

vscode

win系统



[1] C++编译器⁺

[2] vscode⁺

[3] vscode插件⁺



vscode是Microsoft开发的一款编辑器




- ✎ 轻量级软件，安装快速、界面简洁。
- ✎ 操作方便、功能强大。
- ✎ 拓展性强，支持**插件**，拥有庞大的插件库。

可以将vscode打造成强大的C++IDE

- 🔗 与terminal集成，通过交互实现程序的**编译**、**运行**。
- 🔗 通过配置文件配置程序的编译、运行流程。
- 🔗 支持**调试**功能。



部署vscode的主要步骤

-  安装、配置C++编译器。
-  安装、配置vscode。
-  安装、配置vscode插件。



[1]

C++编译器

+

 Windows中主要有两款软件提供了C++编译器

 Cygwin和MinGW。

 MinGW更加轻便，是编辑器通常的配套选择。

下载MinGW

 现在大多是**64位**系统，所以要下载MinGW64。

 可以到SourceForge上进行下载。

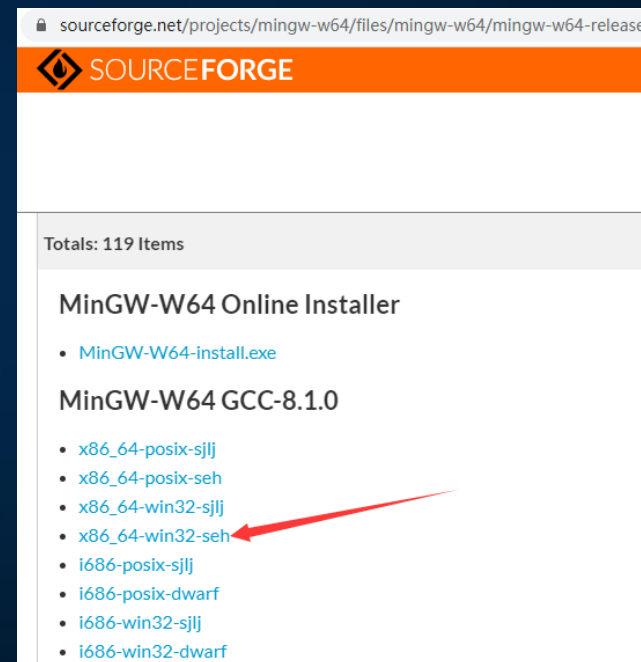
 1. 打开SourceForge上的下载页面。
<https://sourceforge.net/projects/mingw-w64/files/mingw-w64/mingw-w64-release/>

SourceForge：一个开源软件仓库

🚧 可以下载**安装器**或者**压缩包**。
安装器容易遇到网络连接问题，所以下载压缩包更加方便。

👉 2. 下滑页面并找到MinGW的**最新**版本，选择x86_64-win32-seh进行下载。

🚧 x86_64表示64位，win为操作系统。



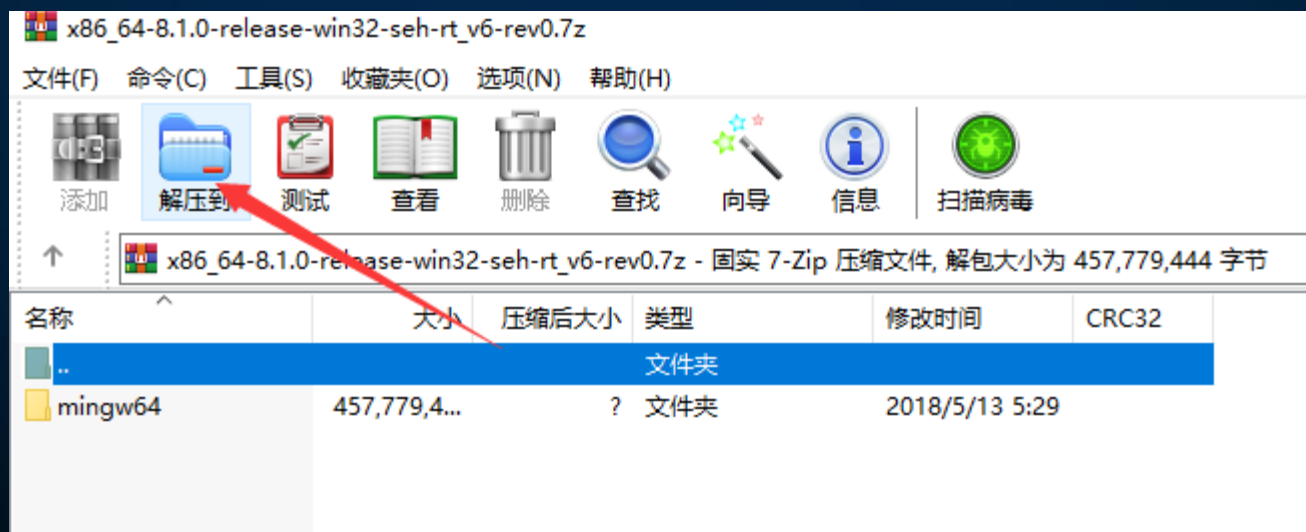
🚧 seh和sjlj是两种不同的**异常处理模型**，通常选用**较新**的seh。

安装MinGW

 将MinGW解压到合适的位置，以用WinRAR进行解压为例。

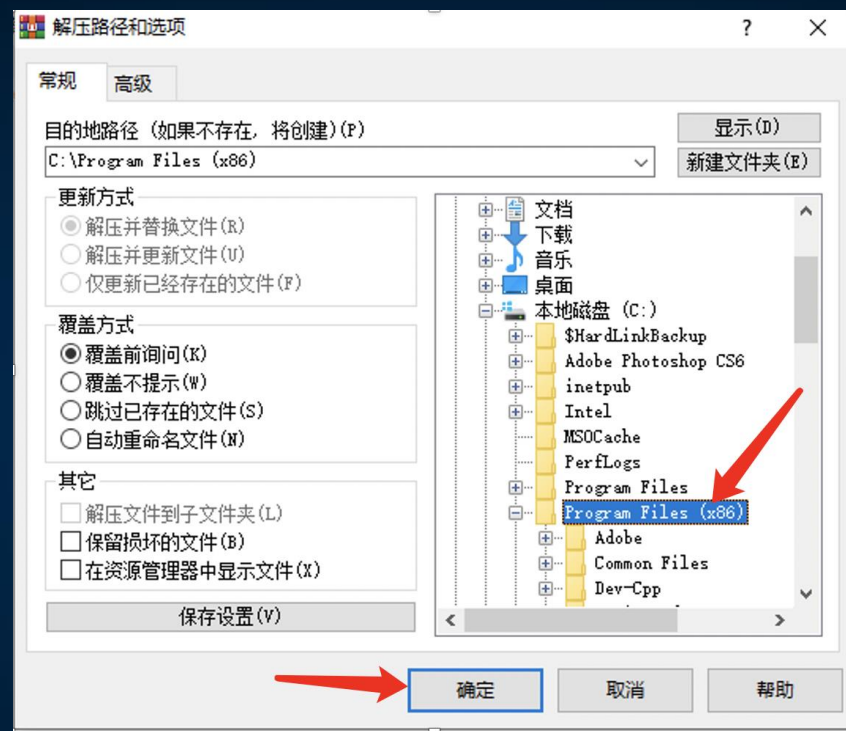
 1. 打开下载的MinGW压缩包。

 2. 点击**解压**相关的按钮，准备进行解压。



WinRAR：一款常用的解压软件

↖ 3. 选择合适的目录，然后点击**确定**进行解压。



🚧 解压到的路径**不允许**包含中文。
通常解压到C盘的Program Files (x86)文件夹中。



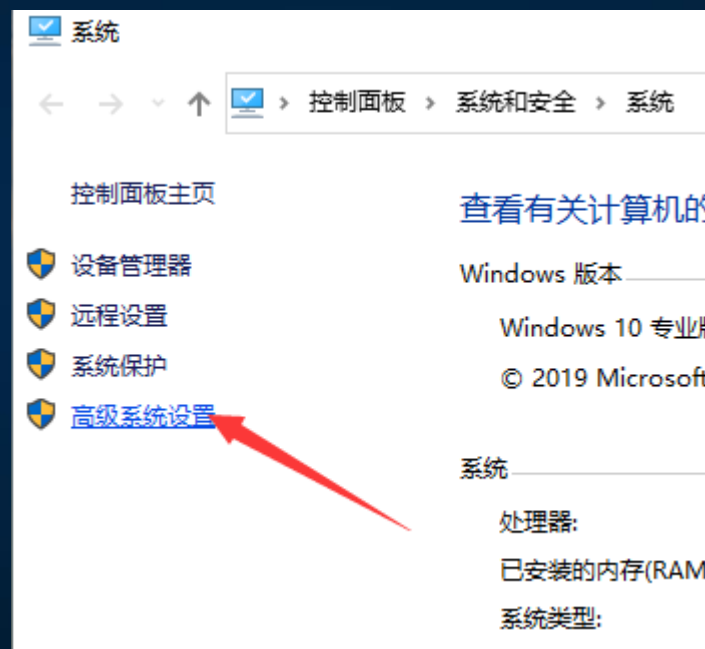
配置环境变量

- 现在在终端中需要使用**完整**路径才能找到编译程序g++。
- 环境变量是程序的**默认**查找目录。
配置好环境变量后，就可以**直接**使用g++命令。
- 不同版本Windows的系统界面可能会略有不同，以Win10为例。

