

# Visual Studio libtorch使用教程

## 预备软件

- libtorch: cpu或gpu版本均可，根据本机环境选择，但要注意下载nightly build版，因stable版头文件引用有错误。
- visual studio: 本教程在vs2015与2019版本下均调试通过，根据本机环境选择。
- \*cmake: 可选，当教程中其他方法均失败情况下尝试。

## 使用流程

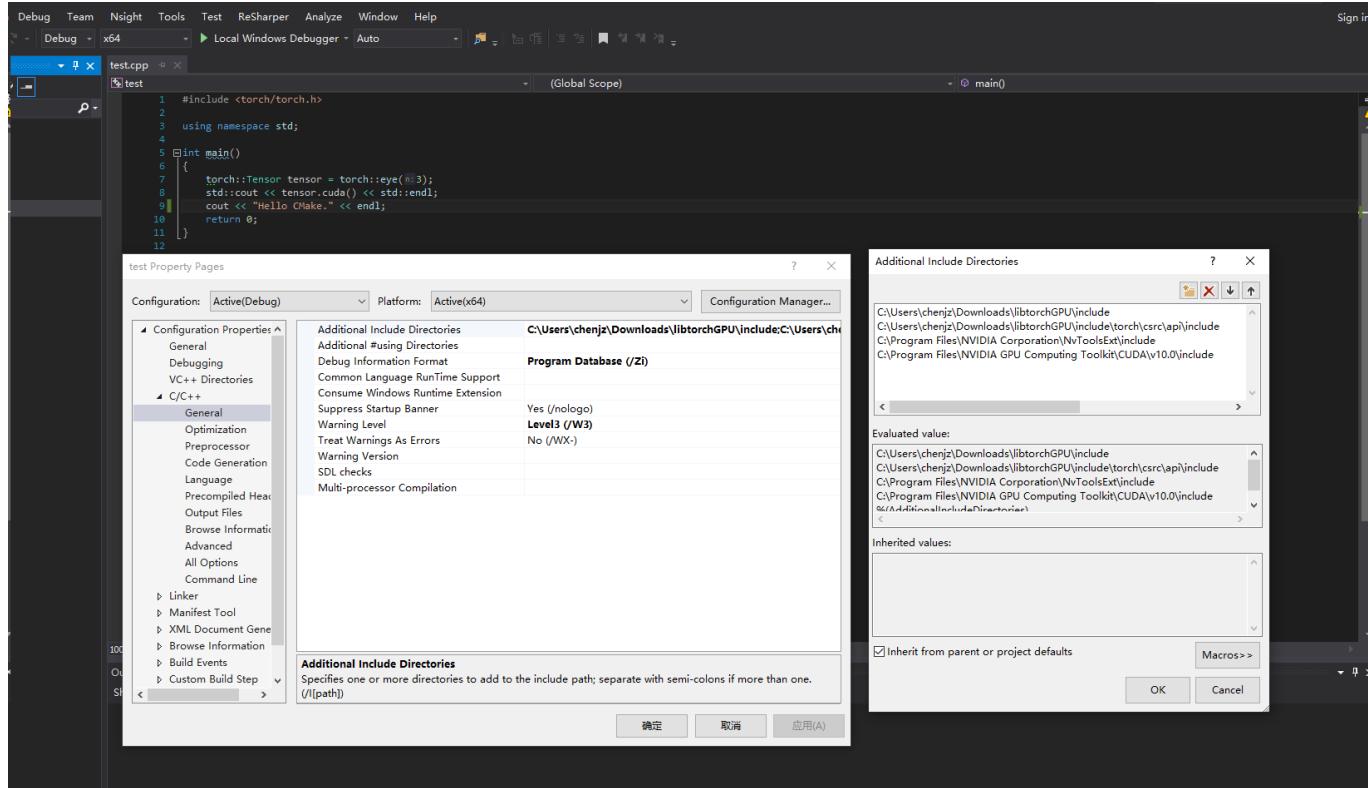
1. 下载对应版本libtorch并解压到对应目录，如在本机为  
