C++软件安装及环境配置

在问题求解课程以及后续会学到的 ICS，OS 等课程，编程任务主要为 C 语言，鉴于大家也许是第一次接触到编程语言，助教给大家准备了一些基本的环境软件配置教程，请根据自己电脑的操作系统查看相应的推荐。

1. Windows 系统

Dev c++

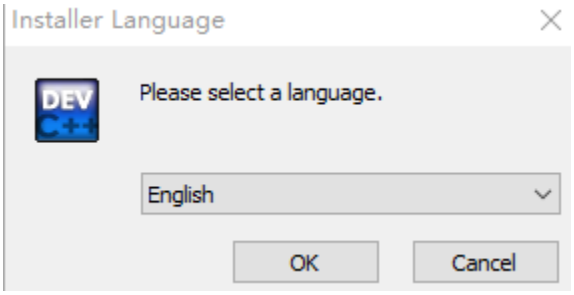
优点：占内存较小，c/c++代码可以直接编译运行，内置 windows 版本的 gcc 编译工具，和 OJ 平台的编译工具同源，因此 Dev c++能编译通过的代码，在 OJ 平台上也能通过，而使用 visual studio 可能会产生一些问题。

缺点：相较于 visual studio，Dev c++的调试功能较为麻烦，没有 visual studio 的便捷性，同时对于多文件项目的编译调试，Dev C++的表现会逊色于 visual studio

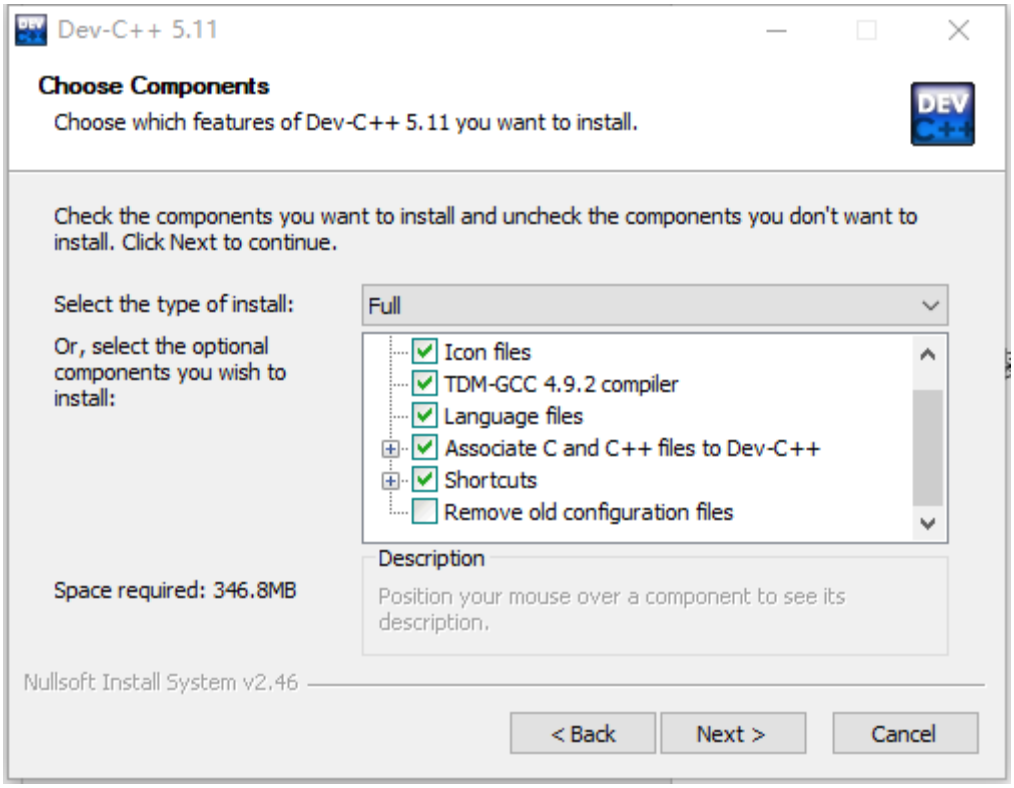
Dev c++安装使用教程：

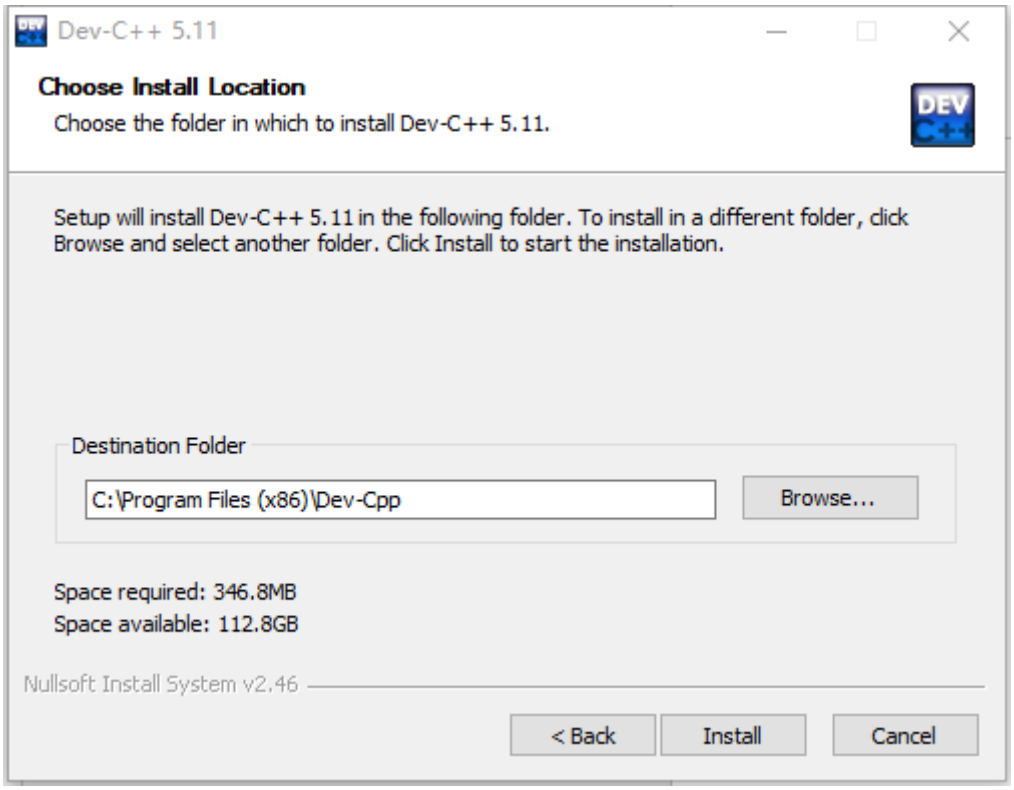
https://pc.qq.com/detail/16/detail_163136.html 提供软件下载，不要傻乎乎地选择“高速下载”!