- ↖ 1. 右键点击资源管理器中的我的电脑，或者桌面上的我的电脑。
- ↖ 2. 点击属性选项。



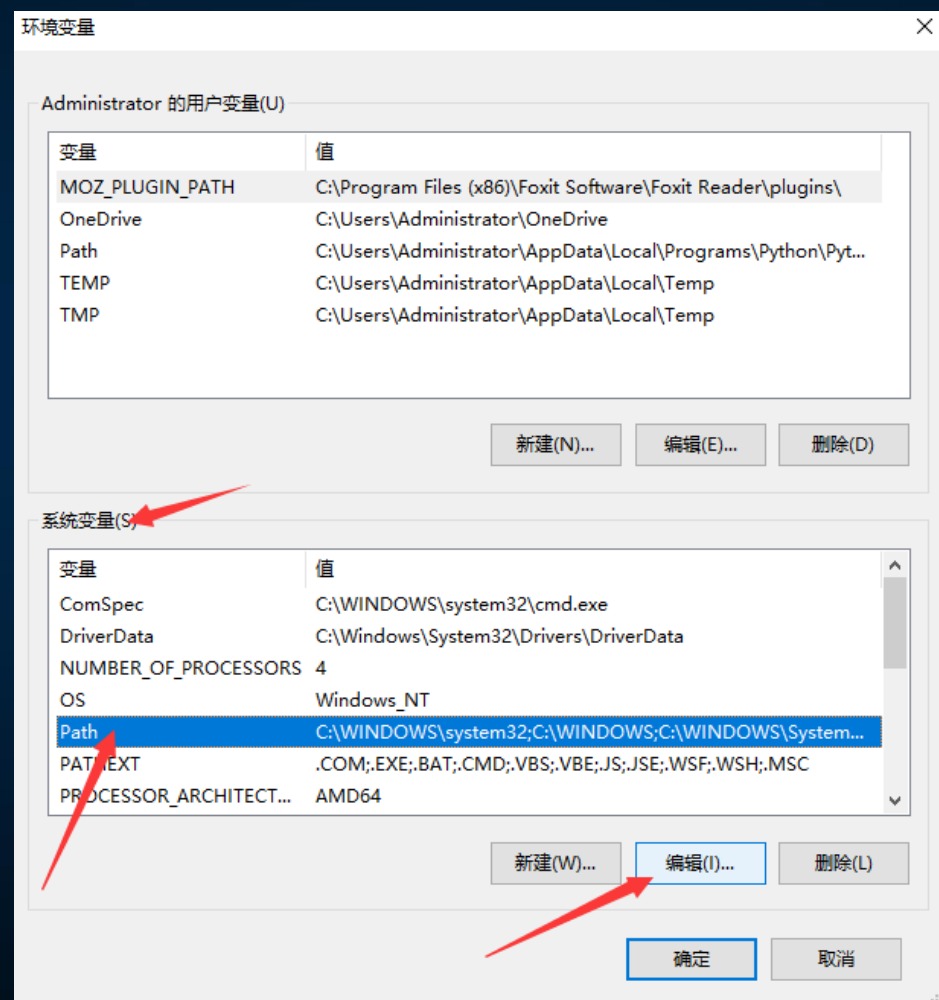
↖ 3. 点击左侧边栏中的高级系统设置。



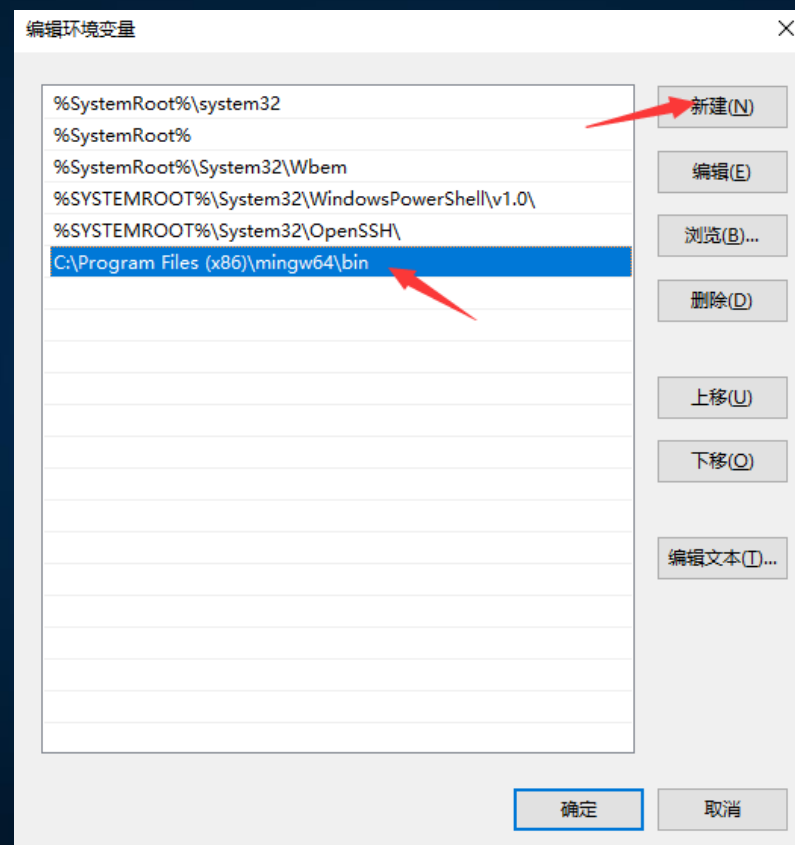
↖ 4. 点击靠下部分中的环境变量按钮。



↖ 5. 选择系统变量中的Path变量。
然后点击右下方的编辑按钮。

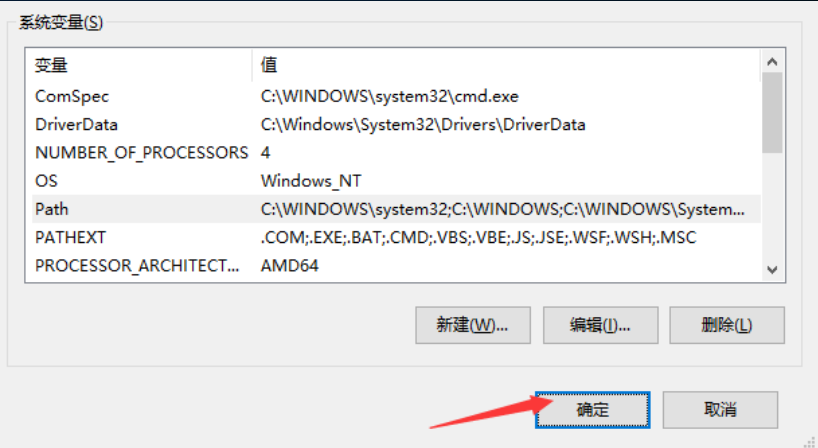
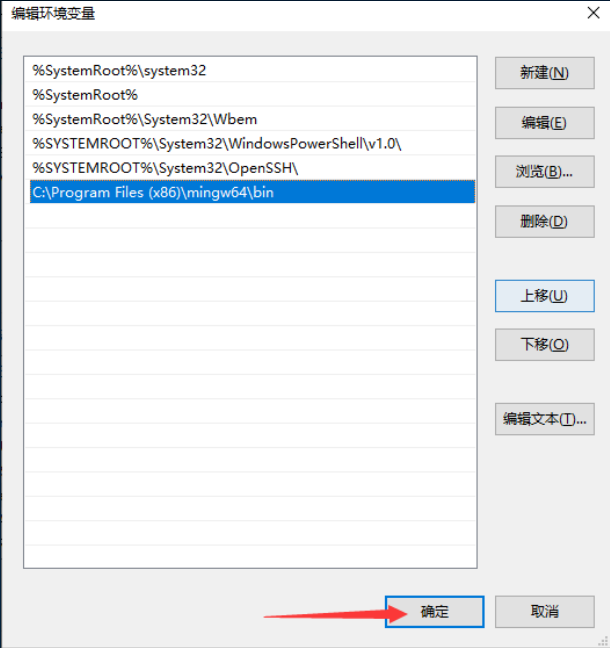


