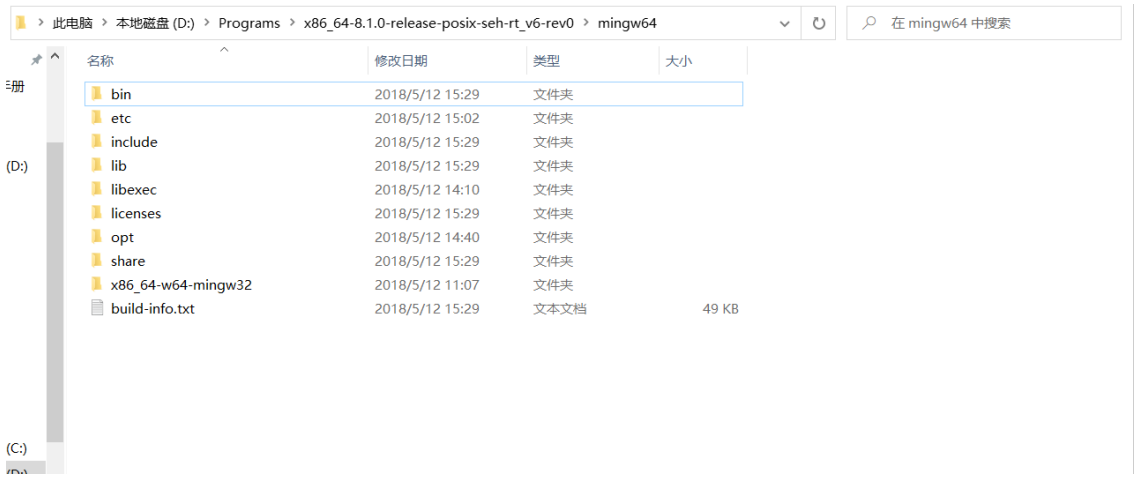
Windows下Visual Studio Code配置c++

下载并配置编译器

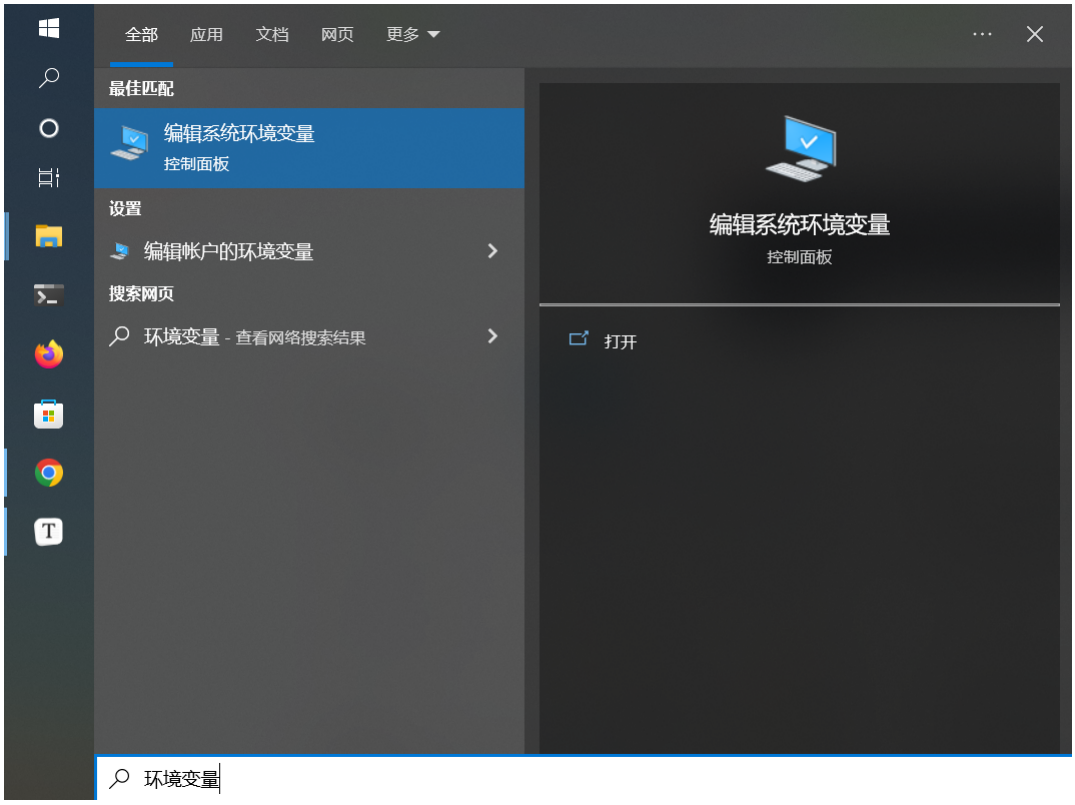
1. 进入mingw 下载地址
2. 选择MinGW-W64 GCC-8.1.0下的x86_64-posix-seh (**直接点击**就可以下载)
3. 把下载的压缩文件解压到你希望的目录下（如`D:\Programs`目录下），进入目录后，目录结构如下所示



