

常用IDE

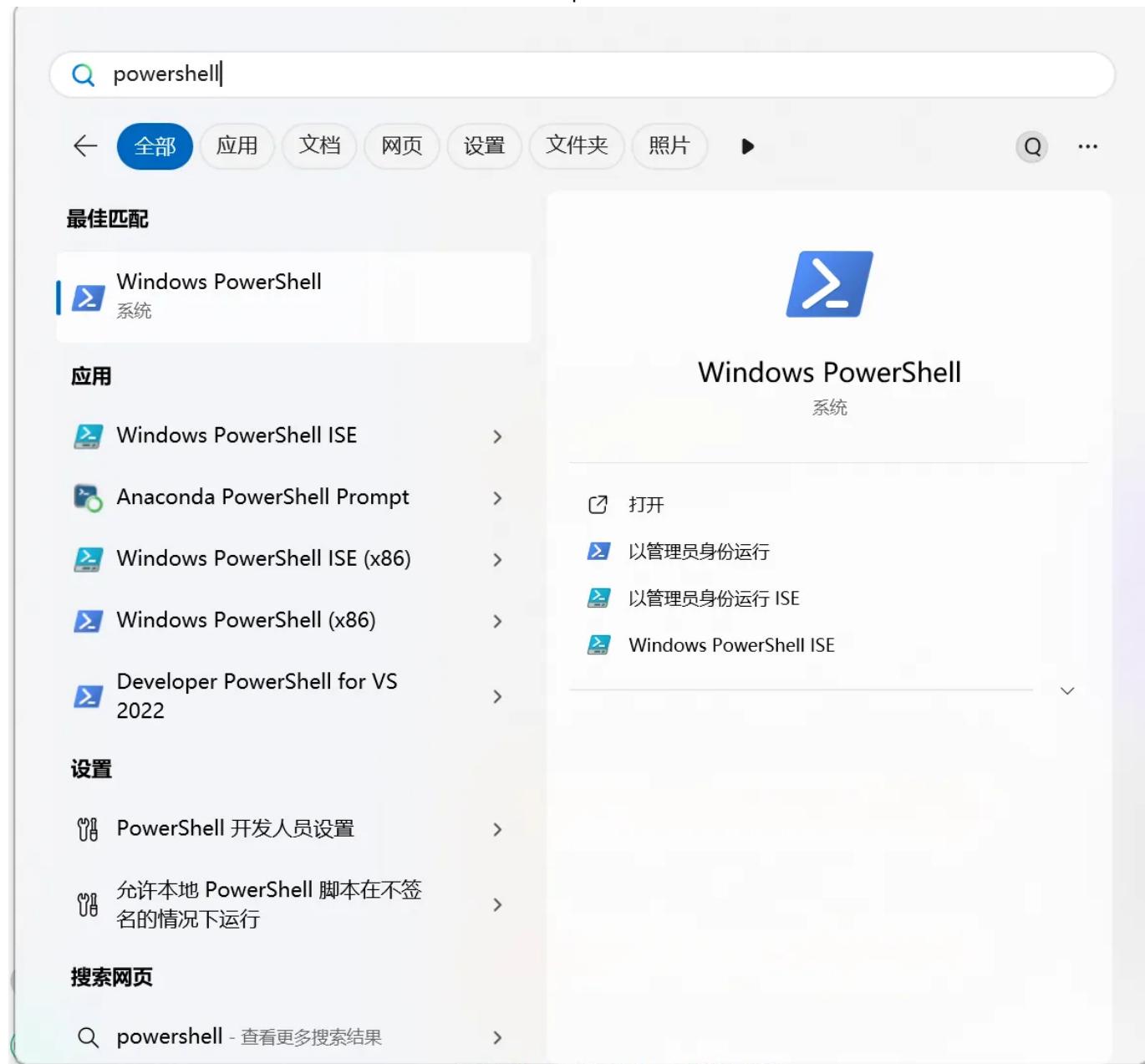
计算机没有通天大道，只有一步一个脚印的坚实

安装不易，软件安装由高品质到低品质

安装 and 运行

第一步就是给电脑添加c++的编译环境，大部分情况新手用的都是一个叫mingw的工具包，我觉得应该很少有人用cmake了，mingw中的GCC是编译c语言的，g++是编译c++的。

有个极简办法，windows系统下有个叫scoop的工具，在开始页面搜索powershell，然后输入scoop install mingw其实就o了，我也推荐另一个叫make的编译工具，scoop install make就可以安装了



```
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
PS C:\Users\lenovo> scoop install mingw
```