下载完成后，开始安装，首先会弹出一个语言选择界面，选择 English，点击 OK

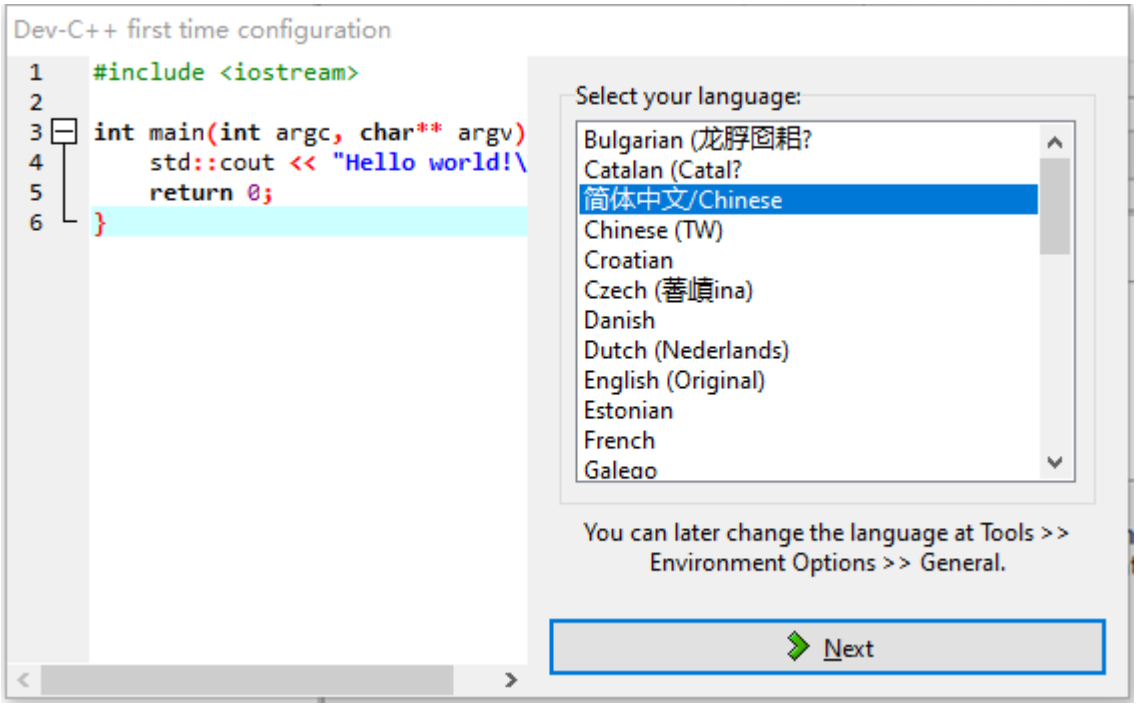


之后选择安装配置以及安装目录，随个人需要调整，默认配置即可满足需求，安装目录建议放在非 C 盘区域。



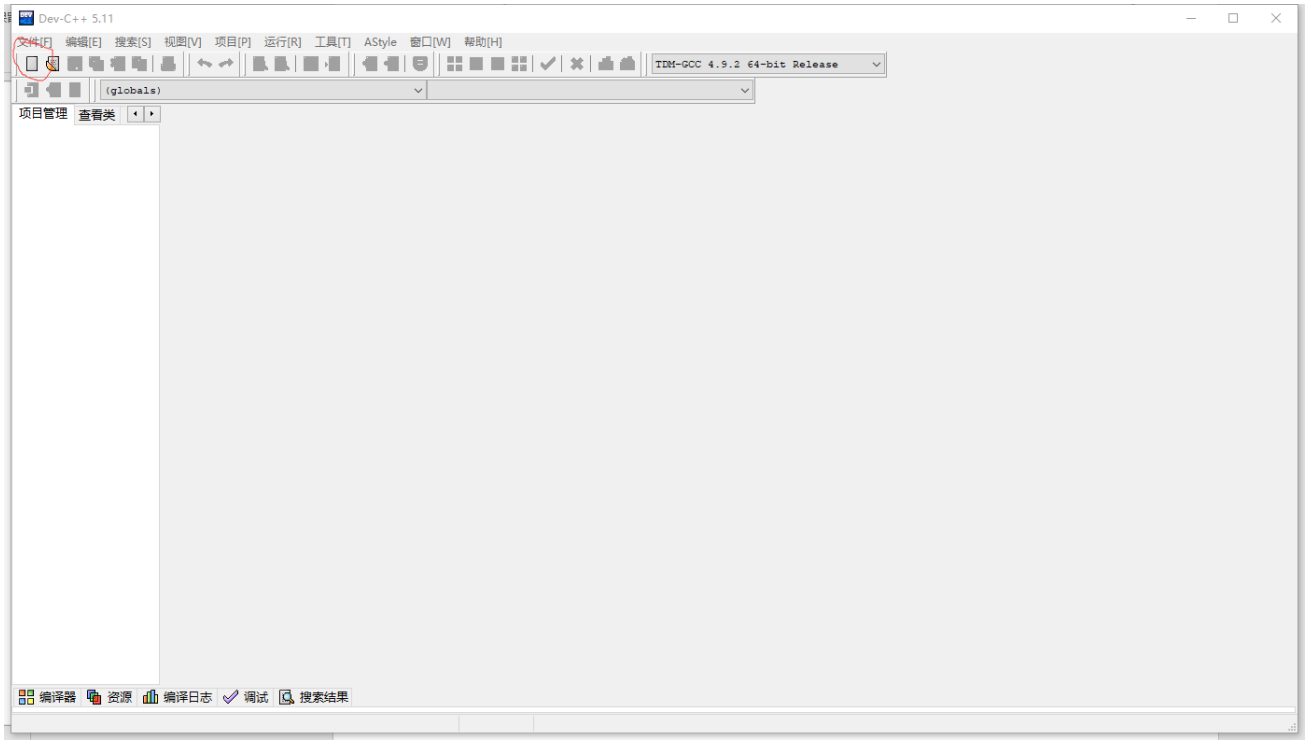


安装完成后运行，语言选择界面，根据需求选择简体中文或英文，接下来根据个人需求进行外观配置，配置完成后，即可使用 Dev C++



使用 Dev C++ 编写第一个 C++程序：

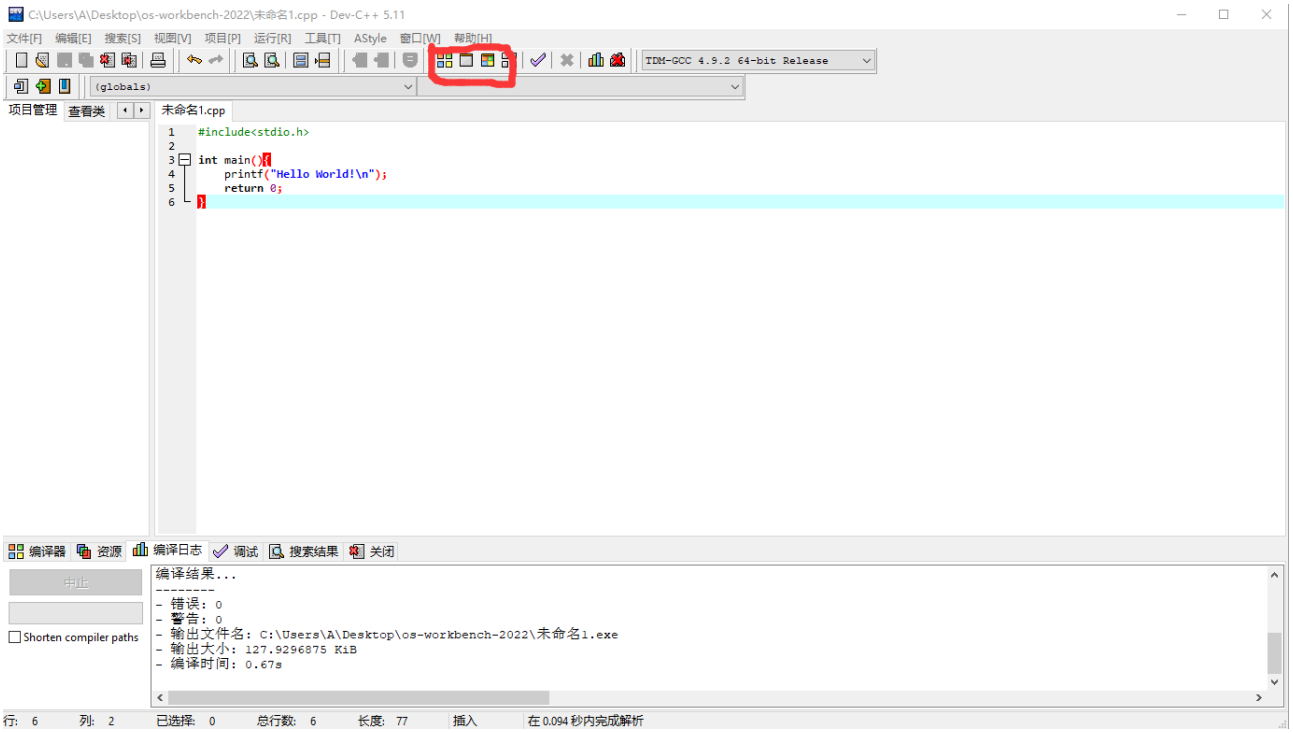
单击左上角红圈圈住的内容，新建文件



输入以下代码

