

# vscode配置C/C++环境

---

## 文章目录

- [准备工作](#)
- - [第一步 下载vscode](#)
  - [第二步 下载mingw](#)
  - [第三步 将mingw添加至系统变量中](#)
  - [第四步 打开VsCode安装一下必要的插件](#)
- [正式开始配置](#)
- - [第一步 新建个存放C/C++文件的文件夹，并新建个cpp文件](#)
  - [第二步 \(1\) 简单的环境配置方法 - run code调试](#)
    - [问题1：run code执行代码出现gcc\(或g++\) :error; no such file or directory的错误](#)
      - [原因分析](#)
      - [解决办法](#)
      - [第一步 打开code runner扩展设置](#)
        - [第二步 进入指令设置界面](#)
        - [第三步 修改编译指令](#)
  - [第二步 \(2\) 复杂的环境配置方法 - gdb调试](#)
    - [第一步 配置编译器环境](#)
    - [第二步 配置task编译任务](#)
    - [第三步 配置launch调试任务](#)
    - [一些问题](#)
      - [问题1](#)
  - [进阶 设置成经典的弹出黑窗运行程序的形式 \(在系统终端中运行程序\)](#)
  - [进阶 设置在vscode内置终端中执行程序](#)

## 准备工作

---

在 Vscode 里配置C/C++运行环境，首先，需要下载 C/C++ 的开发环境，然后将 C/C++ 的开发环境添加至系统变量中。

# 第一步 下载vscode

VsCode大家应该都会下，这里就不提供VsCode的下载与安装教程了

# 第二步 下载mingw

这里采用mingw作为 C/C++ 开发环境，官网链接如下

官网链接：[MinGW官网](#)

This is the official download site for the latest packages originating from the [MinGW.OSDN Project](#), (formerly the MinGW.org Project; however, that domain is no longer associated with this project).

MinGW is a native Windows port of the GNU Compiler Collection (GCC), with freely distributable import libraries and header files for building native Windows applications; includes extensions to the MSVC runtime to support C99 functionality. Although (currently) offering only a 32-bit compiler suite, all of MinGW's software will execute on the 64bit Windows platforms.

MinGW is a registered trademark of Software in the Public Interest Inc., registration number 86017856; this trademark has been registered on behalf of the [MinGW.OSDN Project](#), and its use by any other project is unauthorized.

点这里下载MinGW安装程序

Download [Windows mingw-get-setup.exe](#) (Date: 2017-09-06, Size: 91.00 KB)

Latest Release

- MinGW.OSDN Windows System Libraries (WSL) Windows API (32-bit), Version 5.4.2 (Date: 2021-04-12)
- MinGW.OSDN Windows System Libraries (WSL) C-Runtime Library (32-bit), Version 5.4.2 (Date: 2021-04-12)
- MinGW.OSDN Windows System Libraries (WSL) C-Runtime Library (32-bit DLL), Version 5.4.2 (Date: 2021-04-12)
- MinGW.OSDN Compiler Collection (GCC) GCC-9.2.0 Licence (Date: 2021-02-02)
- MinGW.OSDN Compiler Collection (GCC) GCC-9.2.0 Source Code (Includes MinGW.org patches) (Date: 2021-02-02)

[Download File List](#)

Project Information

- Last Update: 2022-11-11 03:31
- Registered: 2008-11-28 19:42
- Project Ranking: Activeness Ranking: 3 (13707 point)  
Download Ranking: 1 (53439)
- Developer Member: cstraus, earnie, keith, texasgaidheal [Member List](#)

Software Map

- Development Status: 6 - Mature
- Target Users: Developers
- License: BSD 3-Clause License (aka "BSD New" or "BSD Simplified" License), GNU General Public License v2

下载完成后我们会得到这样一个安装程序



双击打开

## MinGW Installation Manager Setup Tool

mingw-get version 0.6.3-pre-20170905-1



Written by Keith Marshall

Copyright © 2009-2013, MinGW.org Project

<http://mingw.org>

This is free software; see the product documentation or source code, for copying and redistribution conditions. There is NO WARRANTY; not even an implied WARRANTY OF MERCHANTABILITY, nor of FITNESS FOR ANY PARTICULAR PURPOSE.

This tool will guide you through the first time setup of the MinGW Installation Manager software (mingw-get) on your computer; additionally, it will offer you the opportunity to install some other common components of the MinGW software distribution.

After first time setup has been completed, you should invoke the MinGW Installation Manager directly, (either the CLI mingw-get.exe variant, or its GUI counterpart, according to your preference), when you wish to add or to remove components, or to upgrade your MinGW software installation.

[View Licence](#)

点这个

[Install](#)

[Cancel](#)

CSDN @DreamSuif

## MinGW Installation Manager Setup Tool

mingw-get version 0.6.3-pre-20170905-1

1.点这里选择一个合适的安装位置，并记住这个位置，之后添加进系统环境的时候会用到



### Step 1: Specify Installation Preferences

#### Installation Directory

C:\MinGW

Change

If you elect to change this, you are advised to avoid any choice of directory which includes white space within the absolute representation of its path name.

#### User Interface Options

Both command line and graphical options are available. The command line interface is always supported; the alternative only if you choose the following option to ...

- ... also install support for the graphical user interface.

Program shortcuts for launching the graphical user interface should be installed ...

- ... just for me (the current user), or ...  ... for all users \* ...

- ... in the start menu, and/or ...  ... on the desktop.

\* selection of this option requires administrative privilege.

2.然后点这里继续

View Licence

Continue

Cancel

CSDN @DreamSuf

## MinGW Installation Manager Setup Tool

mingw-get version 0.6.3-pre-20170905-1



### Step 2: Download and Set Up MinGW Installation Manager

#### Download Progress

Connecting to osdn.net ... \

[progress bar] of [total size] : [current size]

#### Details

静静等待安装完成

View Licence

Continue

Quit

CSDN @DreamSuf

## MinGW Installation Manager Setup Tool

mingw-get version 0.6.3-pre-20170905-1



### Step 2: Download and Set Up MinGW Installation Manager

#### Download Progress

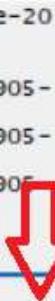
Catalogue update completed; please check 'Details' pane for errors.

Processed 113 of 113 items : 100 %

#### Details

```
mingw-get: *** INFO *** S1  
170905-1-xml.tar.xz  
mingw-get: *** INFO *** S1  
mingw-get: *** INFO *** S1  
1-bin.tar.xz  
mingw-get: *** INFO *** S1  
1-gui.tar.xz  
mingw-get: *** INFO *** S1  
1-lic.tar.xz  
mingw-get: *** INFO *** S1
```

可以点continue进入图形化安装  
界面，但是图形化安装界面太复杂  
了，因此我们直接点quit



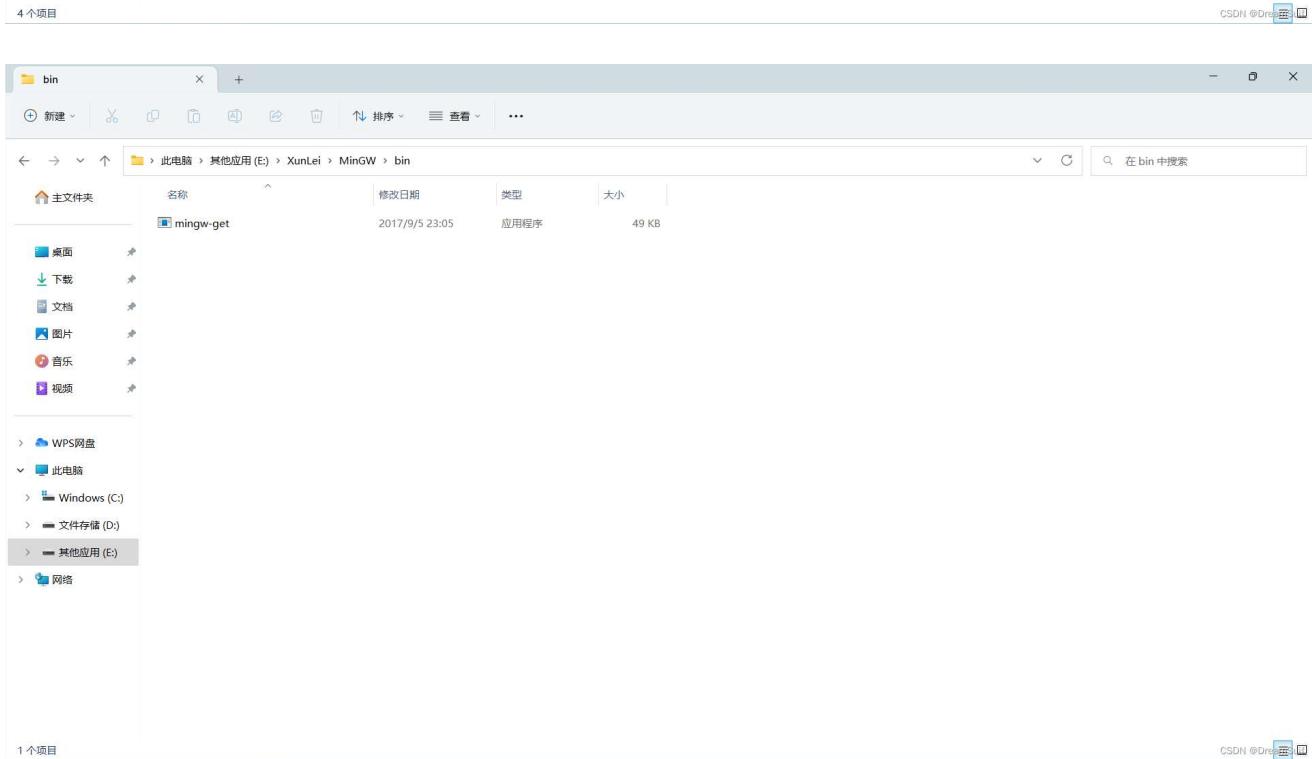
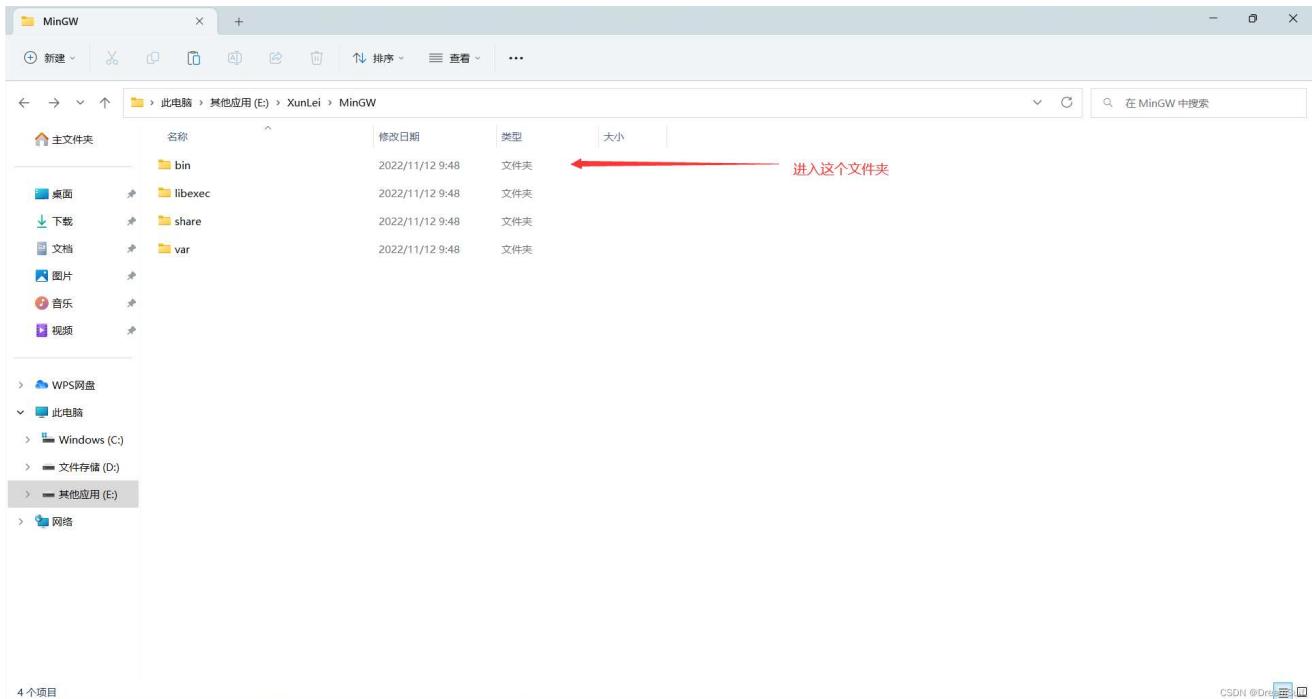
[View Licence](#)

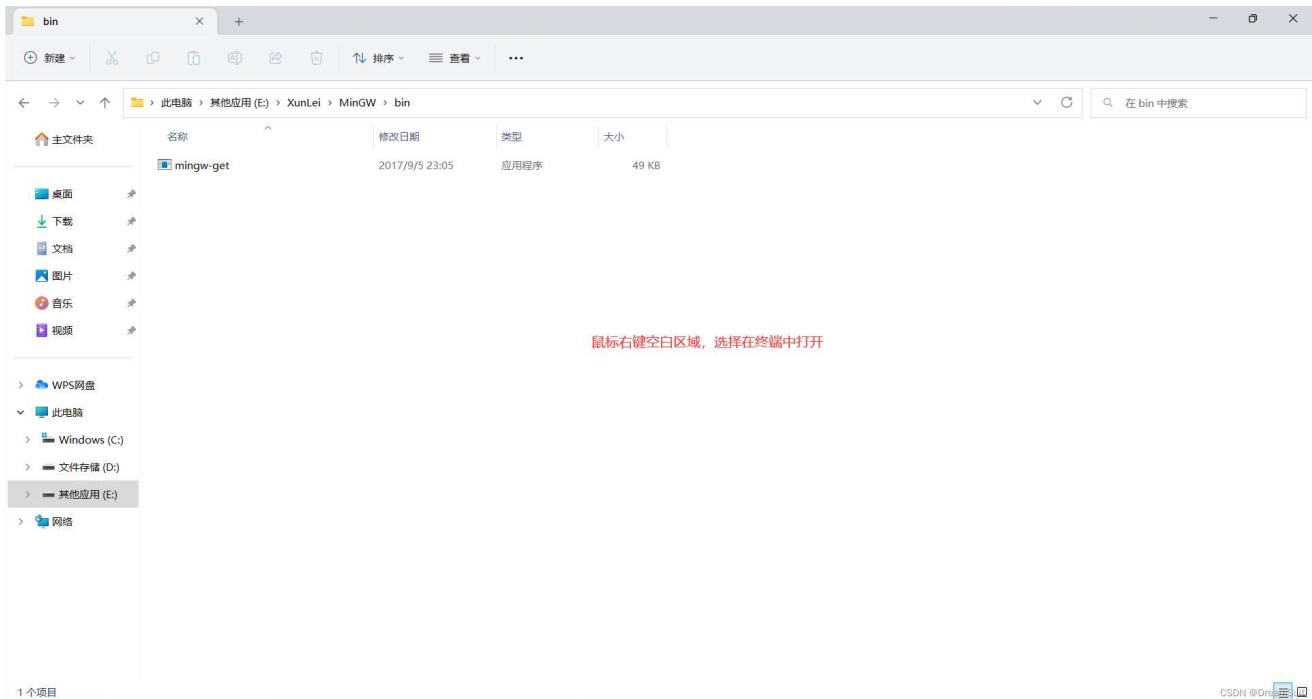
[Continue](#)

[Quit](#)