这个mingw可以看成一个工具包，里面有gcc, g++, 还有一大堆东西，这个装可能会装很多环境，导致你不知道里面有点啥，但是未来可能某个时候就用上了

至于scoop甚至还可以scoop install python直接就装完了，不用去官网，可以说小白必备。

用gcc --version来看看有没有这个东西，有这个东西的版本那就肯定

```
PS C:\Users\lenovo> gcc --version
gcc.exe (x86_64-posix-seh-rev0, Built by MinGW-Builds project) 15.1.0
Copyright (C) 2025 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

PS C:\Users\lenovo> g++ --version
g++.exe (x86_64-posix-seh-rev0, Built by MinGW-Builds project) 15.1.0
Copyright (C) 2025 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

如果要是报错了一大堆红，那说明你的电脑没有scoop，就只能手动装吧，具体可以参考：[博客-如何安装Mingw](#)。由于手动安装scoop的难度比较大，所以如果没有的话，不推荐大家去安装这个东西。

接下来就是安装各种编写代码的地方

Clion

首先打开[JetBrains官网](#)

往下翻找到Clion软件 当然推荐其他的IDE,比如python的pycharm, java的IDEA (除了CLion其他都分专业版和社区版，专业版要付费，但真正开发还得是专业版，对学生来说可以申请学信网认证，申请一年，到期继续续) ~~由于不收费，实际体验下来我觉得Clion是三个里做的比较不好的，由于jetbrains的高度集成，即使不安装Mingw似乎也是可以去直接跑代码的，给了新手一个轻易上手的机会。~~

A rich suite of tools that provide an exceptional developer experience



IntelliJ IDEA
IDE for Java and Kotlin



PyCharm
IDE for Python



ReSharper
Visual Studio extension for .NET



WebStorm
IDE for JavaScript



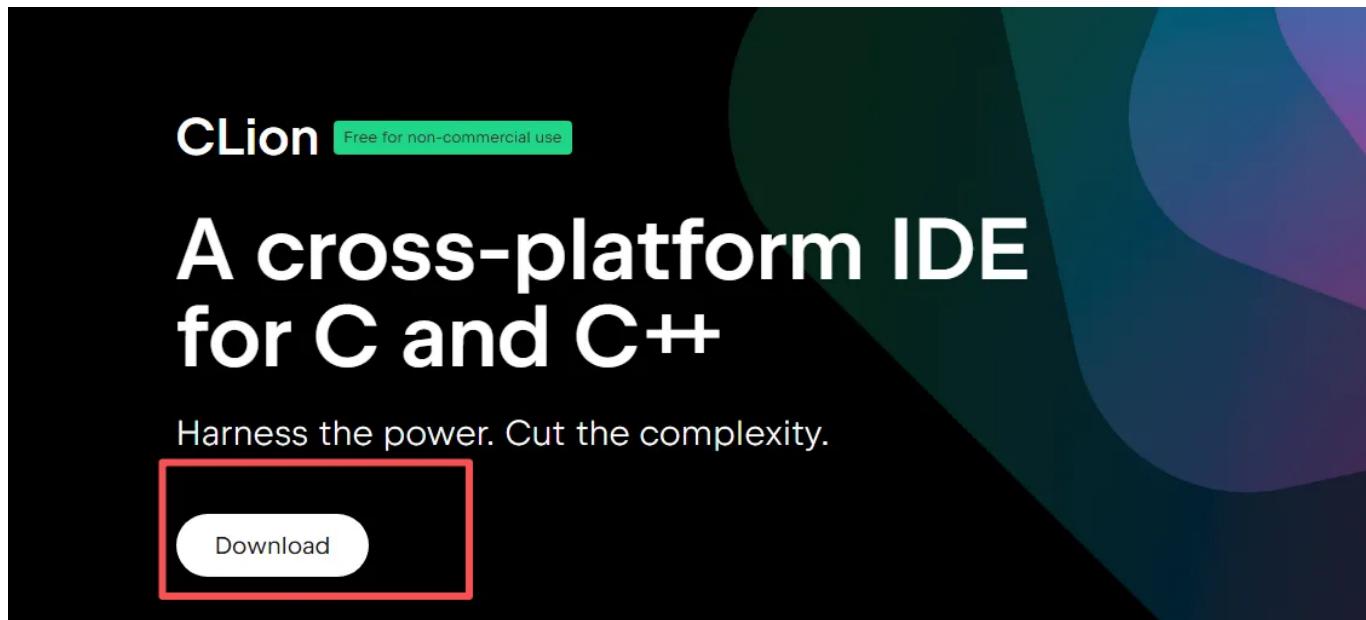
Rider