`C:\Users\chenjz\Downloads\libtorchGPU`。
2. 打开visual studio创建新项目，添加源代码`test.cpp`

```
#include <iostream>
#include <torch/torch.h>

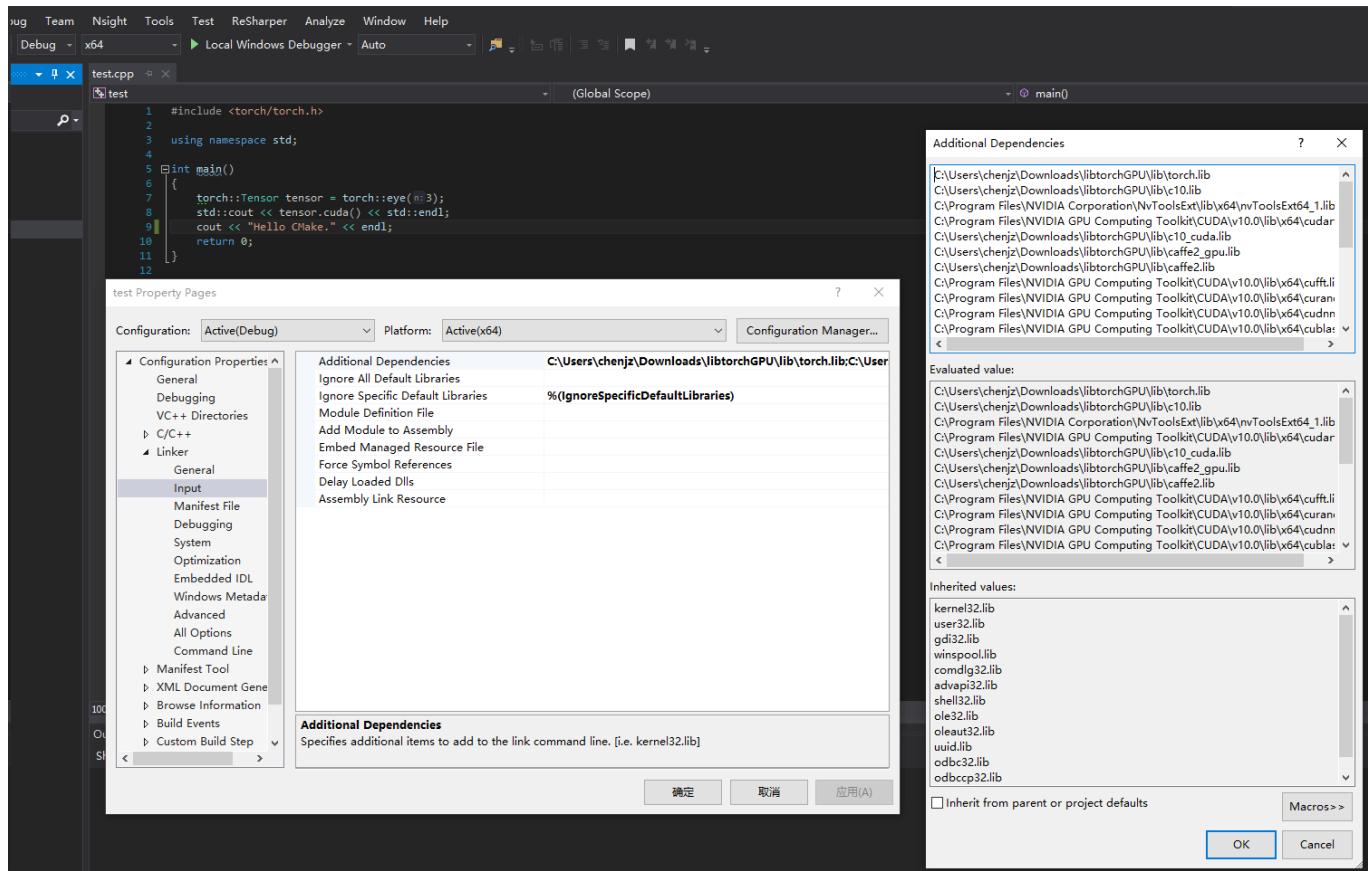
using namespace std;

int main()
{
    torch::Tensor tensor = torch::eye(3);
    std::cout << tensor.cuda() << std::endl;
    return 0;
}
```