CSDN @DreamSuif

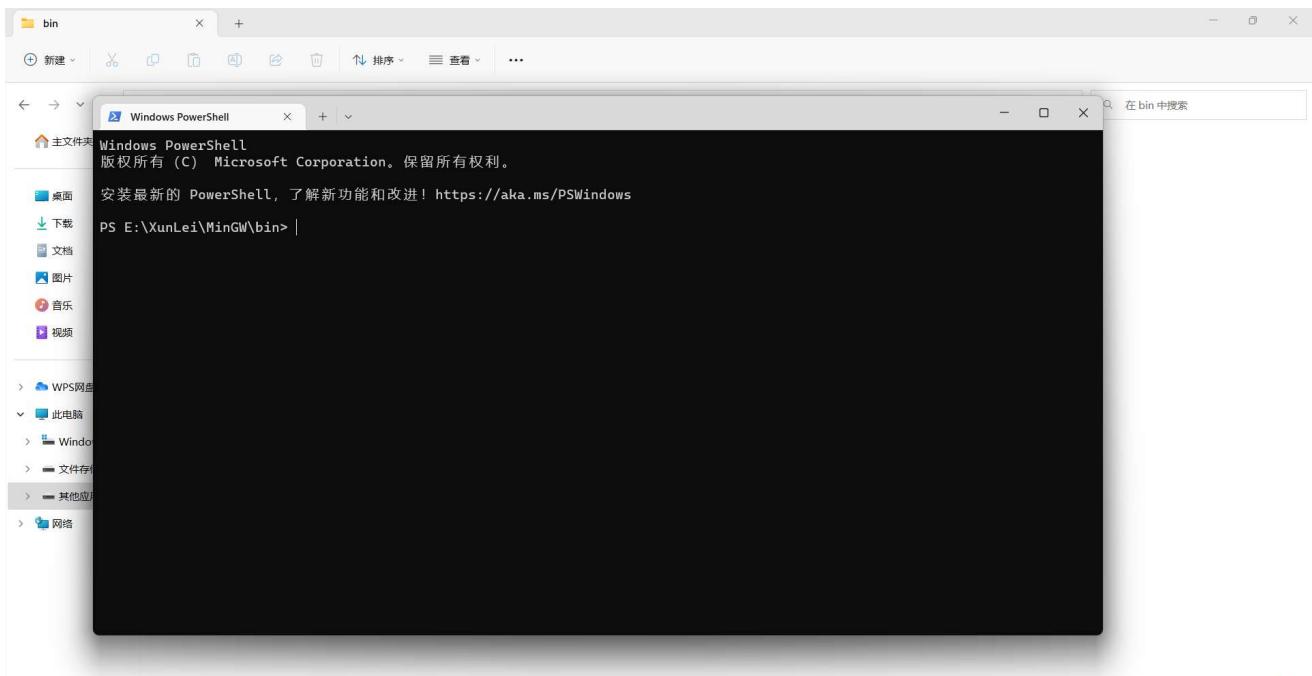
然后我们找到mingw的安装文件夹





1个项目

CSDN @Dre



1个项目

CSDN @Dre

## 依次输入

```
mingw-get install gcc
mingw-get install g++
mingw-get install gdb
```

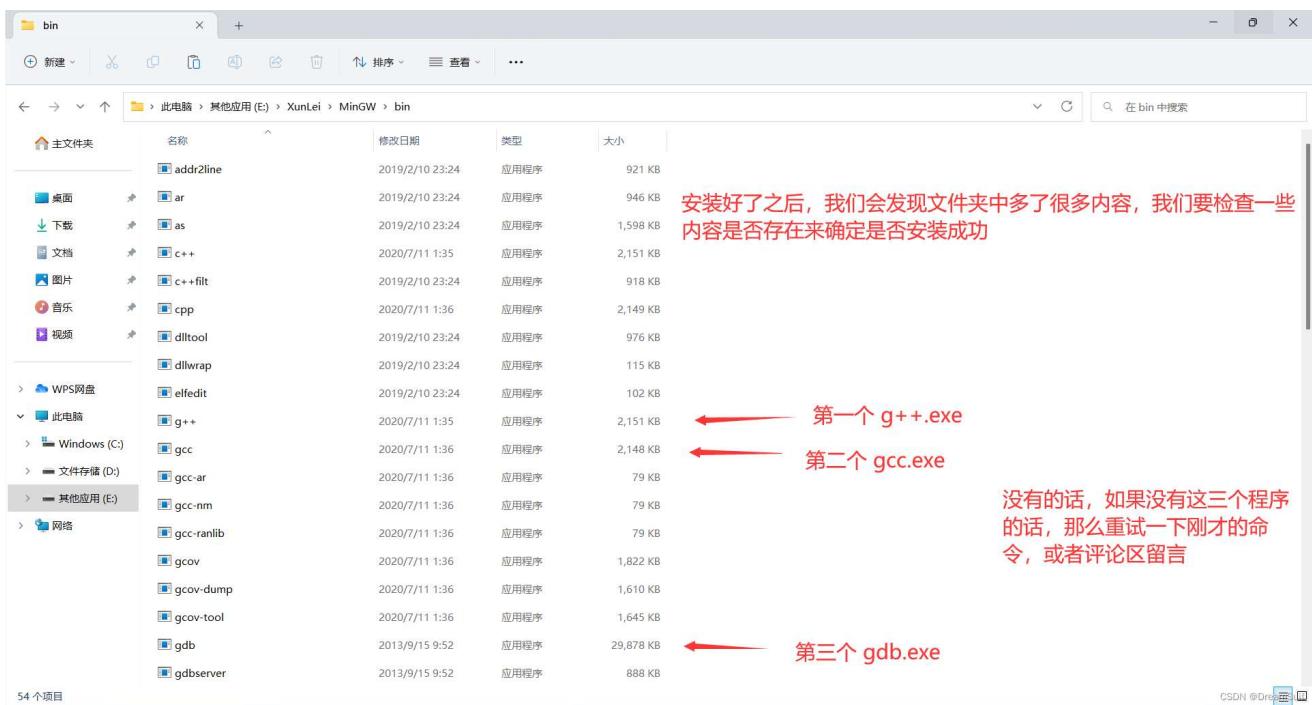
Windows PowerShell  
版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！<https://aka.ms/PSWindows>

因为我这里已经安装过了，所以会提示already installed  
正常情况下，下载速度特别慢(但也比没开会员的百度网盘快)，所以耐心等待

```
PS E:\XunLei\MinGW\bin> mingw-get install gcc
install: gcc-9.2.0-2-mingw32-lic.tar.xz
mingw-get.exe: *** ERROR *** package gcc-9.2.0-2-mingw32-lic.tar.xz is already installed
install: gcc-core-9.2.0-2-mingw32-bin.tar.xz
mingw-get.exe: *** ERROR *** package gcc-core-9.2.0-2-mingw32-bin.tar.xz is already installed
install: gcc-core-4.8.2-2-mingw32-dev.tar.lzma
mingw-get.exe: *** ERROR *** package gcc-core-4.8.2-2-mingw32-dev.tar.lzma is already installed
install: gcc-core-4.8.1-5-mingw32-doc.tar.lzma
mingw-get.exe: *** ERROR *** package gcc-core-4.8.1-5-mingw32-doc.tar.lzma is already installed
install: gcc-9.2.0-2-mingw32-man.tar.xz
mingw-get.exe: *** ERROR *** package gcc-9.2.0-2-mingw32-man.tar.xz is already installed
install: gcc-9.2.0-2-mingw32-info.tar.xz
mingw-get.exe: *** ERROR *** package gcc-9.2.0-2-mingw32-info.tar.xz is already installed
install: gcc-9.2.0-2-mingw32-lang.tar.xz
mingw-get.exe: *** ERROR *** package gcc-9.2.0-2-mingw32-lang.tar.xz is already installed
PS E:\XunLei\MinGW\bin> mingw-get install g++
install: gcc-c++-9.2.0-2-mingw32-bin.tar.xz
mingw-get.exe: *** ERROR *** package gcc-c++-9.2.0-2-mingw32-bin.tar.xz is already installed
install: gcc-c++-4.8.2-2-mingw32-dev.tar.meta
mingw-get.exe: *** ERROR *** package gcc-c++-4.8.2-2-mingw32-dev.tar.meta is already installed
install: gcc-c++-4.8.1-5-mingw32-doc.tar.lzma
mingw-get.exe: *** ERROR *** package gcc-c++-4.8.1-5-mingw32-doc.tar.lzma is already installed
install: gcc-c++-9.2.0-2-mingw32-man.tar.xz
mingw-get.exe: *** ERROR *** package gcc-c++-9.2.0-2-mingw32-man.tar.xz is already installed
PS E:\XunLei\MinGW\bin> mingw-get install gdb
```

CSDN @DreamSuf

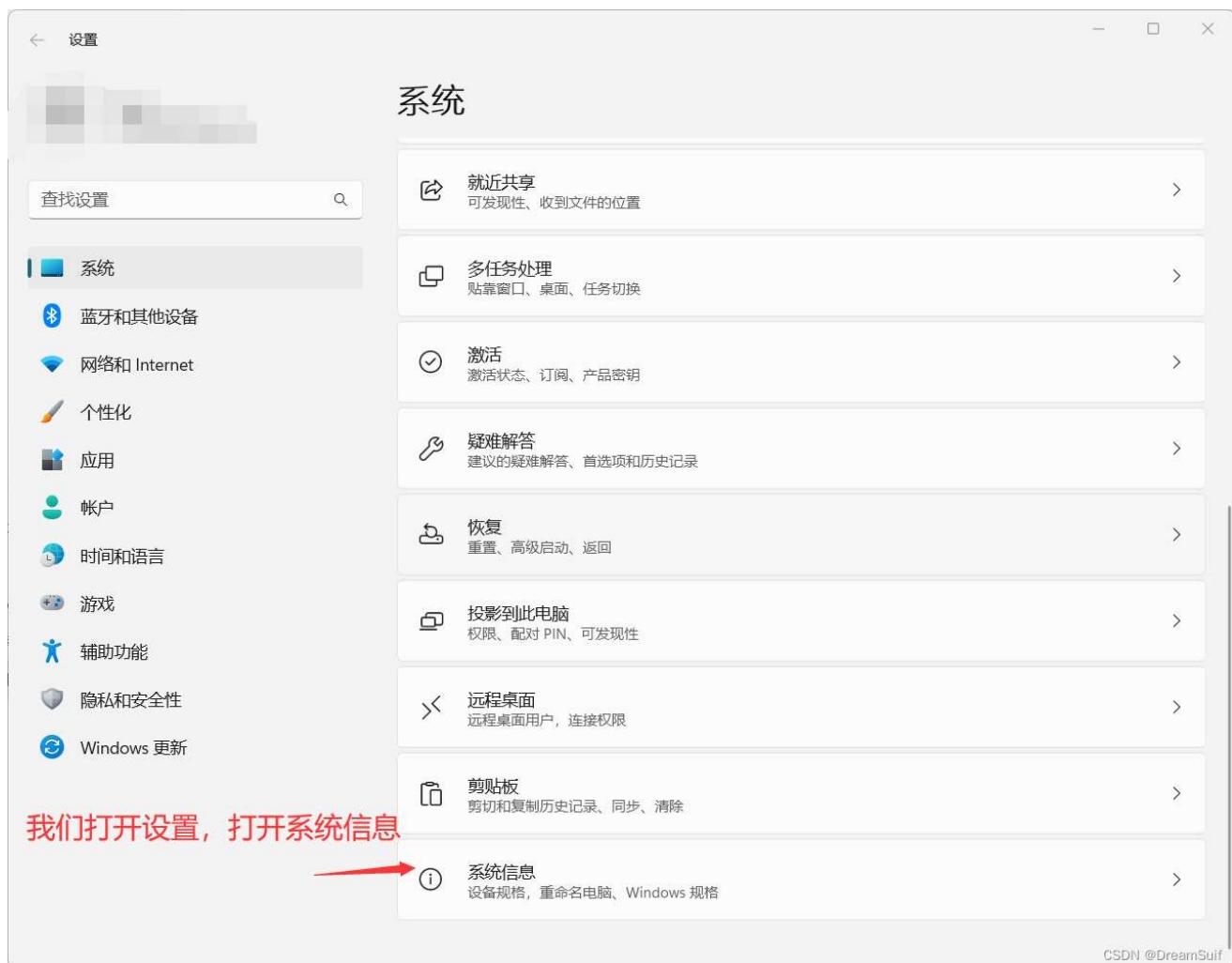


名称	修改日期	类型	大小
addr2line	2019/2/10 23:24	应用程序	921 KB
ar	2019/2/10 23:24	应用程序	946 KB
as	2019/2/10 23:24	应用程序	1,598 KB
c++	2020/7/11 1:35	应用程序	2,151 KB
c++filt	2019/2/10 23:24	应用程序	918 KB
cpp	2020/7/11 1:36	应用程序	2,149 KB
dlttool	2019/2/10 23:24	应用程序	976 KB
dllwrap	2019/2/10 23:24	应用程序	115 KB
elfedit	2019/2/10 23:24	应用程序	102 KB
g++	2020/7/11 1:35	应用程序	2,151 KB
gcc	2020/7/11 1:36	应用程序	2,148 KB
gcc-ar	2020/7/11 1:36	应用程序	79 KB
gcc-nm	2020/7/11 1:36	应用程序	79 KB
gcc-ranlib	2020/7/11 1:36	应用程序	79 KB
gcov	2020/7/11 1:36	应用程序	1,822 KB
gcov-dump	2020/7/11 1:36	应用程序	1,610 KB
gcov-tool	2020/7/11 1:36	应用程序	1,645 KB
gdb	2013/9/15 9:52	应用程序	29,878 KB
gdbserver	2013/9/15 9:52	应用程序	888 KB

然后mingw就下载完成了

## 第三步 将mingw添加至系统变量中

这一步就是告诉电脑，可以在哪个文件夹里找到 C/C++ 的运行环境



## 系统属性

X

计算机名 硬件 高级 系统保护 远程

要进行大多数更改，你必须作为管理员登录。

### 性能

视觉效果，处理器计划，内存使用，以及虚拟内存

设置(S)...

### 用户配置文件

与登录帐户相关的桌面设置

设置(E)...

### 启动和故障恢复

系统启动、系统故障和调试信息

设置(I)...

点环境变量



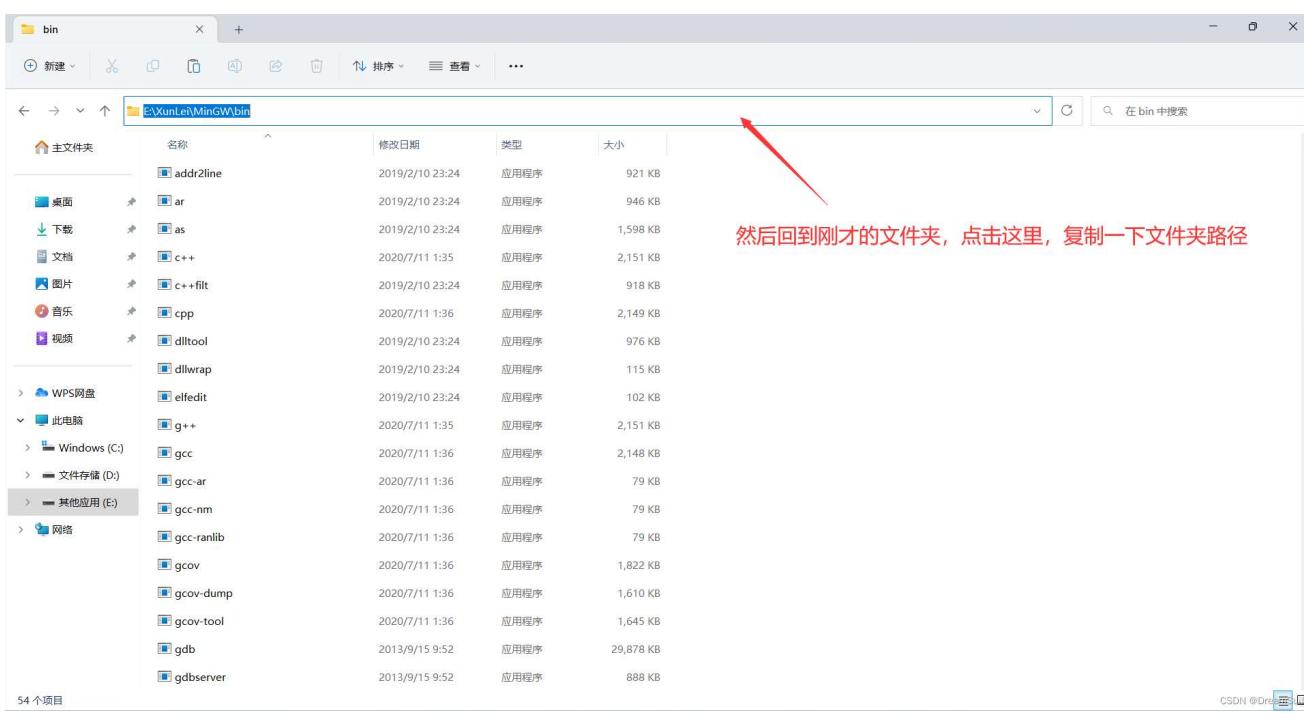
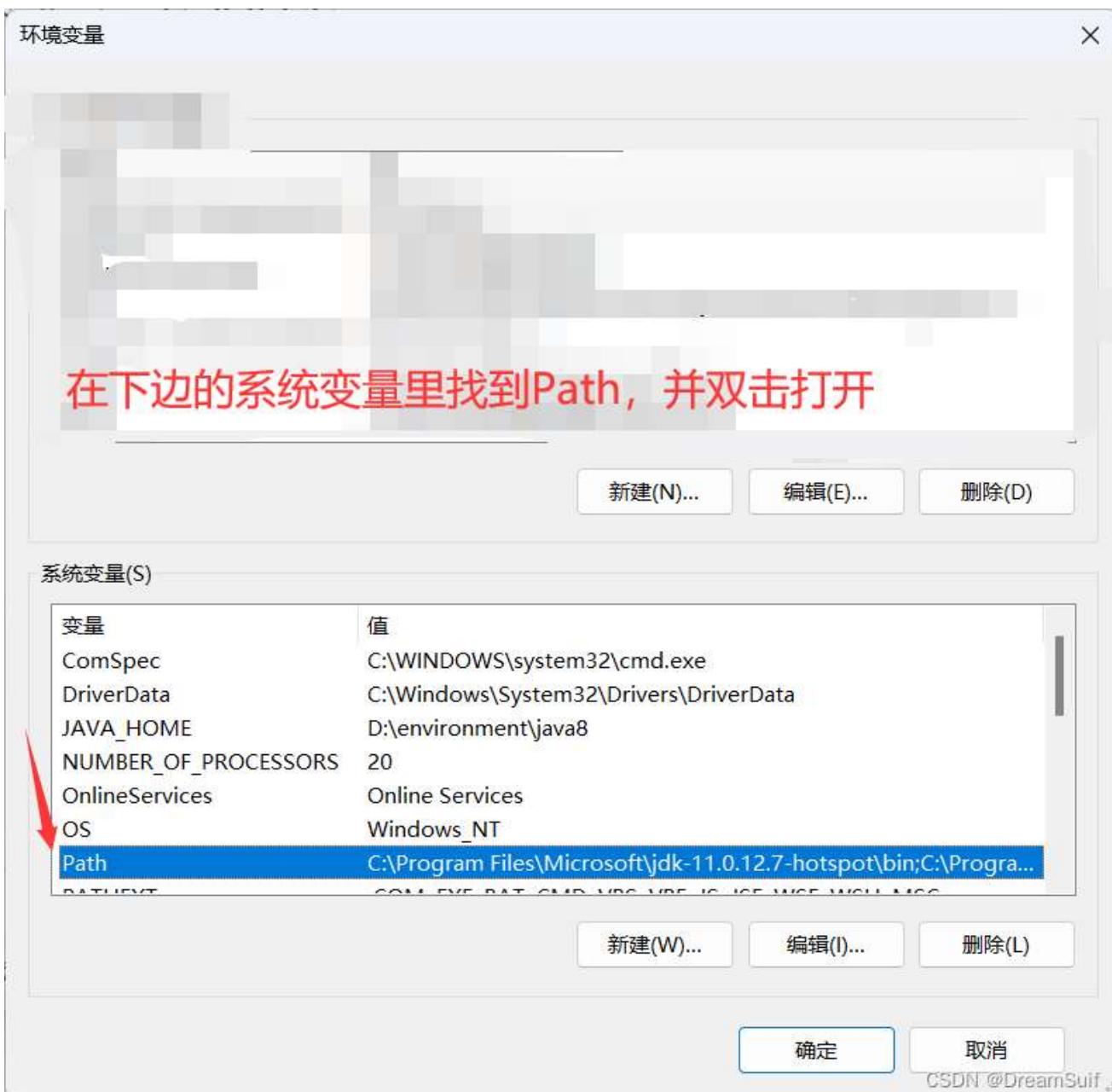
环境变量(N)...