IDE for .NET and game dev



CLion
IDE for C and C++ developers

Explore JetBrains IDEs and code authoring tools →

点击DownLoad



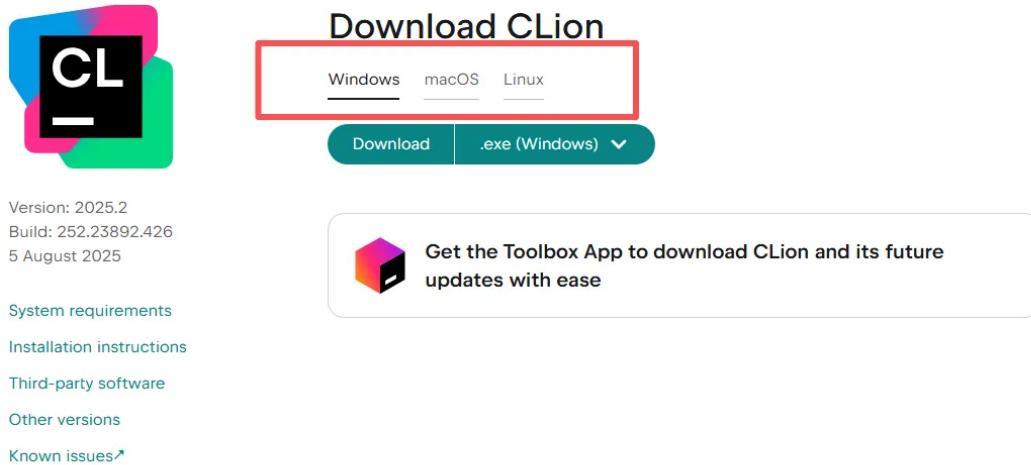
CLion Free for non-commercial use

A cross-platform IDE for C and C++

Harness the power. Cut the complexity.

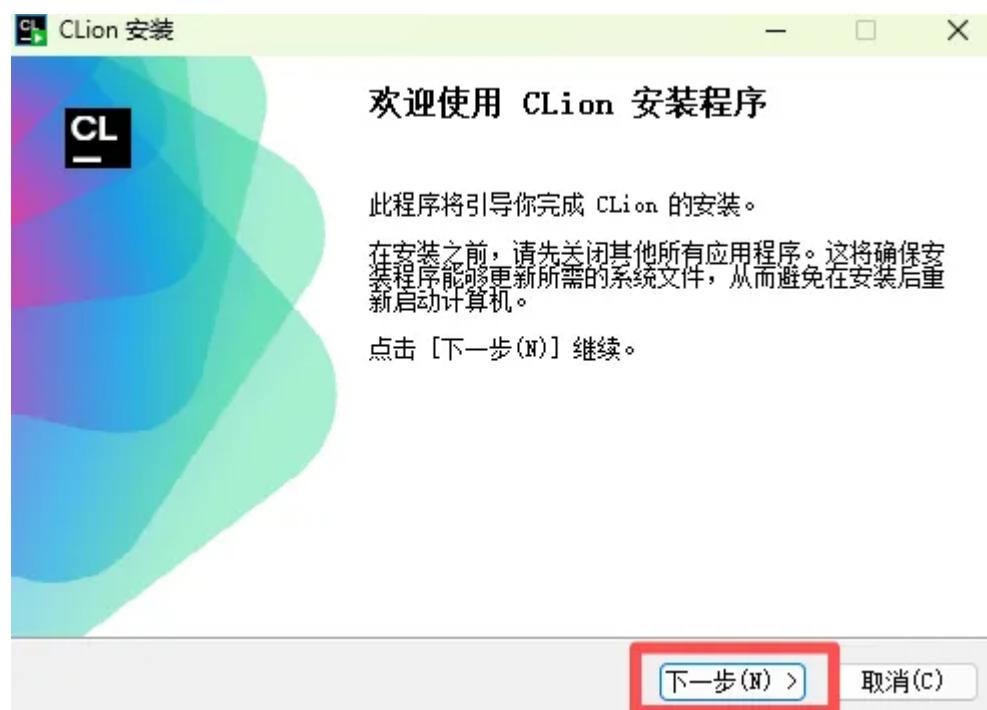
[Download](#)

选择自己的操作系统，现在大部分电脑应该都是win11了，当然不排除你是苹果电脑的可能，或者喜欢用linux开发(暂不推荐)

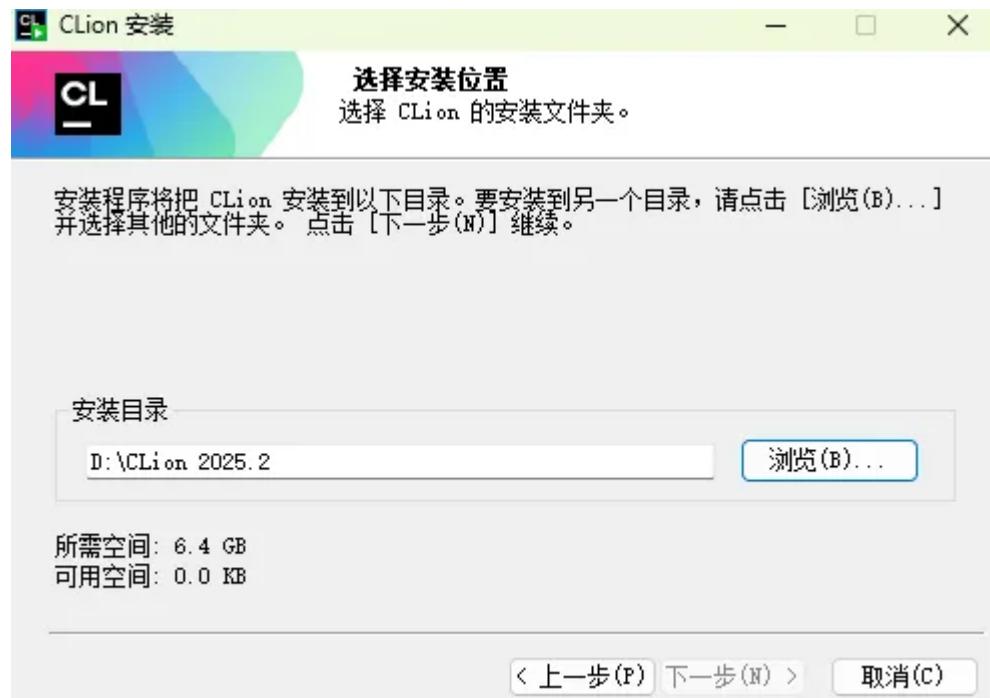


点击download，过一段时间就开始安装exe文件了，(截止2025.8.28最新版是2025.2，我还在用2025.1.3.1)

然后开装



选个安装位置

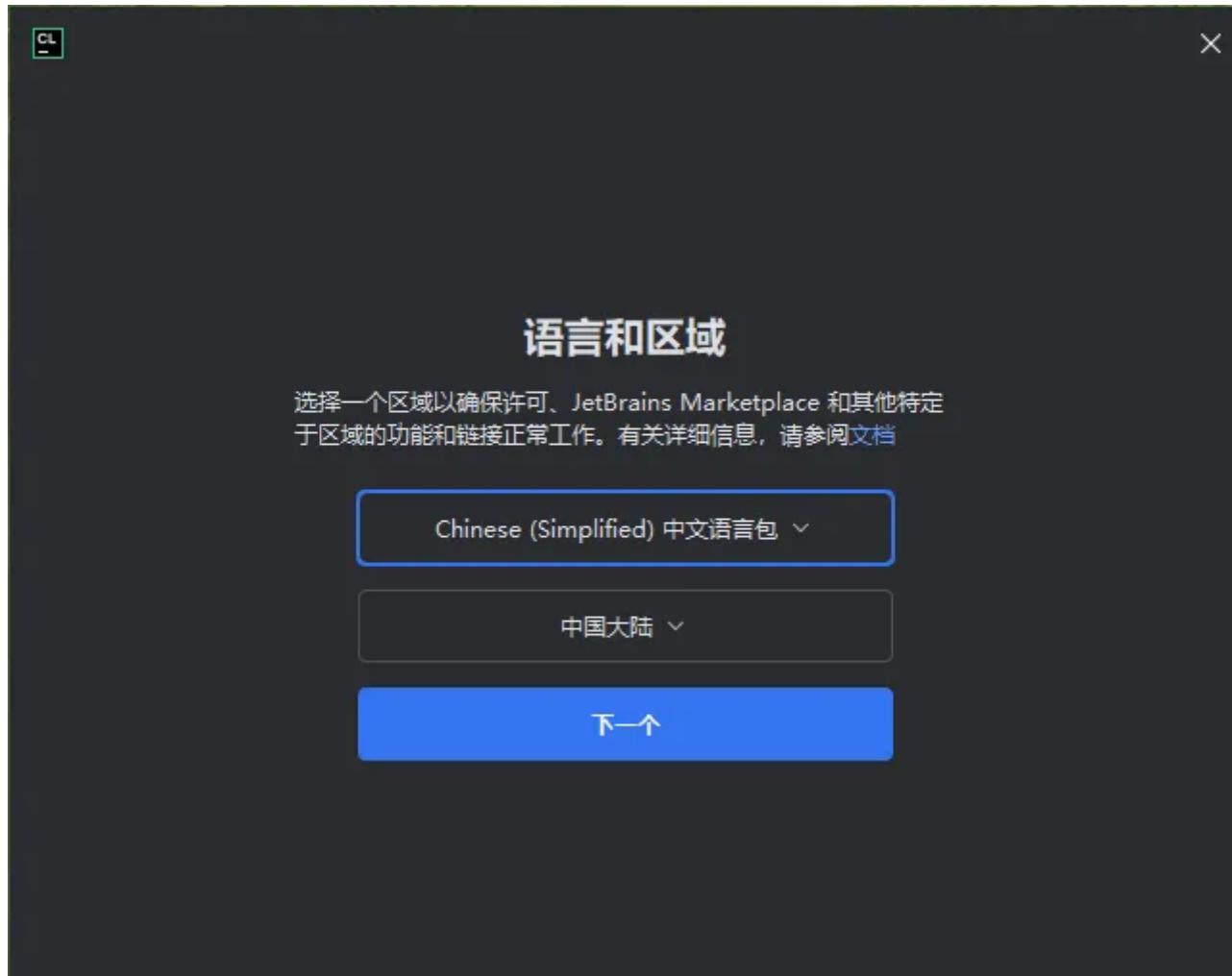


全选即可



