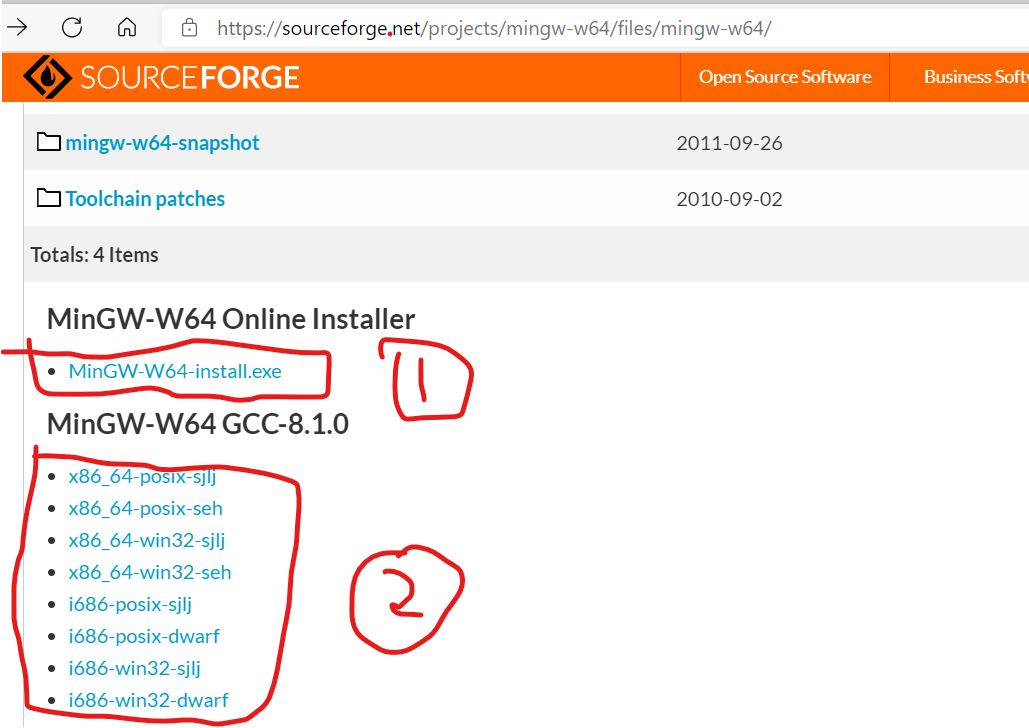
# VSCode下配置OpenCV

## 配置 C++环境

可以选择按章MinGW或者是Visual Studio，另外OpenCV使用CUDA加速只能用Visual Studio进行编译。

### 安装C++编译器

#### 1.1.1 安装MinGW编译器



**下载地址：**<https://sourceforge.net/projects/mingw-w64/files/mingw-w64/>

(架构)Architecture 32位cpu 选择 i686，64位cpu 选择 x86\_64

(线程模式)Threads 如果使用 posix api 选择 posix，如果使用 win32 api 选择win32

(异常处理机制) Exception 有三种，根据机器情况进行选择

SJLJ (setjmp/longjmp) 兼容性好，32位或64位架构都可以使用，但速度慢

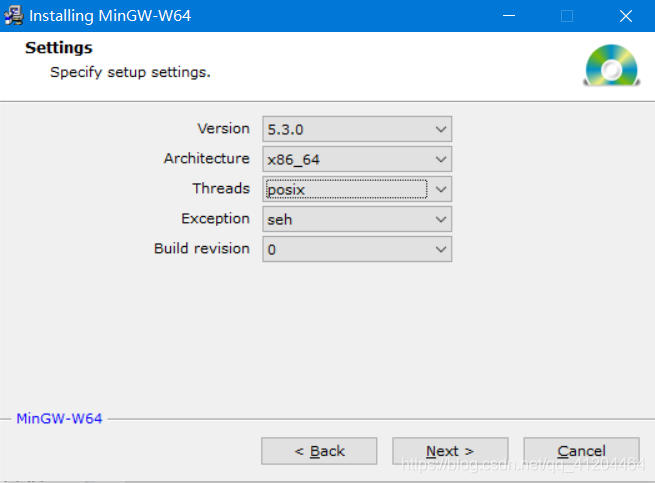
DWARF (DW2, dwarf-2) 速度比SJLJ快，但只能用于32位架构，且有bug，尽量不用