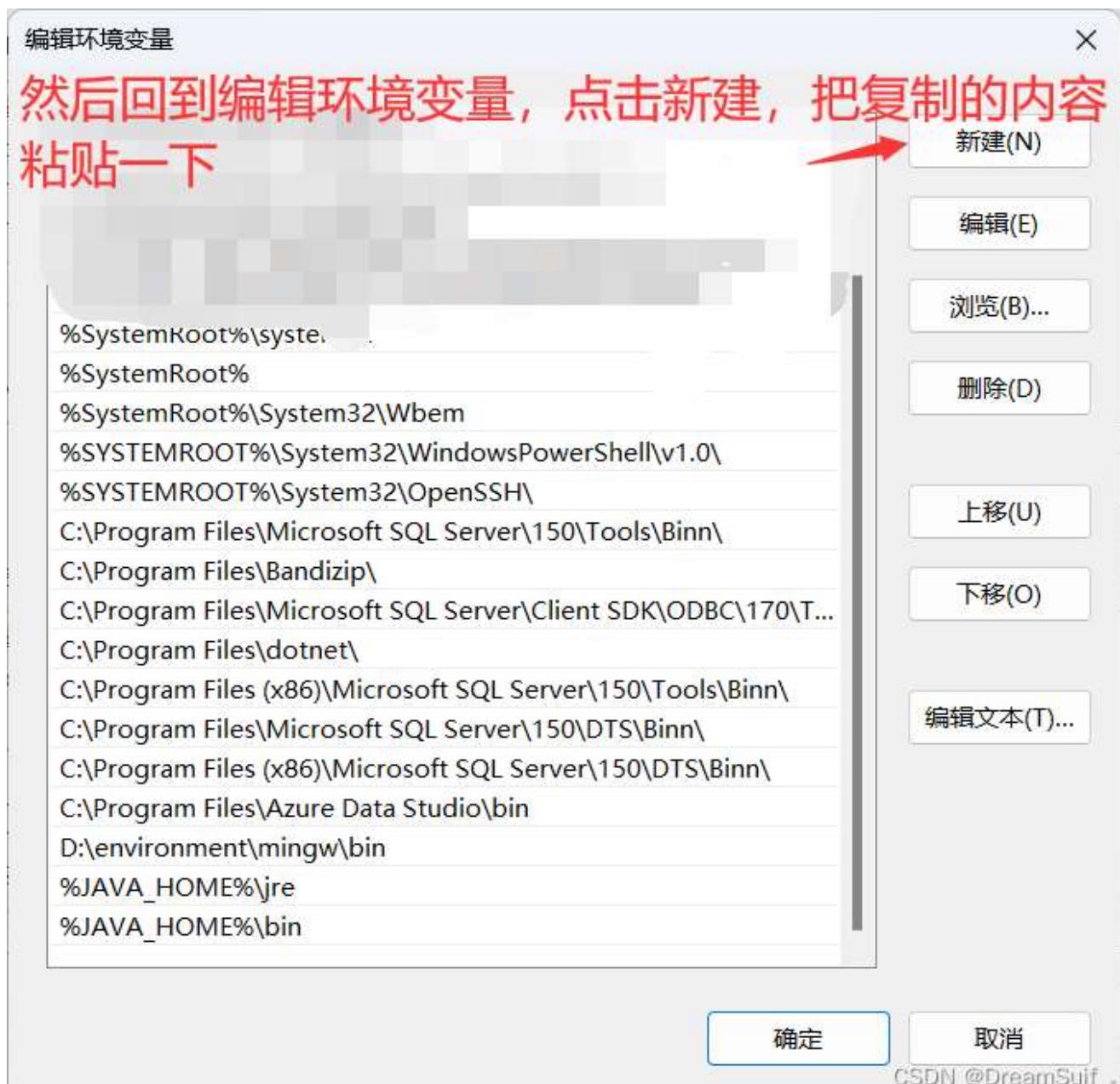
确定

取消

应用(A)

CSDN @DreamSuir  
www.dreamsuir.com





## 编辑环境变量

X

C:\windows\System32\WindowsPowerShell\v1.0\  
C:\windows\System32\OpenSSH\  
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common  
C:\Program Files\NVIDIA Corporation\NVIDIA NvDLISR  
%SystemRoot%\system32  
%SystemRoot%  
%SystemRoot%\System32\Wbem  
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\  
%SYSTEMROOT%\System32\OpenSSH\  
C:\Program Files\Microsoft SQL Server\150\Tools\Binn\  
C:\Program Files\Bandizip\  
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\T...  
C:\Program Files\dotnet\  
C:\Program Files (x86)\Microsoft SQL Server\150\Tools\Binn\  
C:\Program Files\Microsoft SQL Server\150\DTS\Binn\  
C:\Program Files (x86)\Microsoft SQL Server\150\DTS\Binn\  
C:\Program Files\Azure Data Studio\bin  
D:\environment\mingw\bin  
%JAVA\_HOME%\jre  
%JAVA\_HOME%\bin  
E:\XunLei\MinGW\bin

新建(N)

编辑(E)

浏览(B)...

删除(D)

上移(U)

下移(O)

编辑文本(T)...

确定

取消

CSDN @DreamSui

编辑环境变量

```
C:\windows\System32\WindowsPowerShell\v1.0\  
C:\windows\System32\OpenSSH\  
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common  
C:\Program Files\NVIDIA Corporation\NVIDIA NvDLISR  
%SystemRoot%\system32  
%SystemRoot%  
%SystemRoot%\System32\Wbem  
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\  
%SYSTEMROOT%\System32\OpenSSH\  
C:\Program Files\Microsoft SQL Server\150\Tools\Binn\  
C:\Program Files\Bandizip\  
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\T...  
C:\Program Files\dotnet\  
C:\Program Files (x86)\Microsoft SQL Server\150\Tools\Binn\  
C:\Program Files\Microsoft SQL Server\150\DTS\Binn\  
C:\Program Files (x86)\Microsoft SQL Server\150\DTS\Binn\  
C:\Program Files\Azure Data Studio\bin  
D:\environment\mingw\bin  
%JAVA_HOME%\jre  
%JAVA_HOME%\bin  
E:\XunLei\MinGW\bin
```

新建(N)

编辑(E)

浏览(B)...

删除(D)

上移(U)

下移(O)

编辑文本(T)...

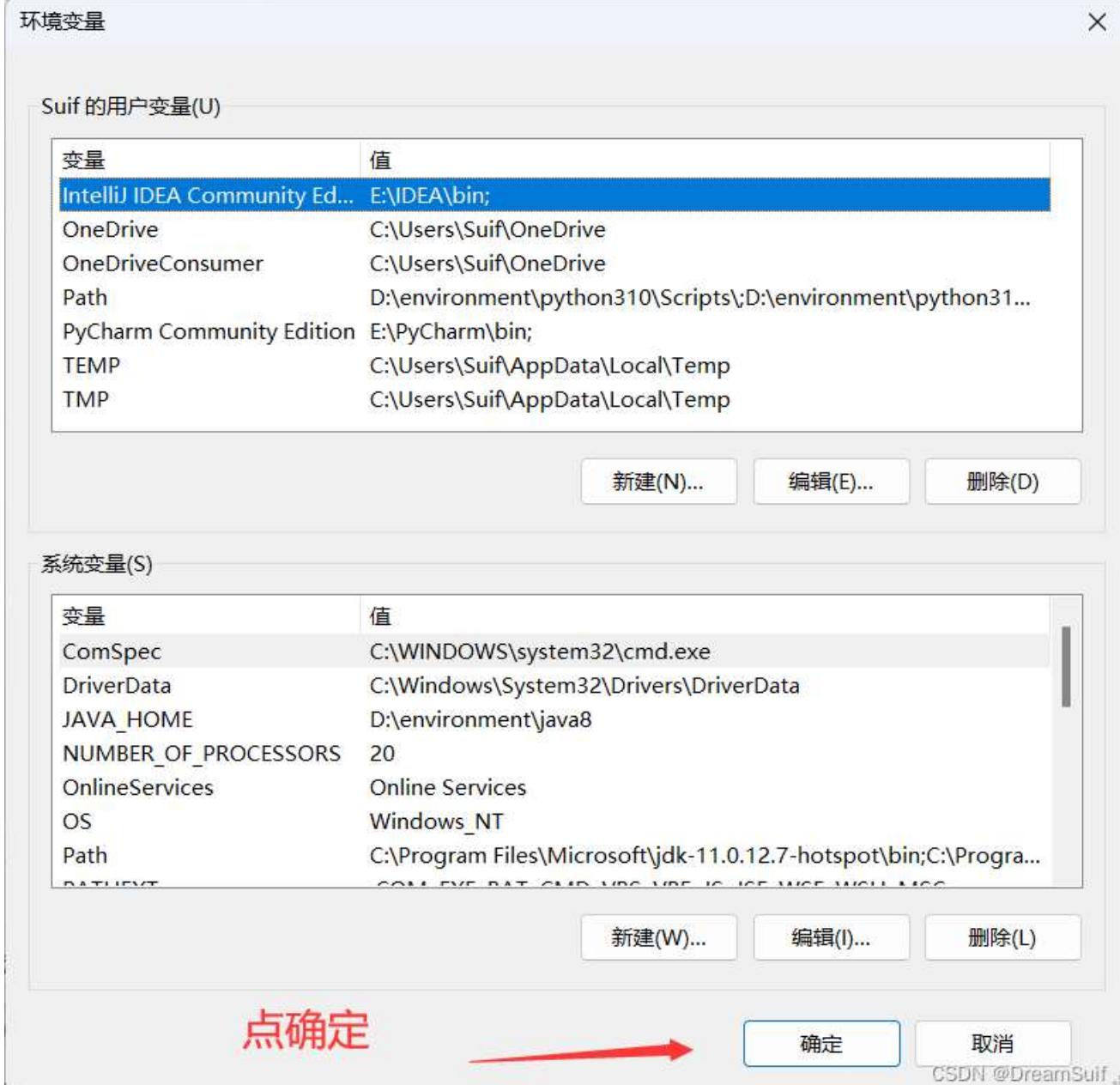
确定

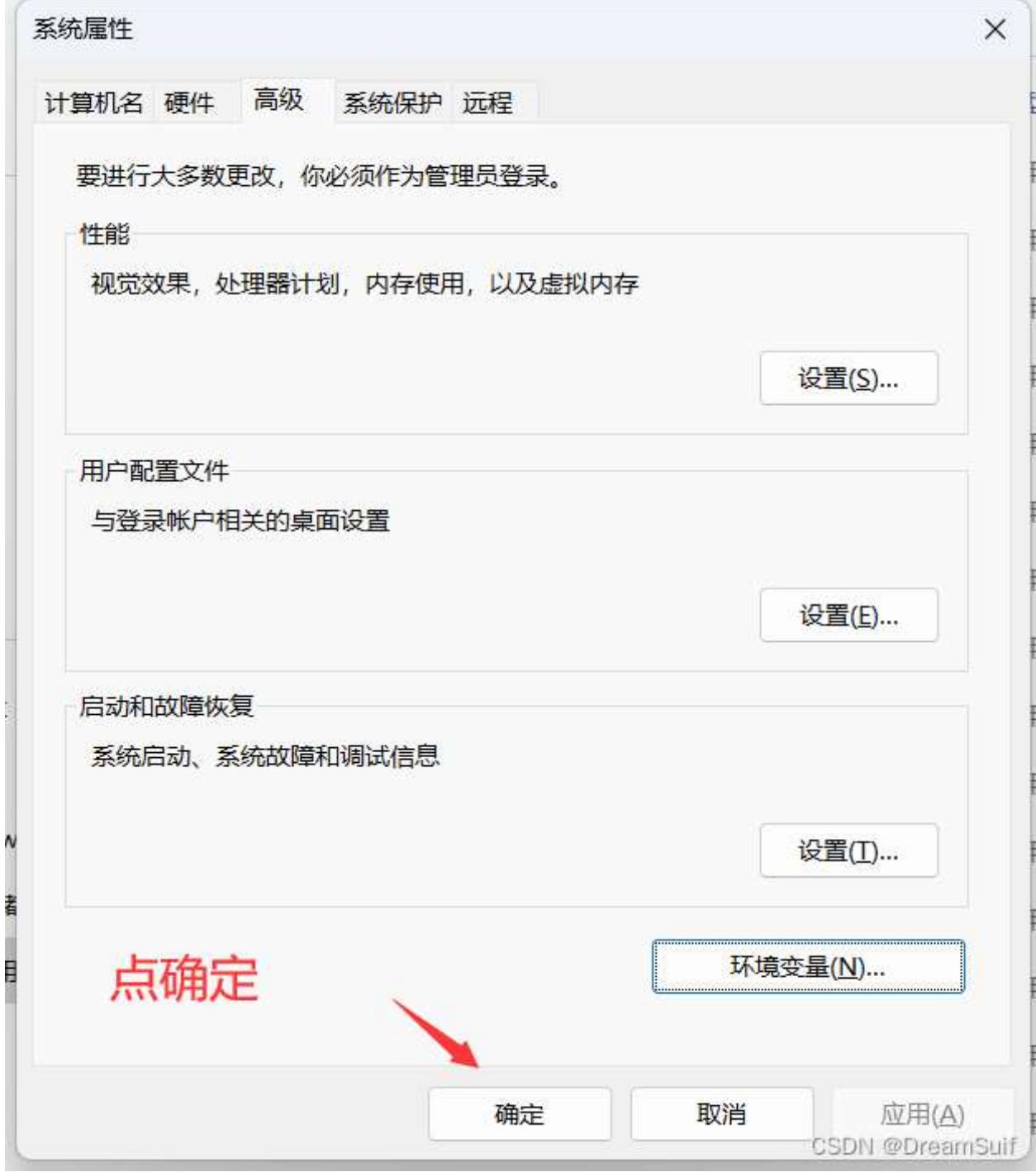
取消

CSDN @DreamSui

然后记得一定要点确定







这样我们的系统环境就配置完成了  
我们可以来检验一下是否配置正确  
按住 win + r 打开以下界面



输入 cmd, 然后点确定

A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.". The window shows the following text:

```
C:\WINDOWS\system32\cmd. + 
Microsoft Windows [版本 10.0.22621.819]
(c) Microsoft Corporation。保留所有权利。
C:\Users\Suif>
```

The status bar at the bottom right says "CSDN @DreamSuif".