然后最后一步点安装，最大的遍历就是不用自己添加PATH，刚才勾选的选项就有，后面就会看到vscode这一缺陷了，然后创建关联里勾选了cpp(c++)和c，这里说一下，**c++的文件中是可以编写c程序的。**

接下来选择语言，同意协议，然后不发送请求一系列经典软件安装操作

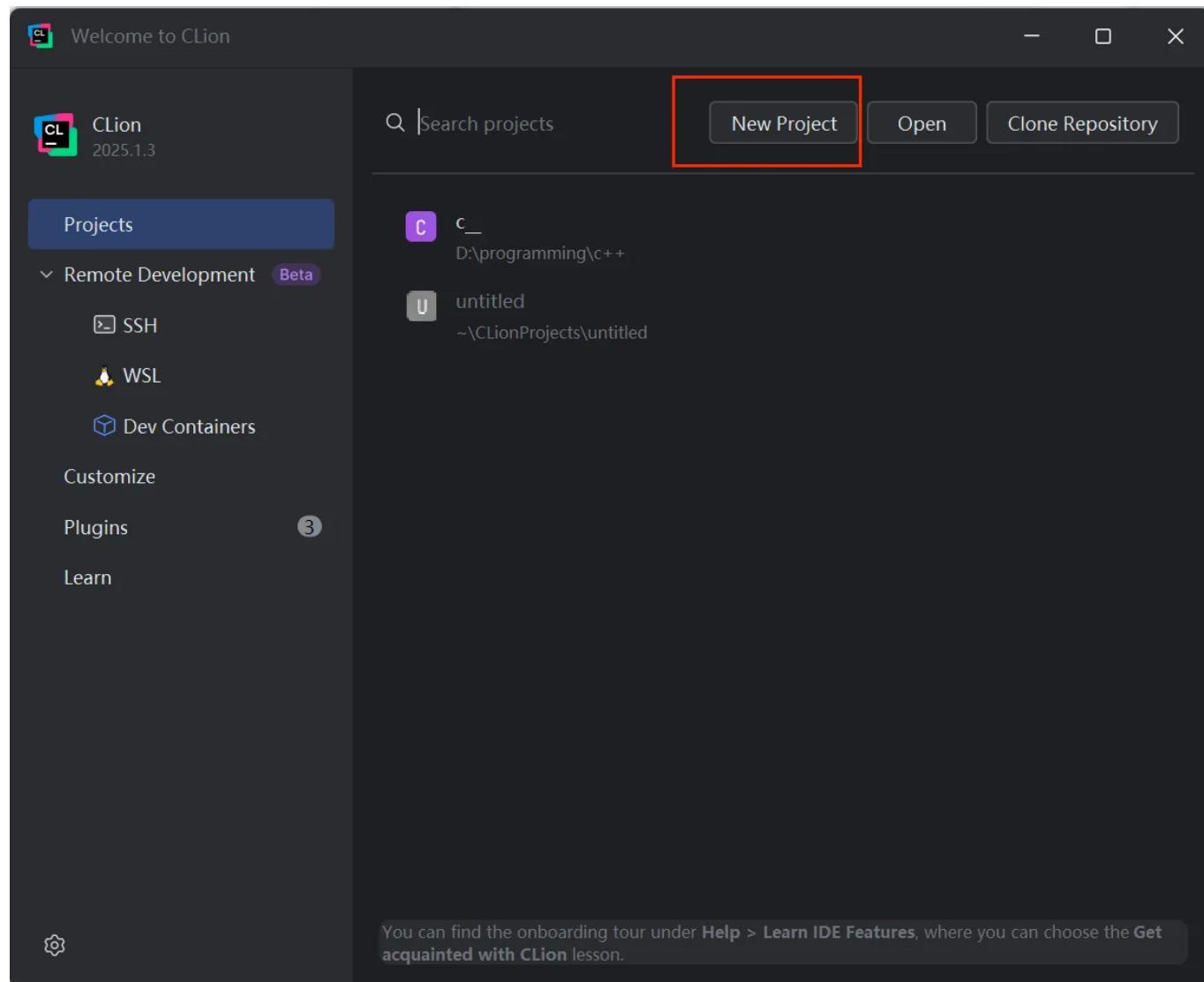


主要是这个界面，选择非商业用途

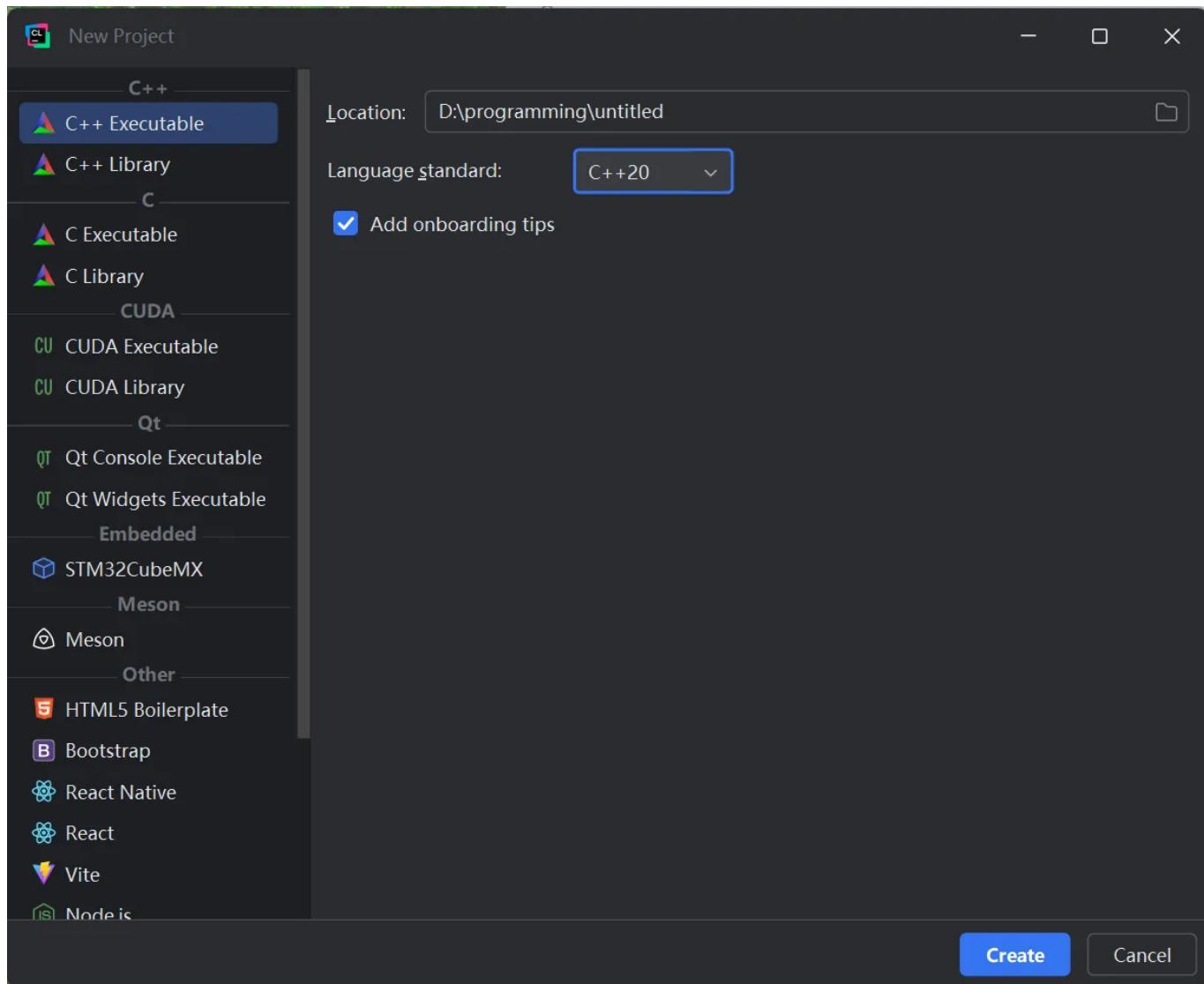


需要自己注册一个账号，微信，谷歌，邮箱等都可以，看自己一般用哪个，这个过程就可以直接省略了

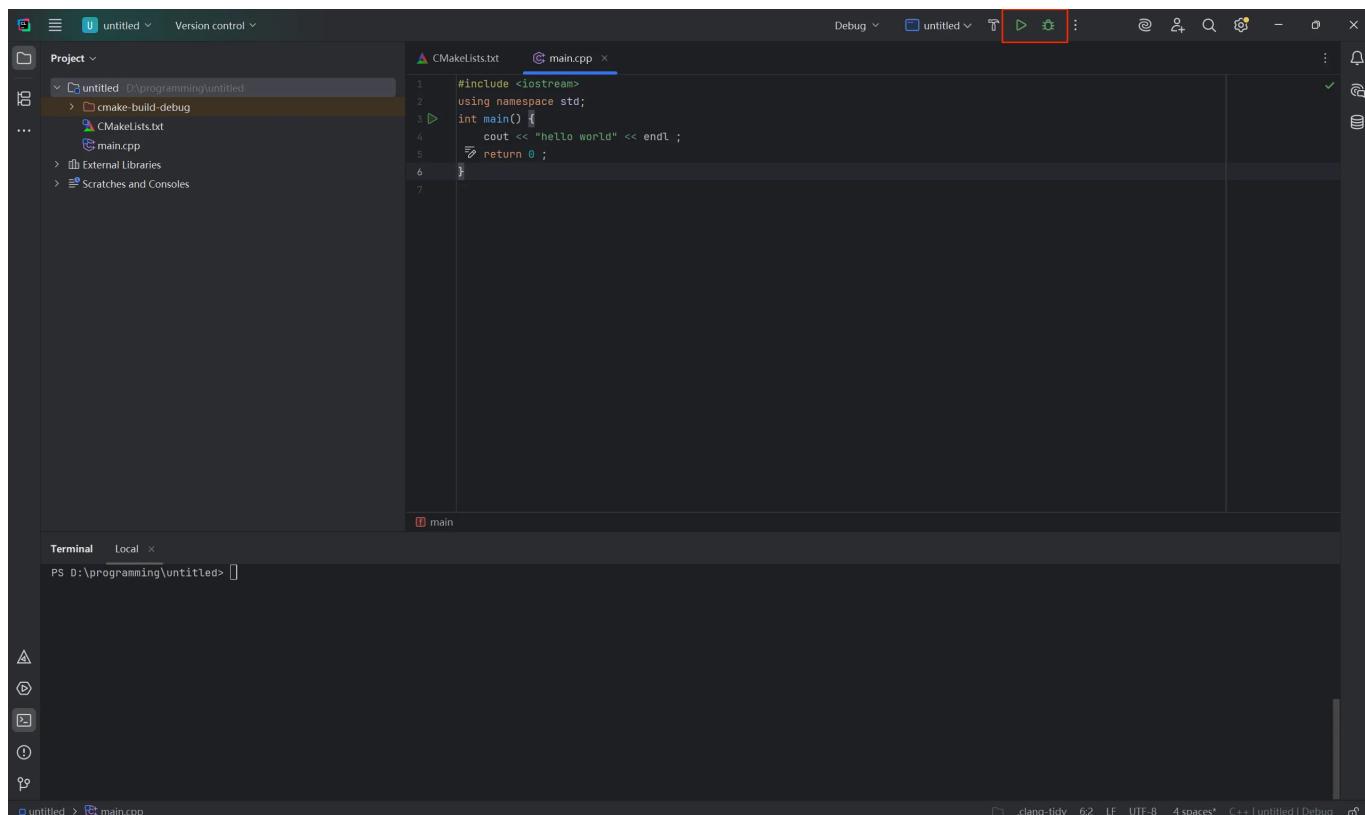
接下来就是**使用环节**了,先new一个Project也就是我们要写的项目（你可以理解成一个大文件下有很多小文件）



选择项目文件存放地址和自己要用的C++版本



等待一会儿自动添加编译器，然后自己就能开始写代码了



如果右上角这两个亮了，证明你配置对了，第一个是运行程序，clion会自动生成cmakelist文件，就是能让程序跑起来，第二个是调试程序(放最后说)

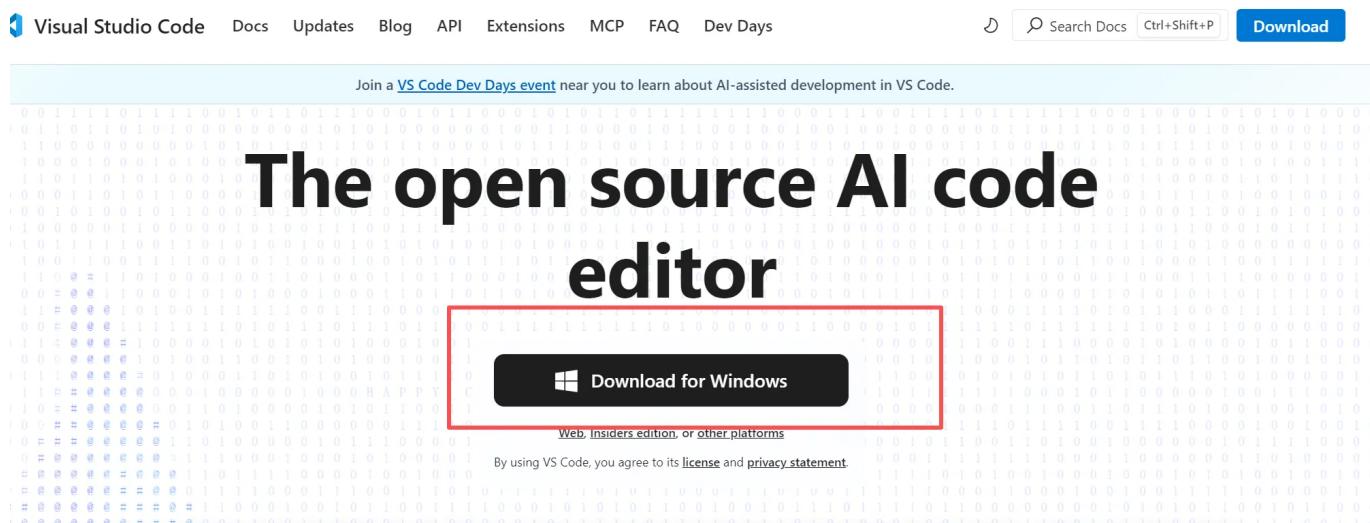
顺便关一下代码提示补全 参考一下这个链接