```
1.  #include<stdio.h>
2.
3.  int main(){
4.      printf("Hello World!\n);
5.      return 0;
```

```
6. }
```



在红框处选择编译运行，将文件保存后，会弹出窗口，输出“Hello World！”即完成配置。

本节参考资料:
https://www.bilibili.com/video/BV19u411v7fz

Visual Studio

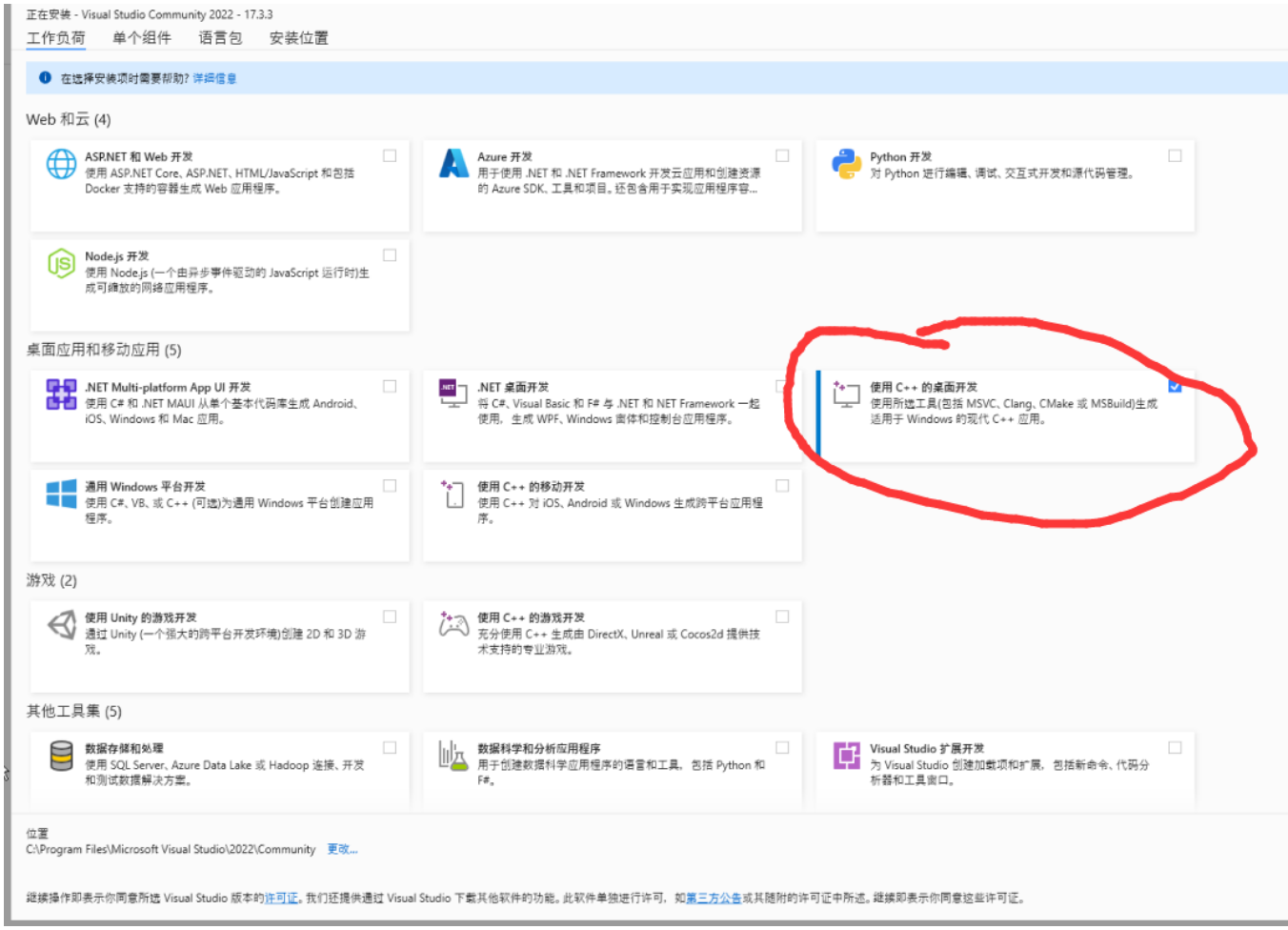
优点：可同时支持其他语言编程，可以创造代码模板，调试方便，支持 git 功能，便于大项目的维护开发，代码自动补全，显示错误和警告。