依次输入

gcc -v

g++ -v

gdb -v

看看是否显示正常结果

```
C:\WINDOWS\system32\cmd. < + >
Microsoft Windows [版本 10.0.22621.819]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\Suif>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=d:/environment/mingw/bin/../libexec/gcc/mingw32/9.2.0/lto-wrapper.exe
Target: mingw32
Configured with: ../src/gcc-9.2.0/configure --build=x86_64-pc-linux-gnu --host=mingw32 --target=mingw32 --disable-win32-registry --with-arch=i586 --with-tune=generic --enable-static --enable-shared --enable-threads --enable-languages=c,c++,objc,obj-c++,fortran,ada --with-dwarf2 --disable-sjlj-exceptions --enable-version-specific-runtime-libs --enable-libgomp --disable-libvtv --with-libiconv-prefix=/mingw --with-libintl-prefix=/mingw --enable-libstdcxx-debug --disable-build-format-warnings --prefix=/mingw --with-gmp=/mingw --with-mpfr=/mingw --with-mpc=/mingw --with-isl=/mingw --enable-nls --with-pkgversion='MinGW.org GCC Build-2'
Thread model: win32
gcc version 9.2.0 (MinGW.org GCC Build-2)

C:\Users\Suif>g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=d:/environment/mingw/bin/../libexec/gcc/mingw32/9.2.0/lto-wrapper.exe
Target: mingw32
Configured with: ../src/gcc-9.2.0/configure --build=x86_64-pc-linux-gnu --host=mingw32 --target=mingw32 --disable-win32-registry --with-arch=i586 --with-tune=generic --enable-static --enable-shared --enable-threads --enable-languages=c,c++,objc,obj-c++,fortran,ada --with-dwarf2 --disable-sjlj-exceptions --enable-version-specific-runtime-libs --enable-libgomp --disable-libvtv --with-libiconv-prefix=/mingw --with-libintl-prefix=/mingw --enable-libstdcxx-debug --disable-build-format-warnings --prefix=/mingw --with-gmp=/mingw --with-mpfr=/mingw --with-mpc=/mingw --with-isl=/mingw --enable-nls --with-pkgversion='MinGW.org GCC Build-2'
Thread model: win32
gcc version 9.2.0 (MinGW.org GCC Build-2)

C:\Users\Suif>g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=d:/environment/mingw/bin/../libexec/gcc/mingw32/9.2.0/lto-wrapper.exe
Target: mingw32
Configured with: ../src/gcc-9.2.0/configure --build=x86_64-pc-linux-gnu --host=mingw32 --target=mingw32 --disable-win32-registry --with-arch=i586 --with-tune=generic --enable-static --enable-shared --enable-threads --enable-languages=c,c++,objc,obj-c++,fortran,ada --with-dwarf2 --disable-sjlj-exceptions --enable-version-specific-runtime-libs --enable-libgomp --disable-libvtv --with-libiconv-prefix=/mingw --with-libintl-prefix=/mingw --enable-libstdcxx-debug --disable-build-format-warnings --prefix=/mingw --with-gmp=/mingw --with-mpfr=/mingw --with-mpc=/mingw --with-isl=/mingw --enable-nls --with-pkgversion='MinGW.org GCC Build-2'
Thread model: win32
gcc version 9.2.0 (MinGW.org GCC Build-2)

C:\Users\Suif>gdb -v
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.

C:\Users\Suif
```

结果正常显示就表示系统环境配置正确，这时候建议重启一下电脑(当然也可以不重启)

# 第四步 打开VsCode安装一下必要的插件

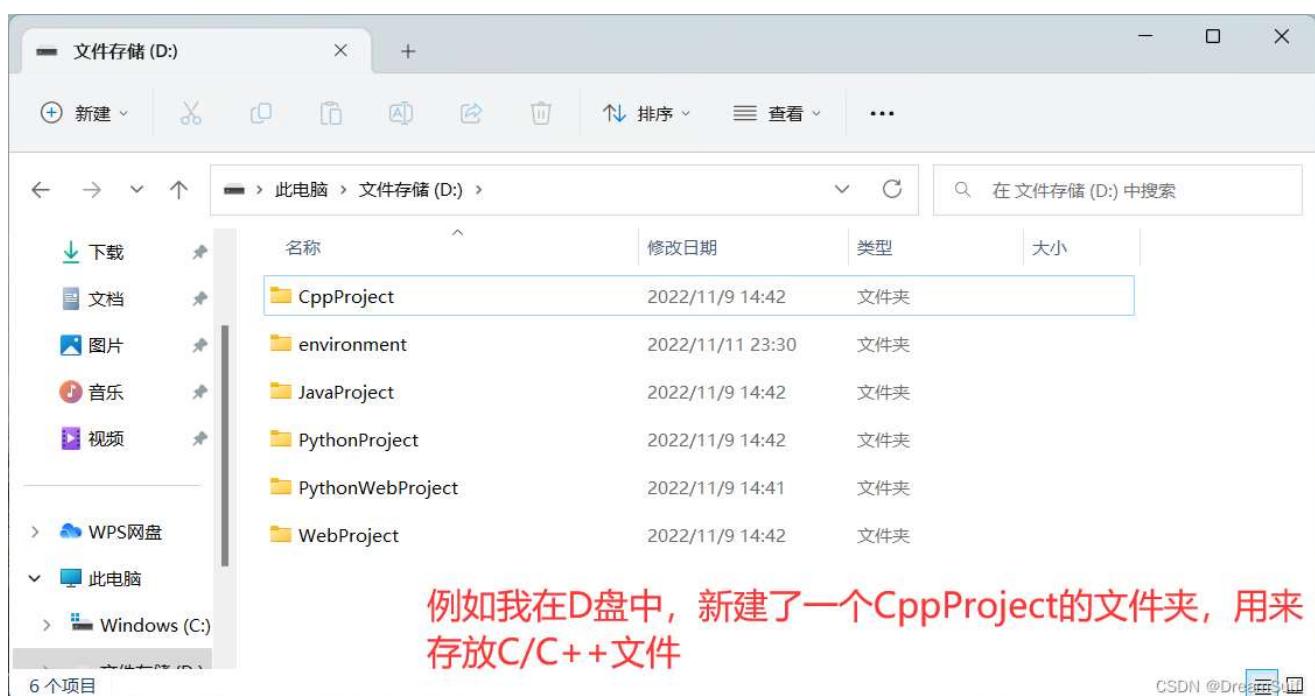
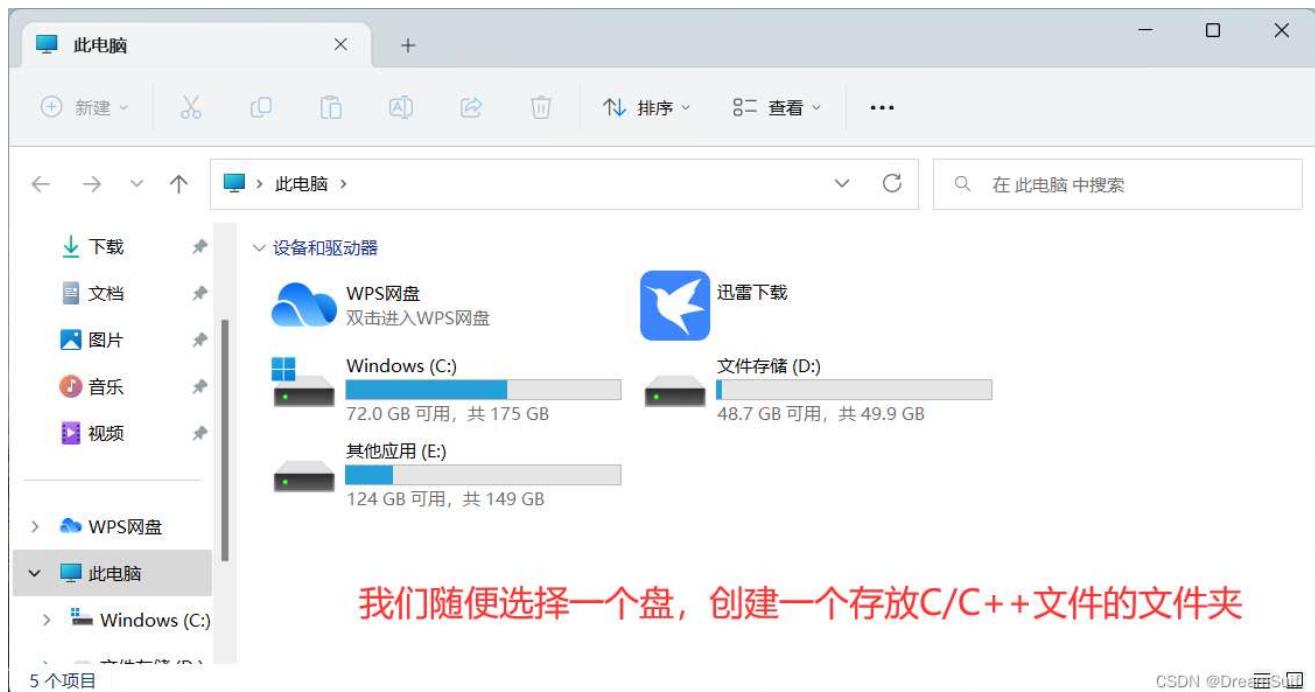




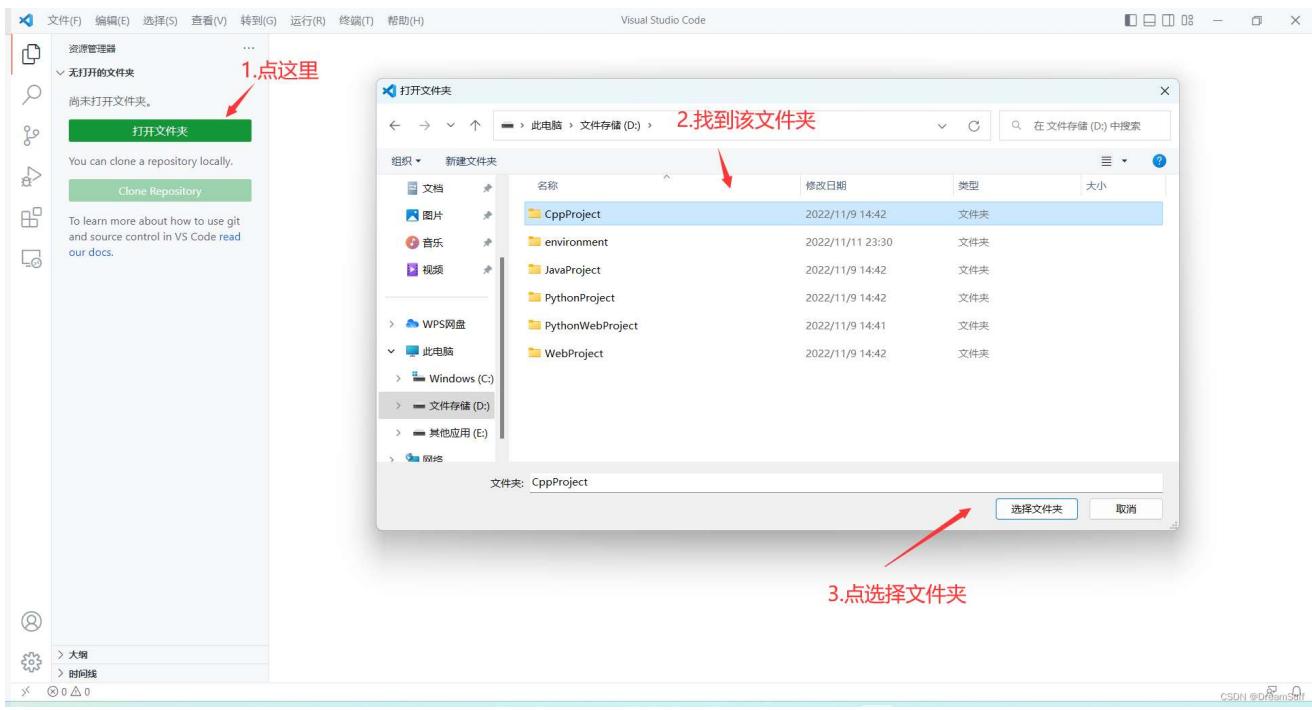
我们的vscode配置 C/C++ 的准备工作就完成了

# 正式开始配置

## 第一步 新建个存放C/C++文件的文件夹，并新建个cpp文件



我们在vscode中打开那个文件夹



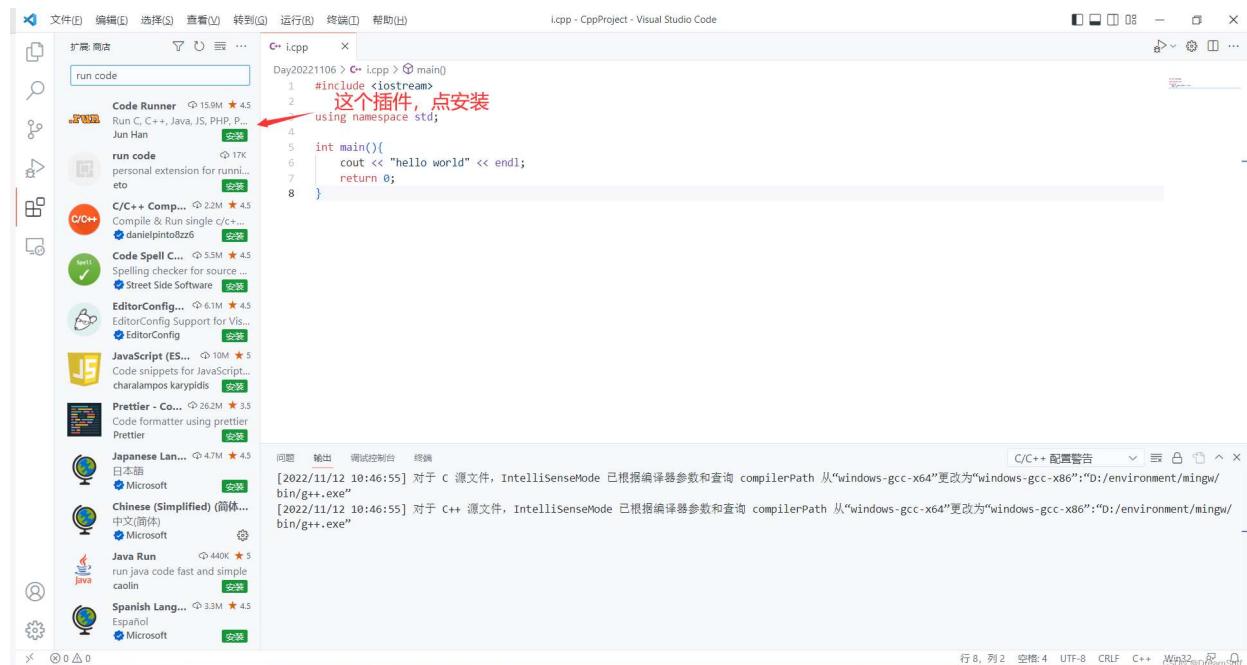
```
#include <iostream>
using namespace std;
int main(){
    int n;
    cin >> n;
    while(n--){
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main(){
    cout << "hello world" << endl;
    return 0;
}
```