- ↖ 6. 点击新建按钮，新建一个条目。
- ↖ 7. 将MinGW中bin目录的**完整**路径填写进新条目中。





⚠ 注意：旧版本Windows中所有的条目是写在一起的，之间用;分隔。

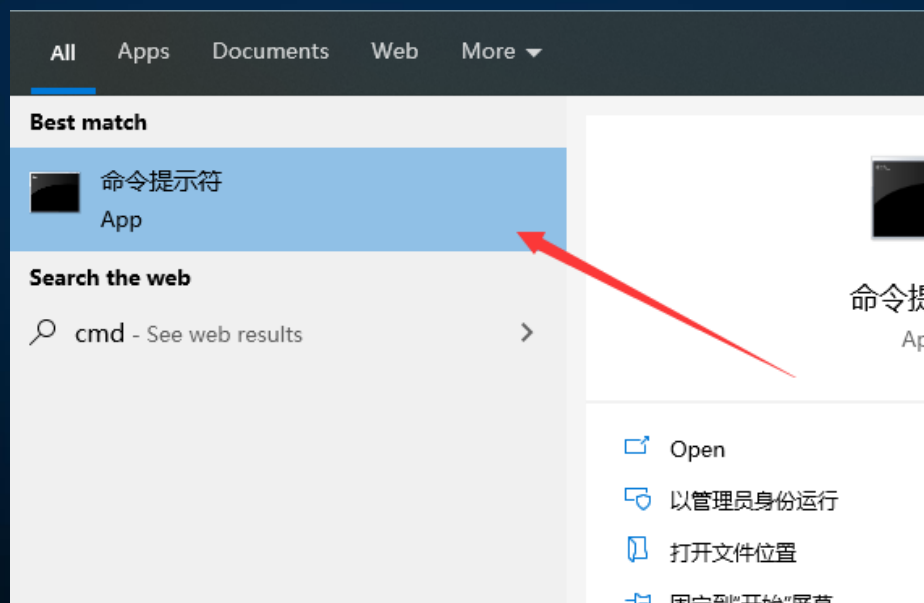
↗ 8. 依次点击所有打开页面的确定按钮，保存设置。



验证环境变量

 现在在终端中应该可以直接使用g++命令了。

-  1. 按下键盘的win键，或者点击屏幕左下方的win按钮，打开开始菜单栏。
-  2. 输入CMD，找到命令提示符后，点击启动终端。



3. 在命令行中输入g++ -v。 (注意g++后有空格)

```
命令提示符
Microsoft Windows [版本 10.0.18363.1082]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=C:/Program Files (x86)/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-w64-mingw32 --prefix=/mingw64 --with-sysroot=/c/mingw810/x86_64-810-win32-seh-rt_v6-rev0/mingw64 --enable-shared --enable-static --disable-multilib --enable-languages=c,c++,fortran,lto --enable-libstdcxx-time=yes --enable-threads=win32 --enable-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-rpath --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-gnu-as --with-gnu-ld --with-arch=nocona --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-static --with-isl=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-pkgversion='x86_64-win32-seh-rev0, Built by MinGW-W64 project' --with-bugurl=https://sourceforge.net/projects/mingw-w64 CFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x86_64-810-win32-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-win32-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-win32-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' LDFLAGS='-pipe -fno-ident -L/c/mingw810/x86_64-810-win32-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: win32
gcc version 8.1.0 (x86_64-win32-seh-rev0, Built by MinGW-W64 project)

C:\Users\Administrator>
```

4. 如果成功输出g++版本信息，则说明环境变量配置成功了。



[2]

vscode

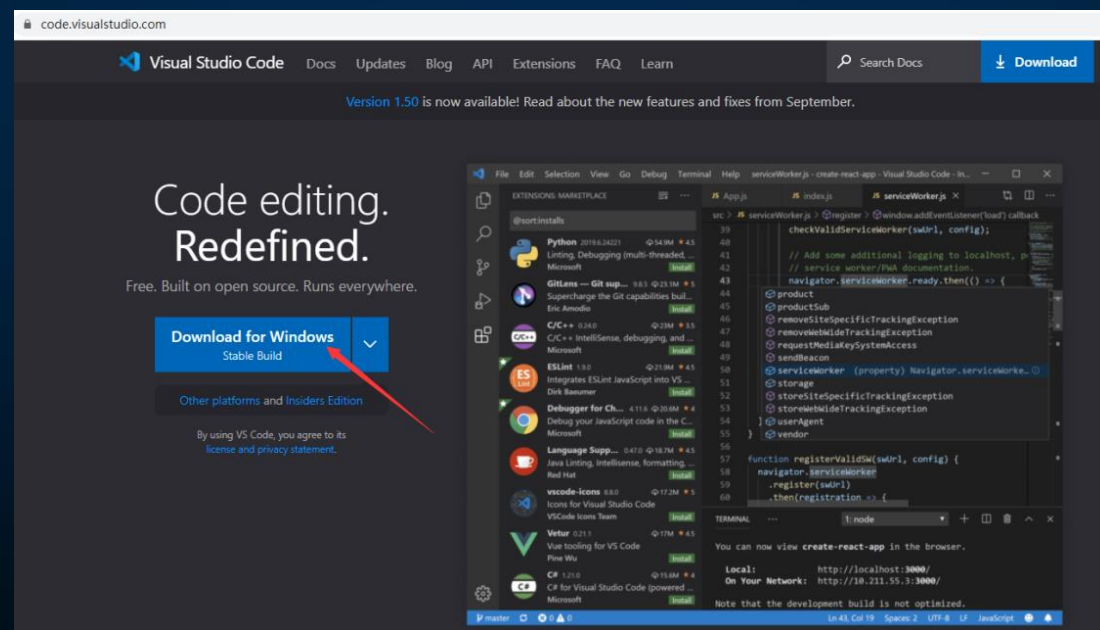
+

下载vscode

 从官网下载**安装包**。

 1. 打开官网页面: <https://code.visualstudio.com/>。

 2. 点击Download按钮。



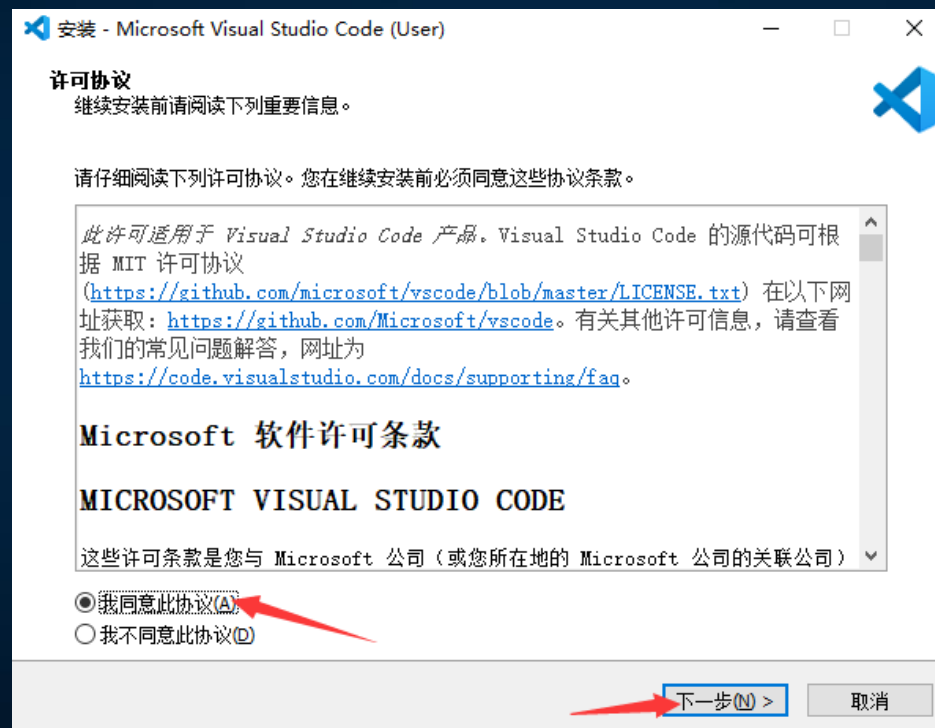
 点击后页面会跳转，然后**自动**开始下载。
如果没有自动下载，可以点击链接**手动**下载。

Thanks for downloading VS Code for Windows!