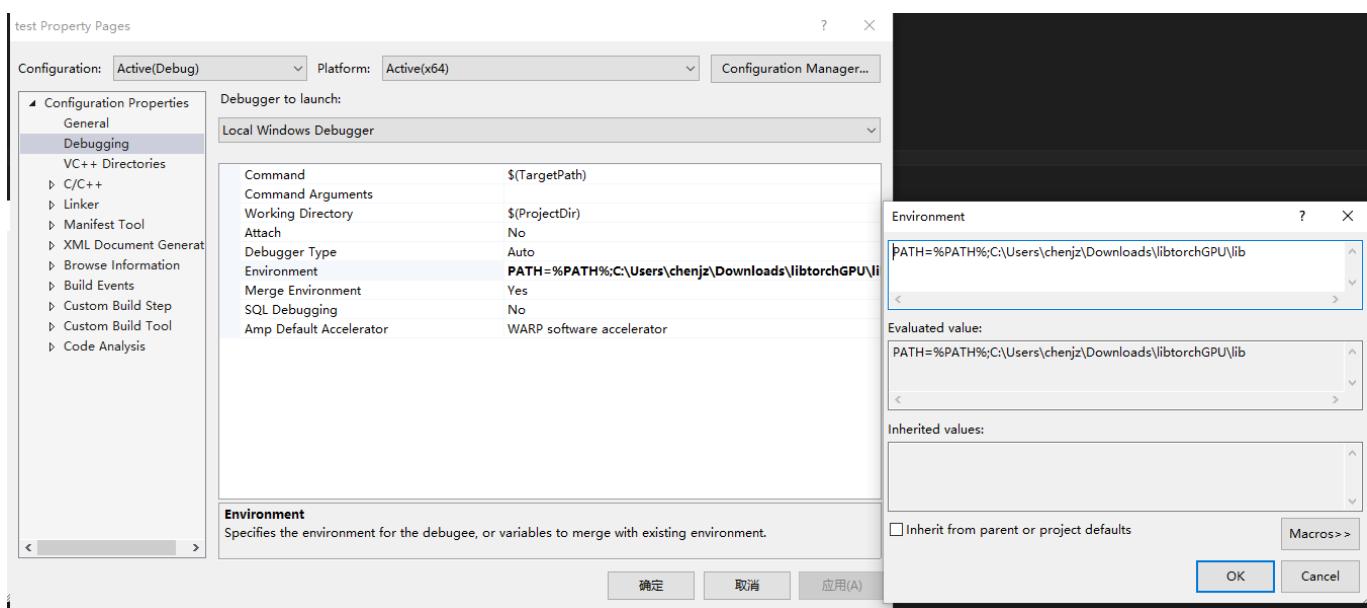
3. 右键项目属性，总共有三项需要配置：



```
C:\Users\chenjz\Downloads\libtorchGPU\include;C:\Users\chenjz\Downloads\libtorchGPU\include\torch\csrc\api\include;C:\Program Files\NVIDIA Corporation\NvToolsExt\include;C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\include;%AdditionalIncludeDirectories
```



```
C:\Users\chenjz\Downloads\libtorchGPU\lib\torch.lib
C:\Users\chenjz\Downloads\libtorchGPU\lib\c10.lib
C:\Program Files\NVIDIA Corporation\NvToolsExt\lib\x64\nvToolsExt64_1.lib
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\lib\x64\cudart.lib
C:\Users\chenjz\Downloads\libtorchGPU\lib\c10_cuda.lib
C:\Users\chenjz\Downloads\libtorchGPU\lib\caffe2_gpu.lib
C:\Users\chenjz\Downloads\libtorchGPU\lib\caffe2.lib
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\lib\x64\cufft.lib
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\lib\x64\curand.lib
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\lib\x64\cudnn.lib
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.0\lib\x64\cublas.lib
```