可能现在意识不到clion的强大，但真正学开很多底层知识的时候发现有很多实用的功能

Vscode

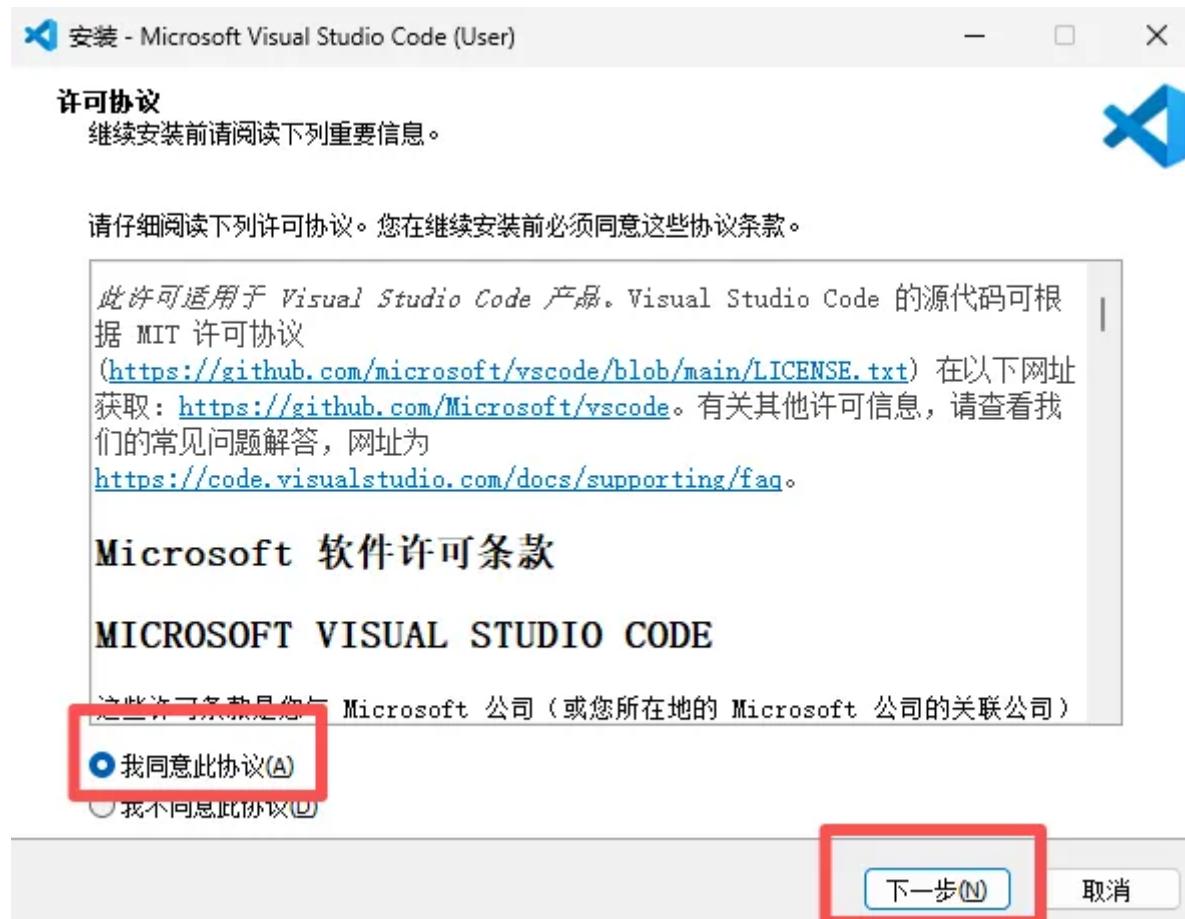
首先打开 [vscode官方链接](#)

点击中间大大的download按钮

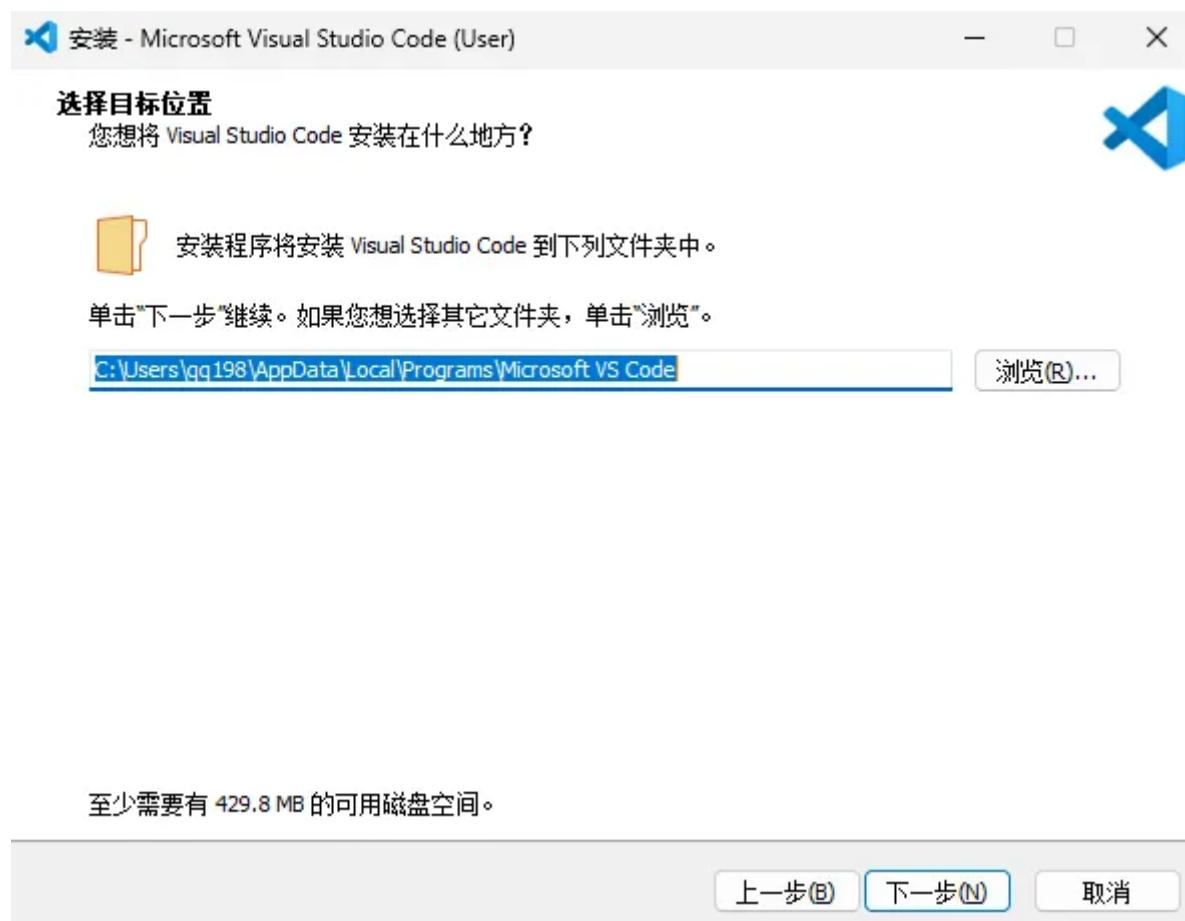


接下来有的可能要选择是什么系统，有的可能就直接跳转到下载了(因为我虚拟机和电脑的界面就不一样，选完windows系统就自动开始下载了) 然后运行下载好的exe

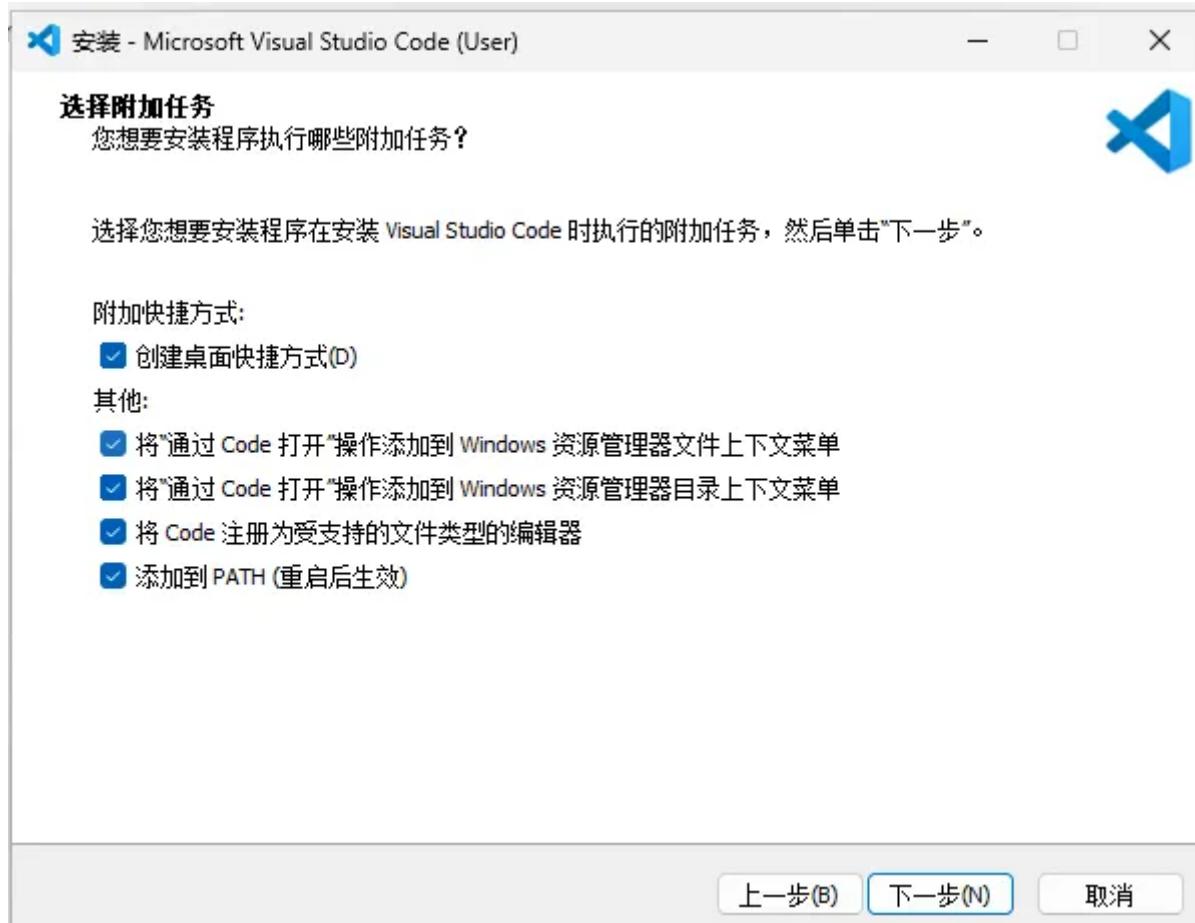
同意，下一步



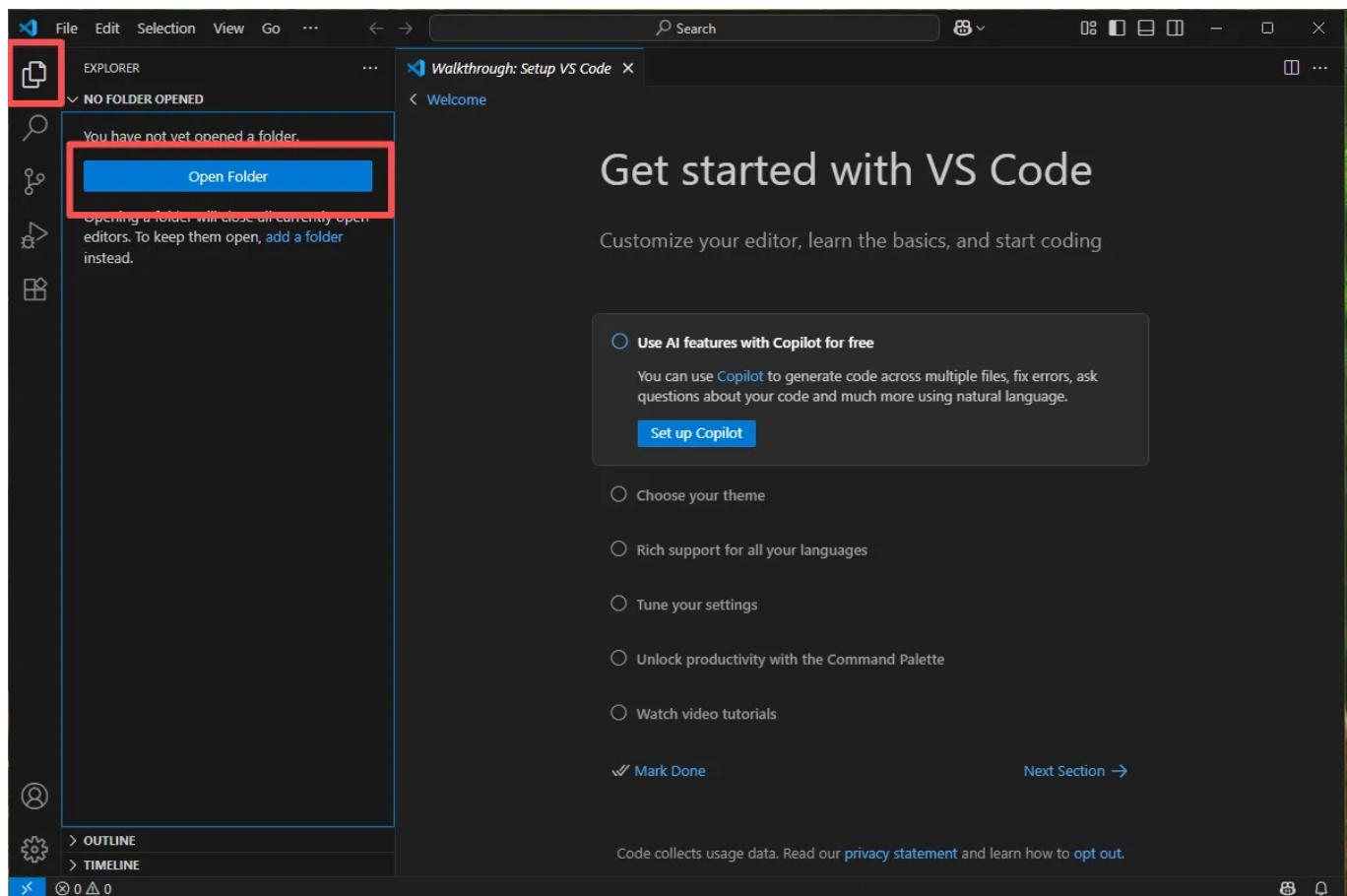
放一个位置



下一步之后再下一步，全选就可以了这里主要是创建一些快捷方式。

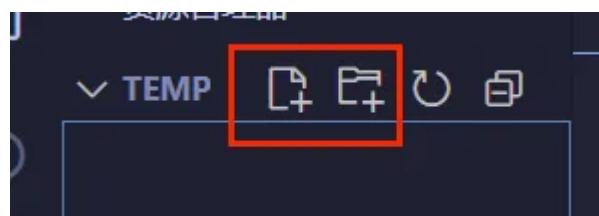


安装好之后



这里就可以打开文件夹了，就是你要写代码的地方，中途会跳出来一个是否信任该文件，信任就可以了。

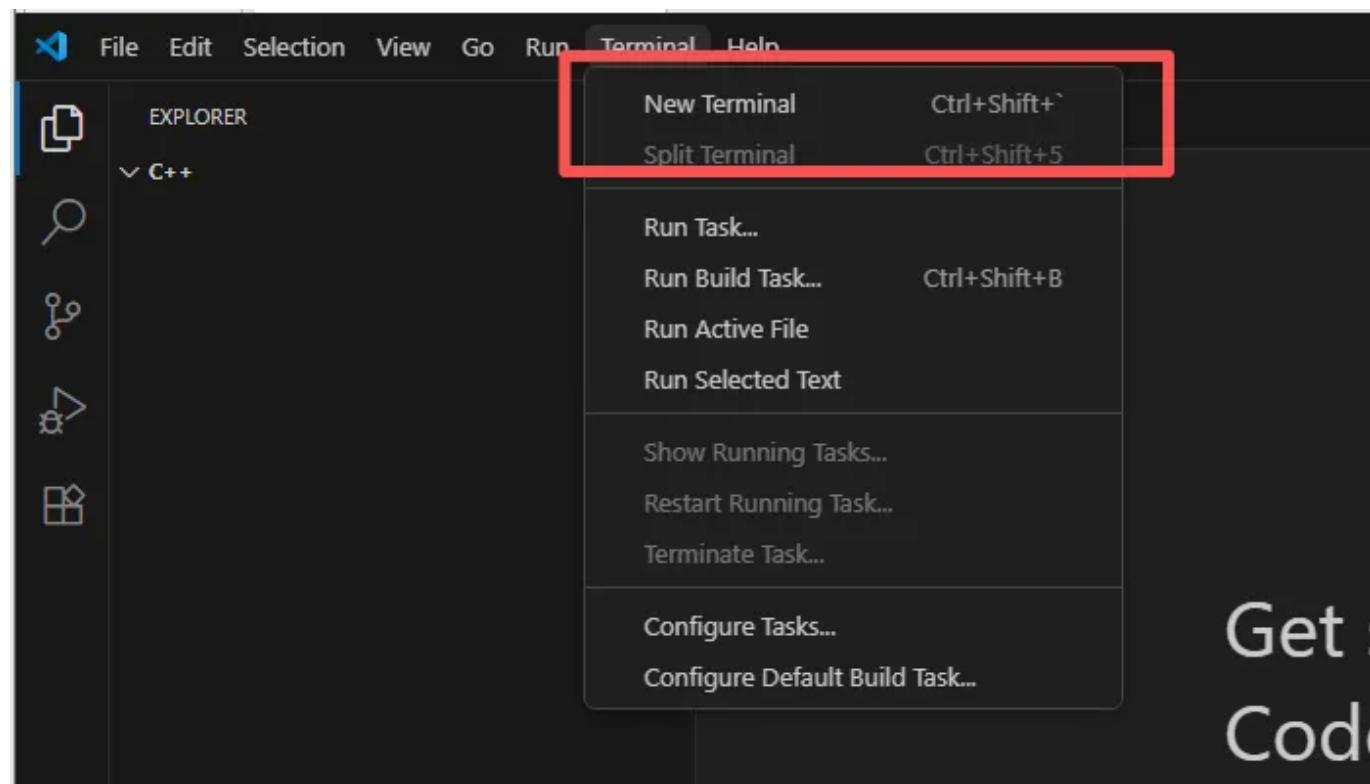
主要用前两个，第一个是新建文件，第二个是新建文件夹



假设我现在写了一段代码