SEH (zero overhead exception) 速度很快，但只能用于64位架构；

(编译版本)Build revision 选择最高的

**注意：在上图中1为在线下载（可能会比较慢），可以在上图中2区域选择对应版本下载**

##### 1.1.1.1在线下载



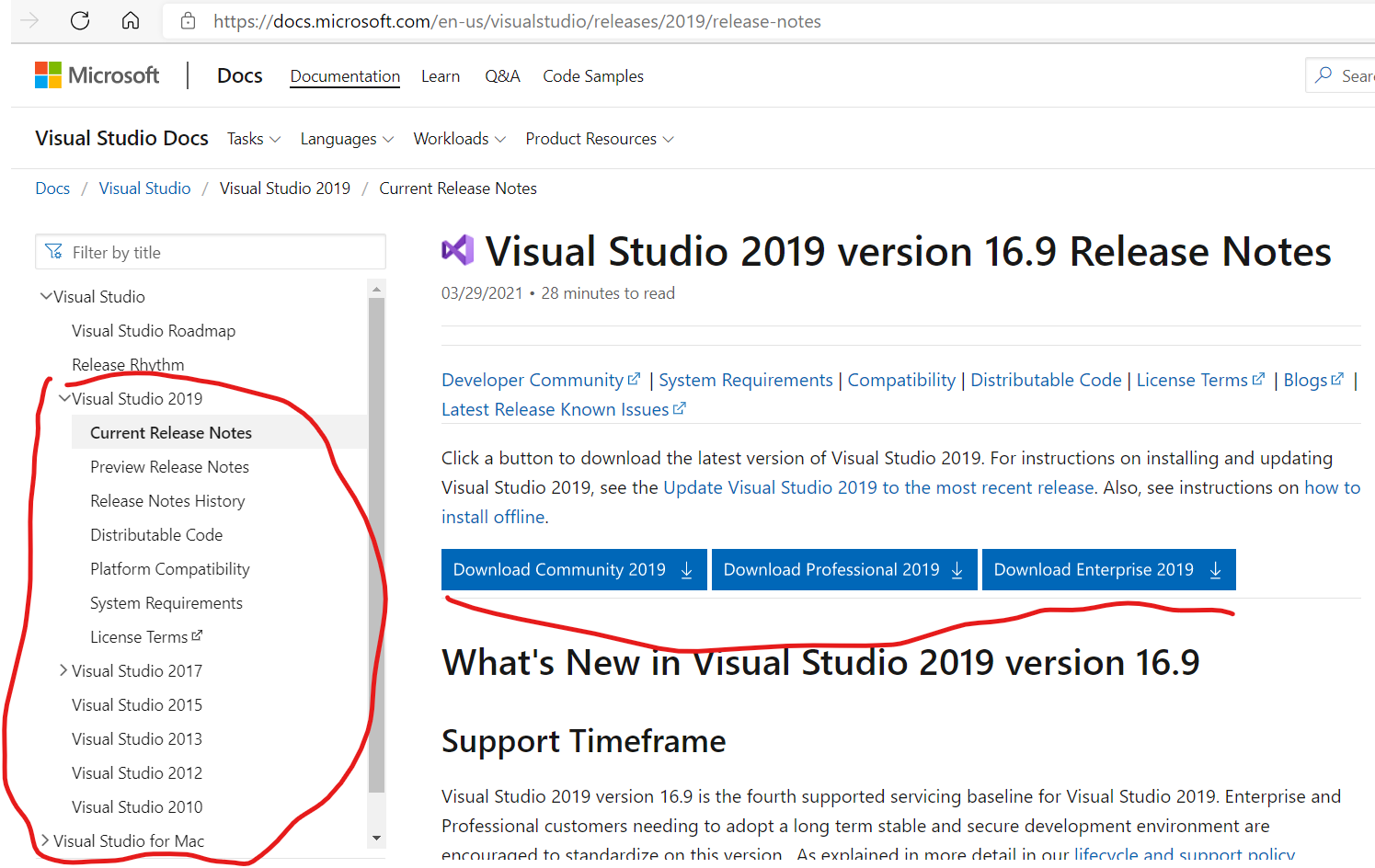
##### 1.1.1.2 下载的对应版本的压缩包

直接解压，然后进入下一步，配置环境

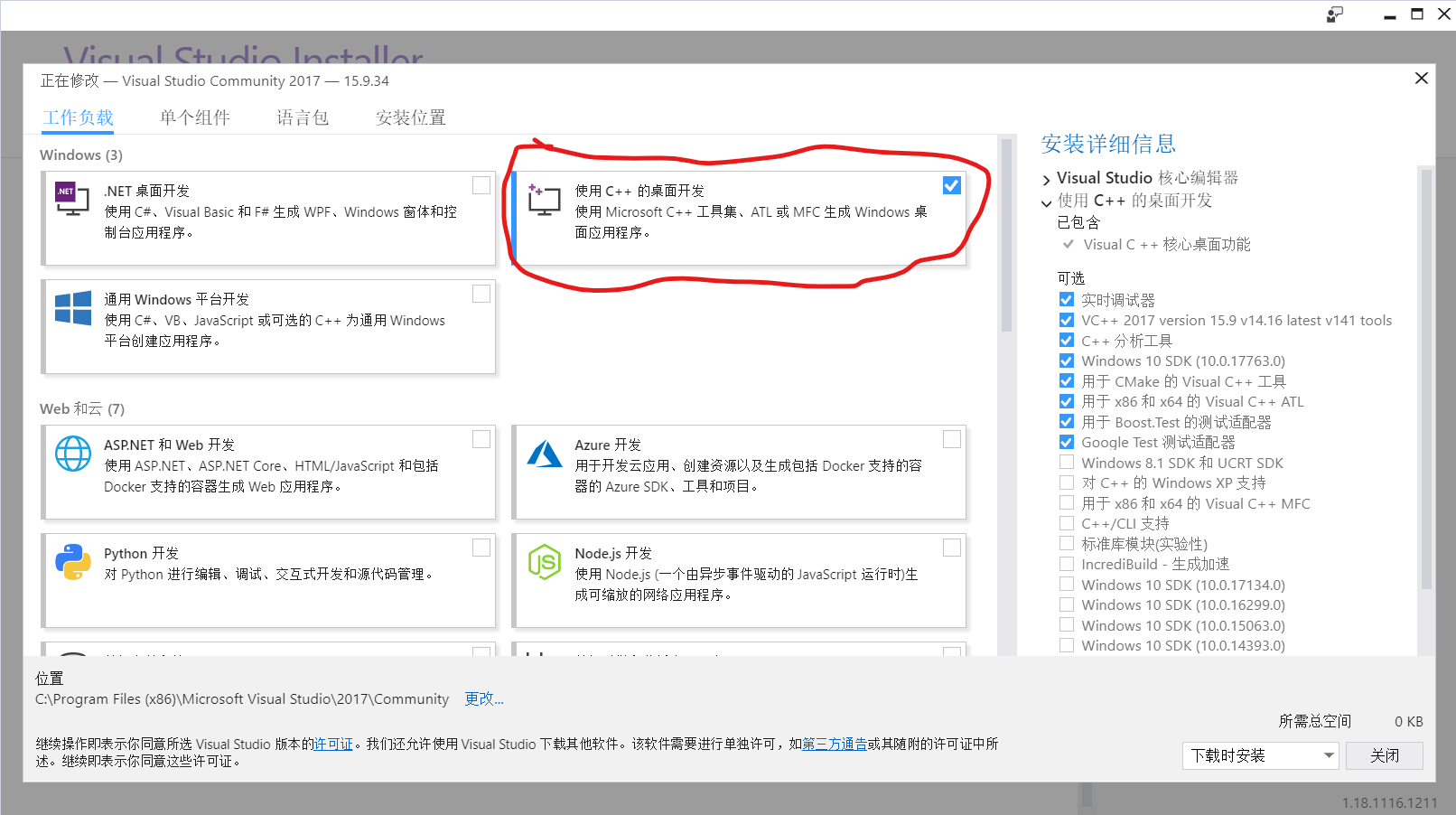
#### 1.1.2 下载安装Visual Studio

下载地址<https://docs.microsoft.com/en-us/visualstudio/productinfo/vs-roadmap>

选择相应的版本下载安装Visual Studio



安装时根据自己的需要选择需要的开发包以及组件，如果只是OpenCV的话，点上【使用c++的桌面开发】即可。

****

下载安装好之后，就可以配置VSCode的环境了。

### 1.2 配置环境变量

#### 1.2.1 Win+I 打开windows设置

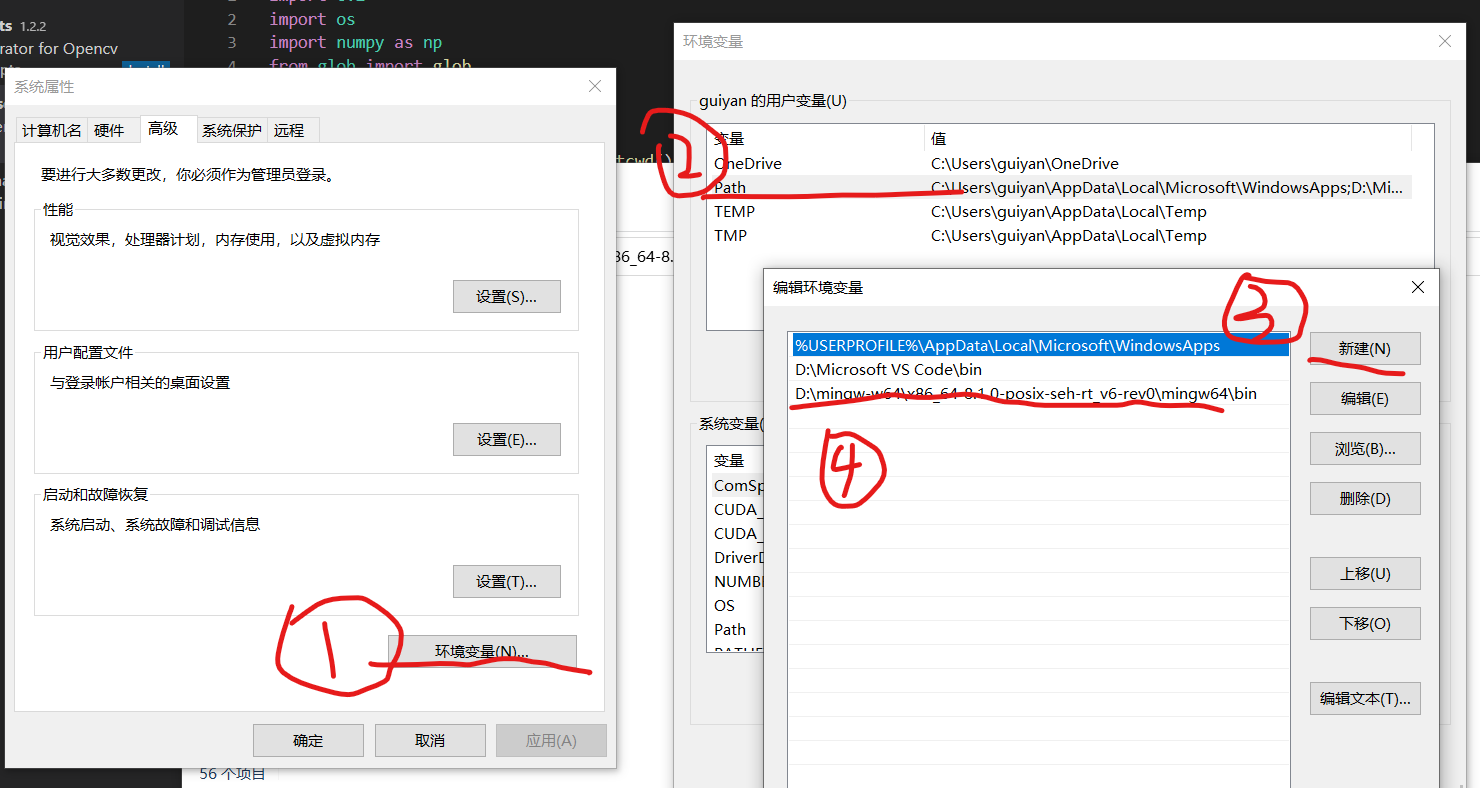




#### 1.2.2 添加对应版本的bin路径

配置MinGW对应的路径

D:\mingw-w64\x86\_64-8.1.0-posix-seh-rt\_v6-rev0\mingw64\bin



配置Visual Studio对应路径：

C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.27023\bin\Hostx64\x64

添加一个INCLUDE环境变量

C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.27023\include

C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\shared

C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\ucrt

C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\um

C:\Program Files (x86)\Windows Kits\10\Include\10.0.17763.0\winrt

D:\OpenCV4\opencv\_cuda\_install\_vc15\_x64\include

D:\OpenCV4\opencv\_cuda\_install\_vc15\_x64\include\opencv2

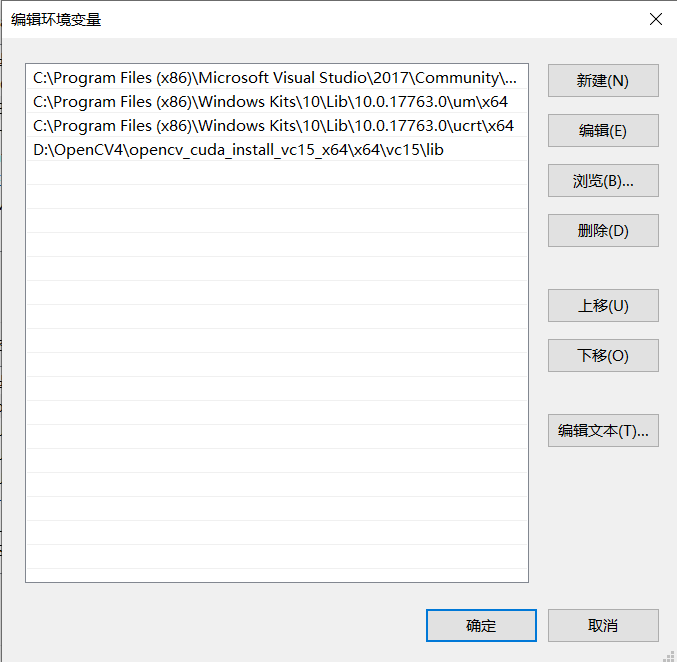
添加一个LIB变量

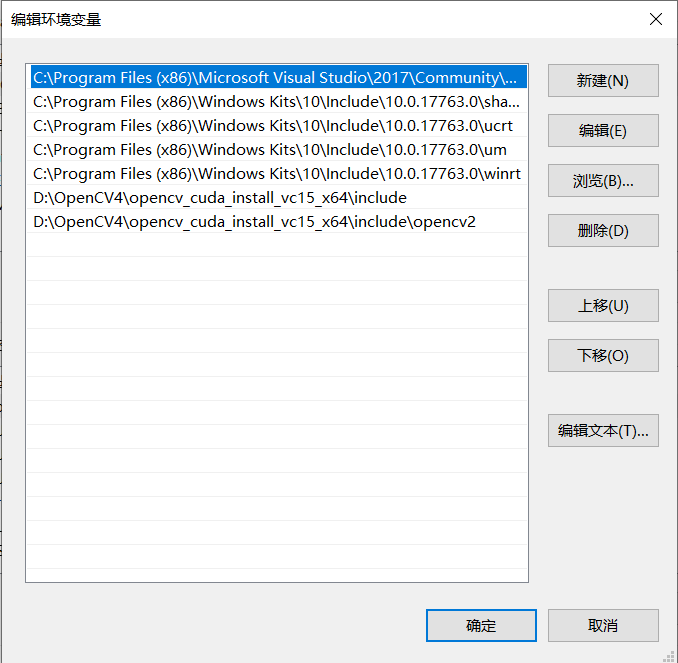
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.27023\lib\x64