Download not starting? Try this [direct download link](#).
Please take a few seconds and help us improve ... [click to take survey](#).

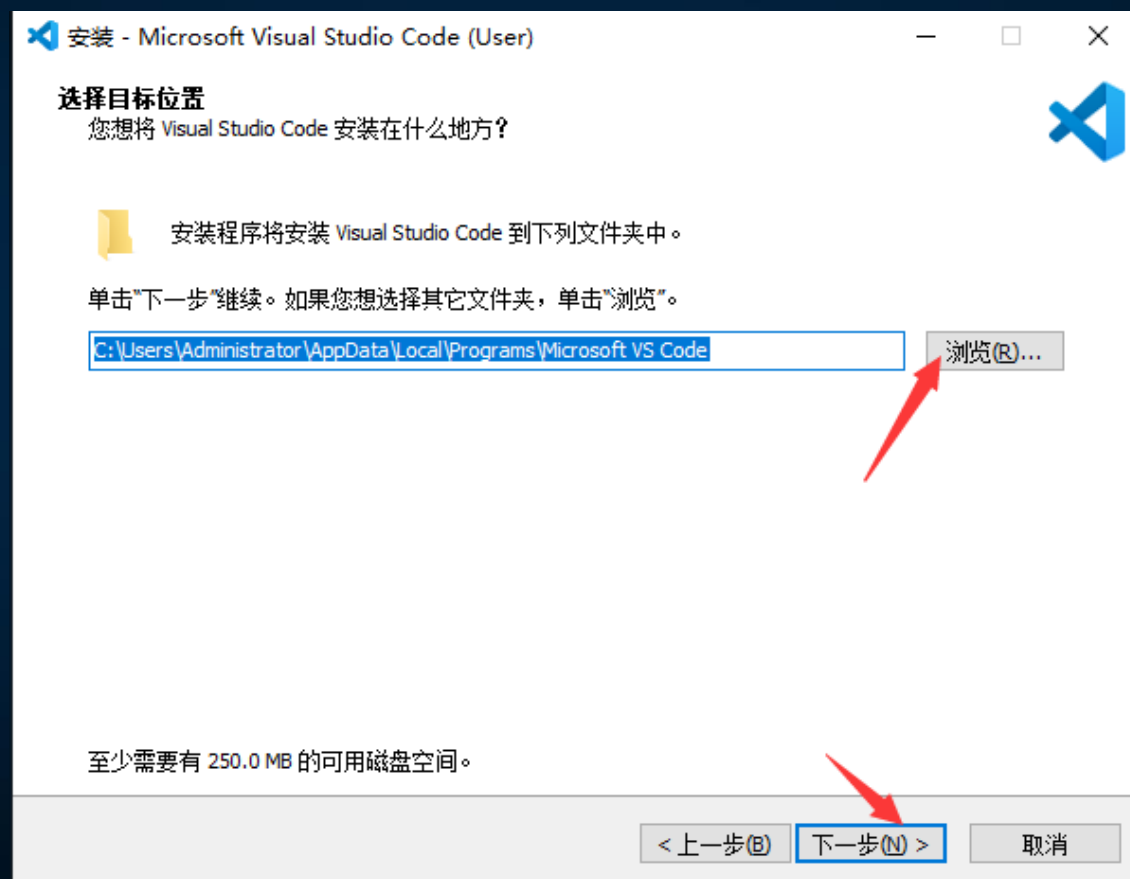
安装vscode

↗ 1. 运行vscode安装包，进入安装流程。

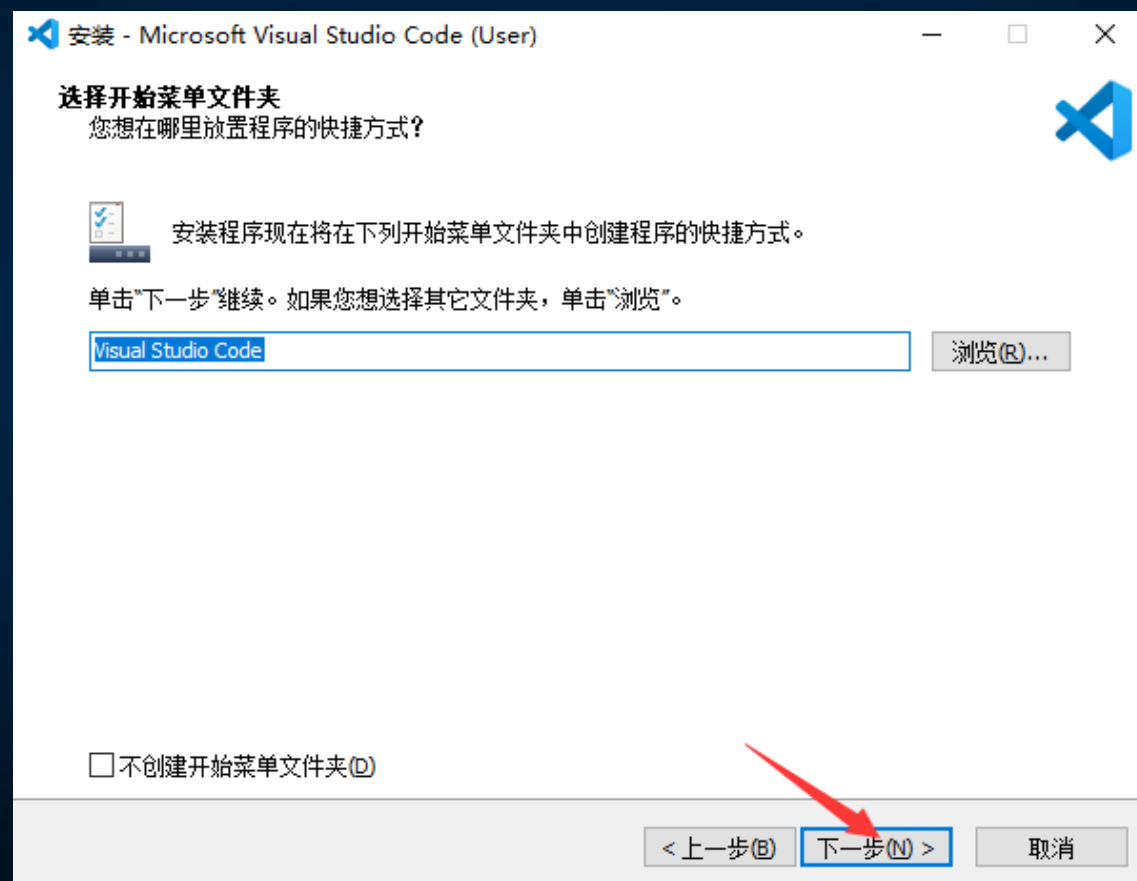


↗ 2. 点击同意协议。然后点击下一步。

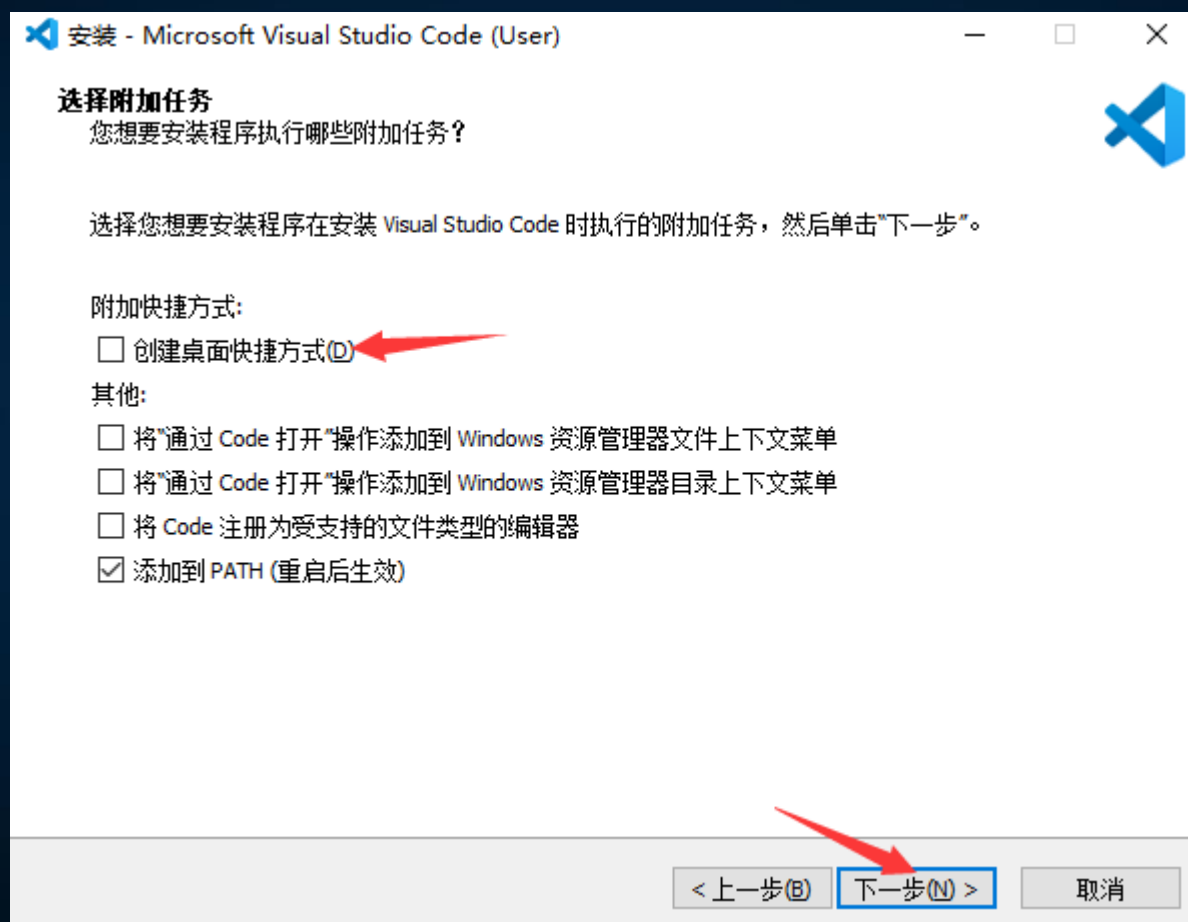