```
C++ hello.cpp > main()
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << "hello world" << endl ;
7     return 0;
8 }
```

如果你翻阅了网上很多教程,这里就开始复制粘贴改各种的配置了。插件的部分我们再说，但是js文件你只会找，然后还有一大堆头疼的问题，还没开始呢就已经弃坑了，这里我想说的就是，**很多计算机课程不联系是学计算机最大的一个问题**，这部分知识其实是计科导实验的部分，不多说了，直接说解决办法



新建一个终端，这就是常说的黑框框（包括第一下的powershell也是）如果你用过vs就会发现他是分离式的，而vscode的是直接展示在下面的

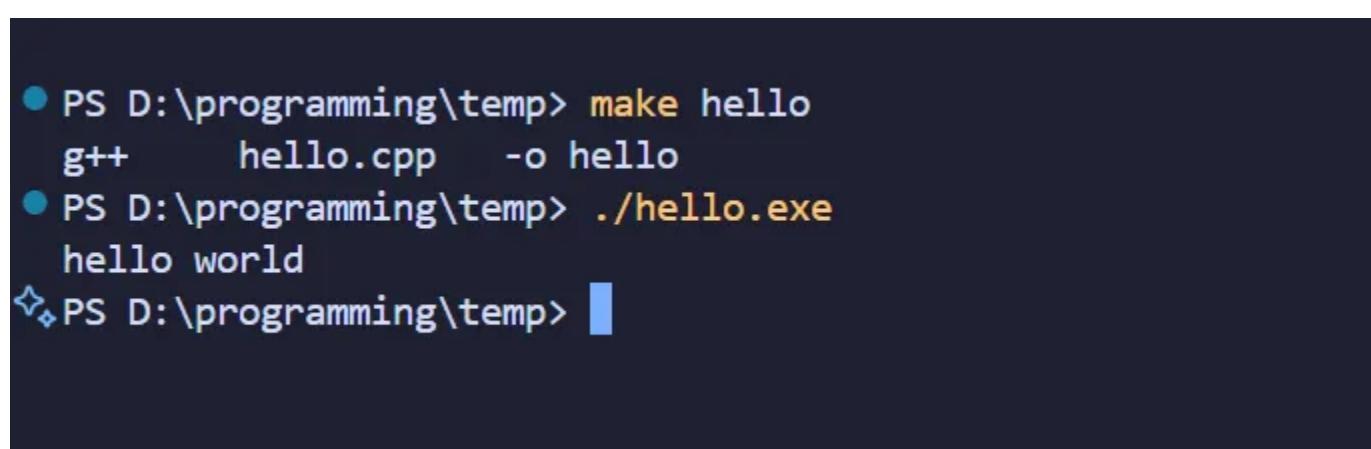
还记得刚开始的mingw么，里面包含了g++，使用