C:\Program Files (x86)\Windows Kits\10\Lib\10.0.17763.0\um\x64

C:\Program Files (x86)\Windows Kits\10\Lib\10.0.17763.0\ucrt\x64

D:\OpenCV4\opencv\_cuda\_install\_vc15\_x64\x64\vc15\lib

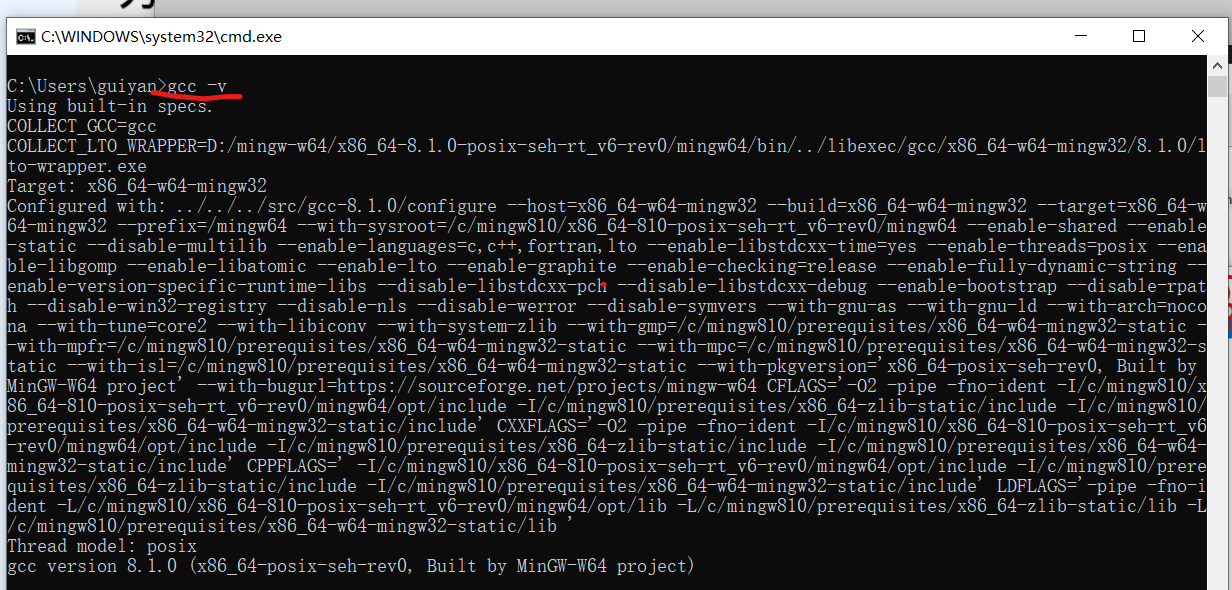


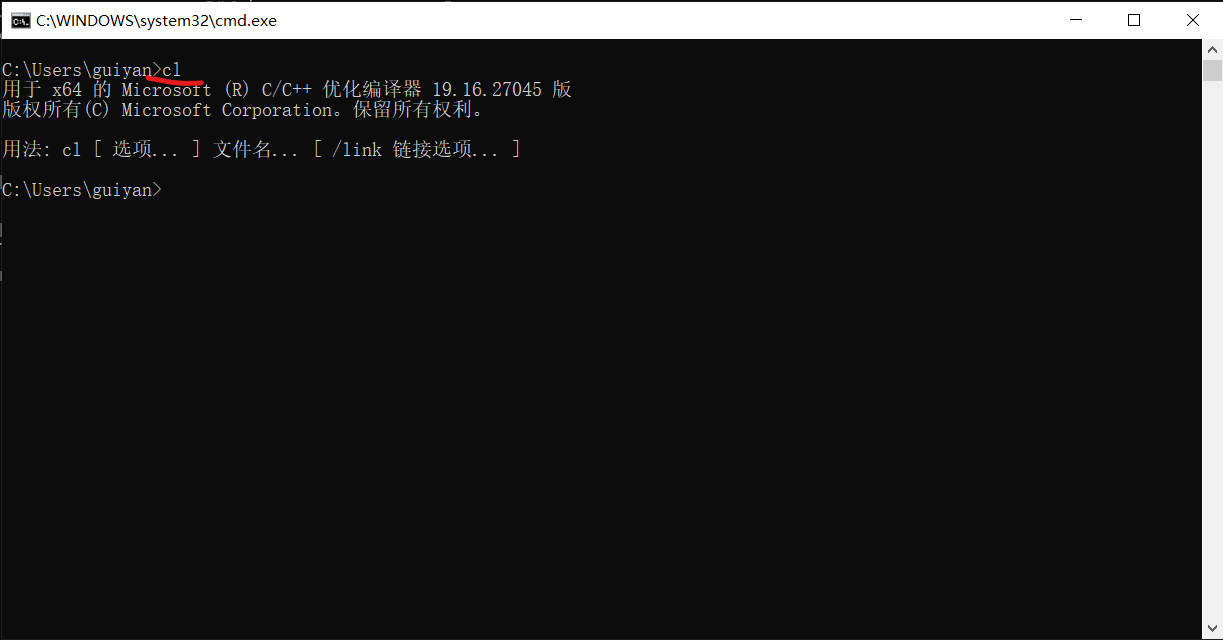


#### Cmd中验证是否正确添加

MinGW 输入gcc -v

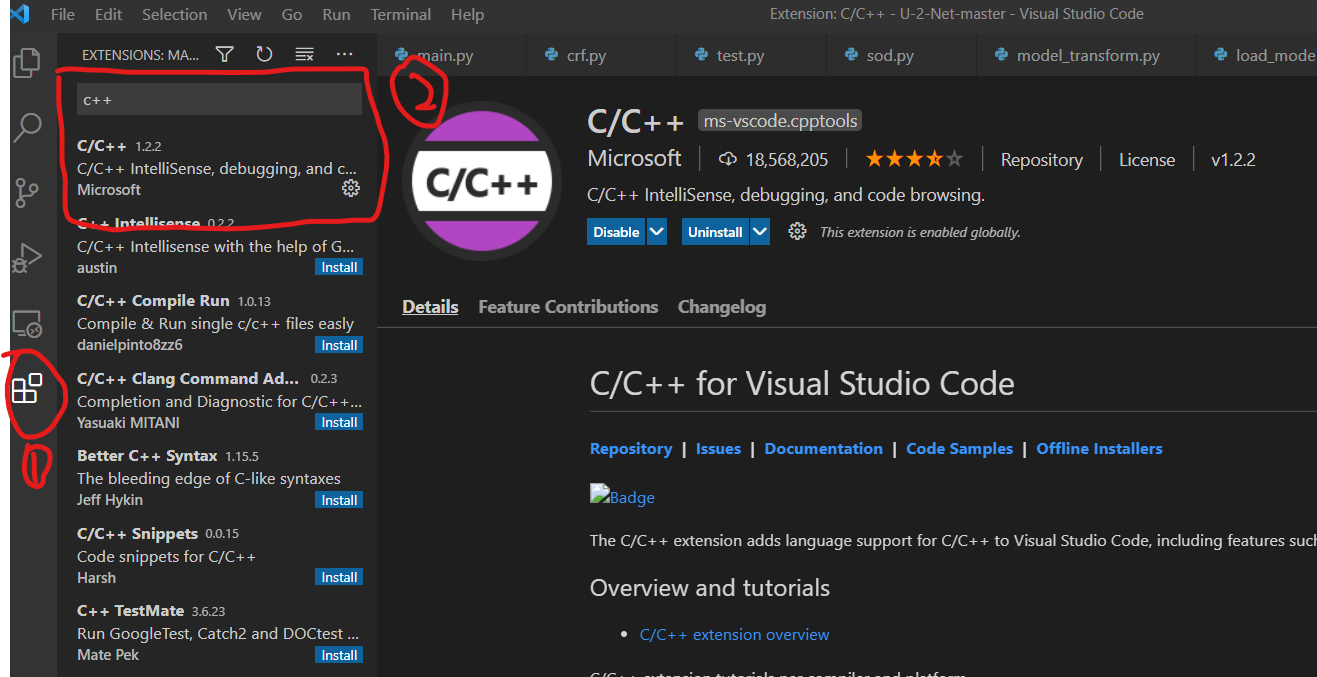
Visual Studio 输入 cl



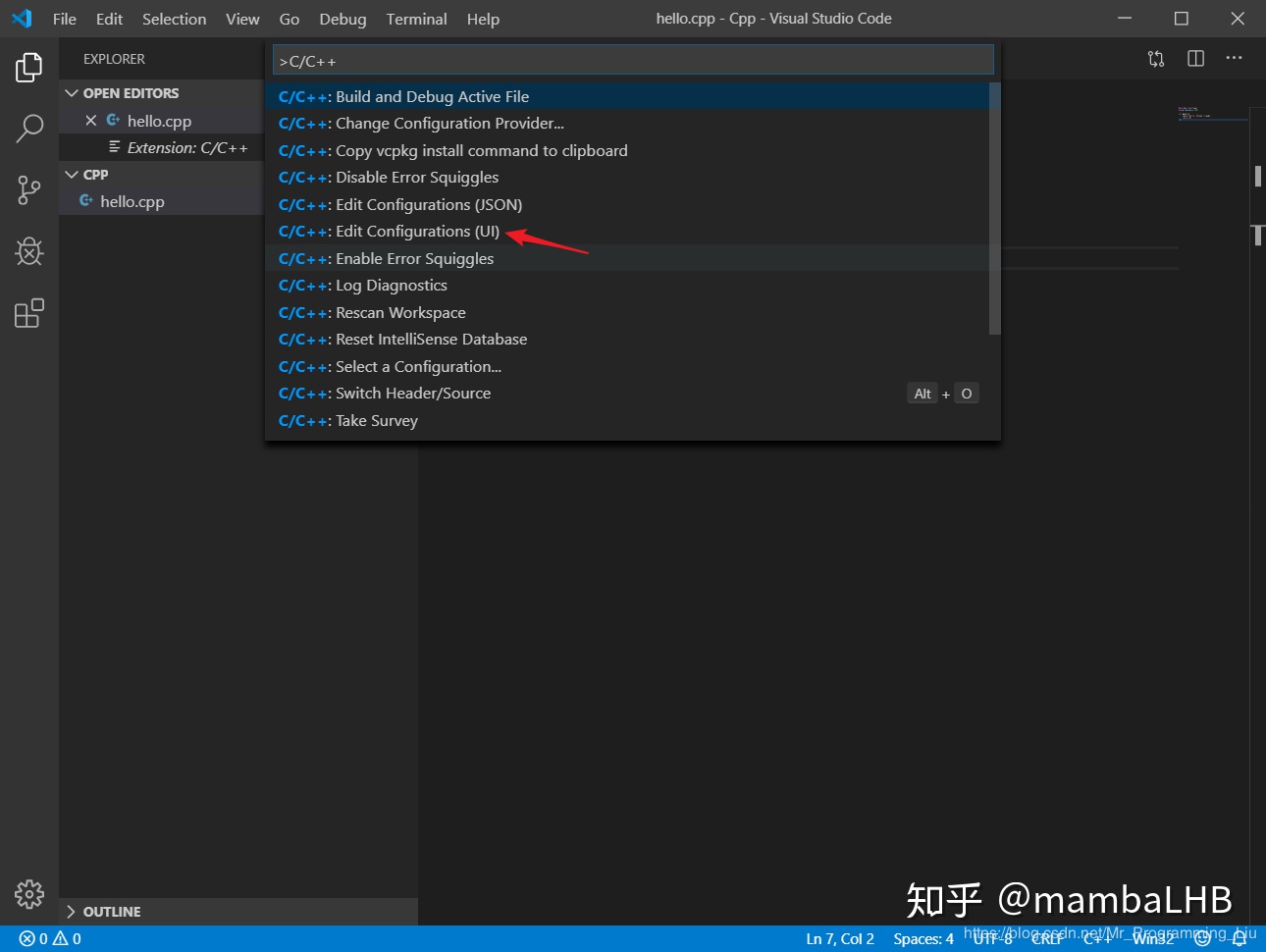


### 1.3 在VSCode中配置环境

#### 1.3.1下载C/C++插件



#### 1.3.2 配置C/C++环境



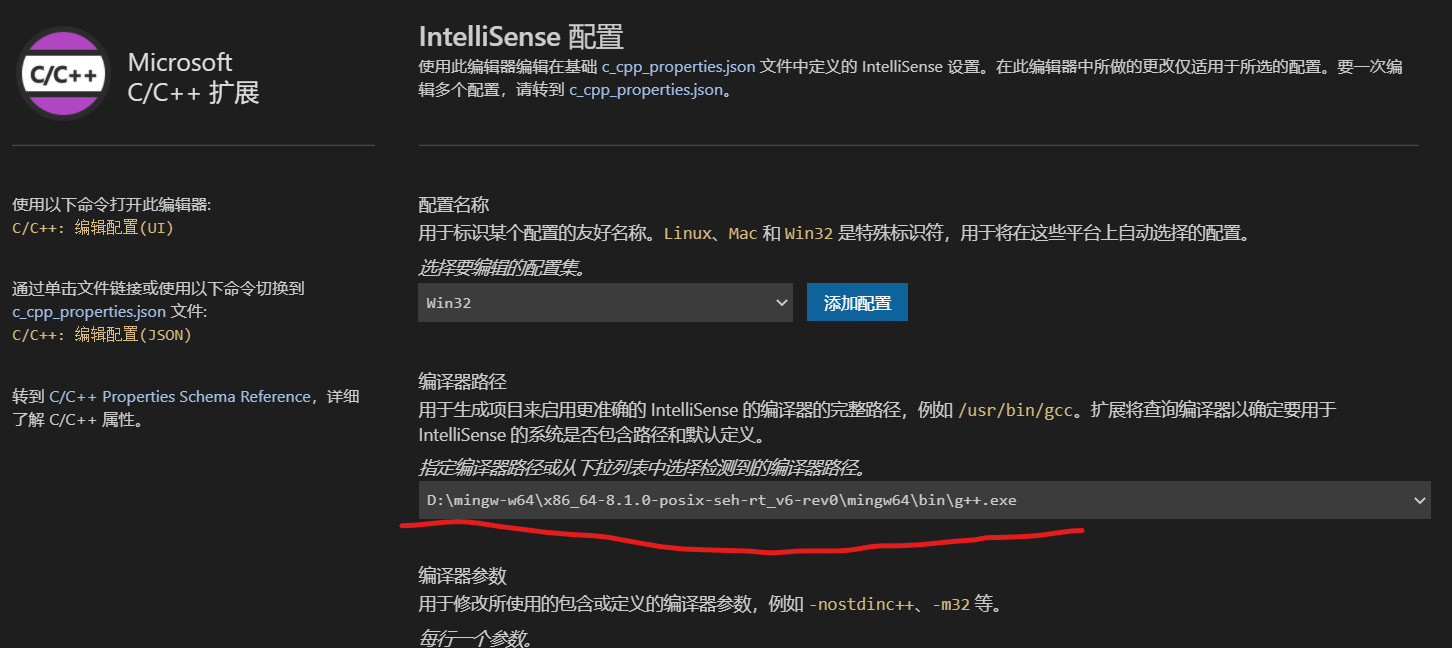
添加编译器路径

MinGW