↖ 3. 选择要安装到的位置，可以使用默认值。然后点击下一步。



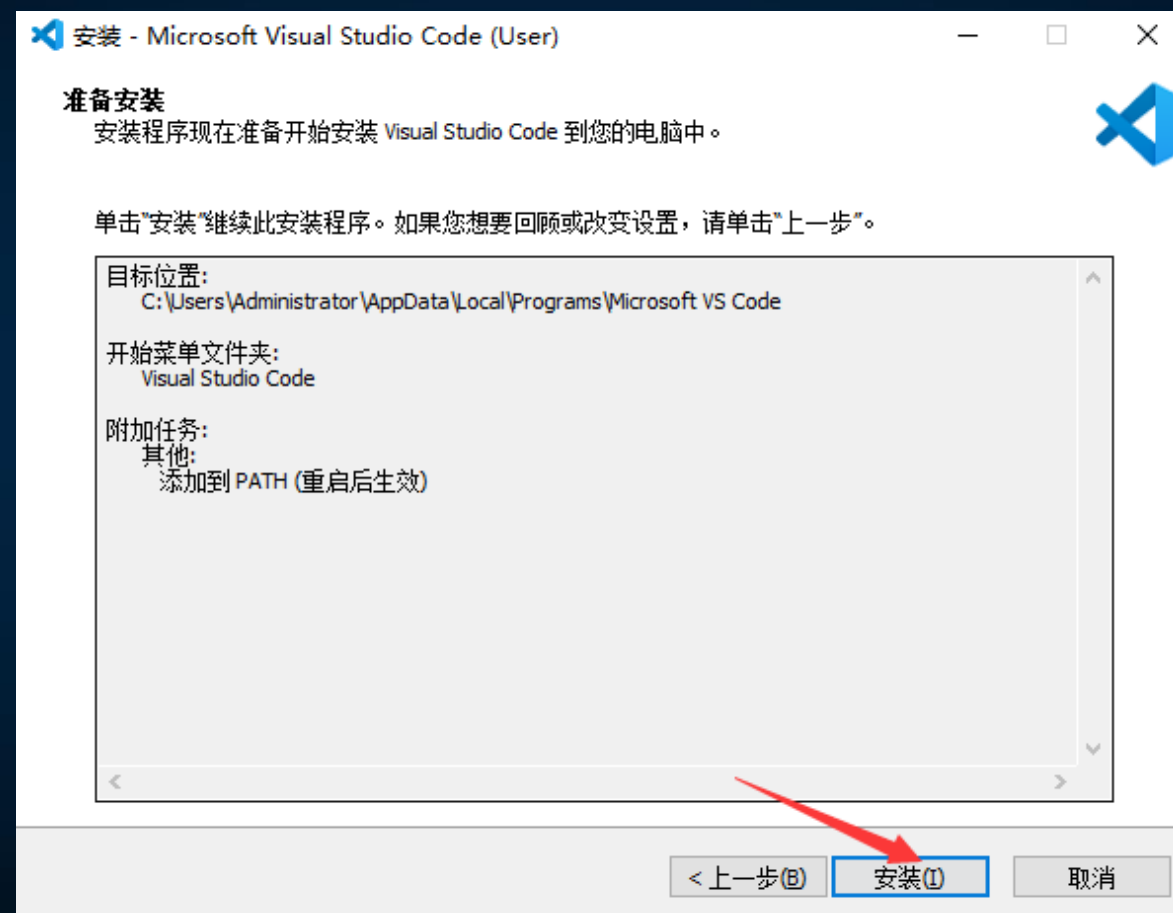
↩ 4. 选择快捷方式的放置位置，使用默认值即可。然后点击下一步。



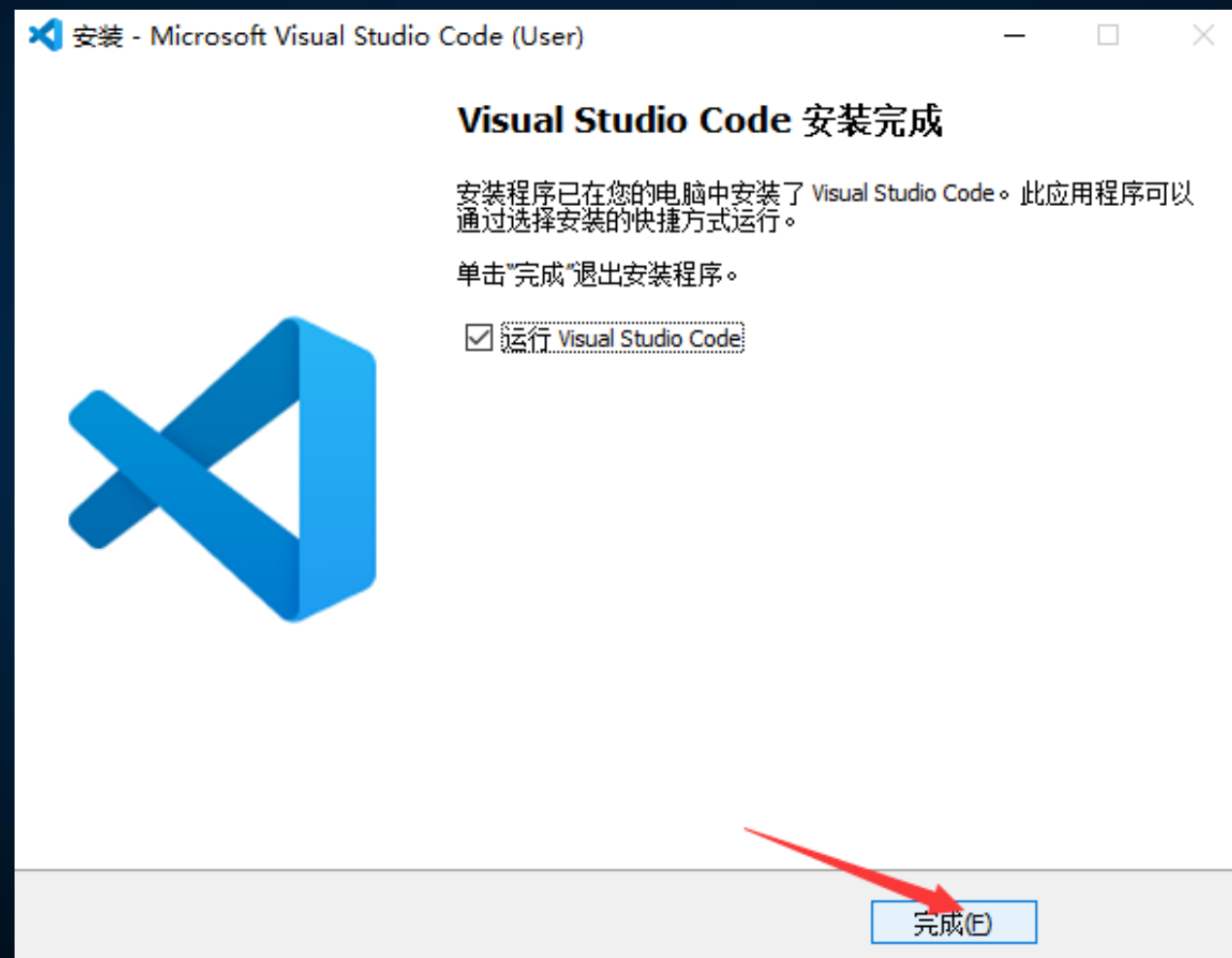
↖ 5. 选择附加任务，根据需要决定是否添加桌面快捷方式。然后点击下一步。




↖ 6. 点击安装进行安装。

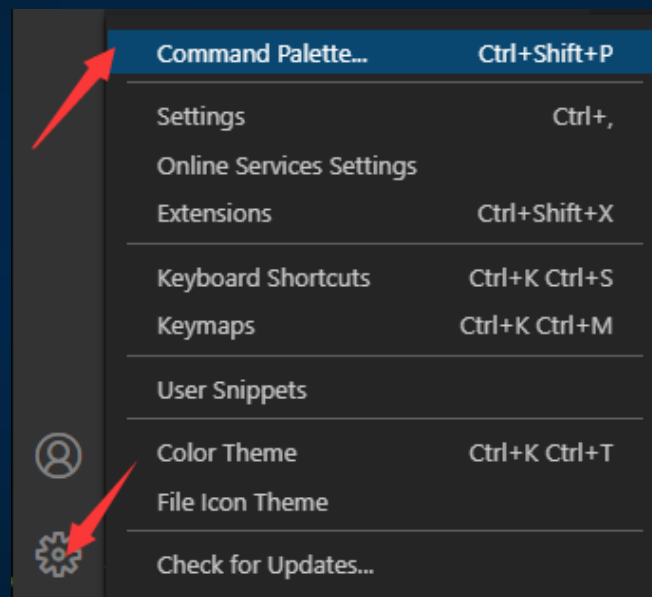


↖ 7. 点击完成。
现在可以打开vscode并进行使用了。