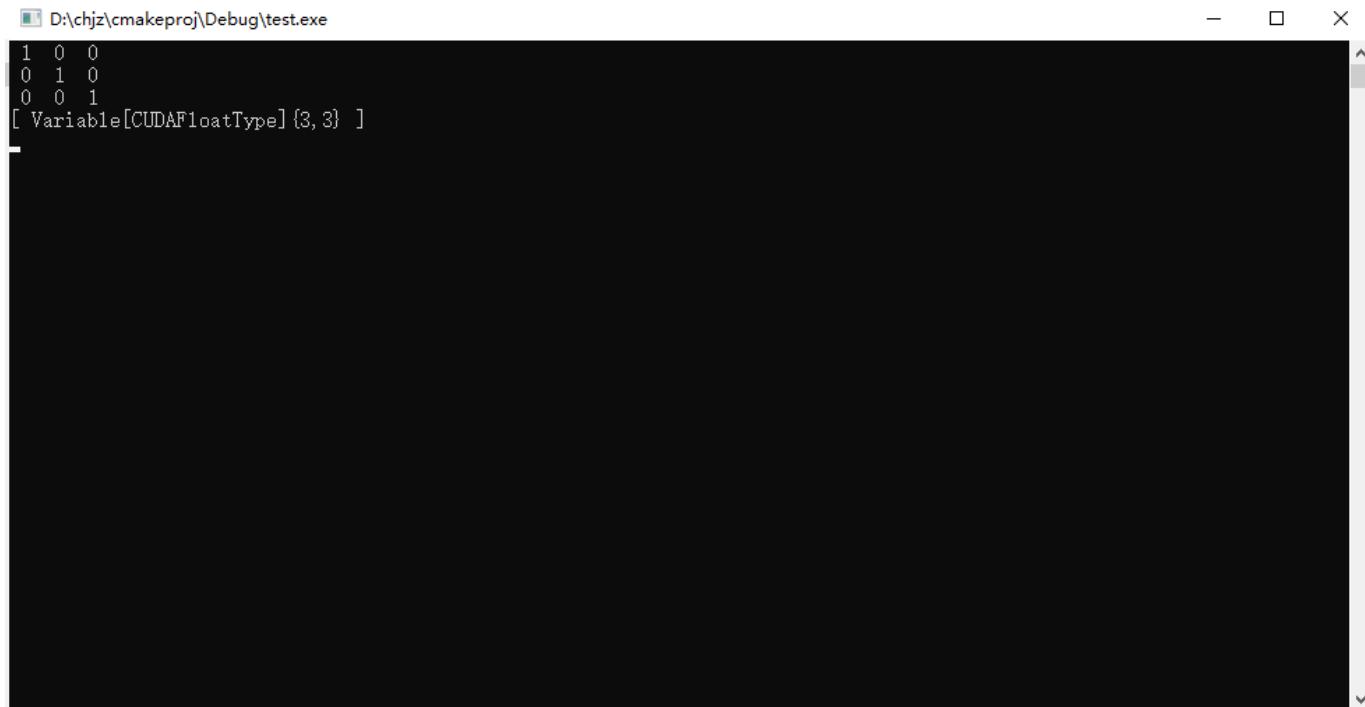
PATH=%PATH%;C:\Users\chenjz\Downloads\libtorchGPU\lib

配置1设置引用目录，配置2设置引用静态库，配置3设置d11引用路径，若使用的是cpu版本可以将配置中有关cuda的部分全部删除。

4. 编译并链接程序：有人build项目时可能会报错C4996，这是msvc为了安全所引用的特性，可以在[项目配置->C/C++>Advanced->Disable Specific Warnings](#)里添加4996。