D:\mingw-w64\x86\_64-8.1.0-posix-seh-rt\_v6-rev0\mingw64\bin\g++.exe

Visual Studio

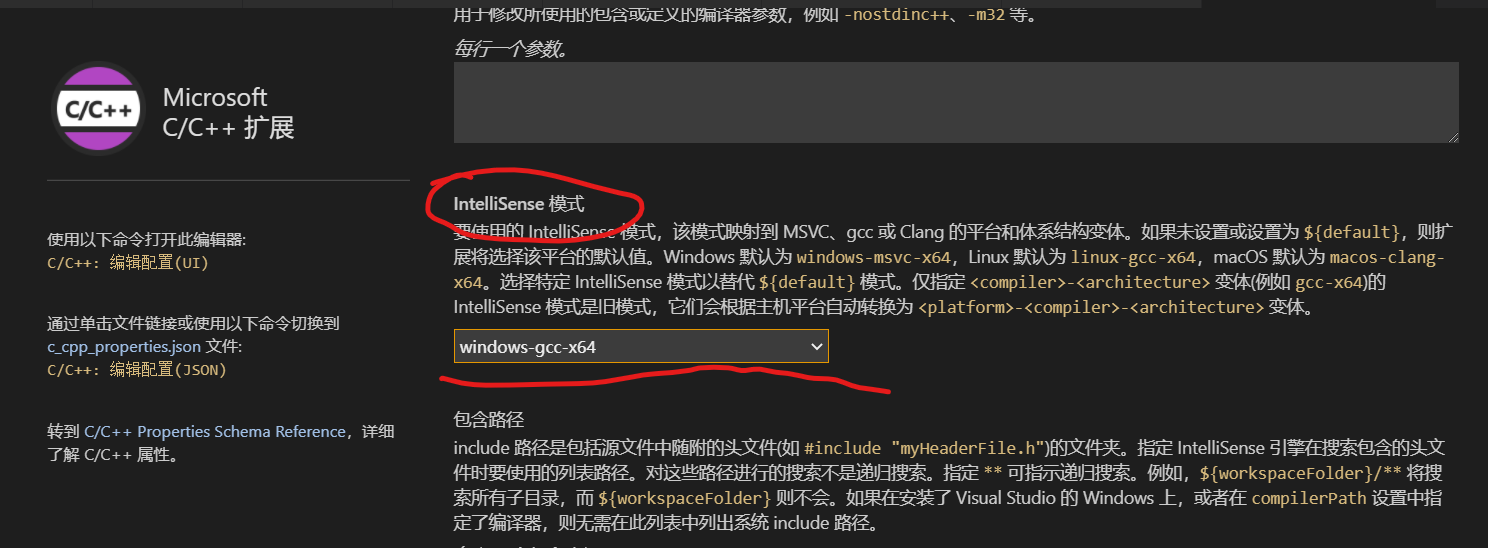
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.27023\include

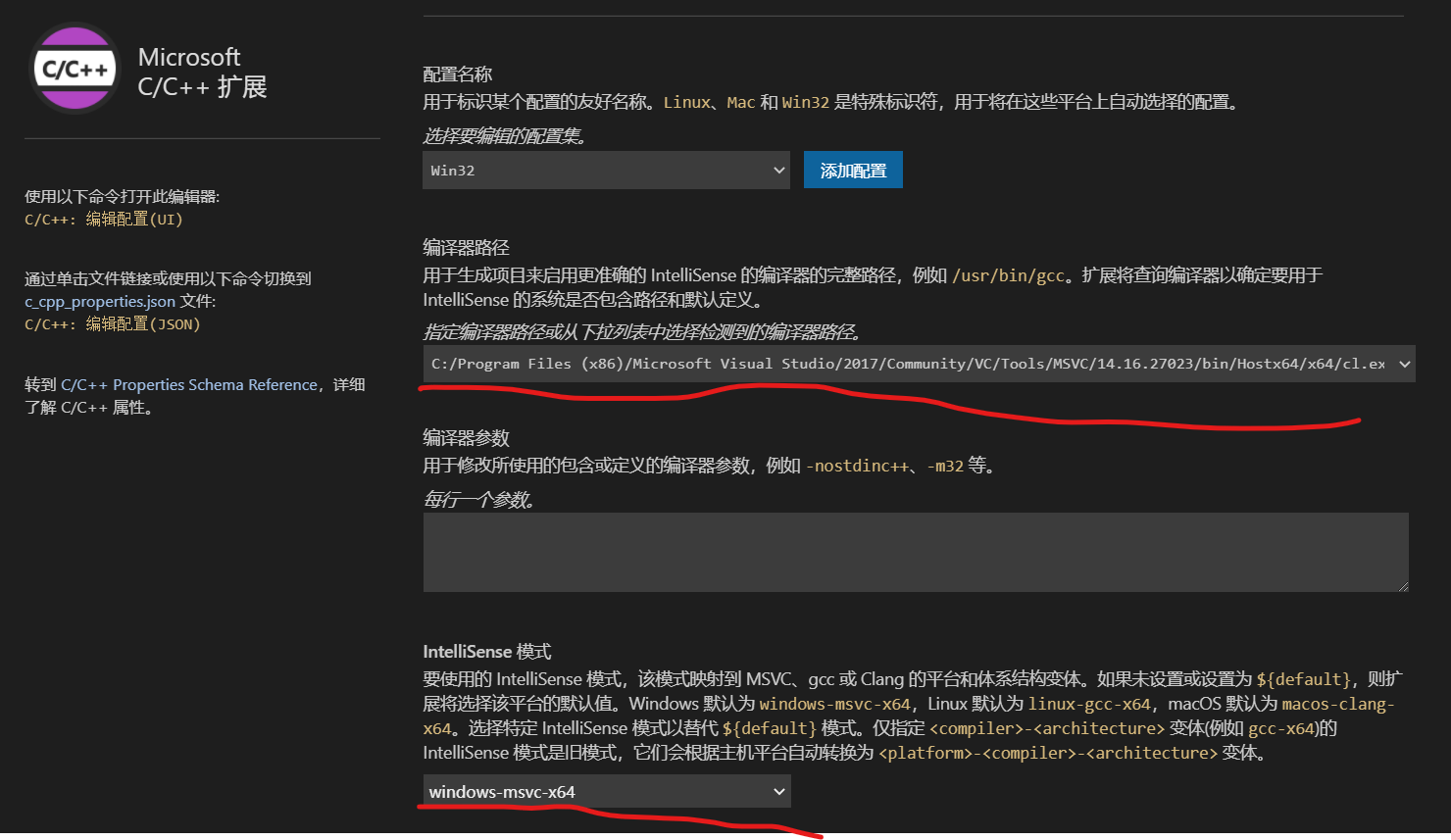


MinGW

选择gcc-x64代码提示模式（注意对应系统）

Visual Studio

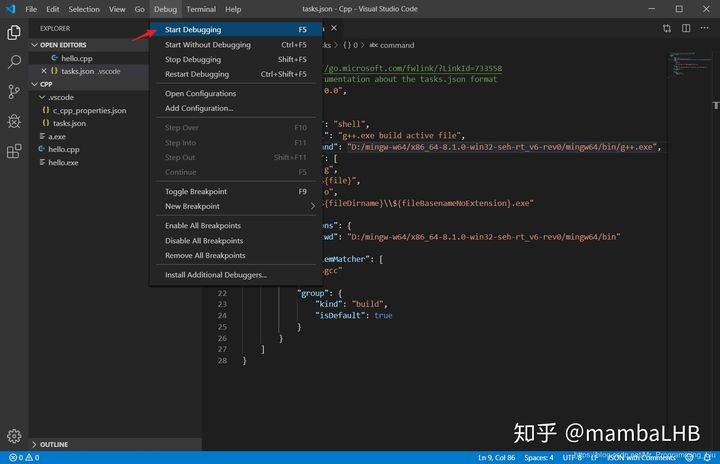




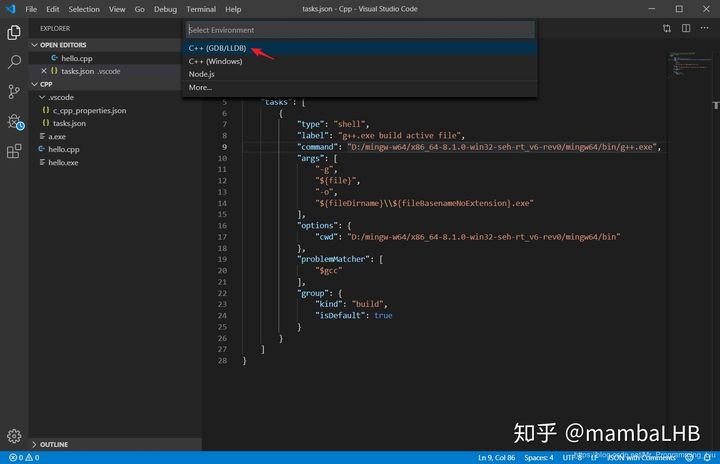
#### 1.3.3 配置调试设置

##### 1.3.3.1 MinGW

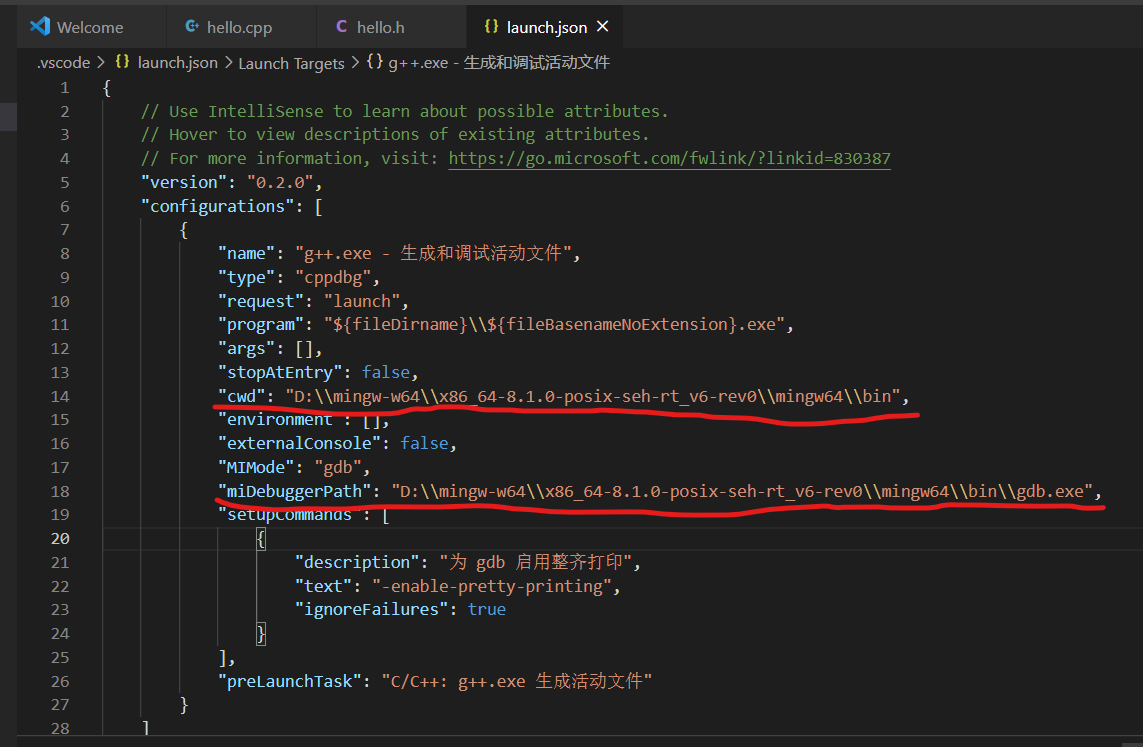
这里主要是为了在.vscode文件夹中产生一个launch.json文件，用来配置调试的相关信息。点击菜单栏的*Debug*-->*Start Debugging*：