4. 把`<Mingw path>\bin`加入到环境变量中，其中`<Mingw path>`是你直接解压的目录，示例：`D:\Programs\x86_64-8.1.0-release-posix-seh-rt_v6-rev0\mingw64\bin`

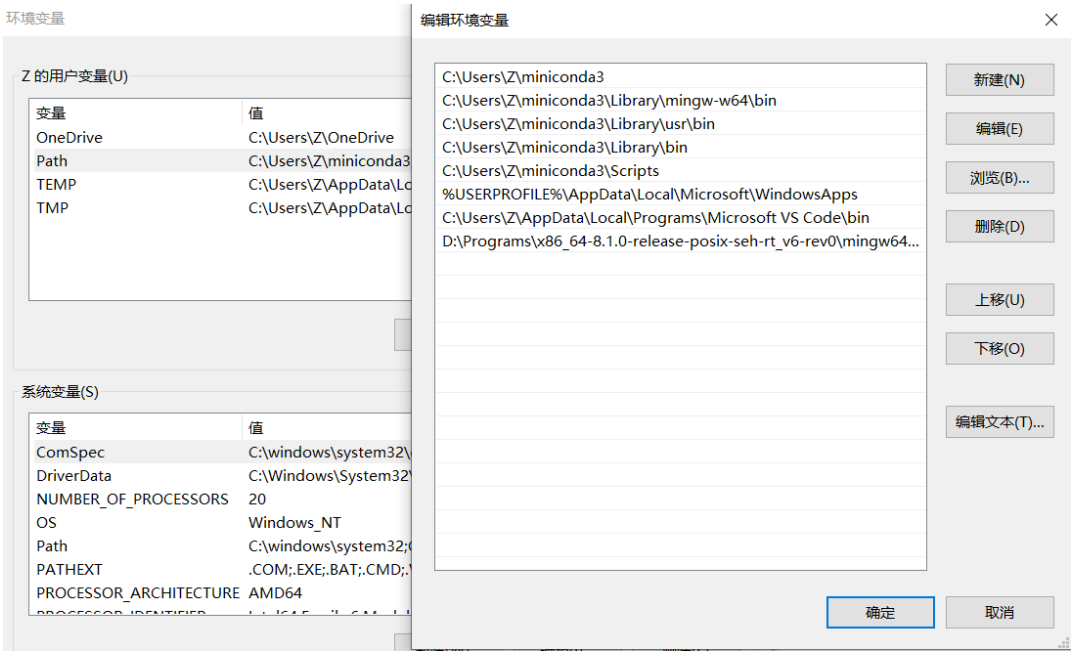
   修改环境变量步骤

   - win系统下，搜索`环境变量`，搜索快捷键`win + Q`，选择`编辑系统环境变量`



   - 点击`环境变量`

- 编辑`用户变量`或者`系统变量`里的`PATH`
- 新建后输入`<Mingw path>\bin`，示例:`D:\Programs\x86_64-8.1.0-release-posix-seh-rt_v6-rev0\mingw64\bin`