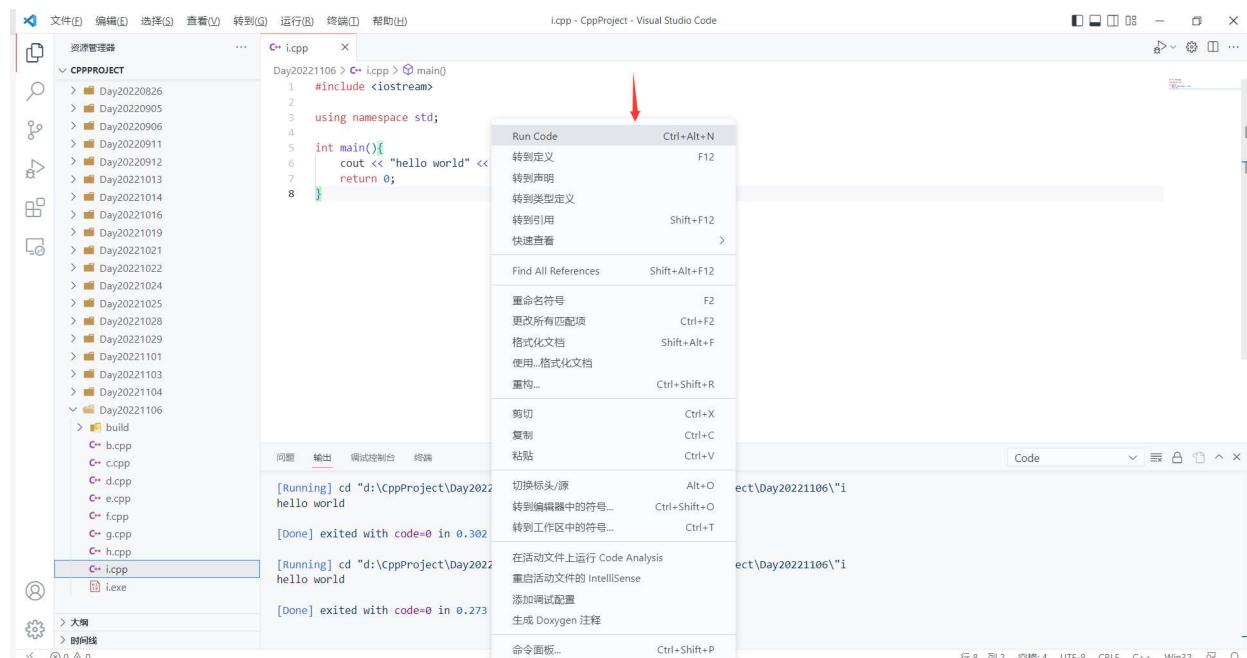
然后我们就可以开始配置环境了，配置的环境只适用于这个大文件夹内

## 第二步 (1) 简单的环境配置方法 - run code调试

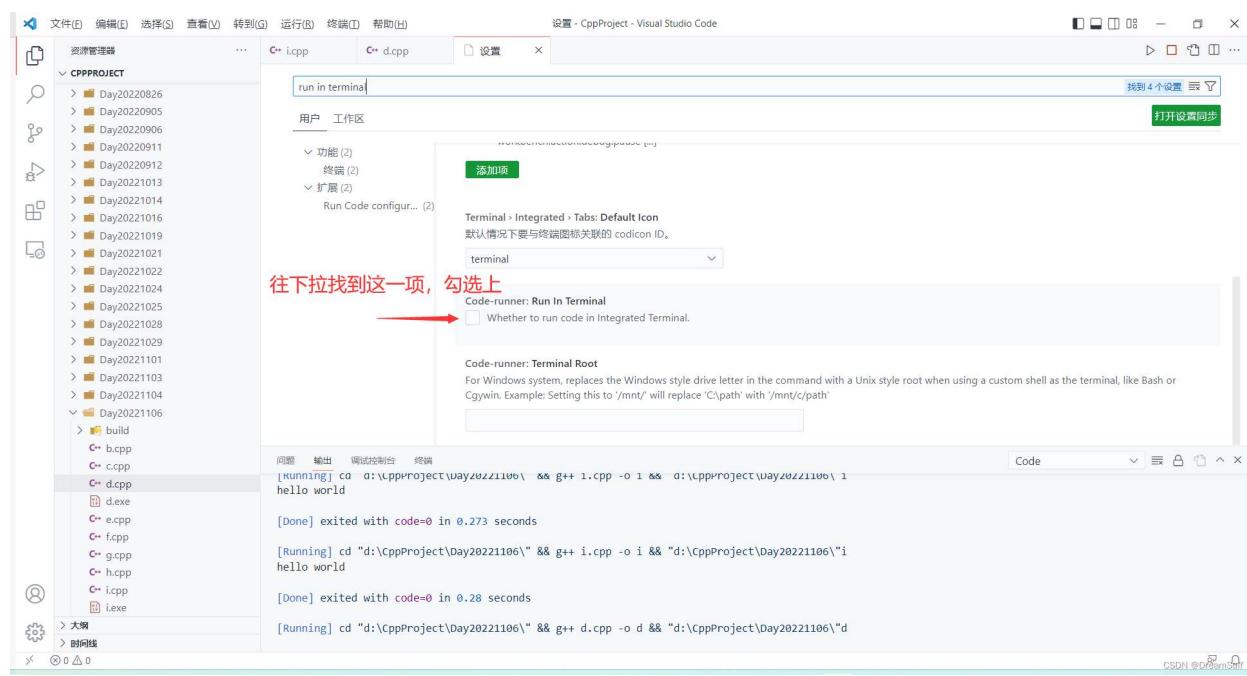
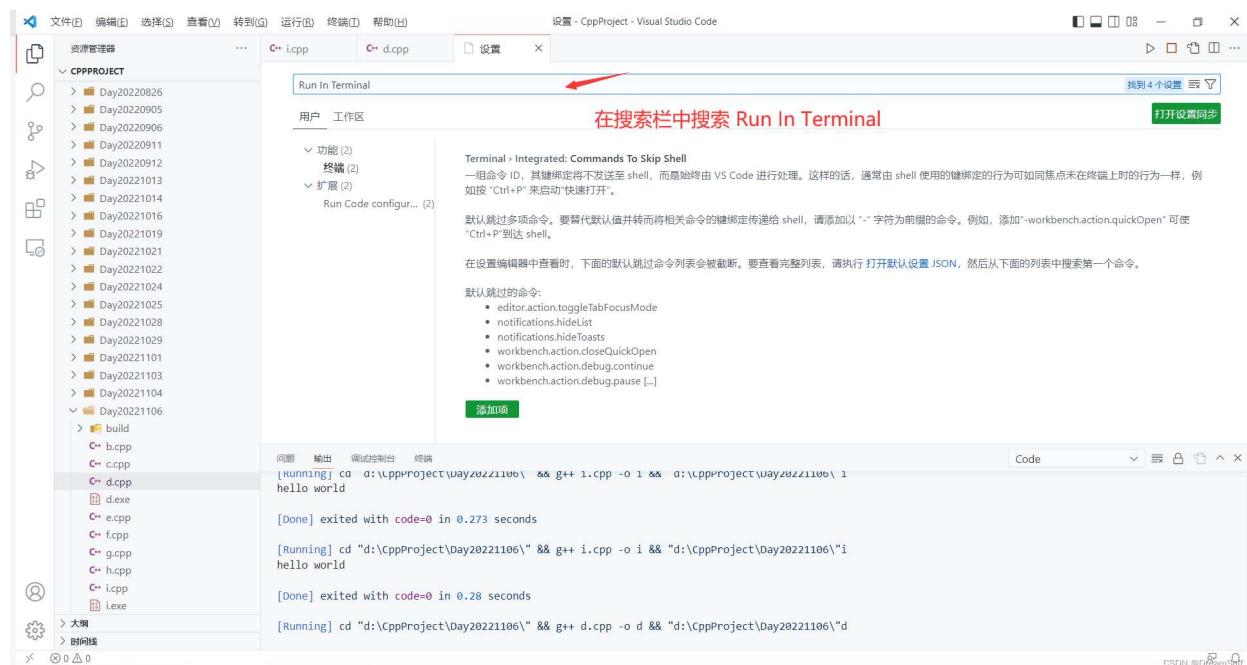
这个方法需要用到一个插件，run code，我们打开扩展商店安装一下run code插件



安装完成了后，我们会发现右键多了个run code



直接点run code就能够编译运行程序了，当然，这样只能得到输出结果，而不能直接在vscode中向程序中输入内容，所以我们还需要一些设置来调整一下run code，我们按 ctrl + , 进入设置（，是一个按键，在M键旁边）



重启一下vscode，这样我们就可以在vscode中向程序输入内容了

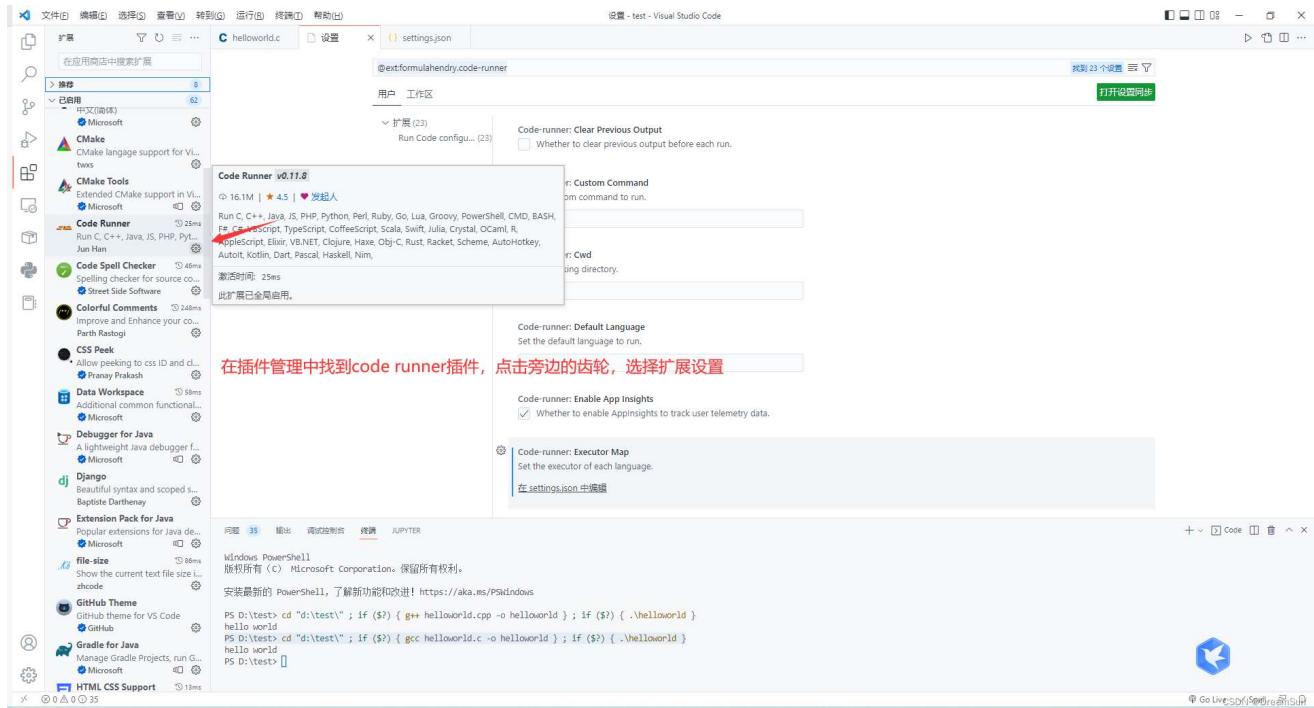
## 问题1：run code执行代码出现gcc(或g++) :error; no such file or directory的错误

### 原因分析

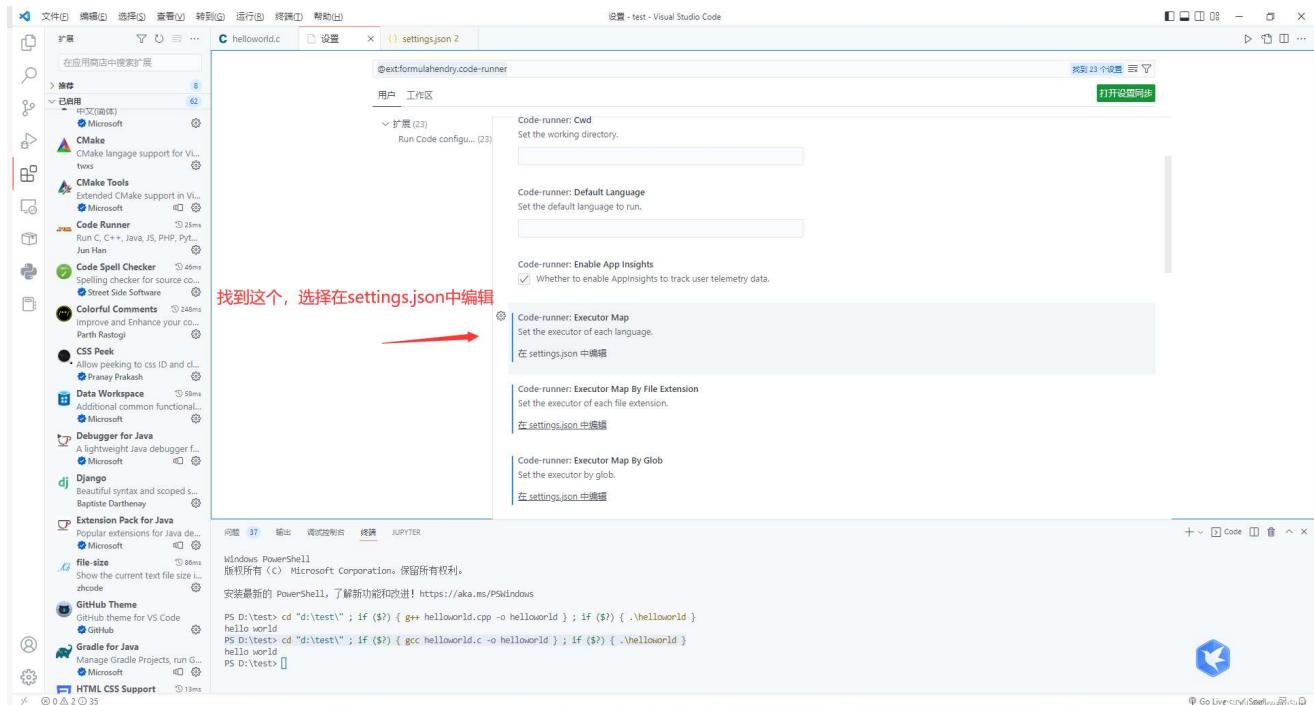
这个错误可能是编译命令执行前没有进入指定的文件夹前导致的，所以我们加一个cd 目标文件夹路径即可

# 解决办法

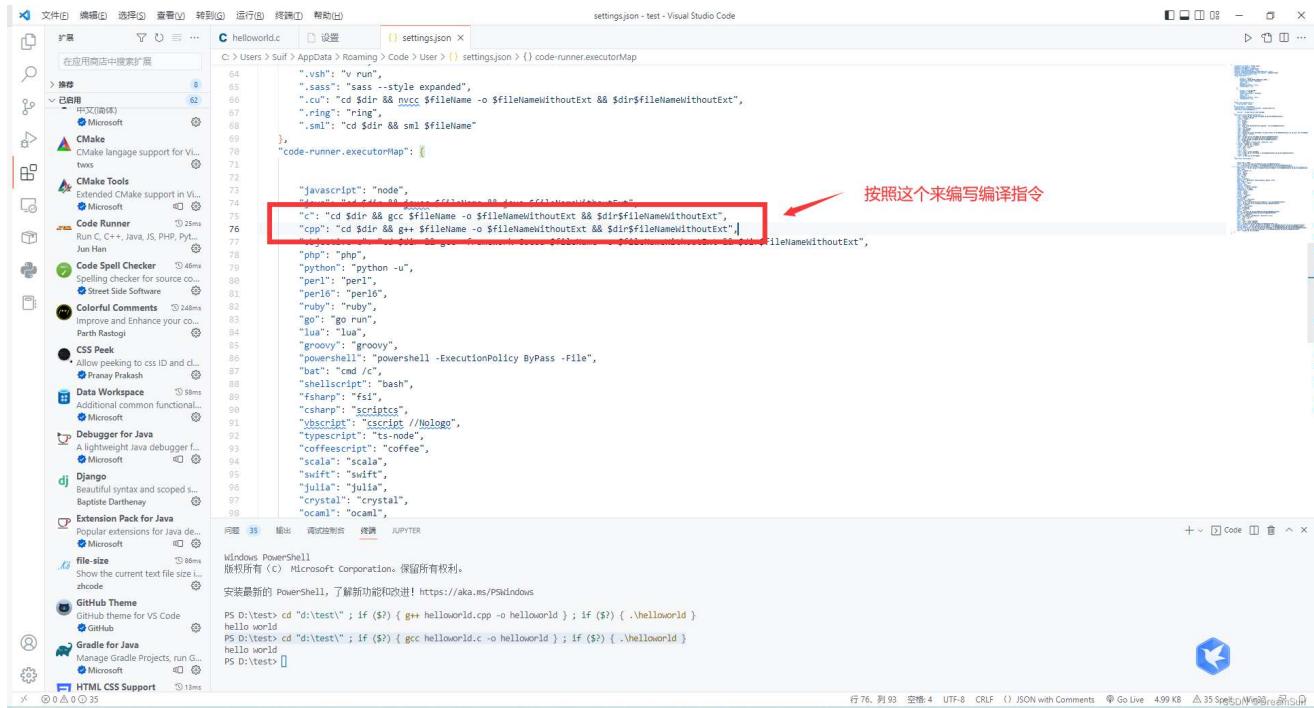
## 第一步 打开code runner扩展设置



## 第二步 进入指令设置界面



### 第三步 修改编译指令



```
C:\Users\Surf>AppData>Roaming>Code>User>settings.json>{}>code-runner.executorMap

64     ".vsh": "v run",
65     ".cu": "cd $dir && nvcc $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",
66     ".m": "cd $dir & clang $fileNameWithoutExt",
67     ".sm": "cd $dir & sm1 $fileName"
68   },
69   "code-runner.executorMap": {
70     ".js": "node",
71     ".py": "python -u",
72     ".c": "cd $dir && gcc $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",
73     ".cpp": "cd $dir && g++ $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",
74     ".java": "javac $fileName && java $fileNameWithoutExt",
75     ".php": "php",
76     ".py": "python",
77     ".perl": "perl",
78     ".perl6": "perl6",
79     ".ruby": "ruby",
80     ".go": "go run",
81     ".lua": "lua",
82     ".groovy": "groovy",
83     ".powershell": "powershell -ExecutionPolicy ByPass -File",
84     ".bat": "cmd /C",
85     ".sh": "sh",
86     ".shscript": "bash",
87     ".fsharp": "fsi",
88     ".csharp": "scrlntcs",
89     ".vbscript": "cscript //Nologo",
90     ".typescript": "ts-node",
91     ".coffeescript": "coffee",
92     ".scala": "scala",
93     ".swift": "swift",
94     ".julia": "julia",
95     ".crystal": "crystal",
96     ".ocaml": "ocaml"
97   }
98 }
```

```
"c": "cd $dir && gcc $fileName -o $fileNameWithoutExt &&  
$dir$fileNameWithoutExt",  
"cpp": "cd $dir && g++ $fileName -o $fileNameWithoutExt &&  
$dir$fileNameWithoutExt",
```