选择C++(GDB/LLDB)：



紧接着会产生一个launch.json的文件：添加gdb.exe路径



##### 1.3.3.2 Visual Studio

使用MinGW类似的方法生成相关，也可以直接创建。

**c\_cpp\_properties.json**

{

    "configurations": [

        {

            "name": "Win32",

            "includePath": [

                "${workspaceFolder}/\*\*",

                "C:\\Program Files (x86)\\Microsoft Visual Studio\\2017\\Community\\VC\\Tools\\MSVC\\14.16.27023\\include\\\*\*",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\include\\\*",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\include\\opencv2"

                // "D:\\vs2015\\VC\\include\\\*\*",

                // "D:\\opencv\\build\\include\\\*\*",

                // "D:\\opencv\\build\\include\\opencv2"

            ],

            "defines": [

                "\_DEBUG",

                "UNICODE",

                "\_UNICODE"

            ],

            "windowsSdkVersion": "10.0.17763.0",

            "compilerPath": "C:/Program Files (x86)/Microsoft Visual Studio/2017/Community/VC/Tools/MSVC/14.16.27023/bin/Hostx64/x64/cl.exe",

            "cStandard": "c17",

            "cppStandard": "c++17",

            "intelliSenseMode": "windows-msvc-x64"

        }

    ],

    "version": 4

}

**tasks.json**

**这里是opencv\_world.dll只需要加入opencv\_world.dll的位置即可。**

{

    "version": "2.0.0",

    "tasks": [

        {

            "type": "cppbuild",

            "label": "C/C++: cl.exe 生成活动文件",

            "command": "cl.exe",

            "args": [

                "/Zi",

                "/EHsc",

                "/nologo",

                "/Fe:",

                "${fileDirname}\\${fileBasenameNoExtension}.exe",

                "${file}",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_core451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_imgproc451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_imgcodecs451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_video451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_ml451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_highgui451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_objdetect451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_flann451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_photo451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_videoio451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_features2d451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_dnn451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_stitching451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_calib3d451.lib",

                "D:\\OpenCV4\\opencv\_cuda\_install\_vc15\_x64\\x64\\vc15\\lib\\opencv\_gapi451.lib",

            ],

            "options": {

                "cwd": "${workspaceFolder}"

            },

            "problemMatcher": [

                "$msCompile"

            ],

            "group": {

                "kind": "build",

                "isDefault": true

            },

            "detail": "编译器: cl.exe"

        }

    ]

}

#### 1.3.4 编写helloworld代码验证是否可以编译运行

#include<iostream>

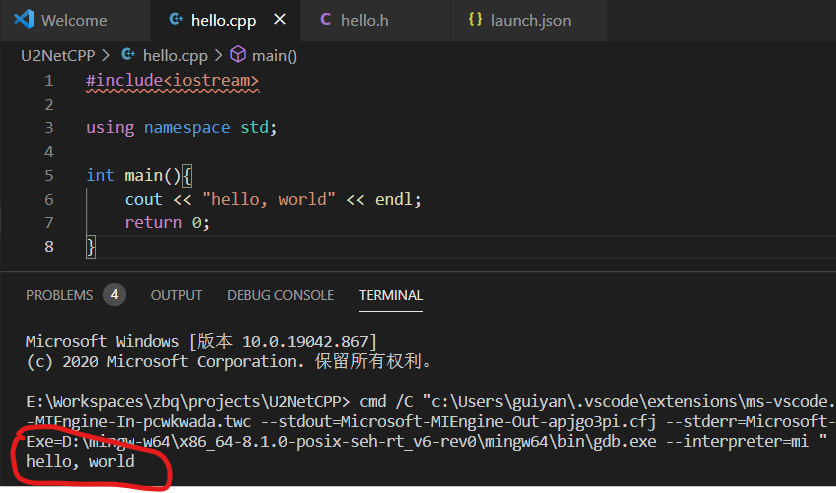
using namespace std;

int main(){

    cout << "hello, world" << endl;

    return 0;

}

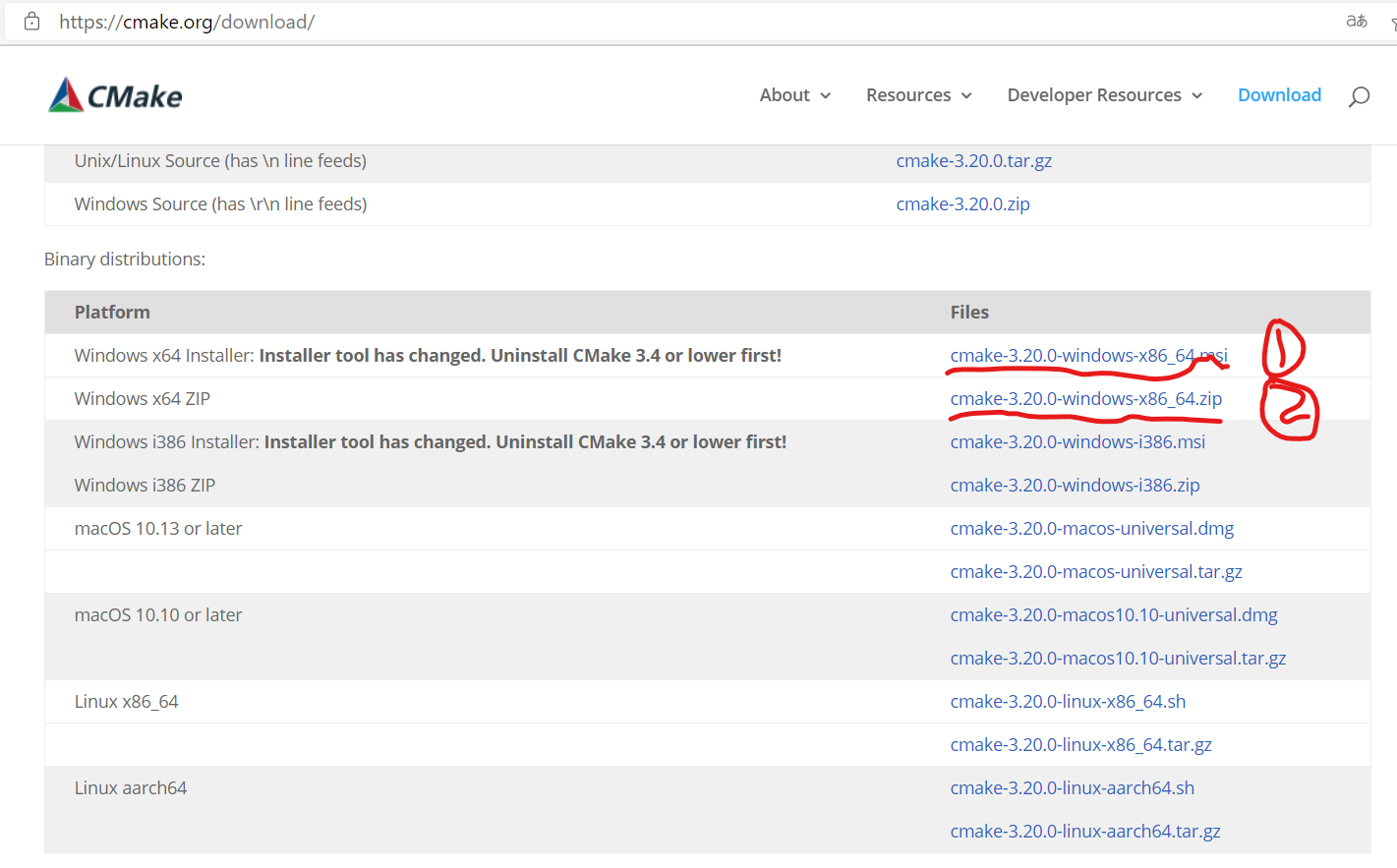


## 配置OpenCV环境

### 2.1 安装CMake

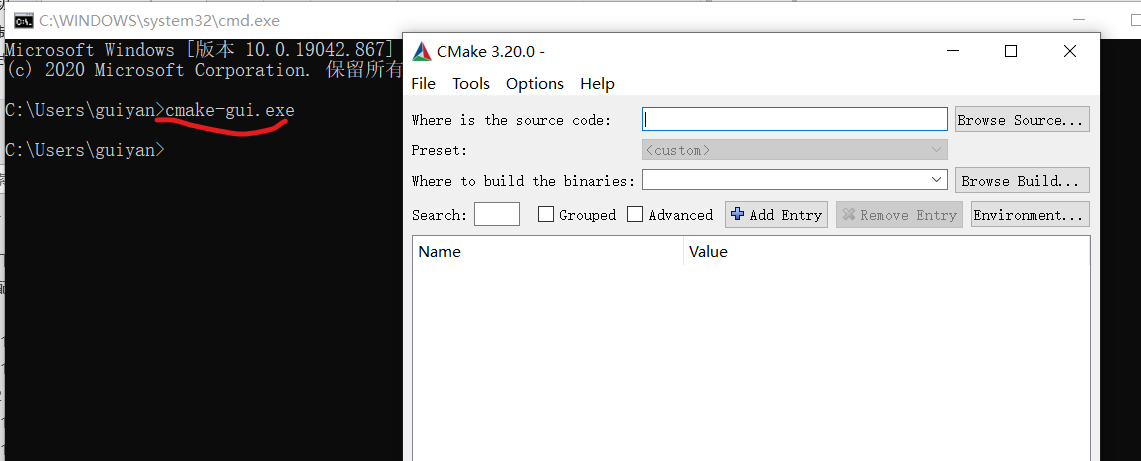
#### 2.1.1 下载安装Cmake

在<https://cmake.org/download/> 下载CMake，用来编译OpenCV，下载msi文件可以免去添加环境变量的步骤，下载zip压缩包，需要自行将CMake目录下的bin目录添加到环境变量中。



