.size() << std::endl;

for (auto i : output)

{

std::cout << i << std::endl;

}

auto policy\_curr = output[0].toTensor();

auto score = output[1].toTensor();

auto curr\_probs = torch::exp(policy\_curr).to(torch::kCPU);

auto score\_output = torch::exp(score).to(torch::kCPU);

std::vector<float> policy(score\_output.data\_ptr<float>(), score\_output.data\_ptr<float>() + score\_output.numel());

std::vector<float> value(score\_output.data\_ptr<float>(), score\_output.data\_ptr<float>() + score\_output.numel());

float score\_sum = 0.0;

for (int i = 0; i < 5; i++) {

score\_sum += (i - 5.0) \* value[i];

}

for (int i = 5; i < 10; i++) {

score\_sum += (i - 4.0) \*value[i];

}

std::cout << curr\_probs << std::endl;

std::cout << score\_sum << std::endl;

}