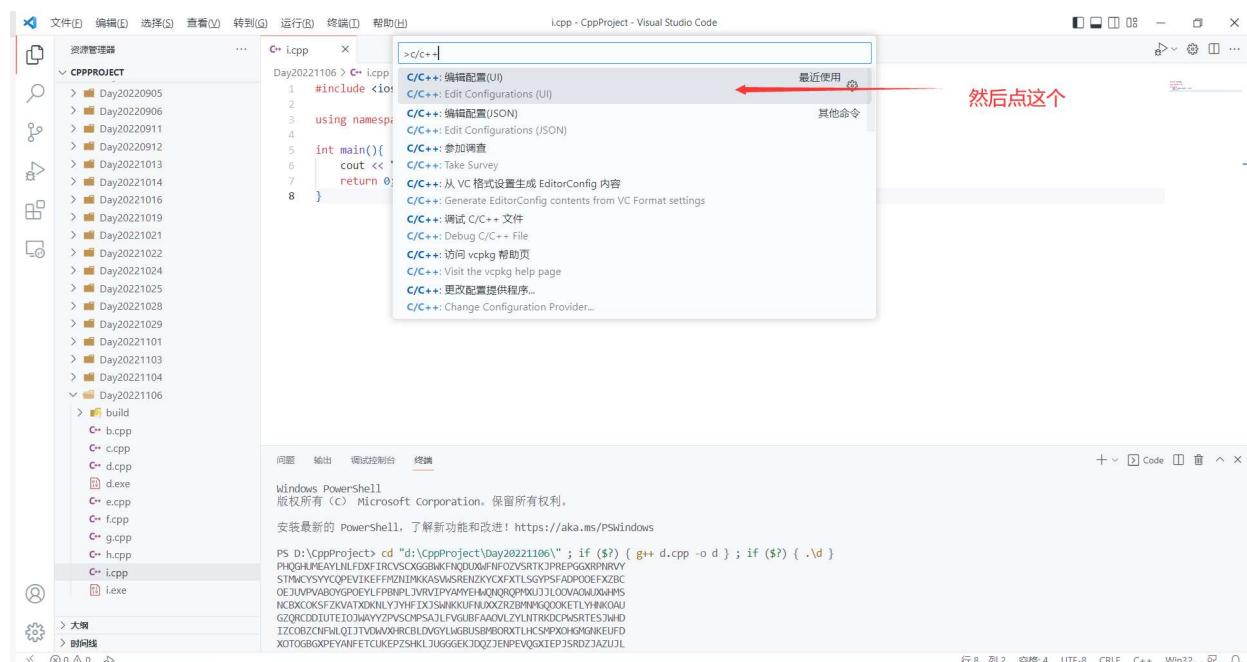
- 1
- 2

## 第二步 (2) 复杂的环境配置方法 - gdb调试

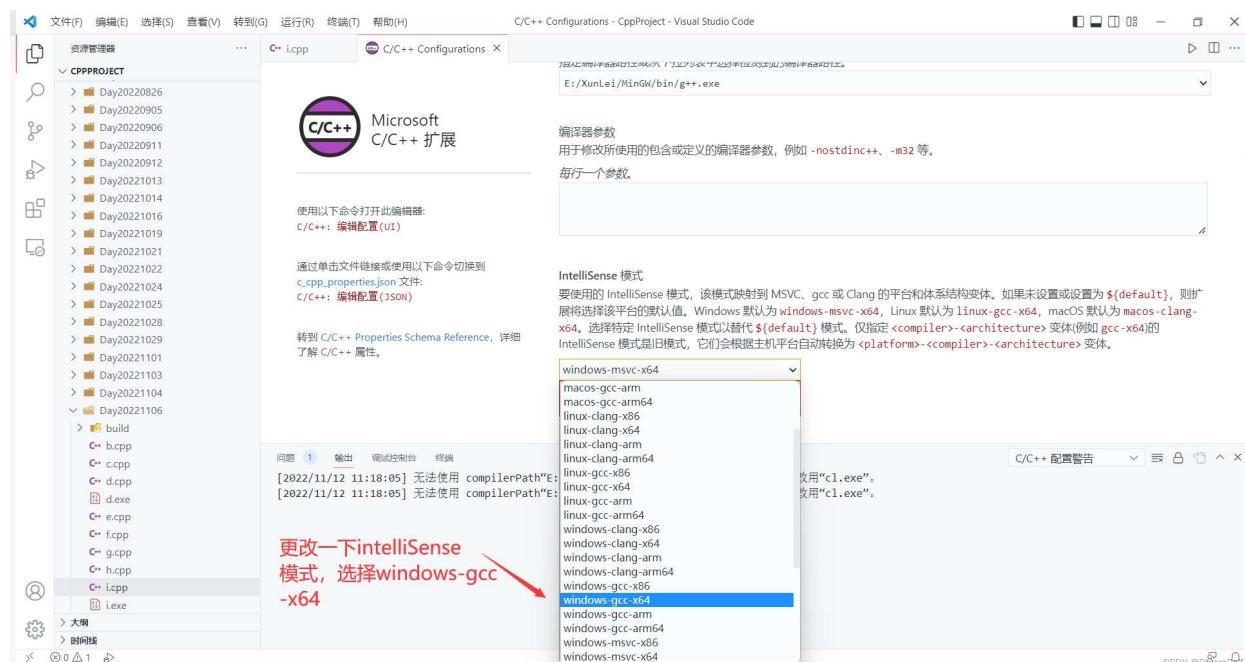
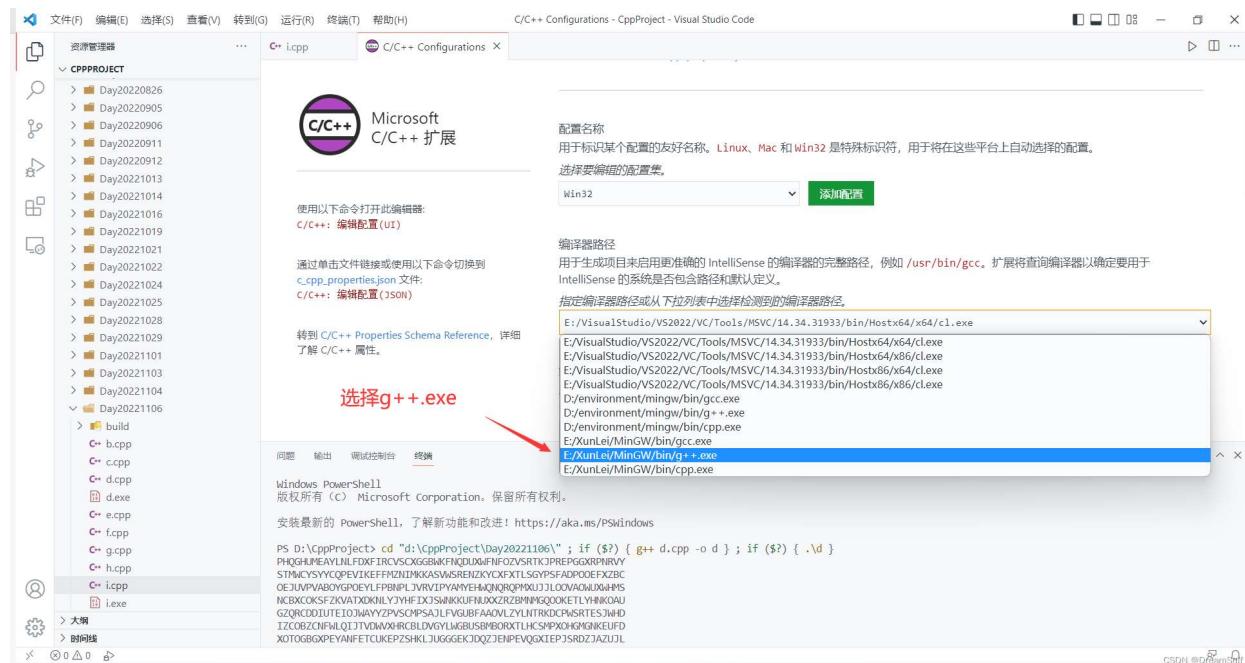
当然，不借助run code的方法我们也有，而且，我个人更推荐这种环境配置方法。

# 第一步 配置编译器环境

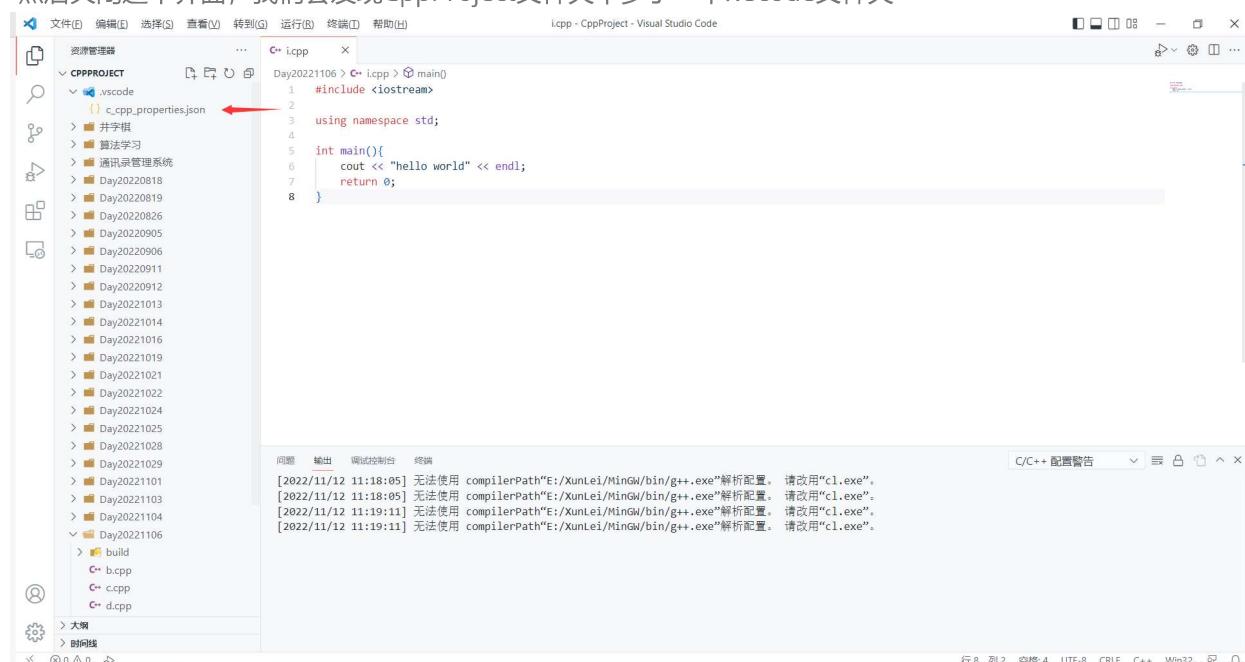
我们按住 **ctrl + shift + p**



我们配置一下编译器路径

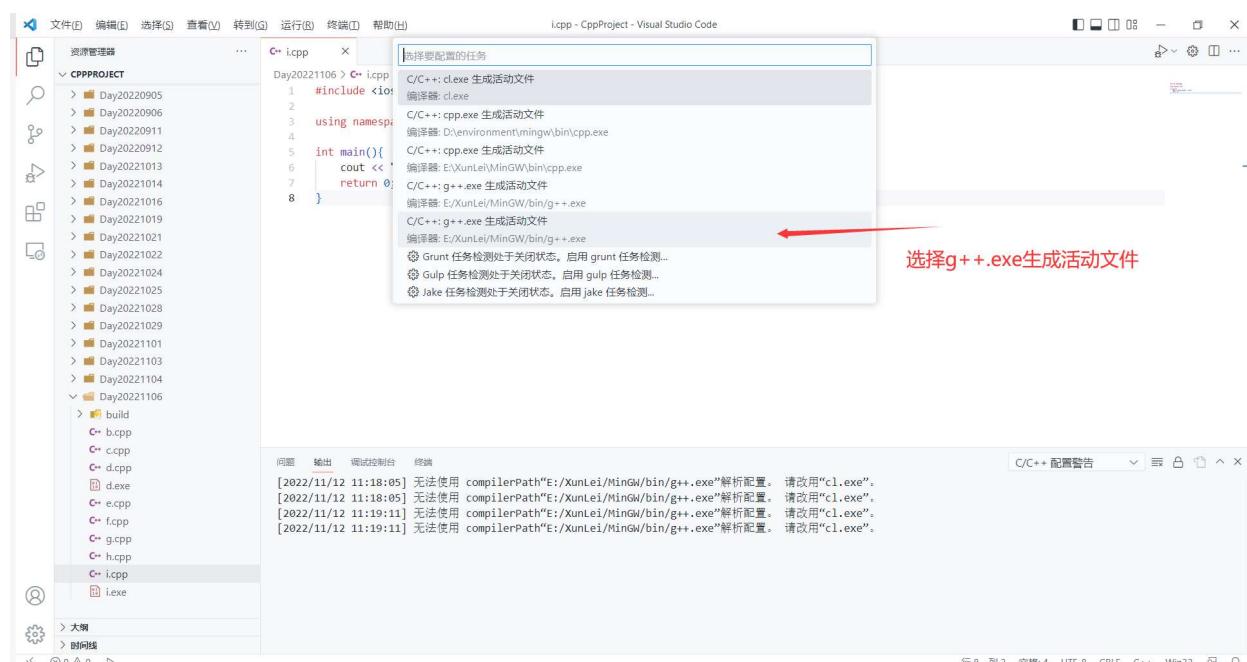
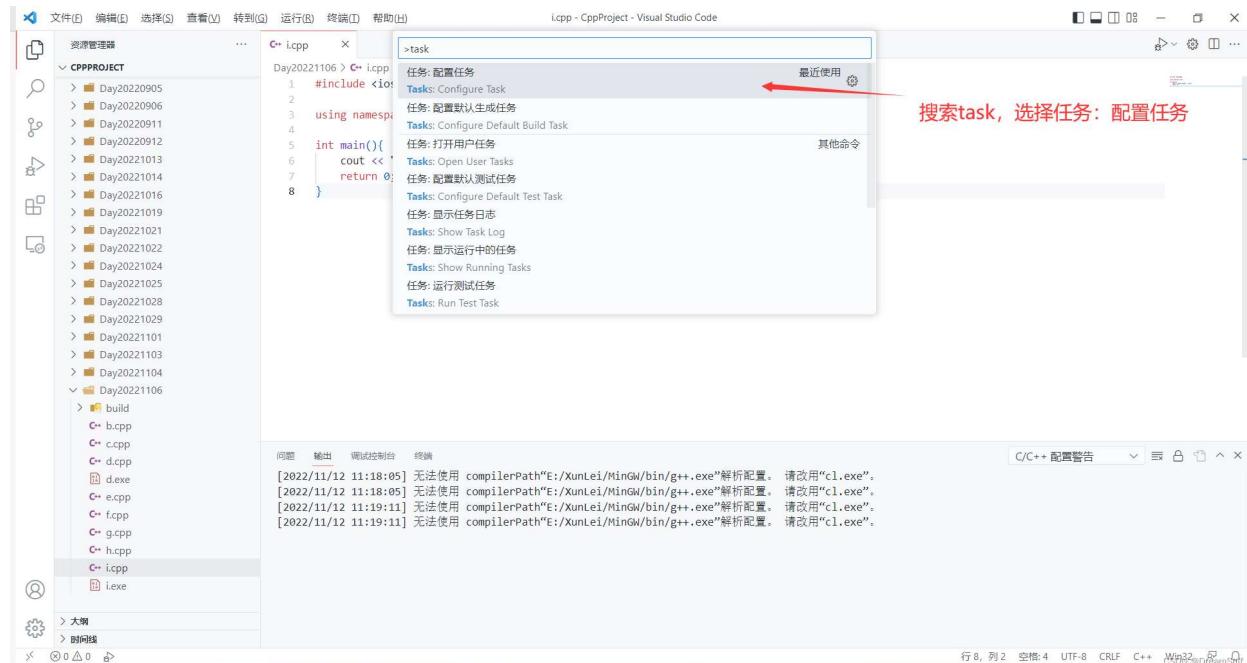


然后关闭这个界面，我们会发现CppProject文件夹中多了一个.vscode文件夹



## 第二步 配置task编译任务

同样，我们按住 **ctrl + shift + p**



此时我们会发现.vscode文件夹中多了tasks.json

简单地介绍一下tasks.json中需要注意的内容

```

{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++ g++.exe 生成 活动文件", /* 编译任务名 */
            "command": "E:/XunLei/MinGW/bin/g++.exe",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}\${fileBasenameNoExtension}.exe" /* 生成的路径exe程序 */
            ],
            "options": {
                "cwd": "E:/XunLei/MinGW/bin"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            }
        }
    ]
}