通过Command Pallete可以搜索功能选项



 点击界面左下角的设置按钮，就能在展开的菜单栏中找到。



 大部分选项上都写了**快捷键**，通过快捷键可以便捷的打开相应的功能。

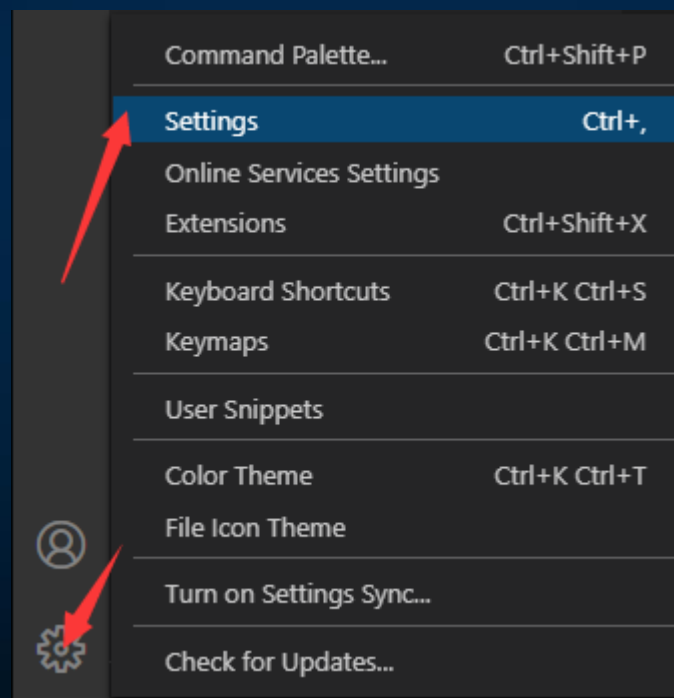
Command Pallete: 命令面板

vscode支持两种**修改设置**的方式

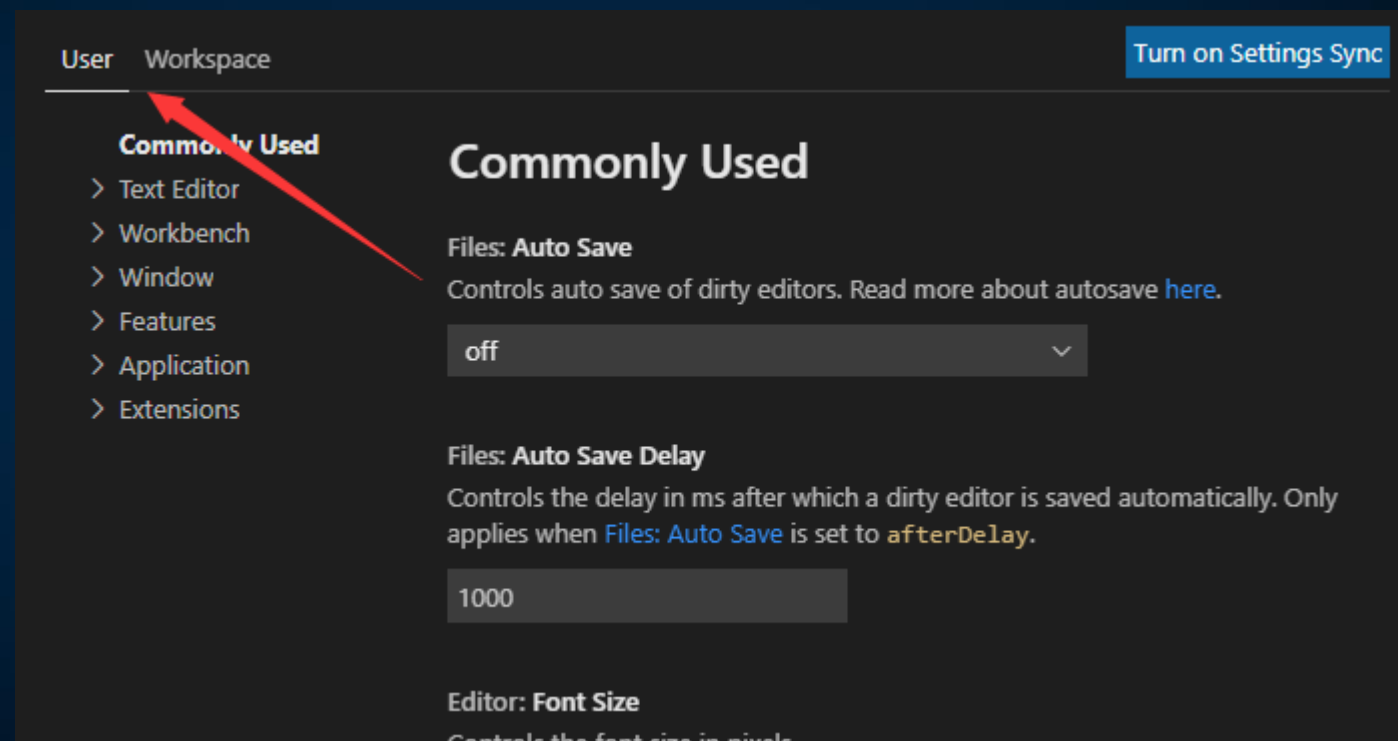
-  **直接**修改**设置文件**settings.json，自由度高，操作较为复杂。
-  通过**界面**设置，自由度低，操作较为友好。
界面设置**本质**上也是修改设置文件。