#### 2.1.2 验证是否安装配置好CMake

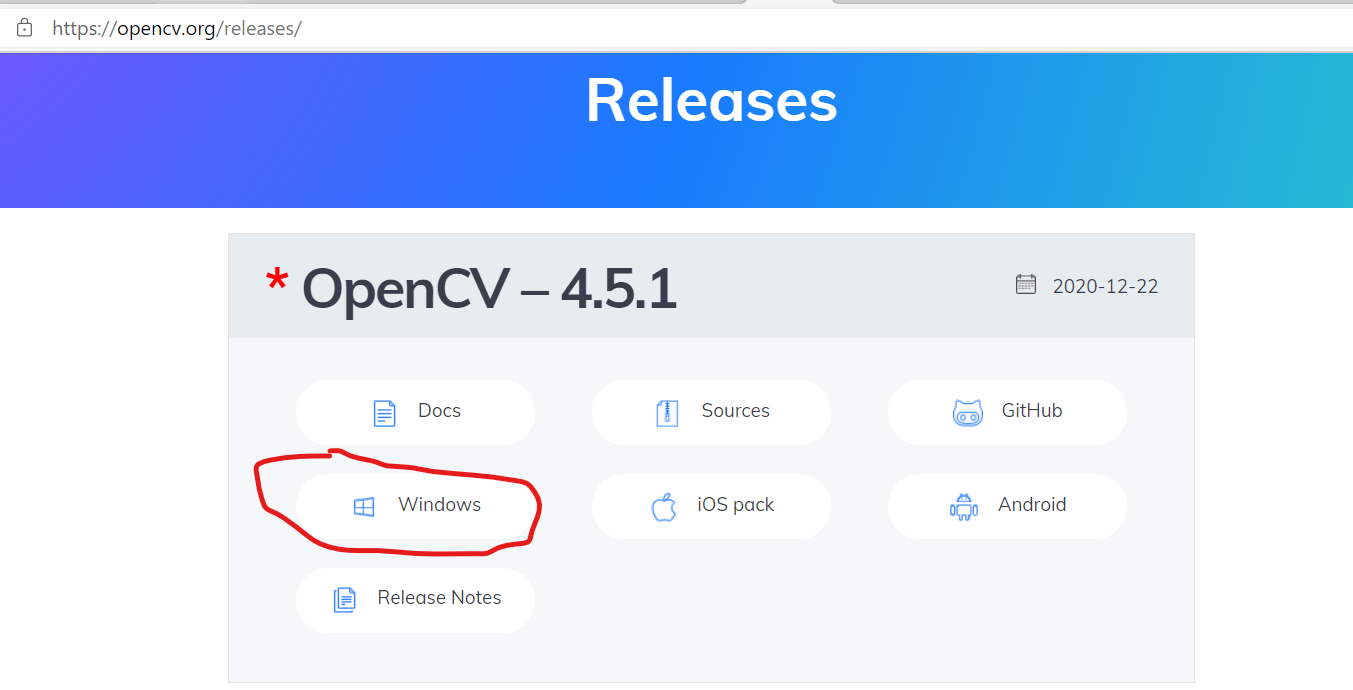
在cmd中输入cmake-gui.exe，是否弹出CMake程序



### 2.2 编译OpenCV

#### 2.2.1 下载解压OpenCV

在<https://opencv.org/releases/>下载，点击exe，解压到一个目录下。

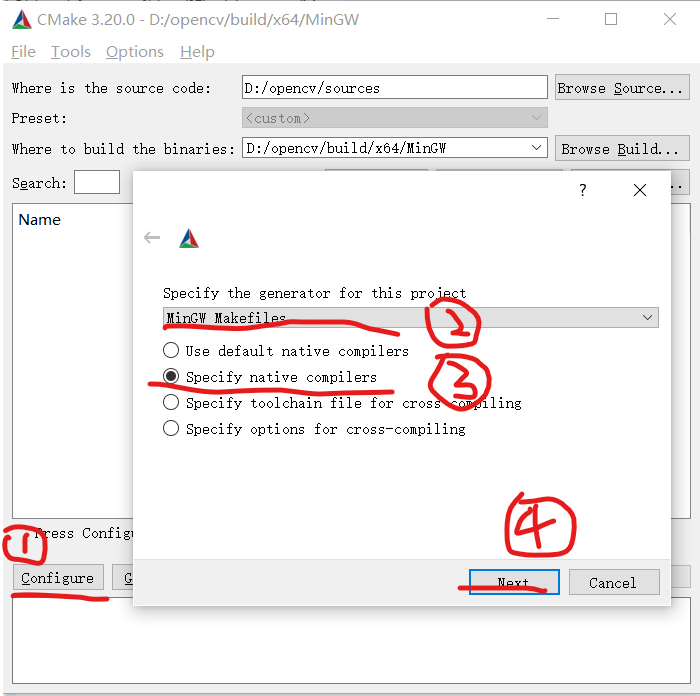


#### 2.2.2 配置并生成OpenCV相关源文件

【where is the source code】在cmake-gui中点击【Browse Source】并选择OpenCV安装的路径下的sources目录。

在**opencv\build\x64**路径下新建一个文件夹（可自起，这里是**MinGW**），用来存放等下编译的文件

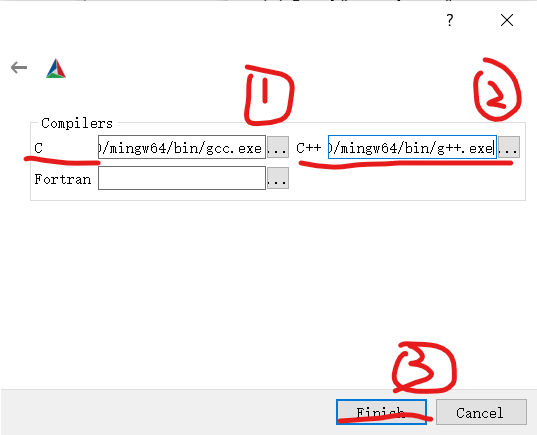
【where to build the binaries】选择上面创建好的文件夹



点击【configure】选择 MinGW 中的 C 和 C++ 编译的exe程序，分别选择自己的mingw安装路径bin文件夹下的gcc与g++

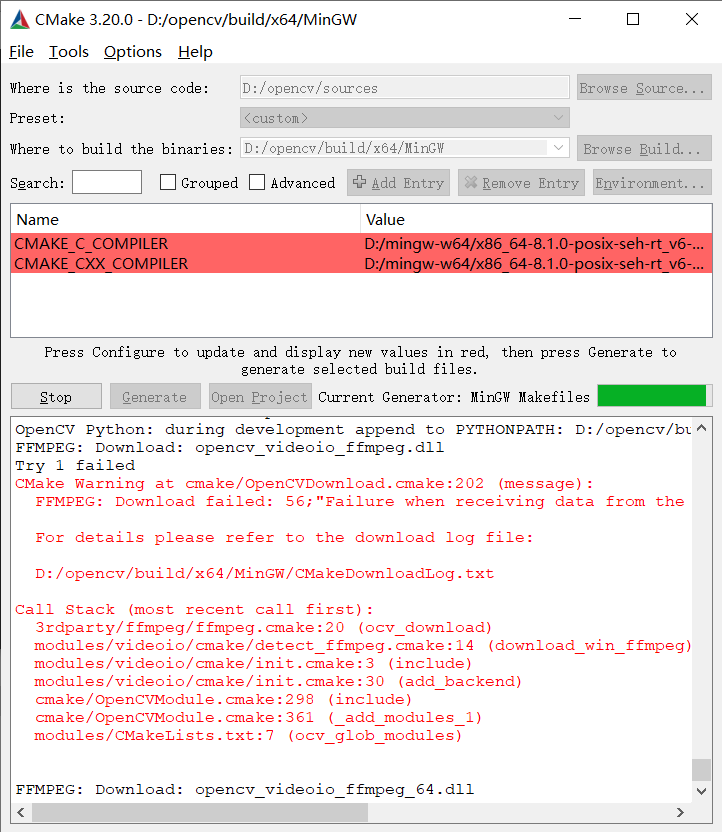
C：x86\_64-w64-mingw32-gcc.exe

C++ : x86\_64-w64-mingw32-g++.exe



设置完毕点击 **Finish**，耐心等待进度条读完，直到输出窗口看到 **Configuring done**

#### 2.2.3 解决opencv\_videoio\_ffmpeg\_64.dll下载错误问题

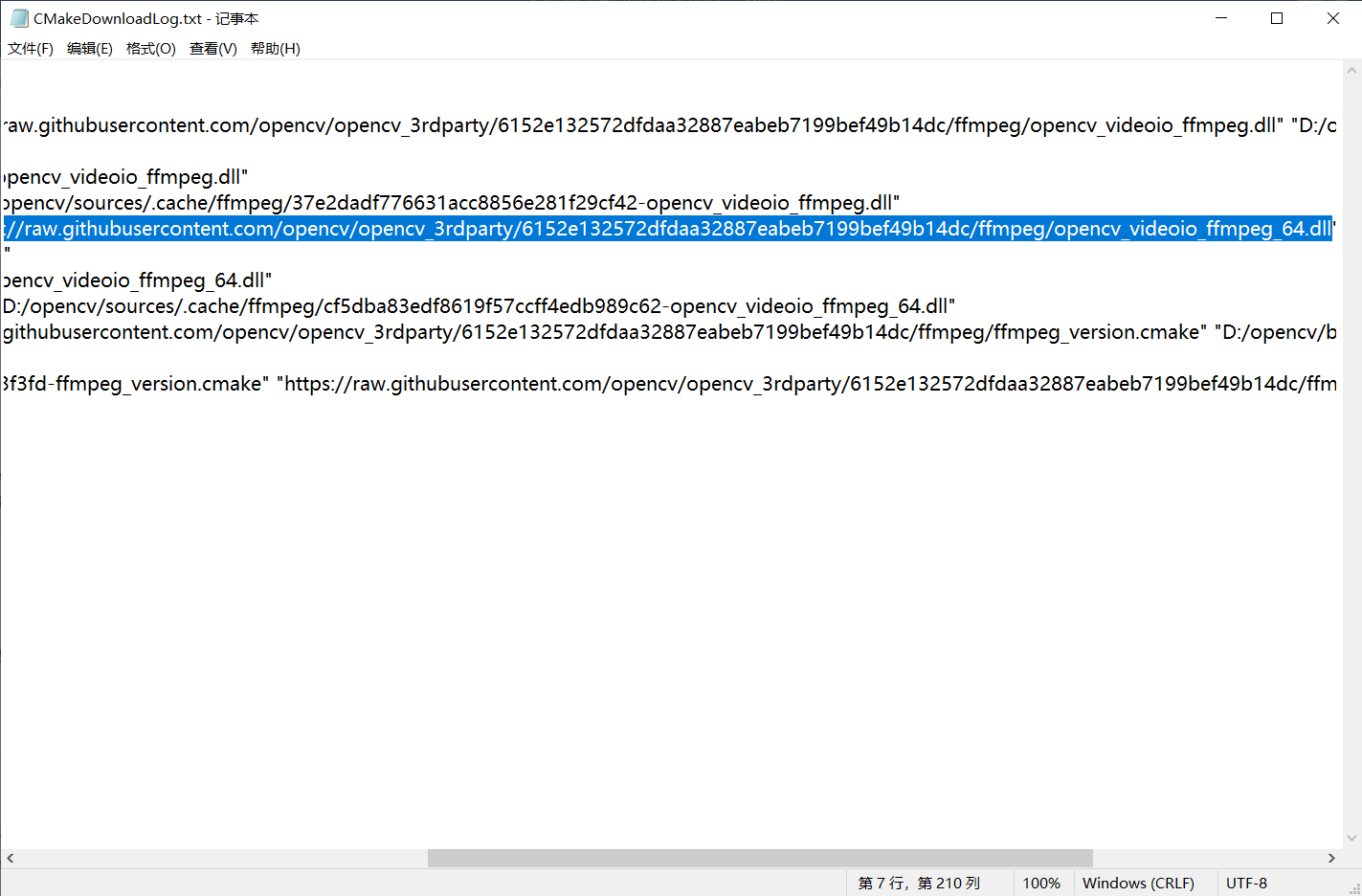


##### 2.2.3.1 下载opencv\_videoio\_ffmpeg动态库文件包括opencv\_videoio\_ffmpeg\_64.dll以及opencv\_videoio\_ffmpeg.dll

编译时需要opencv\_videoio\_ffmpeg\_64.dll，而下载可能比较慢，我们可以找到输出编译结果文件夹（D:\opencv\build\x64\MinGW）中的CMakeDownloadLog.txt（我这里完整的路径是D:\opencv\build\x64\MinGW\CMakeDownloadLog.txt），复制其中github的链接输入到浏览器URL搜索栏中，下载文件，并修改后面的链接下载64位的文件。例如链接如下：

https://raw.githubusercontent.com/opencv/opencv\_3rdparty/6152e132572dfdaa32887eabeb7199bef49b14dc/ffmpeg/opencv\_videoio\_ffmpeg.dll

https://raw.githubusercontent.com/opencv/opencv\_3rdparty/6152e132572dfdaa32887eabeb7199bef49b14dc/ffmpeg/opencv\_videoio\_ffmpeg\_64.dll



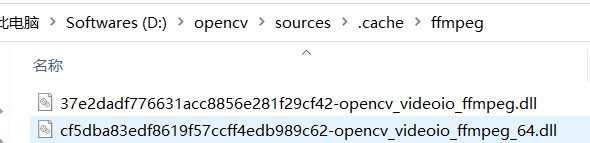
##### 2.2.3.2 将opencv\_videoio\_ffmpeg动态库文件放到指定位置