缺点：本身很占磁盘空间，创建的项目也很占空间，如果想要保存之前的代码文件的话，一个项目可能会有几百 MB，电脑磁盘空间不足的同学慎用。

Visual Studio 安装使用教程

下载地址：https://visualstudio.microsoft.com/zh-hans/vs/

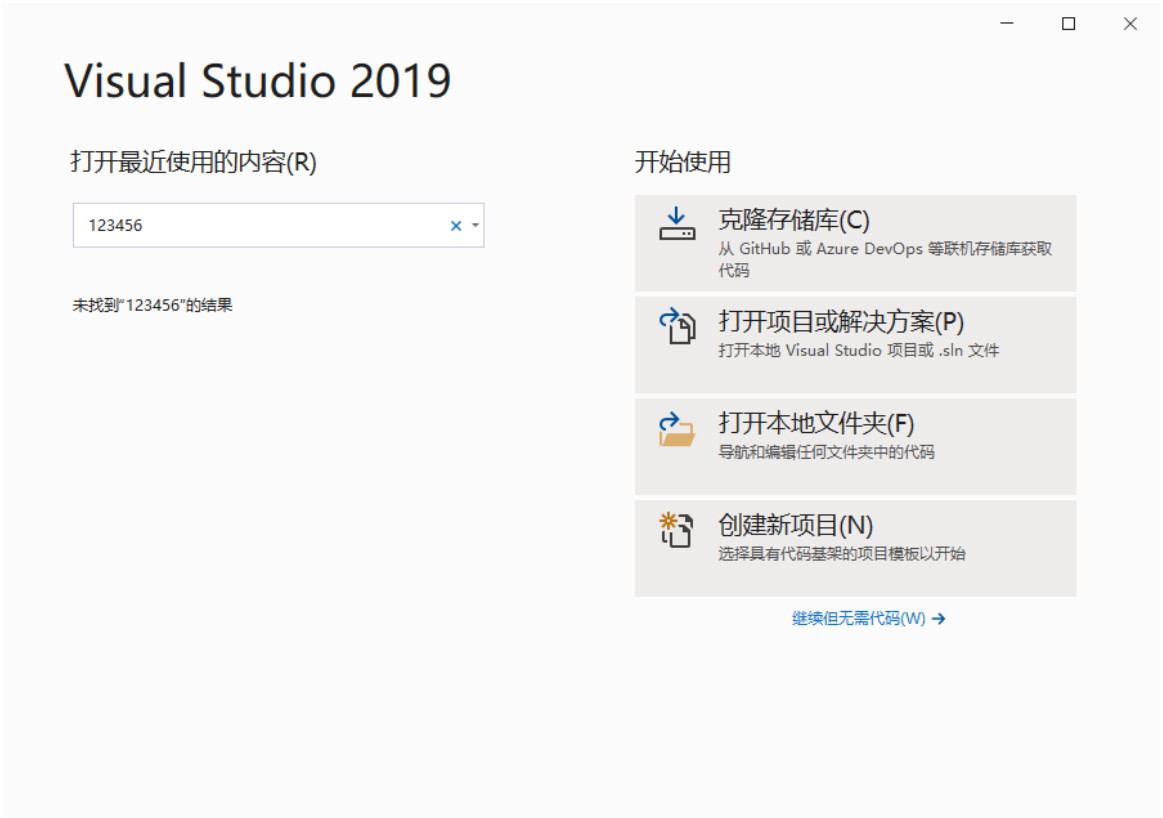
下载完成后，打开软件，等待必需组建下载安装完成后，勾选自己所需内容，这里建议大家只选择“使用 C++的桌面开发