打开vscode的设置界面

- ↖ 1. 点击vscode界面左下角的设置按钮。
- ↖ 2. 点击settings选项。

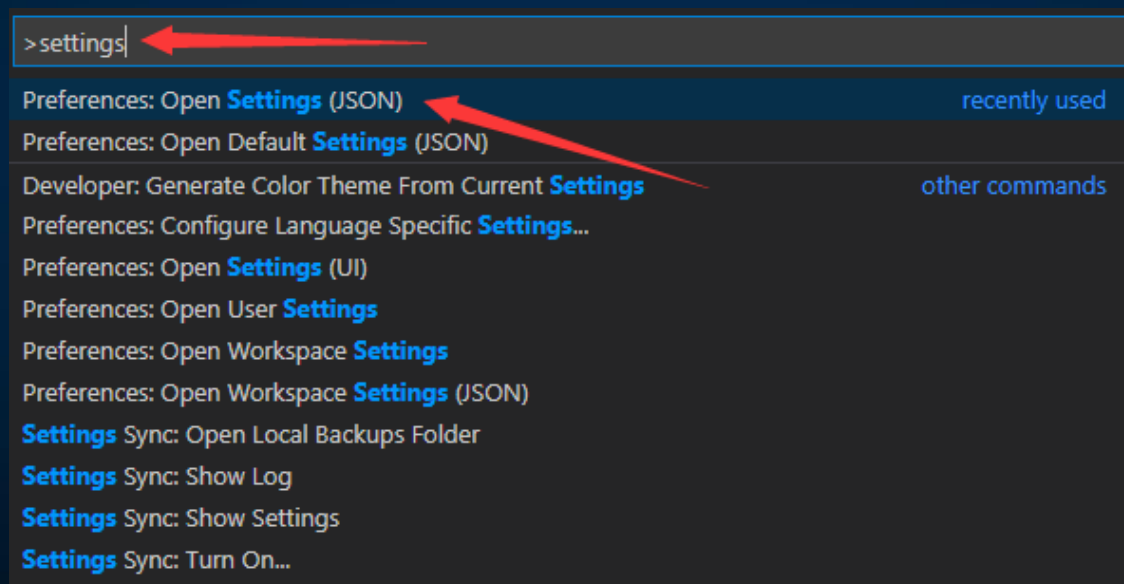


- 🚧 设置分为user和workspace，workspace只有在工作区打开时才显示。
- 🚧 通常只需要进行user设置。



打开vscode的设置文件

1. 打开命令面板。
2. 搜索Preference: Open Settings (JSON)选项并点击。



preference: 首选项、偏好

 设置文件内容格式如下，
里面包含了vscode的设置与插件的设置。

```
{
  "launch": {
    "configurations": [
      {
        "name": "C++ debug (global)",
        "type": "cppdbg",
        "request": "launch",
        "program": "${fileDirname}\\${fileBasenameNoExtension}.exe",
        "cwd": "${fileDirname}",
        "preLaunchTask": "C++ compile"
      }
    ],
  },
  "terminal.integrated.shell.windows": "C:\\Windows\\System32\\cmd.exe",
  "editor.tabSize": 2,
  "C_Cpp.clang_format_style": "{ BasedOnStyle: Google, ColumnLimit: 0}",
  "code-runner.runInTerminal": true,
  "code-runner.executorMap": {
    "cpp": "cd $dir && g++ -std=c++14 \"$fileName\" -o \"$fileNameWithoutExt\\.exe\" && $dir\\$fileNameWithoutExt\\.exe",
  }
}
```

 设置**仅供参考**，最终需要根据实际情况进行配置。

vscode通用配置

 通过设置文件进行设置。

- ↖ 1. 设置终端，powershell较新，是**默认**的终端。
如果运行时遇到了问题，可以切换成**传统**的cmd。

```
"terminal.integrated.shell.windows": "C:\\Windows\\System32\\cmd.exe",
```

- ↖ 2. 设置tab字符宽度为2，与**代码规范**保持一致。

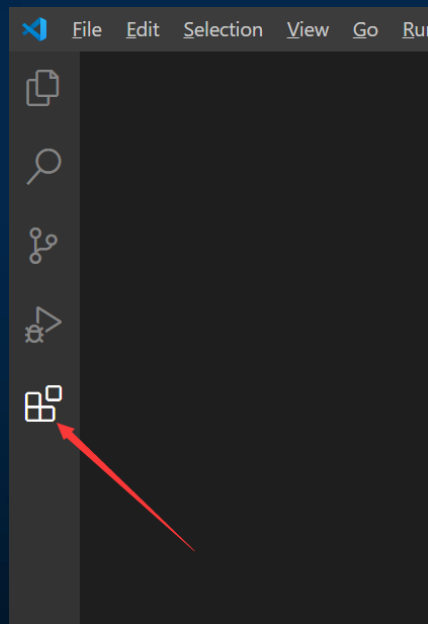
```
"editor.tabSize": 2,
```

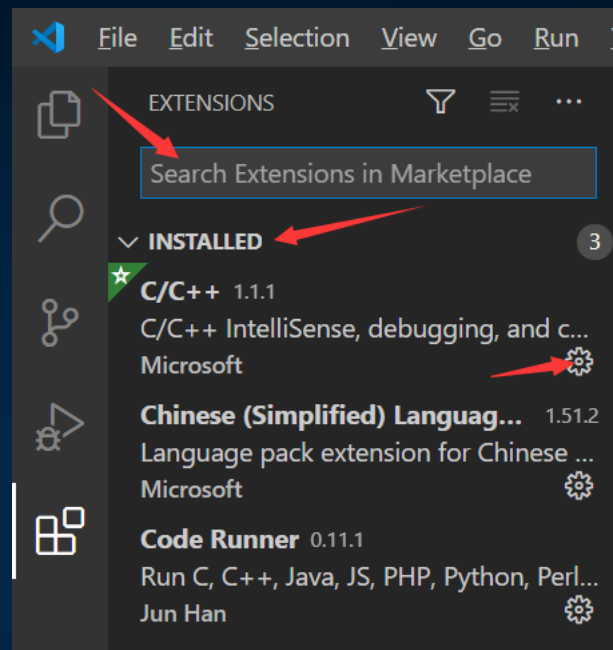
[3]

vscode插件 +

使用插件栏

↖ 1. 点击侧边栏中的插件按钮，打开插件栏。



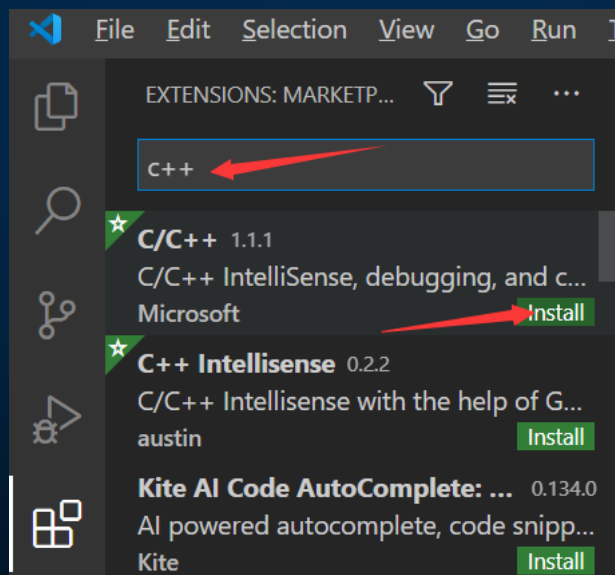


-  **搜索框**中可以对插件进行搜索。
-  已安装的插件会显示在**INSTALLED**栏中。
-  点击插件右下角的**齿轮**图标，可以对插件进行设置。

使用C/C++插件

 可以用于C++程序的识别、查看、调试等。

 1. 在插件搜索栏中搜索C++，找到插件并进行安装。



- ↖ 2. 打开设置文件，加入以下内容。
将代码的**格式化风格**设置为Google。

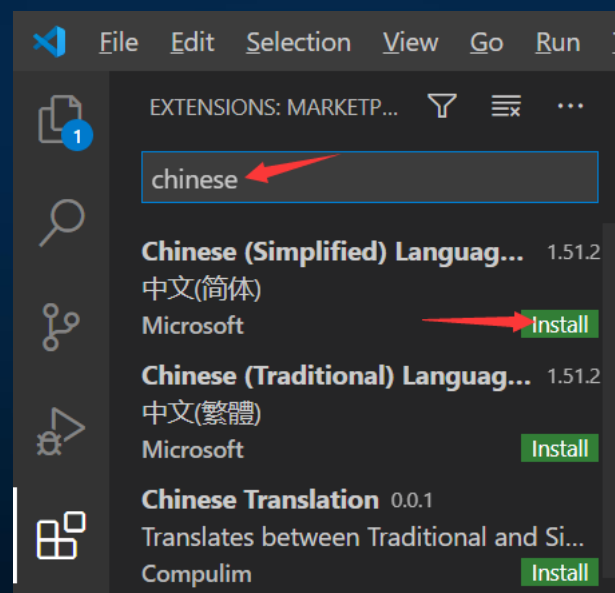
```
"C_Cpp.clang_format_style": "{ BasedOnStyle: Google, ColumnLimit: 0}",
```