- 连续按三个`确定`

5. 测试一下编译器是否安装成功，打开cmd依次输入`g++ -v`，`gcc -v`，`gdb -v`，如果正确显示version则代表安装成功

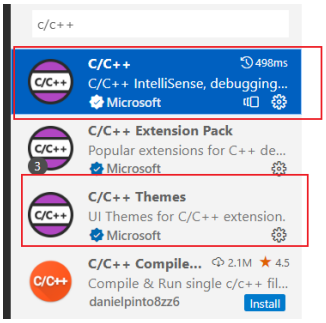
```
C:\Users\Z>g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=D:/Programs/x86_64-8.1.0-release-posix-seh-rt_v6-rev0/mingw64/bin/../libexec/gcc/x86_64-w64-mingw32/
8.1.0/lto-wrapper.exe
Target: x86_64-w64-mingw32
Configured with: ../../../src/gcc-8.1.0/configure --host=x86_64-w64-mingw32 --build=x86_64-w64-mingw32 --target=x86_64-w
64-mingw32 --prefix=/mingw64 --with-sysroot=/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64 --enable-shared --enable
-static --disable-multilib --enable-languages=c,c++,fortran,lto --enable-libstdcxx-time=yes --enable-threads=posix --ena
ble-libgomp --enable-libatomic --enable-lto --enable-graphite --enable-checking=release --enable-fully-dynamic-string --
enable-version-specific-runtime-libs --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-bootstrap --disable-rpat
h --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-gnu-as --with-gnu-ld --with-arch=noco
na --with-tune=core2 --with-libiconv --with-system-zlib --with-gmp=/c/mingw810/prerequisites/x86_64-w64-mingw32-static -
-with-mpfr=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-mpc=/c/mingw810/prerequisites/x86_64-w64-mingw32-s
tatic --with-isl=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-pkgversion='x86_64-posix-seh-rev0, Built by
MinGW-W64 project' --with-bugurl=https://sourceforge.net/projects/mingw-w64 CFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x
86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/
prerequisites/x86_64-w64-mingw32-static/include' CXXFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x86_64-810-posix-seh-rt_v6
-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-
mingw32-static/include' CPPFLAGS=' -I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prere
quisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' LDFLAGS='-pipe -fno-i
dent -L/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/lib -L/c/mingw810/prerequisites/x86_64-zlib-static/lib -L
/c/mingw810/prerequisites/x86_64-w64-mingw32-static/lib '
Thread model: posix
gcc version 8.1.0 (x86_64-posix-seh-rev0, Built by MinGW-W64 project)
```

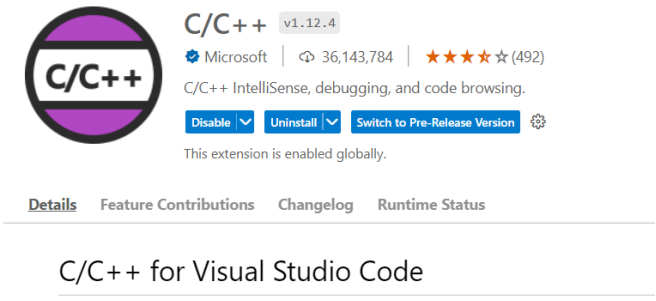
```
C:\Users\Z>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=D:/Programs/x86_64-8.1.0-release-posix-seh-rt_v6-rev0/mingw64/bin/../libexec/gcc/x86_64-w64-mingw32/
8.1.0/lto-wrapper.exe
Target: x86_64-w64-mingw32
Configured with: ../../../src/gcc-8.1.0/configure --host=x86_64-w64-mingw32 --build=x86_64-w64-mingw32 --target=x86_64-w
64-mingw32 --prefix=/mingw64 --with-sysroot=/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64 --enable-shared --enable
-static --disable-multilib --enable-languages=c,c++,fortran,lto --enable-libstdcxx-time=yes --enable-threads=posix --ena
ble-libgomp --enable-libatomic --enable-lto --enable-graphite --enable-checking=release --enable-fully-dynamic-string --
enable-version-specific-runtime-libs --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-bootstrap --disable-rpat
h --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-gnu-as --with-gnu-ld --with-arch=noco
na --with-tune=core2 --with-libiconv --with-system-zlib --with-gmp=/c/mingw810/prerequisites/x86_64-w64-mingw32-static -
-with-mpfr=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-mpc=/c/mingw810/prerequisites/x86_64-w64-mingw32-s
tatic --with-isl=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-pkgversion='x86_64-posix-seh-rev0, Built by
MinGW-W64 project' --with-bugurl=https://sourceforge.net/projects/mingw-w64 CFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x
86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/
prerequisites/x86_64-w64-mingw32-static/include' CXXFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x86_64-810-posix-seh-rt_v6
-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-
mingw32-static/include' CPPFLAGS=' -I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prere
quisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' LDFLAGS='-pipe -fno-i
dent -L/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/lib -L/c/mingw810/prerequisites/x86_64-zlib-static/lib -L
/c/mingw810/prerequisites/x86_64-w64-mingw32-static/lib '
Thread model: posix
gcc version 8.1.0 (x86_64-posix-seh-rev0, Built by MinGW-W64 project)
```

```
C:\Users\Z>gdb -v
GNU gdb (GDB) 8.1
Copyright (C) 2018 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 "x86_64-w64-mingw32".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
```