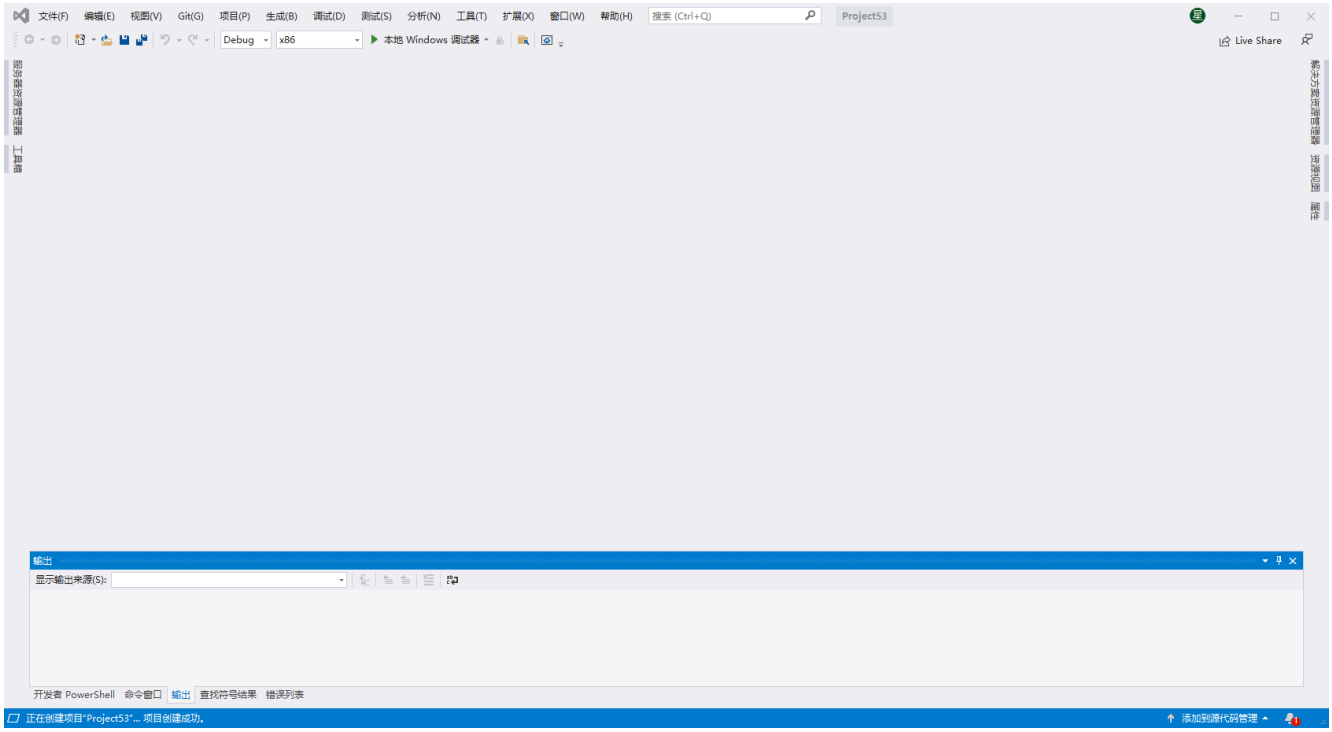
等待下载安装完成，由于文件较大，建议大家不要下载到系统盘。

使用 Visual Studio 编写第一个 C++程序：

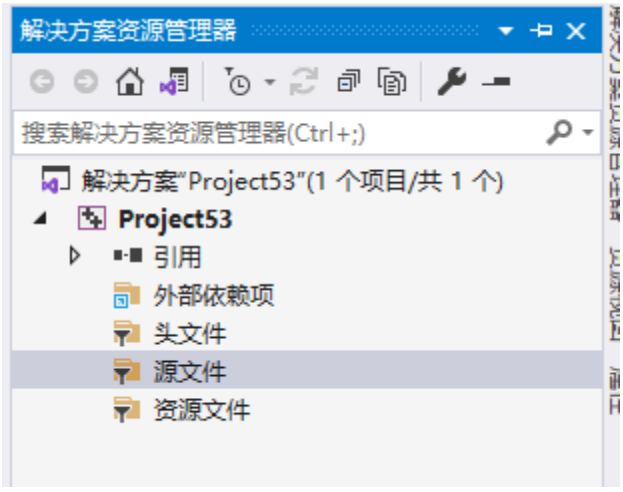
打开 Visual Studio 程序，点击启动，创建新项目，选择空项目



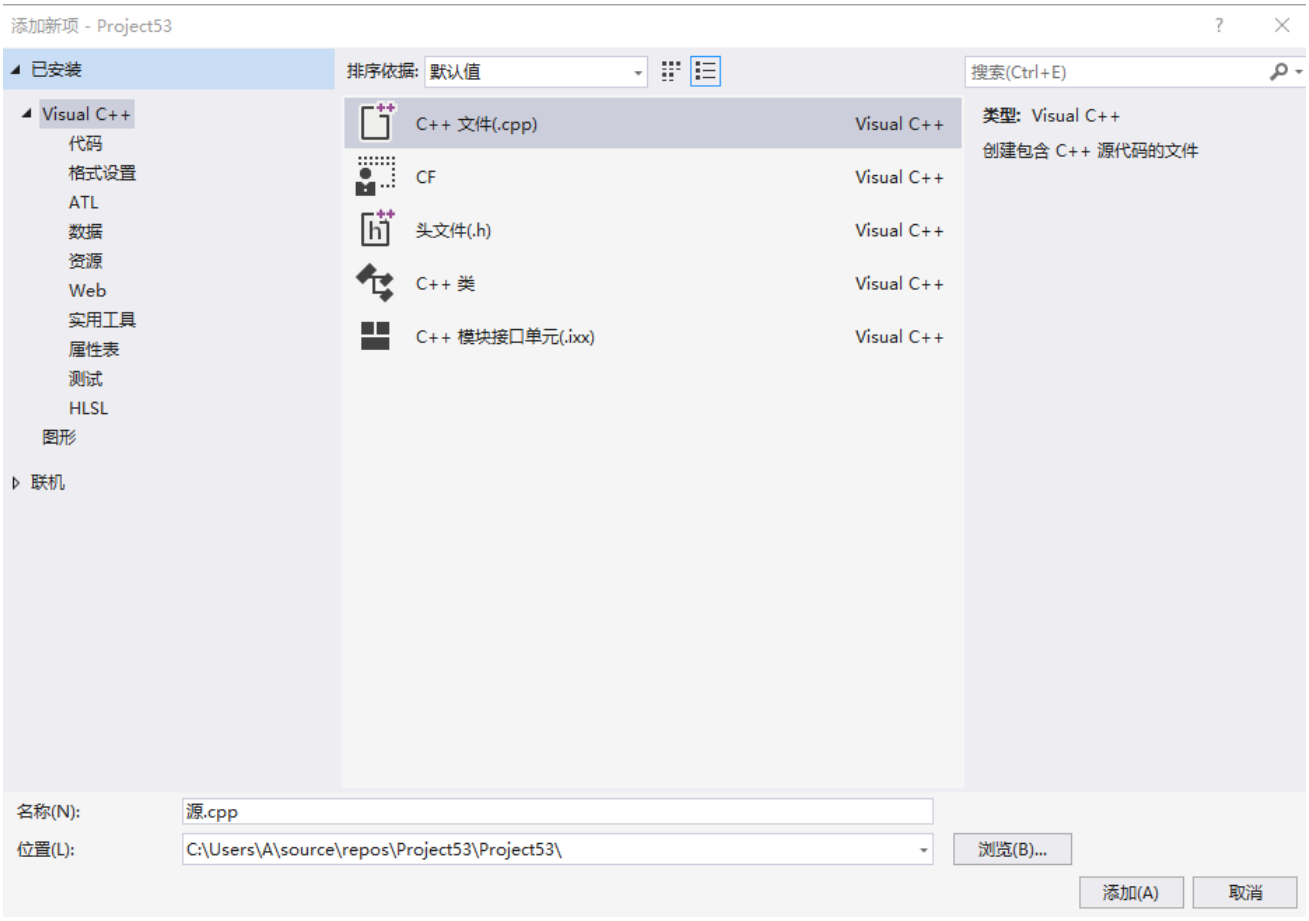
调整代码存放位置，不建议放在系统盘中，点击创建，进入以下界面



点击右边栏的解决方案资源管理器，右击源文件->添加->新建项,选择默认的 C++文件，修改命名，点击添加



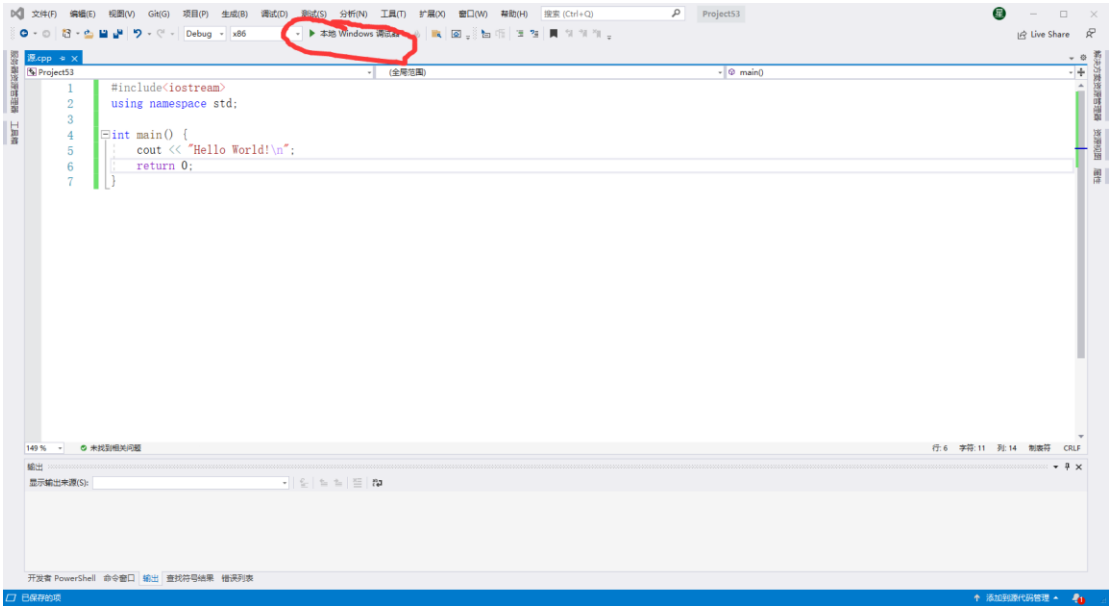