- 🚧 在编辑C++程序时按下组合键Alt + Shift + F进行格式化。

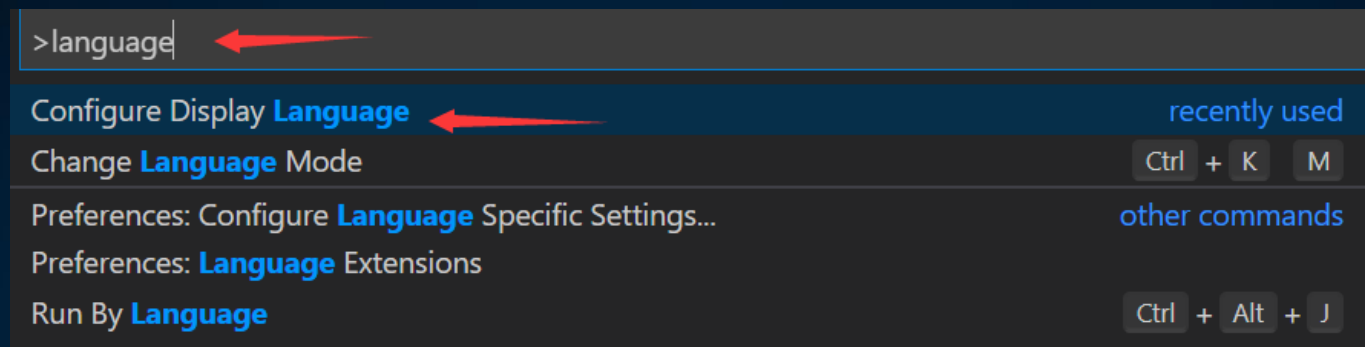
使用Chinese Language插件

 可以将vscode的界面设置成中文。

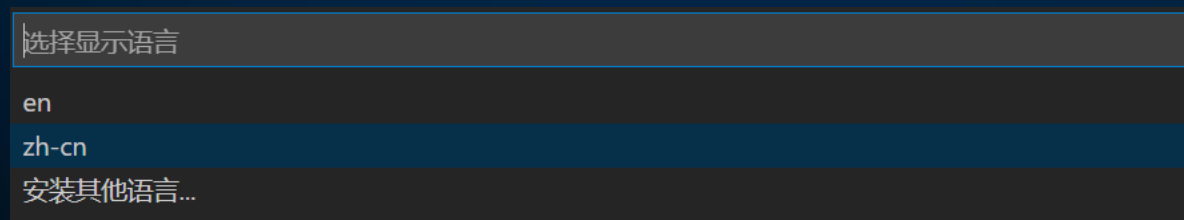
 1. 在插件搜索栏中搜索chinese，找到插件并进行安装。



↖ 2. 打开命令面板，搜索Configure Display Language选项并点击。



↖ 3. 点击zh-cn，重启vscode后界面将会被切换成中文。

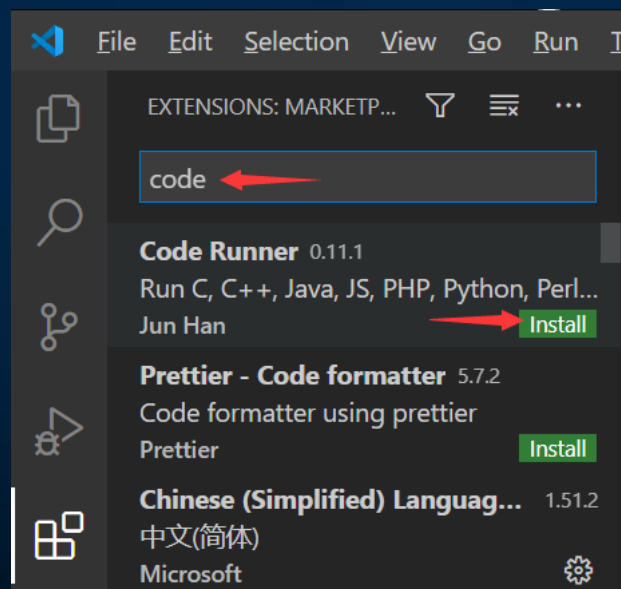


Configure Display Language: 配置显示语言

使用Code Runner插件

 可以在**集成终端**中运行代码并进行交互。

 1. 在插件搜索栏中搜索code，找到插件并进行安装。



↖ 2. 打开设置文件，加入以下内容。

```
"code-runner.runInTerminal": true,  
"code-runner.executorMap": {  
  "cpp": "cd $dir && g++ -std=c++14 \"$fileName\" -o \"$fileNameWithoutExt\".exe && $dir \"$fileNameWithoutExt\".exe",  
}
```

📌 runInTerminal: 在集成终端中运行，**不**打开新的终端。

📌 executorMap: 设置了C++程序的编译、运行命令。

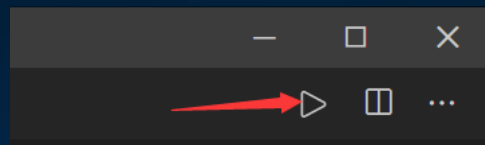
📌 \$之后的内容为**占位符**，会根据**文件信息**进行**替换**。

\$dir: 文件所属目录。

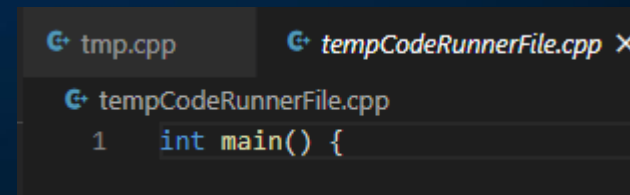
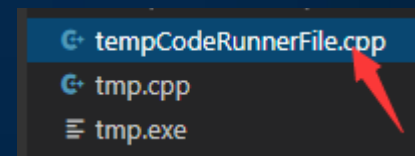
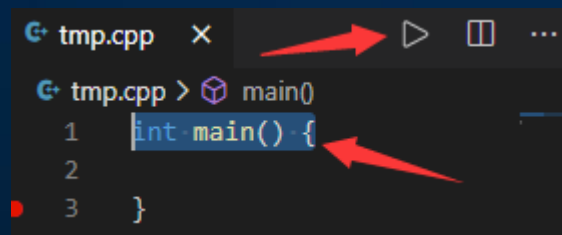
\$fileNameWithoutExt: 不带扩展名的文件名。

🔗 注意：这里的编译运行和**调试**中的编译运行是两套**不同**的流程。

🚧 打开C++程序，右上角出现**运行**按钮。
点击后会显示**集成终端**并运行程序。



🚧 如果在**框选**了一段程序的情况下按运行按钮，
会根据框选内容生成一份**新**的文件，并运行新文件。

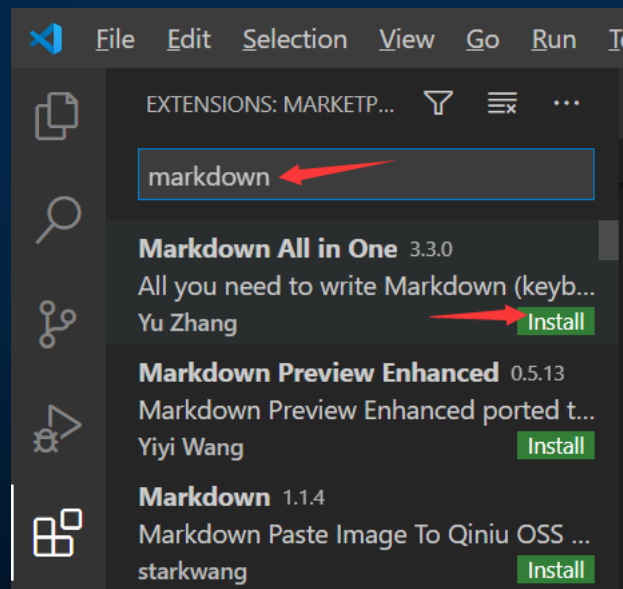



🔗 所以运行的时候注意**不要**框选任何程序片段。

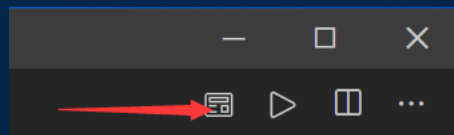
使用Markdown插件

 可以对md格式的文件进行渲染。

 1. 在插件搜索栏中搜索markdown，找到插件并进行安装。



 打开md文件，右上角出现渲染按钮。
点击后会显示渲染后的文档。



谢谢