配置vscode

1. 从官网下载vscode (默认下载配置应该会把vscode的bin加入环境变量中去)
2. 安装`c/c++`插件

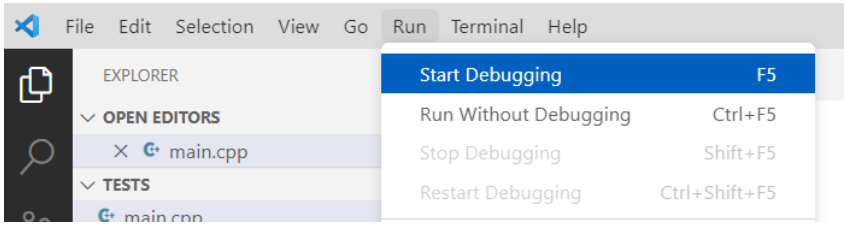




3. 随便写一个cpp

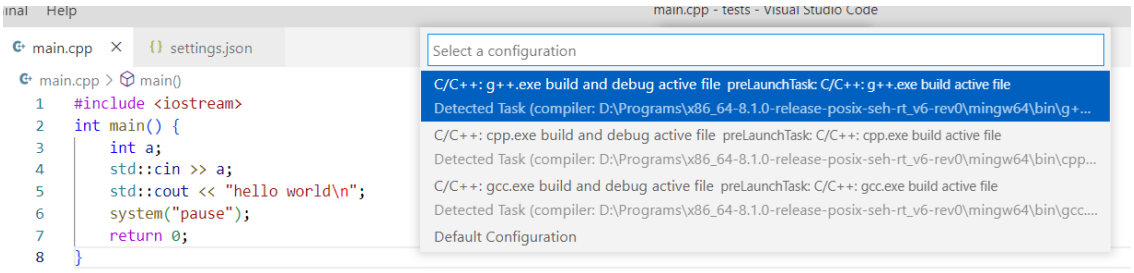
```cpp
#include <iostream>
int main() {
    std::cout << "hello world\n";
    return 0;
}
```

4. 点击`run` `start debugging`

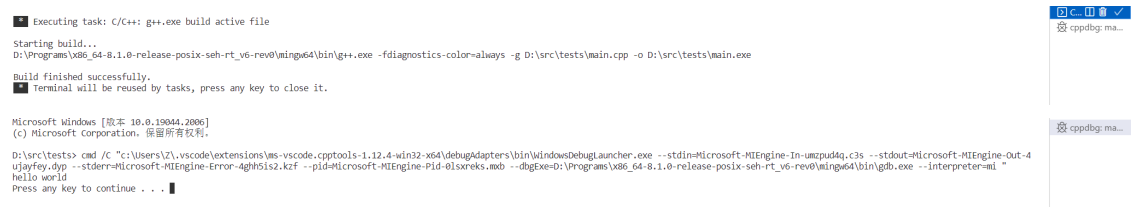


5. 选择`C++(GDB/LLDB)`

6. configuration选择第一个`C/C++:g++.exe build and debug active file`



7. 最后会在默认的terminal输出`hello world`



note: 如果vscode默认的terminal不是`powershell`或者`cmd`。`ctrl + shift + p`，搜索`Select Default Profile`，选择`Command Prompt`或者`PowerShell`。

可能遇到的问题

1. 输入报错

```cpp
#include <iostream>
int main() {
    int a;
    std::cin >> a;
    std::cout << "hello world\n";
    return 0;
}
```

上面的代码在debug console中输入数字后会**报错** *Unable to perform this action because the process is running*。如果要输入，**请直接在teminal中输入**

2. vscode run出错

可能是默认tasks和launch的配置问题，直接修改`.vscode`目录下的`tasks.json`和`launch.json`

tasks.json

```
{
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: g++.exe build active file",
            "command": "D:\\Programs\\x86_64-8.1.0-release-posix-seh-rt_v6-
rev0\\mingw64\\bin\\g++.exe", // 修改成自己的mingw目录
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "Task generated by Debugger."
        }
    ],
    "version": "2.0.0"
}
```

launch.json

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(gdb) Launch",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/${fileBasenameNoExtension}.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
```

```
            "MIMode": "gdb",
            "miDebuggerPath": "D:\\Programs\\x86_64-8.1.0-release-posix-seh-
rt_v6-rev0\\mingw64\\bin\\gdb.exe", // 修改成自己的mingw目录
            "preLaunchTask": "C/C++: g++.exe build active file",
            "setupCommands": [
                {
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                },
                {
                    "description":  "Set Disassembly Flavor to Intel",
                    "text": "-gdb-set disassembly-flavor intel",
                    "ignoreFailures": true
                }
            ]
        }
    ]
}
```

调试多个cpp文件

修改 *tasks.json* 的参数，将 `${file}` 改为 `${fileDirname}\\*.cpp` ，在运行任务时使用文件夹 `${workspaceFolder}` 下的所有 `.cpp` 文件作为 `CPP` 源文件

task.json

```
{
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: g++.exe build active file",
            "command": "D:\\Programs\\x86_64-8.1.0-release-posix-seh-rt_v6-
rev0\\mingw64\\bin\\g++.exe", // 修改成自己的mingw目录
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${fileDirname}\\*.cpp",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "Task generated by Debugger."
```

```
        }
    ],
    "version": "2.0.0"
}
```