”



输入以下代码，点击本地 Windows 调试器，即可运行，在终端输出 Hello World！即成功。

```
#include<iostream>
using namespace std;

int main() {
    cout << "Hello World!\n";
    return 0;
}
```



MACOS 和 Windows 都能用

VS Code

优点：海量的插件提供丰富的功能，同时软件轻量级，不占内存，同时代码可以直接运行，调试。

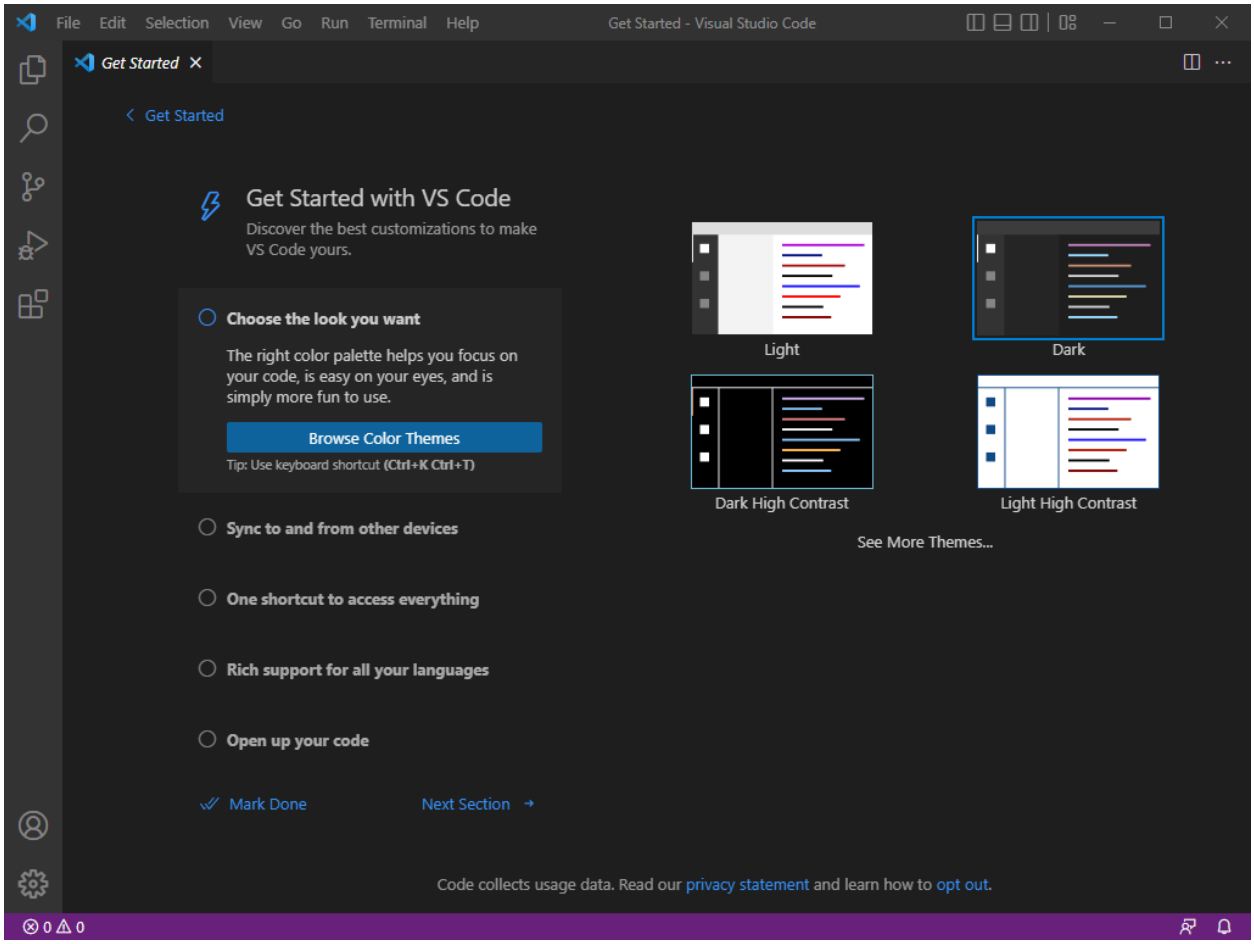
缺点：配置过程很麻烦，需要自己在网络上找一些合适的插件，配置好的 VS code 和没配置的 VS code 使用上会差别很大。