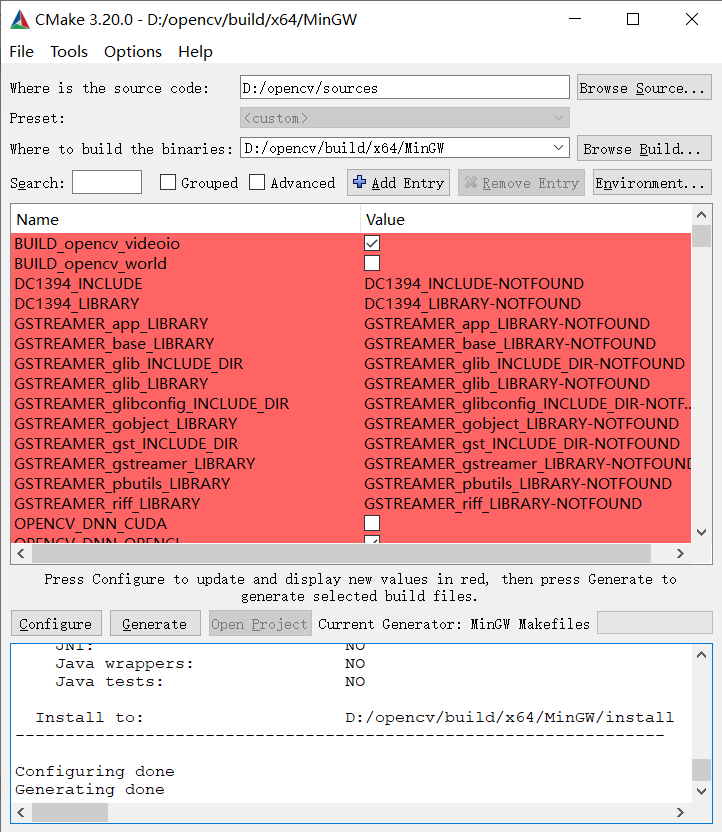
将两个opencv\_videoio\_ffmpeg动态库文件放到opencv源文件的.cache\ffmpeg （我这里是在D:\opencv\sources\.cache\ffmpeg）

这时候先打开任务管理器结束cmake-gui，然后再将刚刚复制过来的文件对应改名为已存在的两个未下载成功的dll文件

再重新启动cmake-gui点击configure，等待，直到看到Configuring done

再点击Generate。





##### 2.2.3.3 另一种获取下载URL方法

进入到OpenCV的源文件目录下的3rdparty\ffmpeg 路径下（我这里完整目录为D:\opencv\sources\3rdparty\ffmpeg）

打开目录下ffmpeg.cmake文件和ffmpeg-download.ps1.in，用ffmpeg.cmake文件中对应的FFMPEG\_BINARIES\_COMMIT值代替ffmpeg-download.ps1.in文件中url的参数名，如下图所示，如：

ffmpeg.cmake：

ocv\_update(FFMPEG\_BINARIES\_COMMIT "6152e132572dfdaa32887eabeb7199bef49b14dc")

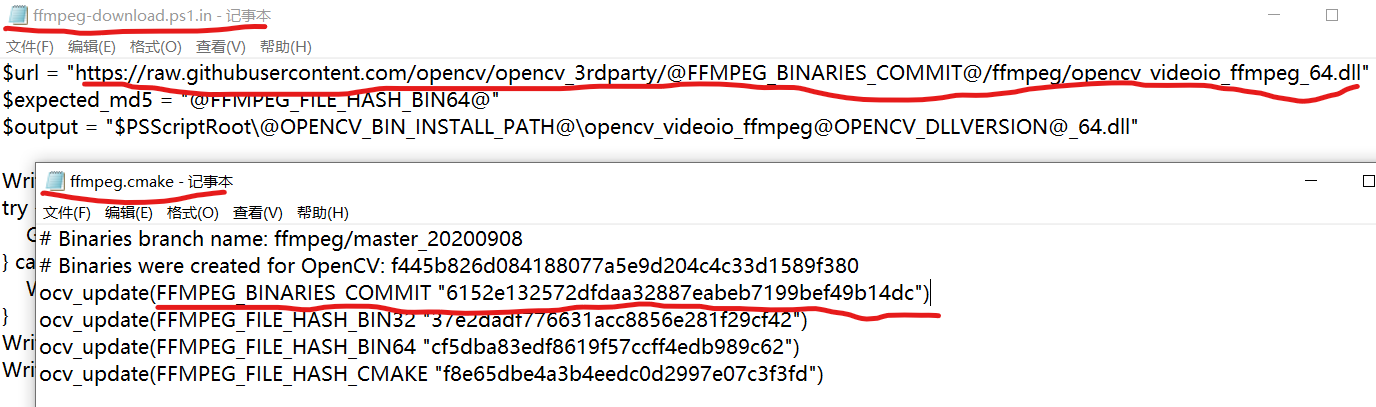
ffmpeg-download.ps1.in：

$url = "https://raw.githubusercontent.com/opencv/opencv\_3rdparty/@FFMPEG\_BINARIES\_COMMIT@/ffmpeg/opencv\_videoio\_ffmpeg\_64.dll"

最后的链接为

https://raw.githubusercontent.com/opencv/opencv\_3rdparty/6152e132572dfdaa32887eabeb7199bef49b14dc/ffmpeg/opencv\_videoio\_ffmpeg.dll

https://raw.githubusercontent.com/opencv/opencv\_3rdparty/6152e132572dfdaa32887eabeb7199bef49b14dc/ffmpeg/opencv\_videoio\_ffmpeg\_64.dll



输入到浏览器URL搜索栏中，下载文件。

**再进行2.4.2步**

#### 2.2.4 编译OpenCV

打开命令行，输入使用 cd 命令切换到 生成OpenCV源文件的路径下（我这里是D:\opencv\build\x64\MinGW），

输入 minGW32-make 命令，这个时候大概花费 30 分钟左右的时间来编译，这个具体的时间和根据电脑的性能有关的。

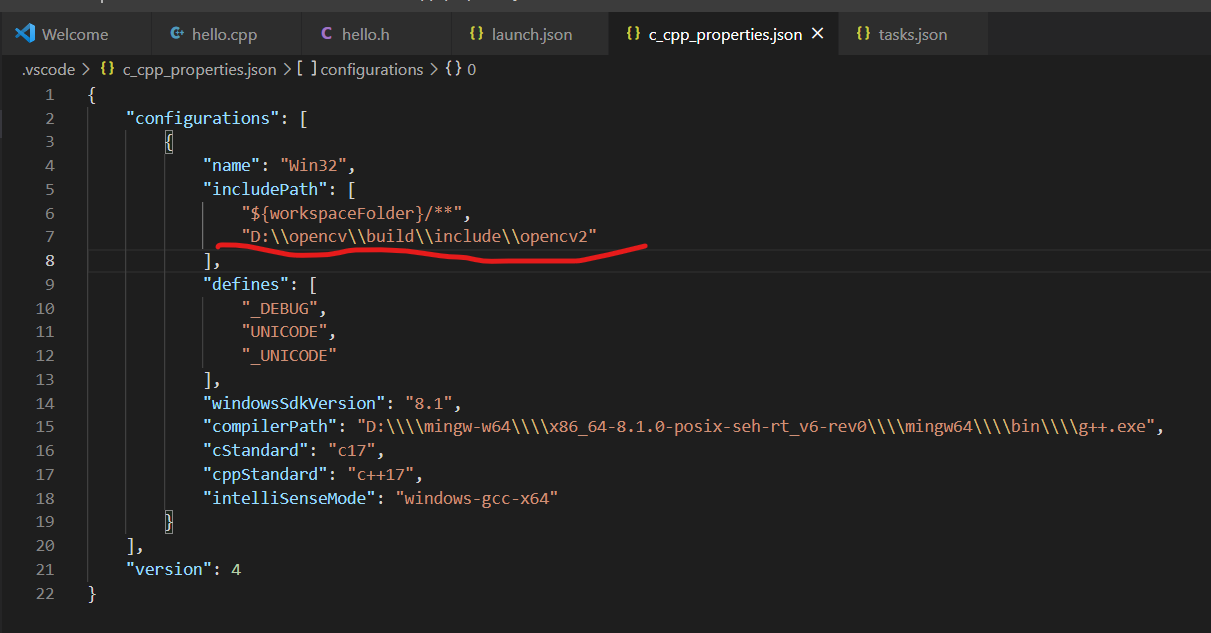
#### 2.2.5 配置OpenCV环境

将OpenCV编译好的bin目录添加到环境变量，（我这里的完整路径是D:\opencv\build\x64\MinGW\bin）

### 2.3 在VSCode中配置OpenCV

#### 2.3.1 配置includePath

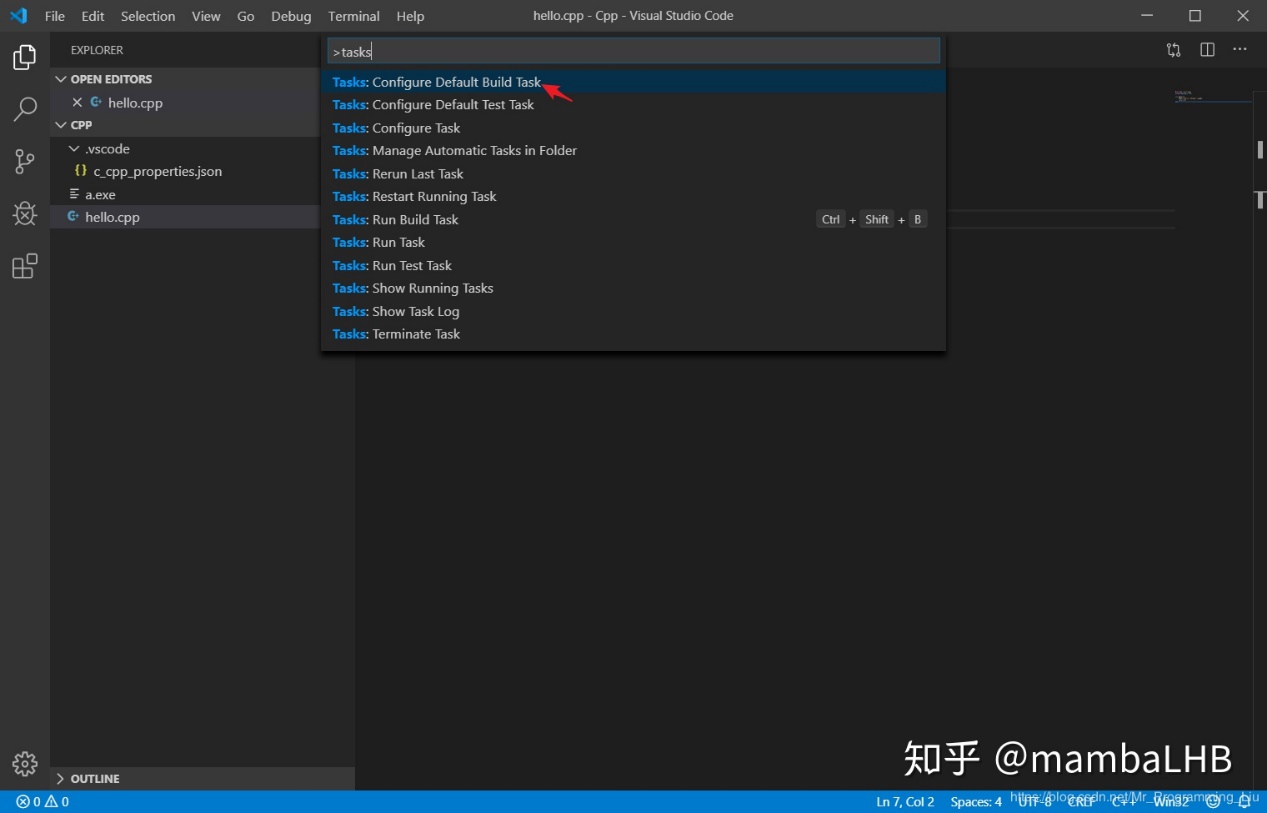
将OpenCV中build\include文件夹中的头文件配置到VSCode中c++的配置中，（这里我的完整的目录是D:\opencv\build\include\opencv2）