```

## 第三步 配置launch调试任务

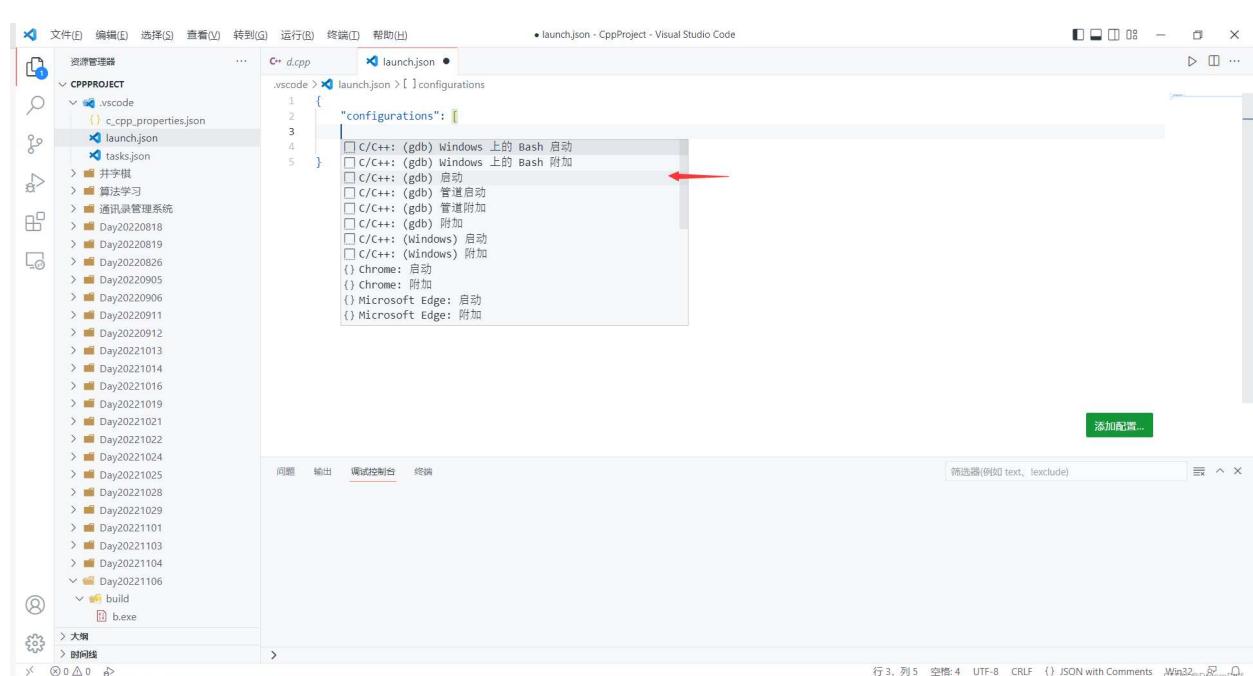
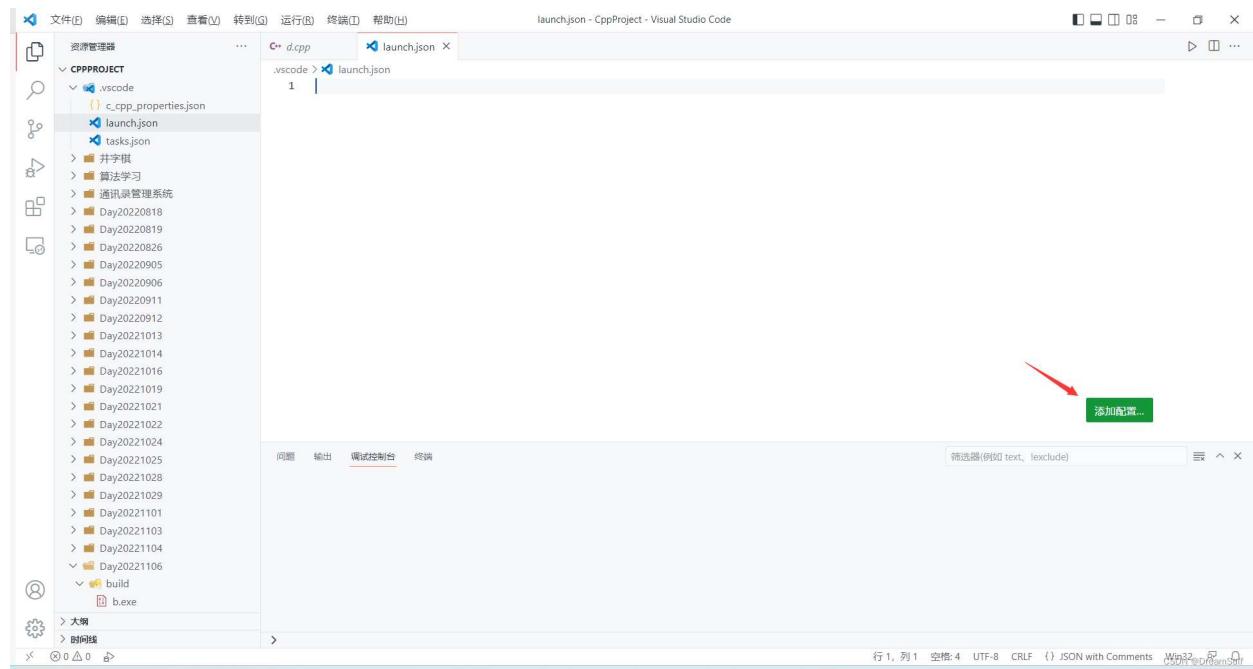
因为C/C++插件不会自动生成launch.json，因此launch.json需要我们自己编写，在.vscode文件夹内新建一个launch.json文件

```

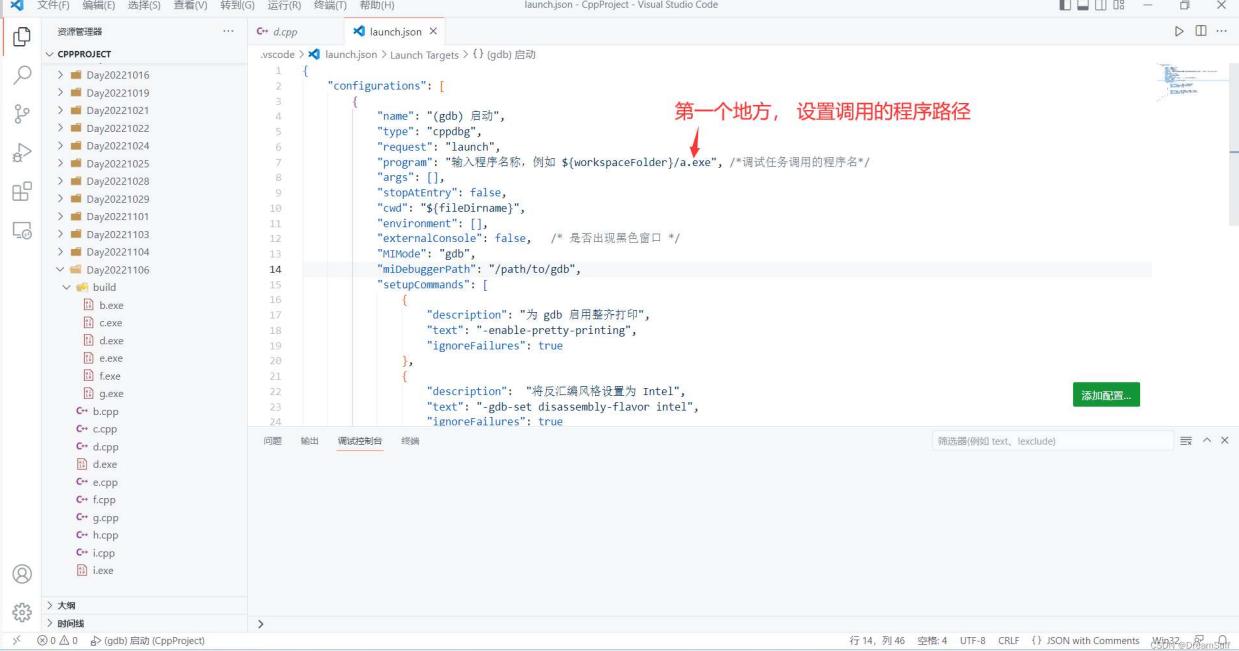
res = 0;
for(int i=1;i<29;i++){
    for(int j=1;j<59;j++){
        int k=0;
        bool flag = true;
        for(; k<4; k++){
            int x = i + dir[k][0], y=i+dir[k][1];
            if(c[x][y]<=c[i][j]){
                flag = false;
                break;
            }
        }
        if(flag) res++;
    }
}
cout << maxx << endl;
cout << res << endl;
for(int i=0; i<30;i++){
    for(int j=0; j<60;j++){
        cout << c[i][j];
    }
}

```

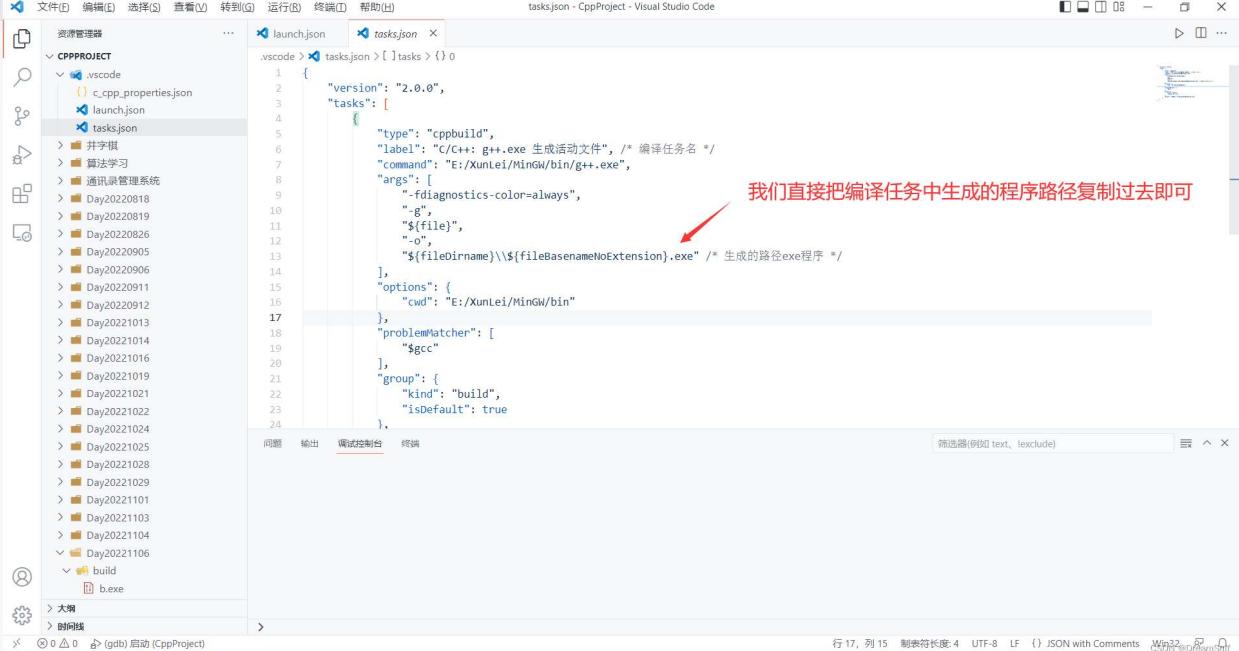
我们会发现，这里有个添加配置



## 第一处



```
1  {
2    "configurations": [
3      {
4        "name": "(gdb) 启动",
5        "type": "cppdbg",
6        "request": "launch",
7        "program": "${workspaceFolder}/a.exe", /* 调试任务调用的程序名 */
8        "args": [],
9        "stopAtEntry": false,
10       "cwd": "${fileDirname}",
11       "environment": [],
12       "externalConsole": false, /* 是否出现黑色窗口 */
13       "MIMode": "gdb",
14       "miDebuggerPath": "/path/to/gdb",
15       "setupCommands": [
16         {
17           "description": "为 gdb 启用整齐打印",
18           "text": ".enable-pretty-printing",
19           "ignoreFailures": true
20         },
21         {
22           "description": "将反汇编风格设置为 Intel",
23           "text": ".gdb-set disassembly-flavor intel",
24           "ignoreFailures": true
25       }
26     ]
27   }
28 }
```



```
1  {
2    "version": "2.0.0",
3    "tasks": [
4      {
5        "type": "cppbuild",
6        "label": "C/C++ g++_exe 生成活动文件", /* 编译任务名 */
7        "command": "E:/XunLei/MinGW/bin/g++.exe",
8        "args": [
9          "-fdiagnostics-color=always",
10         "-g",
11         "${file}",
12         "-o",
13         "${fileDirname}\${fileBasenameNoExtension}.exe" /* 生成的路径exe程序 */
14       ],
15       "options": {
16         "cwd": "E:/XunLei/MinGW/bin"
17       },
18       "problemMatcher": [
19         "$gcc"
20       ],
21       "group": {
22         "kind": "build",
23         "isDefault": true
24       }
25     }
26   }
27 }
```

```

{
  "configurations": [
    {
      "name": "(gdb) 启动",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}\\${fileBasenameNoExtension}.exe", /*调试任务调用的程序名*/
      "args": [],
      "stopAtEntry": false,
      "cwd": "${fileDirname}",
      "environment": [],
      "externalConsole": false, /* 是否出现黑色窗口 */
      "MIMode": "gdb",
      "mimDebuggerPath": "/path/to/gdb",
      "setupCommands": [
        {
          "description": "为 gdb 启用整齐打印",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        },
        {
          "description": "将反汇编风格设置为 Intel",
          "text": "-gdb-set disassembly-flavor intel",
          "ignoreFailures": true
        }
      ]
    }
  ]
}

```

## 第二处

第二处,gdb.exe调试程序路径

```

{
  "configurations": [
    {
      "name": "(gdb) 启动",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}\\${fileBasenameNoExtension}.exe", /*调试任务调用的程序名*/
      "args": [],
      "stopAtEntry": false,
      "cwd": "${fileDirname}",
      "environment": [],
      "externalConsole": false, /* 是否出现黑色窗口 */
      "MIMode": "gdb",
      "mimDebuggerPath": "/path/to/gdb", // 第二处,gdb.exe调试程序路径
      "setupCommands": [
        {
          "description": "为 gdb 启用整齐打印",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        },
        {
          "description": "将反汇编风格设置为 Intel",
          "text": "-gdb-set disassembly-flavor intel",
          "ignoreFailures": true
        }
      ]
    }
  ]
}

```

我们直接复制g++.exe路径，然后把g++改成gdb.exe即可

```

1  {
2      "version": "2.0.0",
3      "tasks": [
4          {
5              "type": "cppbuild",
6              "label": "C/C++ g++_exe 生成活动文件", /* 编译任务名 */
7              "command": "E:/XunLei/MinGW/bin/g++.exe",
8              "args": [
9                  "-fdiagnostics-color=always",
10                 "-g",
11                 "${file}",
12                 "-o",
13                 "${fileDirname}\${fileBasenameNoExtension}.exe" /* 生成的路径exe程序 */
14             ],
15             "options": {
16                 "cwd": "E:/XunLei/MinGW/bin"
17             },
18             "problemMatcher": [
19                 "$gcc"
20             ],
21             "group": {
22                 "kind": "build",
23                 "isDefault": true
24             }
25         }
26     ]
27 }

```

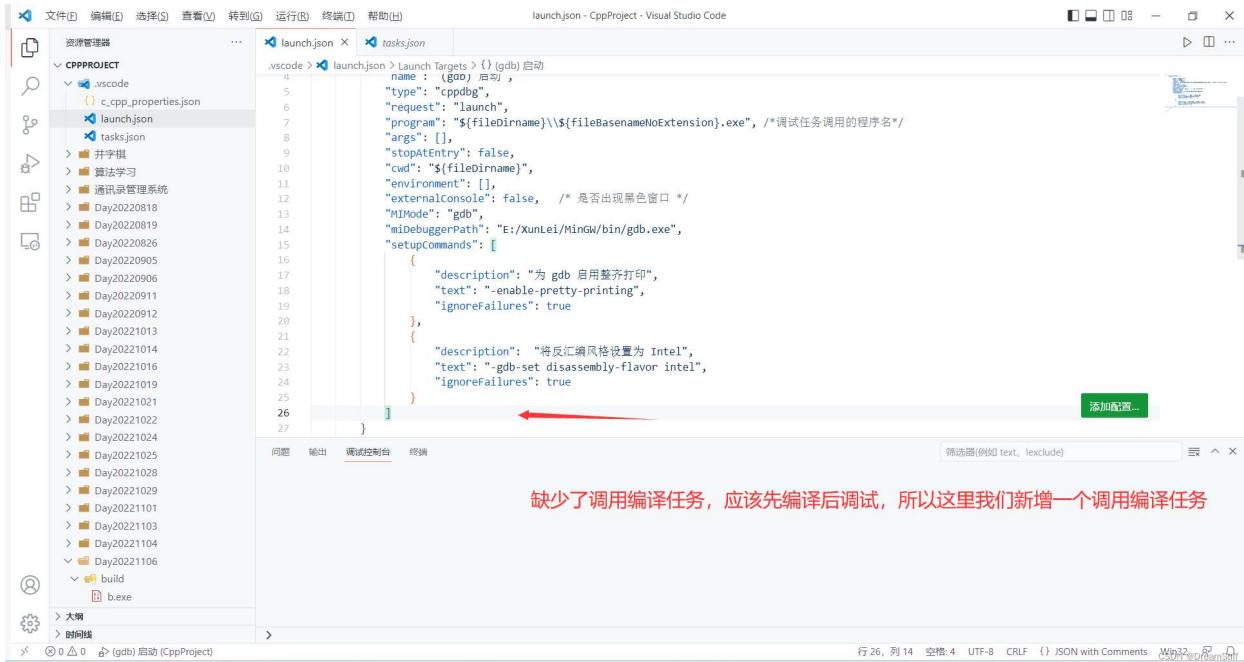
就像这样

```

1  {
2      "configurations": [
3          {
4              "name": "(gdb) 启动",
5              "type": "cppdbg",
6              "request": "launch",
7              "program": "${fileDirname}\${fileBasenameNoExtension}.exe", /* 调试任务调用的程序名 */
8              "args": [],
9              "stopAtEntry": false,
10             "cwd": "${fileDirname}",
11             "environment": [],
12             "externalConsole": false, /* 是否出现黑色窗口 */
13             "MIMode": "gdb",
14             "miDebuggerPath": "E:/XunLei/MinGW/bin/gdb.exe",
15             "setupCommands": [
16                 {
17                     "description": "为 gdb 启用整齐打印",
18                     "text": ".enable-pretty-printing",
19                     "ignoreFailures": true
20                 },
21                 {
22                     "description": "将反汇编风格设置为 Intel",
23                     "text": ".gdb-set disassembly-flavor intel",
24                     "ignoreFailures": true
25                 }
26             ]
27 }
28 ]
29 }

```

### 第三处



launch.json - CppProject - Visual Studio Code

```
vscode > launch.json > tasks.json
```

```
name : "(gdb) 后缀",
"request": "launch",
"program": "${fileDirname}\\${fileBasenameNoExtension}.exe", /*调试任务调用的程序名*/
"args": [],
"stopAtEntry": false,
"cwd": "${fileDirname}",
"environment": [],
"externalConsole": false, /*是否出现黑色窗口 */
"MIMode": "gdb",
"midebuggerPath": "E:/XunLei/MinGW/bin/gdb.exe",
"setupCommands": [
    {
        "description": "为 gdb 启用整齐打印",
        "text": ".enable.pretty-printing",
        "ignoreFailures": true
    },
    {
        "description": "将反汇编风格设置为 Intel",
        "text": ".gdb-set disassembly-flavor intel",
        "ignoreFailures": true
    }
]
```

缺少了调用编译任务，应该先编译后调试，所以这里我们新增一个调用编译任务

```

{
    "version": "0.1",
    "configurations": [
        {
            "name": "gdb 启动 (CppProject)",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/b.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "miDebuggerPath": "E:/XunLei/MinGW/bin/gdb.exe",
            "setupCommands": [
                {
                    "description": "为 gdb 启用整齐打印",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                },
                {
                    "description": "将反汇编风格设置为 Intel",
                    "text": ".gdb-set disassembly-flavor intel",
                    "ignoreFailures": true
                }
            ],
            "preLaunchTask": ""
        }
    ]
}

```

调试前调用任务，我们直接把编译任务中的label复制过来就行

```

{
    "version": "0.1",
    "configurations": [
        {
            "name": "gdb 启动 (CppProject)",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/b.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "miDebuggerPath": "E:/XunLei/MinGW/bin/gdb.exe",
            "setupCommands": [
                {
                    "description": "为 gdb 启用整齐打印",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                },
                {
                    "description": "将反汇编风格设置为 Intel",
                    "text": ".gdb-set disassembly-flavor intel",
                    "ignoreFailures": true
                }
            ],
            "preLaunchTask": "C/C++: g++.exe 生成活动文件"
        }
    ]
}

```

保存，重启一下编辑器，我们就可以按F5调试程序了

注意：任何文件做出更改后都需要保存，新内容才会生效，我们可以按 **ctrl + s** 快速保存文件

## 一些问题

### 问题1

问：为什么按F5后出现这个报错信息，没有输出结果

```
PS D:\CppProject> & 'c:\Users\Suif\.vscode\extensions\ms-vscode.cppTools-1.12.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-v11lhtf.3a4' '--stdout=Microsoft-MIEngine-Out-u2kfqvz.ta' '--stderr=Microsoft-MIEngine-Error-dpduza1.v2m' '--pid=Microsoft-MIEngine-Pid-dcmv2i.rtr' '--dbgExe=D:\environment/mingw/bin/gdb.exe' '--interpret=<file>' @DreamSuif
```