VS Code 安装，基本配置及使用教程

注：使用 Windows10 的安装过程作为例子，MACOS 请参考
https://www.zhihu.com/collection/655664111

下载地址：https://code.visualstudio.com/Download

下载完成后按照步骤执行安装，安装完成后打开会显示以下界面

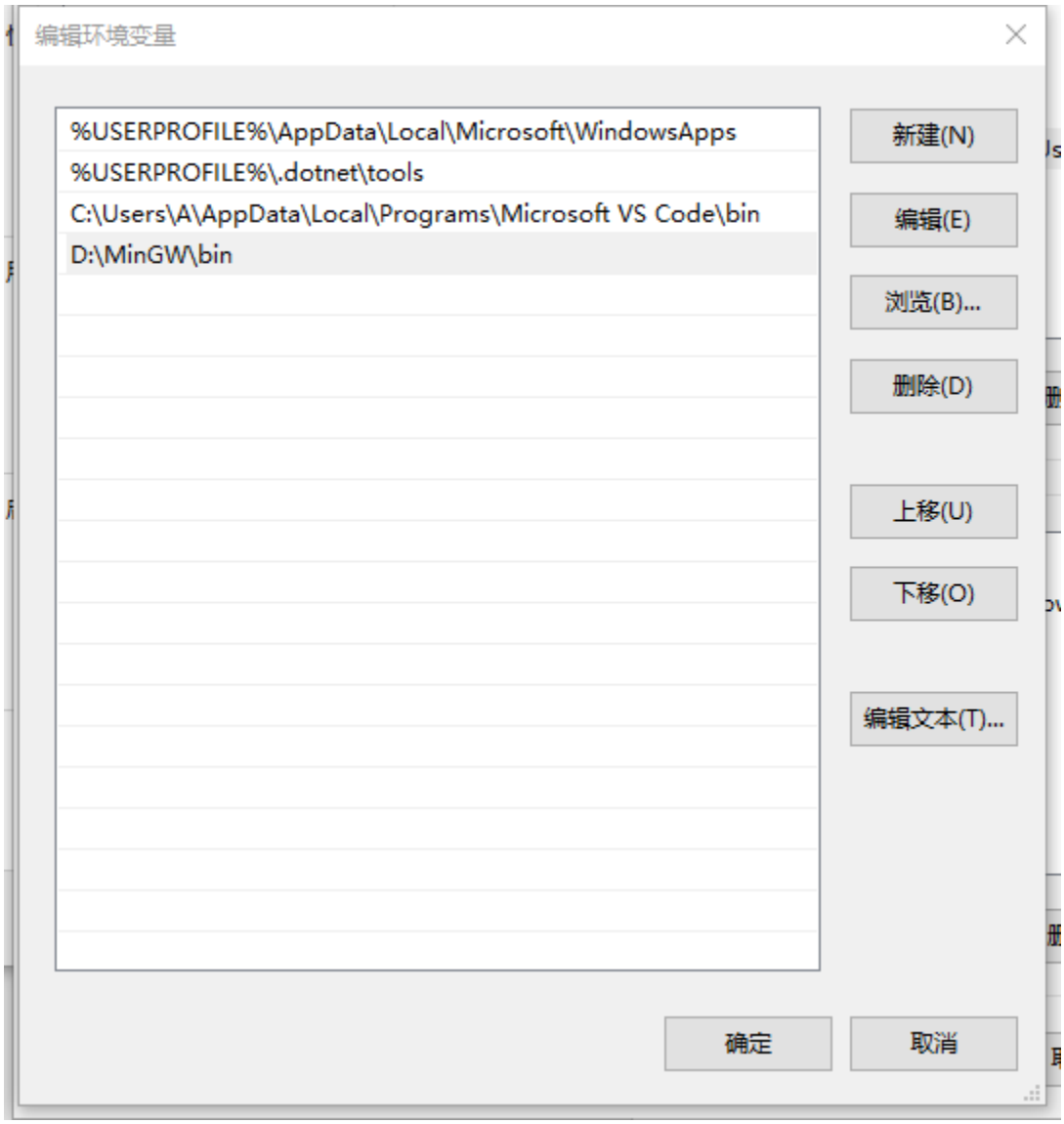


点击左侧栏第五个按钮，进入插件栏，如需中文可以搜索 Chinese，搜索结果中的中文语言包自行下载安装。搜索 C/C++并安装


File Edit Selection View Go Run
Extension: Chinese (Simplified) (简体中文) Language Pack for Visual Studio Cod...
EXTENSIONS: MARKET...
Chinese
Chinese (Simplified) (简体...
中文(简体)
Microsoft
Chinese (Tradi... 944K 5
中文(繁體)
Microsoft
Install
Chinese Transla... 18K 5
Translates between Tradition...
Compulim
Install
ESLint Chinese ... 24K 5
eslint中文规则提示插件
maggie
Install
Chinese Lorem 17K 4
Traditional Chinese Lorem
Kevin Yang
Install
Chinese Lorem 18K
简体中文的乱数假文
catlair
Install
Chinese Translator 2K 5
Translate between Tradition...
mollykannn
Install
Chinese Lorem Ipsum 4K
Chinese Lorem Generator fo...
Galen Dai
Install
Chinese Support fo... 5K
Enables grammar checking ...
Adam Voss
Install
Chinese Colors 298
预设颜色补全——中国传统色
taiyuuki
Install
Quick Chinese ... 493 5
Get Started
Extension: Chinese (Simplified) (简体中文) Language Pack for Visual Studio Code
Chinese (Simplified) (简体中文) La
Microsoft
18,015,922
(44)
Language pack extension for Chinese (Simplified)
Uninstall
This extension is enabled globally.
Details
Feature Contributions
Changelog
适用于 VS Code 的中文（简体）语言包
此中文（简体）语言包为 VS Code 提供本地化界面。
使用方法
通过使用"Configure Display Language"命令显式设置 VS Code 显示语言，可以替代默认 UI 语言。 按下"Ctrl+Shift+P"组合键以显示"命令面板"，然后键入"display"以筛选并显示"Configure Display Language"命令。按"Enter"，然后会按区域设置显示安装的语言列表，并突出显示当前语言设置。选择另一个"语言"以切换 UI 语言。 请参阅文档并获取更多信息。
参与
Categories
Language Packs
Extension Resources
Marketplace
Repository
Microsoft
More Info
Released 2018/4/24
on 01:35:37
Last 2022/8/31
In order to use VS Code in Chinese Simplified, VS Code needs to restart.
Restart
Don't Restart
0 0

安装完成后，新建一个文件夹，文件夹名称不能带有英文，同时下载 MinGW，下载地址为 https://sourceforge.net/projects/mingw/