\*在已有项目里添加libtorch同样需要以上三步。

运行成功！



```
1 0 0
0 1 0
0 0 1
[ Variable[CUDAFloatType] {3, 3} ]
```

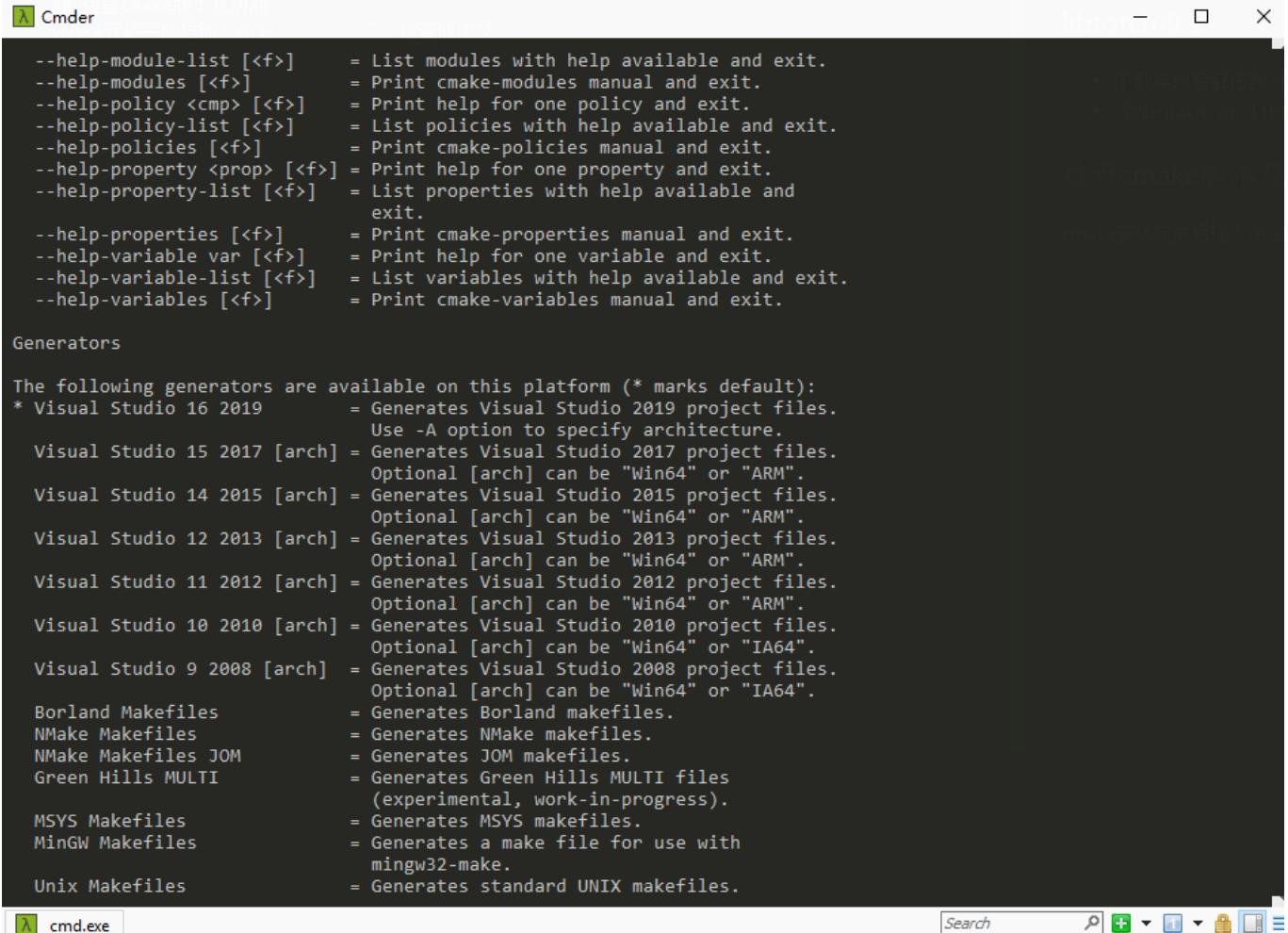
## 在上述方法失效的情况下从cmake重新创建工程

---

以上方法依赖于libtorch现有的项目结构，故可能不适用于未来版本。在这种时候需要从cmake项目开始重新生成sln项目。

### 检查cmake版本与功能

输入命令`cmake -h`，检查generator选项，样例输出如下图



```

Cmder
--help-module-list [<f>]      = List modules with help available and exit.
--help-modules [<f>]          = Print cmake-modules manual and exit.
--help-policy <cmp> [<f>]     = Print help for one policy and exit.
--help-policy-list [<f>]       = List policies with help available and exit.
--help-policies [<f>]         = Print cmake-policies manual and exit.
--help-property <prop> [<f>]   = Print help for one property and exit.
--help-property-list [<f>]    = List properties with help available and
                               exit.
--help-properties [<f>]        = Print cmake-properties manual and exit.
--help-variable var [<f>]      = Print help for one variable and exit.
--help-variable-list [<f>]    = List variables with help available and exit.
--help-variables [<f>]        = Print cmake-variables manual and exit.

Generators

The following generators are available on this platform (* marks default):
* Visual Studio 16 2019           = Generates Visual Studio 2019 project files.
                                    Use -A option to specify architecture.
Visual Studio 15 2017 [arch]     = Generates Visual Studio 2017 project files.
                                    Optional [arch] can be "Win64" or "ARM".
Visual Studio 14 2015 [arch]     = Generates Visual Studio 2015 project files.
                                    Optional [arch] can be "Win64" or "ARM".
Visual Studio 12 2013 [arch]     = Generates Visual Studio 2013 project files.
                                    Optional [arch] can be "Win64" or "ARM".
Visual Studio 11 2012 [arch]     = Generates Visual Studio 2012 project files.
                                    Optional [arch] can be "Win64" or "ARM".
Visual Studio 10 2010 [arch]     = Generates Visual Studio 2010 project files.
                                    Optional [arch] can be "Win64" or "IA64".
Visual Studio 9 2008 [arch]      = Generates Visual Studio 2008 project files.
                                    Optional [arch] can be "Win64" or "IA64".
Borland Makefiles              = Generates Borland makefiles.
NMake Makefiles                = Generates NMake makefiles.
NMake Makefiles JOM             = Generates JOM makefiles.
Green Hills MULTI              = Generates Green Hills MULTI files
                                (experimental, work-in-progress).
MSYS Makefiles                 = Generates MSYS makefiles.
MinGW Makefiles                = Generates a make file for use with
                                mingw32-make.
Unix Makefiles                 = Generates standard UNIX makefiles.

cmd.exe