答：这并不是报错信息，而是编译调试命令，输出的结果在调试控制台中显示。

```

    #include <iostream>
    using namespace std;
    int main()
    {
        cout << "Hello world" << endl;
        return 0;
    }

```

输出内容在这里

## 进阶 设置成经典的弹出黑窗运行程序的形式（在系统终端中运行程序）

有的小伙伴可能觉得调试控制台不太好用，想要vscode编译运行c/c++程序时弹出终端，在终端中运行程序，这点其实也很简单，我们只需要改变launch.json中的 program 和 args 中的内容，然后设置一下黑窗弹出即可。

```

{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(gdb) 启动",
            "type": "cppdbg",
            "request": "launch",
            "program": "c:\Windows\System32\cmd.exe",
            "args": [
                "-c",
                "${filebasename}\${filebasenameNoExtension}.exe",
                "&",
                "pause"
            ],
            "stopAtEntry": false,
            "cwd": "${fileDirname}\${filebasename}",
            "environment": [],
            "externalConsole": true,
            "MIMode": "gdb",
            "miDebuggerPath": "D:/environment/mingw/bin/gdb.exe",
            "setupCommands": [
                {
                    "description": "为 gdb 启用整齐打印",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                },
                {
                    "description": "将反汇编风格设置为 intel",
                    "text": "-gdb-set disassembly-flavor intel",
                    "ignoreFailures": true
                }
            ],
            "preLaunchTask": "C/C++: Build Task"
        }
    ]
}

```

把program调用任务改为调用系统终端，因为系统终端的位置基本都在这里，所以可以直接照着抄就行。

args中的内容是传递给终端的命令，这个意思是运行指定位置的程序，待程序运行完成后暂停。  
只需要把第二行的程序路径改成和task.json中相同的程序路径即可其他的照抄

就像这样，我们调用系统终端，执行运行程序的命令，为了不让终端执行完程序后自动退出，所以我们还加了一个 pause命令

生成的程序路径，直接复制到launch.json中即可

```

1  "tasks": [
2    {
3      "type": "cppbuild",
4      "label": "C/C++构建",
5      "command": "D:/environment/mingw/bin/g++.exe",
6      "args": [
7        "-fdiagnostics-color=always",
8        "-g",
9        "${file}",
10       "-o",
11       "${fileDirname}\\build\\${filebasenameNoExtension}.exe",
12     ],
13     "options": {
14       "cwd": "D:/environment/mingw/bin"
15     },
16     "problemMatcher": [
17       "gcc"
18     ],
19     "group": {
20       "kind": "build",
21       "isDefault": true
22     },
23   },
24   {
25     "detail": "调试器生成的任务。"
26   }
27 ],
28 }

```

这一项改为true，才会弹出系统终端窗口，否则你是看不到黑窗的

```

1  "configurations": [
2    {
3      "name": "(gdb) 启动",
4      "type": "cppdbg",
5      "request": "launch",
6      "program": "${fileDirname}\\build\\${filebasenameNoExtension}.exe",
7      "args": [
8        "-g",
9        "-p"
10      ],
11      "externalConsole": true,
12      "stopAtEntry": false,
13      "cwd": "${fileDirname}",
14      "environment": [],
15      "setupCommands": [
16        {
17          "description": "为 gdb 启用整齐打印",
18          "text": "-enable-pretty-printing",
19          "ignoreFailures": true
20        },
21        {
22          "description": "将反汇编风格设置为 Intel",
23          "text": "-gdb-set disassembly-flavor intel",
24          "ignoreFailures": true
25        }
26      ]
27    }
28  ]

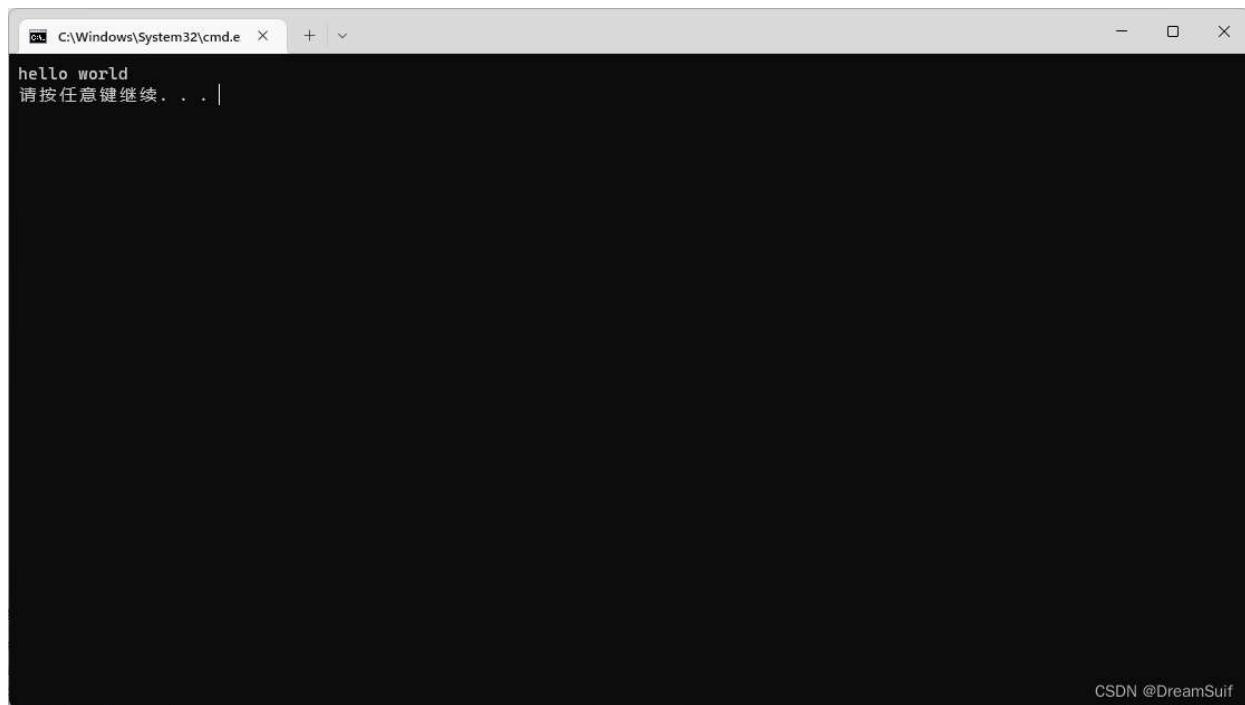
```

我的program 和 args 中的内容：

```
"program": "C:\\Windows\\System32\\cmd.exe",
"args": [
    "/c",
    "${fileDirname}\\${fileBasenameNoExtension}.exe",
    "&",
    "pause"
],
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7

完成后的效果



CSDN @DreamSuif

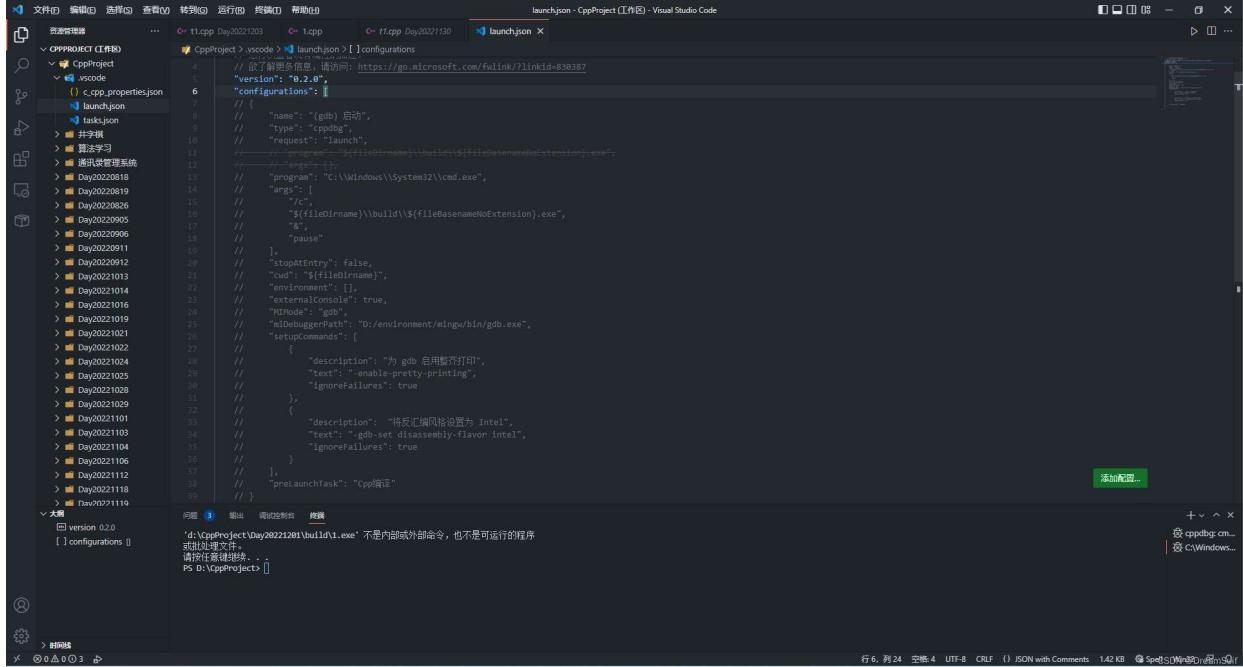
## 进阶 设置在vscode内置终端中执行程序

有的小伙伴既不想用run code等插件，又想要在vscode内置终端中运行程序，这种方法我们当然也有！

我们只需要改变launch.json中的内容即可。（因为gdb调试不能在vscode内置终端中执行，所以我们得选择window启动调试任务）

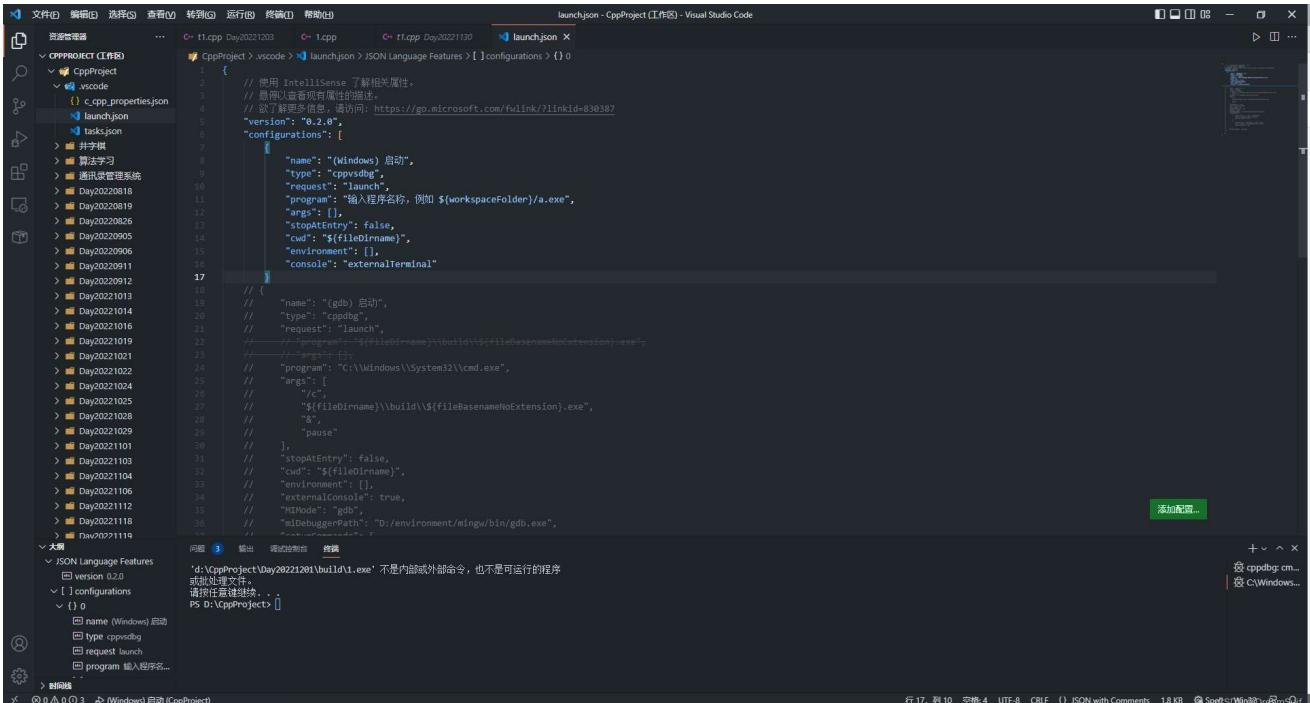
先把configurations中的内容注释掉（因为后面配置过程中需要用到之前的一些内容），全选

configurations中的内容，按下快捷键ctrl + / 即可快速注释选中内容，



```
// 欲了解更多信息，请访问: https://go.microsoft.com/fwlink/?LinkId=830387
"version": "0.2.0",
"configurations": [
    {
        "name": "(gdb) 启动",
        "type": "cppdbg",
        "request": "launch",
        "program": "${fileDirname}\\build\\${fileBasenameNoExtension}.exe",
        "args": [
            "-c"
        ],
        "stopAtEntry": false,
        "cwd": "${fileDirname}",
        "environment": [],
        "externalConsole": true,
        "MIMode": "gdb",
        "miDebuggerPath": "D:/environment/mingw/bin/gdb.exe",
        "setupCommands": [
            {
                "description": "为 gdb 启用整齐打印",
                "text": "enable pretty-printing",
                "ignoreFailures": true
            }
        ],
        "description": "将反汇编风格设置为 Intel",
        "text": "gdb-set disassembly-flavor intel",
        "ignoreFailures": true
    },
    {
        "preLaunchTask": "Cpp编译"
    }
]
```

然后我们点击添加配置，选择windows启动，我们就会得到这样的一些内容：



```
// 使用 Intellisense 了解相关属性。
// 欲了解更多信息，请访问: https://go.microsoft.com/fwlink/?LinkId=830387
"version": "0.2.0",
"configurations": [
    {
        "name": "(Windows) 启动",
        "type": "cppvsdbg",
        "request": "launch",
        "program": "${workspaceFolder}/a.exe",
        "args": [],
        "stopAtEntry": false,
        "cwd": "${fileDirname}",
        "environment": [],
        "console": "externalTerminal"
    },
    {
        "name": "(gdb) 启动",
        "type": "cppdbg",
        "request": "launch",
        "program": "${fileDirname}\\build\\${fileBasenameNoExtension}.exe",
        "args": [
            "-c"
        ],
        "stopAtEntry": false,
        "cwd": "${fileDirname}",
        "environment": [],
        "externalConsole": true,
        "MIMode": "gdb",
        "miDebuggerPath": "D:/environment/mingw/bin/gdb.exe",
        "setupCommands": [
            {
                "description": "为 gdb 启用整齐打印",
                "text": "enable pretty-printing",
                "ignoreFailures": true
            }
        ],
        "description": "将反汇编风格设置为 Intel",
        "text": "gdb-set disassembly-flavor intel",
        "ignoreFailures": true
    },
    {
        "preLaunchTask": "Cpp编译"
    }
]
```

program, args中的内容改成和原来的program, args中的内容一样，我们直接复制粘贴过来即可（这个内容不需要和我的一样，我只是给个示范）

```

{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(Windows) 启动",
            "type": "cppvsdbg",
            "request": "launch",
            "program": "C:\Windows\System32\cmd.exe",
            "args": [
                "/c",
                "${fileDirname}\build\${fileBasenameNoExtension}.exe",
                "-g",
                "pause"
            ],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "console": "externalTerminal"
        }
    ]
}

```

同样的我们需要调试前调用编译任务生成文件

```

{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(Windows) 启动",
            "type": "cppvsdbg",
            "request": "launch",
            "program": "C:\Windows\System32\cmd.exe",
            "args": [
                "/c",
                "${fileDirname}\build\${fileBasenameNoExtension}.exe",
                "-g",
                "pause"
            ],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "console": "externalTerminal",
            "preLaunchTask": "Cpp编译"
        }
    ]
}

```

然后我们只需要改变console中的内容即可控制是在系统终端中运行程序还是在vscode终端中运行程序

该项值为 externalTerminal 则是在系统终端中运行程序

该项值为 integratedTerminal 则是在vscode终端中运行程序

```

    "configurations": [
        {
            "name": "(Windows) 启动",
            "type": "cppvsdbg",
            "request": "launch",
            "program": "C:\Windows\System32\cmd.exe",
            "args": [
                "/c",
                "${fileDirname}\build\${fileBasenameNoExtension}.exe",
                "-g",
                "pause"
            ],
            "environment": [],
            "console": "integratedTerminal",
            "preLaunchTask": "Cpp编译"
        }
    ]
}

```

我们把该项值改为 integratedTerminal 即可在vscode终端中运行程序

```

    "configurations": [
        {
            "name": "(Windows) 启动",
            "type": "cppvsdbg",
            "request": "launch",
            "program": "C:\Windows\System32\cmd.exe",
            "args": [
                "/c",
                "${fileDirname}\build\${fileBasenameNoExtension}.exe",
                "-g",
                "pause"
            ],
            "environment": [],
            "console": "integratedTerminal",
            "preLaunchTask": "Cpp编译"
        }
    ]
}

```

我们把该项值改为 integratedTerminal 即可在vscode终端中运行程序

## 最终效果：

The screenshot shows the Visual Studio Code interface with a C++ project open. The code editor displays a file named `1.cpp` containing the following code:

```
#include <stdio.h>
int main(){
    puts("hello world");
    return 0;
}
```

The terminal at the bottom shows the output of the program:

```
hello world
```

The status bar at the bottom right indicates the file is 77B in size and has 778 characters.