下载完成后安装 MinGW，将安装位置添加到环境变量中，Win10 的操作如下，右击 Windows 图标，点击系统，在页面的右边栏找到高级系统设置->高级->环境变量，编辑 PATH 环境变量，将 MinGW 安装路径加入环境变量中，如图，我这里安装在 D 盘。PATH 环境变量的作用，有兴趣的可以自行搜索了解，大二选 ICS 之后需要了解。



返回 VScode，创建名称为".vscode"的子文件夹，在子文件夹里添加四个文件"launch.json","setting.json","tasks.json","c_cpp_properties.json"。

Launch.json 的配置文件，注意 MinGW 路径修改为你的路径

```
{
    "version": "0.2.0",
    "configurations": [
        {
        // 配置 VS Code 调试行为：
        "name": "GDB Debug", // 设置在启动配置下拉菜单中显示调试配置的名称。
        "preLaunchTask": "Compile", // 调试会话开始前要运行的任务。
        "type": "cppdbg", // 设置要使用的基础调试器。使用 GDB 或 LLDB 时必须是 cppdbg 。
        "request": "launch", // 设置启动程序还是附加到已经运行的实例。启动或附加 ( launch | attach ).
```

```
        "program": "${fileDirname}/${fileBasenameNoExtension}.exe", //
调试器将启动或附加的可执行文件的完整路径。
        "externalConsole": true, // 设置是否显示外部控制台。
        "logging": { // 用于确定应该将哪些类型的消息记录到调试控制台。
            "exceptions": true, // 是否应将异常消息记录到调试控制台。默认为
真。
            "moduleLoad": false, // 是否应将模块加载事件记录到调试控制台。默
认为真。
            "programOutput": true, // 是否应将程序输出记录到调试控制台的可选
标志。默认为真。
            "engineLogging": false, // 是否应将诊断引擎日志记录到调试控制
台。默认为假。
            "trace": false, // 是否将诊断适配器命令跟踪记录到调试控制台。默
认为假。
            "traceResponse": false // 是否将诊断适配器命令和响应跟踪记录到调
试控制台。默认为假。
        },
        // 配置目标应用程序:
        "args": [], // 设置调试时传递给程序的命令行参数。
        "cwd": "${workspaceFolder}", // 设置调试器启动的应用程序的工作目
录。
        "environment": [], // 设置调试时添加到程序环境中的环境变量，例如:
[ { "name": "squid", "value": "clam" } ]。
        // 自定义 GDB 或者 LLDB:
        "windows": {
            "MIMode": "gdb", // 指定 VS Code 连接的调试器，必须为 gdb 或者
lldb。
            "miDebuggerPath": "D:/MinGw/bin/gdb.exe" // 调试器的路径，修改
为你的安装路径
        },
        "miDebuggerArgs": "", // 传递给调试器的附加参数
        "stopAtEntry": false, // 设置调试器是否停止在目标的入口（附加时忽
略）。默认值为 false。
        "setupCommands": [{ // 执行下面的命令数组以设置 GDB 或 LLDB
            "description": "Enable pretty-printing for gdb",
            "text": "-enable-pretty-printing", // 鼠标悬停查看变量的值，需
要启用 pretty-printing 。
            "ignoreFailures": true // 忽略失败的命令，默认为 false 。
        }]
    }]
}
```

Setting.json 的配置文件:

```
{
    "workbench.editorAssociations": [

    ],
    "files.associations": {
        "iostream": "cpp"
    }
}
```

Tasks.json 的配置文件

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "Compile",
            "type": "shell",
            "windows": {
                "command": "g++",
                "args": [
                    "-g",
                    "-std=c++11",
                    "\"${file}\"",
                    "-o",
                    "\"${fileDirname}\\${fileBasenameNoExtension}.exe\""
                ]
            },
            "group": "build",
            "presentation": {
                "reveal": "silent",
                "focus": false,
                "echo": false,
                "panel": "new"
            },
            "problemMatcher": {
                "owner": "cpp",
                "fileLocation": [
                    "relative",
                    "${workspaceFolder}"
                ],
                "pattern": {
                    "regexp":
"^(.*):(\\d+):(\\d+):\\s+(warning|error):\\s+(.*)$",
                    "file": 1,
```

```
                    "line": 2,
                    "column": 3,
                    "severity": 4,
                    "message": 5
                }
            }
        },
        {
            "type": "cppbuild",
            "label": "C/C++: g++.exe 生成活动文件",
            "command": "D:\\MinGW\\bin\\g++.exe",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "调试器生成的任务。"
        }
    ]
}
```

C_cpp_properties.json 配置，注意 MinGW 安装路径

```
{
    "configurations": [
        {
            "name": "MinGW",
            "intelliSenseMode": "gcc-x64",
            "compilerPath": "D:/MinGw/bin/gcc.exe",
            "cStandard": "c11",
            "cppStandard": "c++17",
            "includePath": [
                "${workspaceFolder}/**",
```

```
                "D:/MinGw/include",
                "D:/mingw/lib/gcc/mingw32/6.3.0/include/c++",
                "D:/mingw/lib/gcc/mingw32/6.3.0/include/c++/tr1",
                "D:/mingw/lib/gcc/mingw32/6.3.0/include/c++/mingw32"
            ],
            "defines": [
                "_DEBUG",
                "UNICODE",
                "_UNICODE",
                "__GNUC__=7",
                "__cdecl=__attribute__((__cdecl__))"
            ],
            "browse": {
                "path": [
                    "${workspaceFolder}/**",
                    "D:/MinGw/include",
                "D:/mingw/lib/gcc/mingw32/6.3.0/include/c++",
                "D:/mingw/lib/gcc/mingw32/6.3.0/include/c++/tr1"
                ],
                "limitSymbolsToIncludedHeaders": true,
                "databaseFilename": ""
            }
        }
    ],
    "version": 4
}
```

完成基本配置之后，新建 cpp 文件，输入以下代码

```
#include<iostream>
using namespace std;

int main() {
    cout << "Hello World!\n";
    system("pause");
    return 0;
}
```

点击上边栏的运行—>启动调试，最后输出 Hello World!即算成功。
本节参考资料
https://blog.csdn.net/agentky/article/details/108802620

注：学术诚信在本门课程中非常受重视，OJ 平台的代码将会进行查重，并且不定期检

查代码成功通过评测的同学的思路，以线上检查为主，如发现剽窃行为，单次被发现，扣除当次作业分数，多次被发现或期末机考作弊，OJ 分数将被判为不及格。

- 学术诚信(什么事情能做，什么不能)

- 如何正确求助: 提问的智慧和别像弱智一样提问
- 以上内容选自蒋炎岩老师的 PA 手册