```

## 手动编写CMakeLists.txt文件，示例如下

```

cmake_minimum_required (VERSION 3.8)

set(CMAKE_CXX_STANDARD 11)

find_package(Torch REQUIRED)

add_executable (CMakeProject "CMakeProject.cpp" "CMakeProject.h")

target_link_libraries(CMakeProject ${TORCH_LIBRARIES})

```

## 使用cmake自带的generator生成vs项目，以vs2015为例

```

cmake -DCMAKE_PREFIX_PATH=C:\Users\chenjz\Downloads\libtorchGPU -
DCMAKE_BUILD_TYPE=Debug -G"Visual Studio 14 2015 Win 64" .

```

若上述操作无误项目文件夹应该如下图，`Project.sln`为生成项目。

名称	修改日期	类型	大小
CMakeFiles	2019/4/5 20:46	文件夹	
ALL_BUILD.vcxproj	2019/4/5 20:46	VC++ Project	45 KB
ALL_BUILD.vcxproj.filters	2019/4/5 20:46	VC++ Project Fil...	1 KB
cmake_install.cmake	2019/4/5 20:46	CMAKE 文件	2 KB
CMakeCache.txt	2019/4/5 20:46	文本文档	15 KB
CMakeLists.txt	2019/4/4 20:59	文本文档	1 KB
CMakeProject.cpp	2019/4/5 9:28	C++ 源文件	1 KB
CMakeProject.h	2019/4/4 19:42	C Header 源文件	1 KB
CMakeProject.vcxproj	2019/4/5 20:46	VC++ Project	57 KB
CMakeProject.vcxproj.filters	2019/4/5 20:46	VC++ Project Fil...	1 KB
Project.sln	2019/4/5 20:46	Visual Studio Sol...	4 KB
ZERO_CHECK.vcxproj	2019/4/5 20:46	VC++ Project	44 KB
ZERO_CHECK.vcxproj.filters	2019/4/5 20:46	VC++ Project Fil...	1 KB

## 项目说明

Project.sln共包含三个项目：ALL\_BUILD, CMakeProject, ZERO\_CHECK, 其中CMakeProject是我们想要的。

The screenshot shows the Visual Studio interface. The Solution Explorer on the left lists three projects: ALL\_BUILD, CMakeProject, and ZERO\_CHECK. The CMakeProject node under Source Files contains a CMakeProject.cpp file. The code editor on the right displays the contents of CMakeProject.cpp:

```

1 // // CMakeProject.cpp : Defines the entry point for the application.
2 //
3
4 #include "CMakeProject.h"
5 #include <torch/torch.h>
6
7 using namespace std;
8
9 int main()
10 {
11     torch::Tensor tensor = torch::eye(3);
12     std::cout << tensor << std::endl;
13     cout << "Hello CMake." << endl;
14     return 0;
15 }
16

```

CMakeProject下的External Dependencies里包含了所有与libtorch有关的头文件，配置好d11路径后若无错误就可以直接编译运行了。

## 在HM中调用代码

我在github上提供了一个HM16.20的cmake项目，大家可以在对应位置修改CMakeLists.txt来在指定项目调用libtorch。

- 在C++中加载pytorch模型教程[https://pytorch.org/tutorials/advanced/cpp\\_export.html](https://pytorch.org/tutorials/advanced/cpp_export.html)

- HM cmake工程[https://github.com/chjz1024/HM\\_vc2015](https://github.com/chjz1024/HM_vc2015)