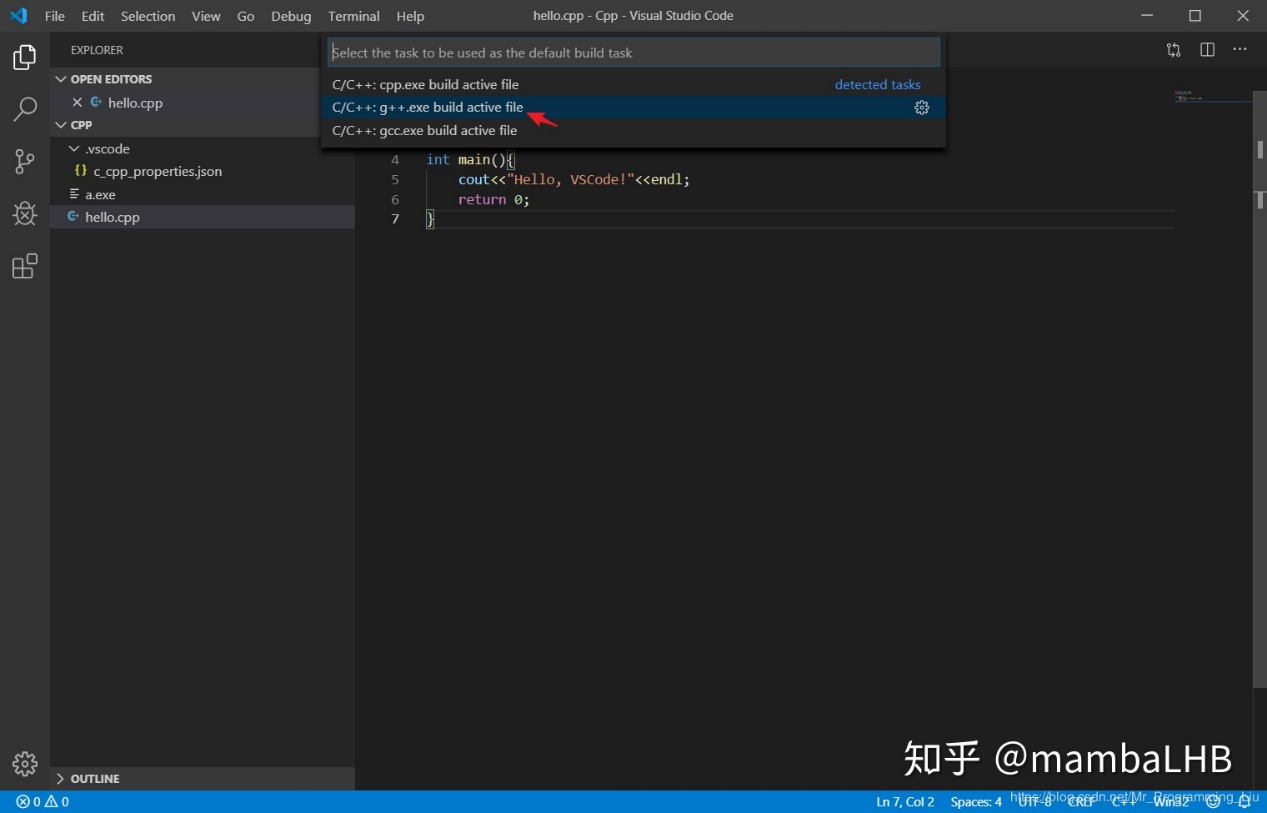


#### 2.3.2 配置compilerPath

##### 2.3.2.1 配置一个构建任务

按**Ctrl+Shift+P**打开命令面板，输入>**Task**，选择 Configuration Task即可创建tasks.json文件。





##### 2.3.2.2 在Tasks.json中添加opencv头文件、库文件以及MinGW库文件路径

将如下内容添加到编译的args参数中

"-I", "D:\\opencv\\build\\include",

"-I", "D:\\opencv\\build\\include\\opencv2",

"-L", "D:\\opencv\\build\\x64\\MinGW\\lib",

"-l", "opencv\_core451",

"-l", "opencv\_imgproc451",

"-l", "opencv\_imgcodecs451",

"-l", "opencv\_video451",

"-l", "opencv\_ml451",

"-l", "opencv\_highgui451",

"-l", "opencv\_objdetect451",

"-l", "opencv\_flann451",

"-l", "opencv\_imgcodecs451",

"-l", "opencv\_photo451",

"-l", "opencv\_videoio451",

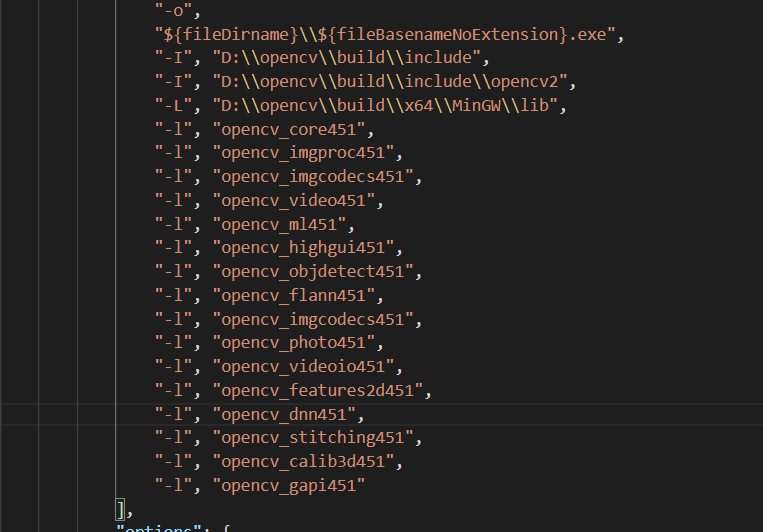
"-l", "opencv\_features2d451",

"-l", "opencv\_dnn451",

"-l", "opencv\_stitching451",

"-l", "opencv\_calib3d451",

"-l", "opencv\_gapi451"



#### 2.3.3 验证是否配置好

创建一个C++源文件，输入如下代码，测试是否配置正确。

#include <opencv.hpp>

using namespace cv;

int main(){

    Mat img = imread(r"E:\Workspaces\zbq\projects\U-2-Net-master\test\_image\01.jpg");

    imshow("image", img);

    waitKey(0);

    return 0;

}

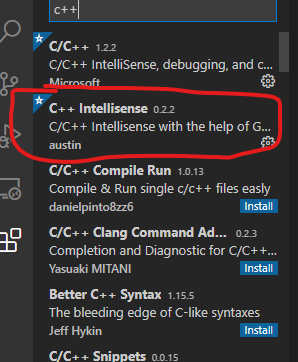
注意：将launch.json中的externalConsole设置为true并重启VSCode，不然不会弹出额外的命令行，导致opencv窗口无法显示。



#### 2.3.4 C++智能提示

##### 2.3.4.1 下载插件

下载C++ Intellisense插件



##### 2.3.4.2 配置settings.json文件

将如下内容添加到settings.json文件中

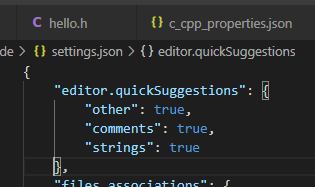
"editor.quickSuggestions": {

"other": true,

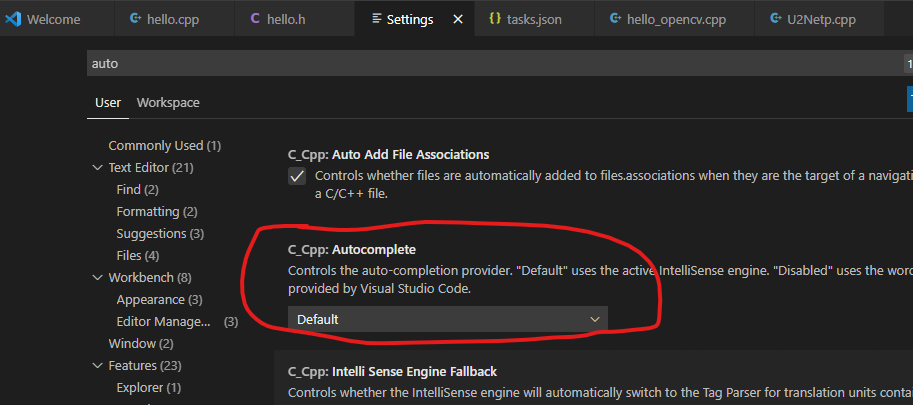
"comments": true,

"strings": true

},

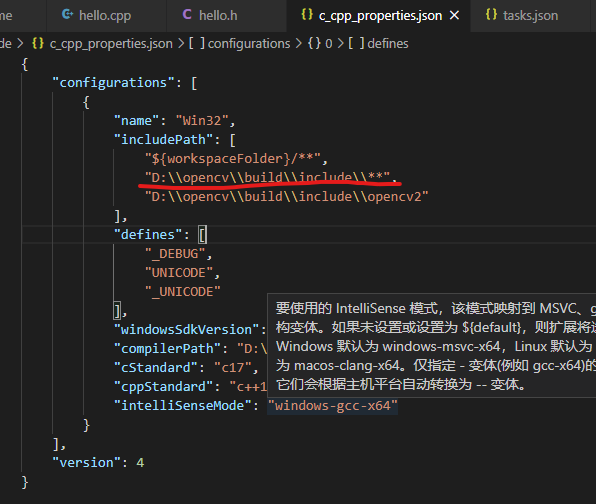


另外，注意autocomplete属性需要设置为Default



##### 2.3.4.3 配置c\_cpp\_properties.json文件

在c\_cpp\_properties.json文件中添加下面带有两个星号\*\*的这一句。



重启VSCode，智能提示就可以用了。

## 附页

### A 参考链接

VSCode配置C/C++环境 - mambaJava的文章 - 知乎

<https://zhuanlan.zhihu.com/p/87864677>

vs2015+OpenCV的安装配置与测试（演示）

<https://blog.csdn.net/qq_38755175/article/details/84988368>

windows系统【VSCode搭建OpenCV、C++开发环境】

<https://blog.csdn.net/qq_41204464/article/details/105472091>

Windows下CMake安装教程

<https://blog.csdn.net/u011231598/article/details/80338941>

再整理：Visual Studio Code(vscode)下的基于C++的OpenCV的最新搭建攻略解析

<https://www.cnblogs.com/czlhxm/p/13848278.html>

解决“检测到 #include 错误。请更新 includePath”问题和“无法打开 源 文件 "avr/pgmspace.h"”问题（VSCode中进行Arduino开发）

<https://blog.csdn.net/wh201906/article/details/103878857>

VScode编辑C++文件时自动提示没了

<https://blog.csdn.net/weixin_43337806/article/details/109126252>

OpenCV开发笔记（十一）：OpenCV编译支持Gpu(cuda) 加速开发之win-qt-mingw32编译

<https://blog.csdn.net/qq21497936/article/details/102725669>

Windows 10下vscode使用visual studio 2017的MSVC配置C/C++编译环境

<https://blog.csdn.net/heilone6688/article/details/91050508>

VS Code：使用VS的cl.exe编译运行C/C++程序

<https://blog.csdn.net/qq_34801642/article/details/105453161>

window10下使用VScode编译运行opencv（使用MSVC）

<https://blog.csdn.net/weixin_41115751/article/details/89880333>

VSCode配置C++环境（MSVC）

https://blog.csdn.net/qq\_38981614/article/details/99629597