```
g++ hello.cpp // g++ 加上文件名字  
./a.exe      // 运行文件使用./生成的exe名字  
.\\a.exe     // 都一样的
```

一般来说g++生成的都是a.exe 也可以用make (前提是用scoop装了)

```
make hello // 不加文件后缀名  
./hello.exe // 运行  
.\\hello.exe // 也可以
```

make会直接生成其所对应的exe文件



```
● PS D:\programming\temp> make hello  
g++      hello.cpp    -o hello  
● PS D:\programming\temp> ./hello.exe  
hello world  
◆ PS D:\programming\temp> █
```

鄙人觉得make比g++好用。这就是命令行式与程序进行交互，这才是一个程序员需要干的，学以致用。不是大部分时间配环境都需要ctrl c/v

当然不配置各种js文件，另辟蹊径的路也是有bug的

```
g++ hello.cpp
```

```
C:/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.2.0/../../..//x86_64-w64-mingw32/bin/ld.exe:  
C:/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.2.0/../../..//x86_64-w64-  
mingw32/lib//lib/libmingw32.a(lib64_libmingw32_a-crtexewin.o):crtexewin.c:(.text.startup+0xb4): undefined  
reference to `WinMain'
```

然后报了一大堆不懂的错误，这里涉及到了很多隐晦而又底层的东西，我希望你能够在未来的学习旅途中，逐步去理解为什么会发生这样的问题。我们这里只讲解解决方案

将main()改成WinMain()即可

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

```
int WinMain()
{
    cout << "hello world" ;
    return 0 ;
}
```

看到这里其实已经老眼昏花了感觉，但是说实话计算机没有好走的路。

这里的具体操作参考了哈佛大学的cs50课程，有兴趣可以去b站上看一看，主要是教编程入门(C + Pyhont + HTML,CSS,JS初步)的，也是命令行+程序的开发，希望能够有所帮助。

以上所有指令适用于CLion中，但是CLion中点下按钮就能自动进行G++能一系列操作，而网上的各种教程也只是帮你配置如何点按钮自动执行罢了。

另一种方法就是配环境，如果不喜欢命令行的操作，不喜欢和powershell打交道，那就去[博客-保姆级配置vscode](#)，可以说五天计算机，三天配环境。

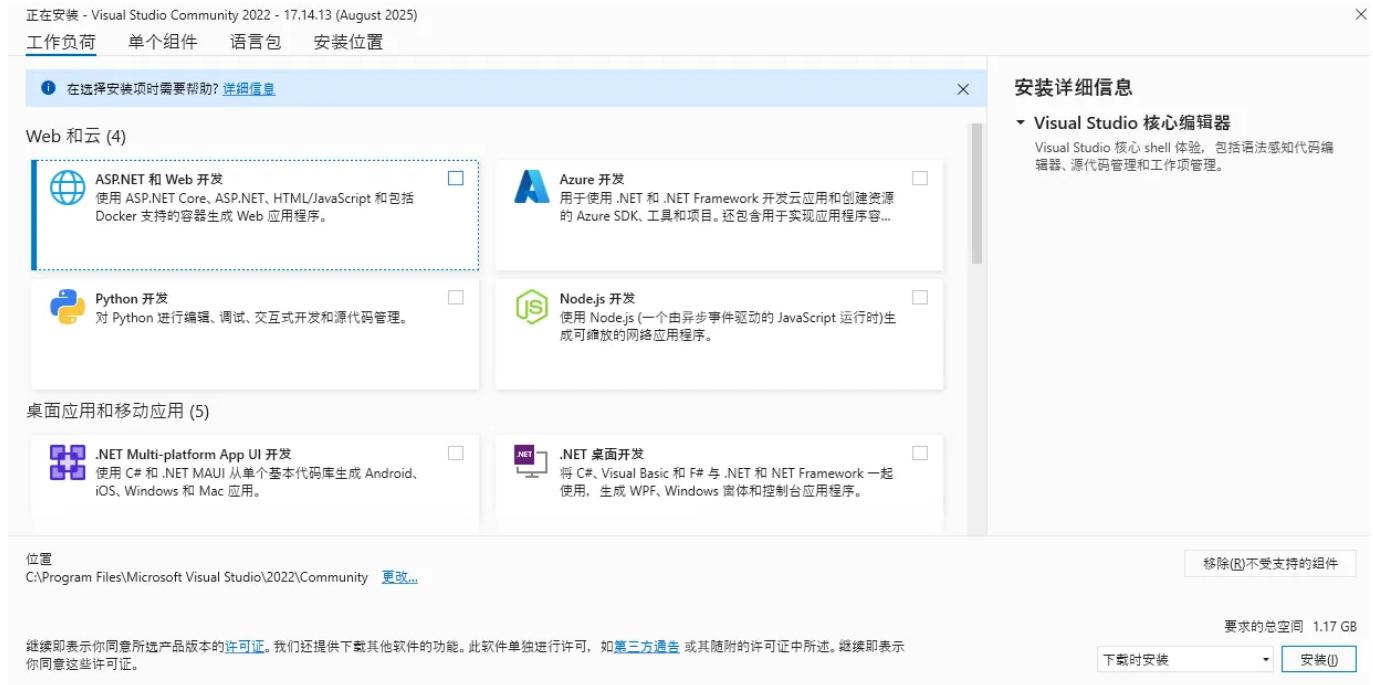
这里如果大家选择走命令行的话，调试会很痛苦，如果选择走配环境的话，那么配起来会很痛苦。

Visual Studio

依旧打开[官方链接](#)，选择社区的免费下载

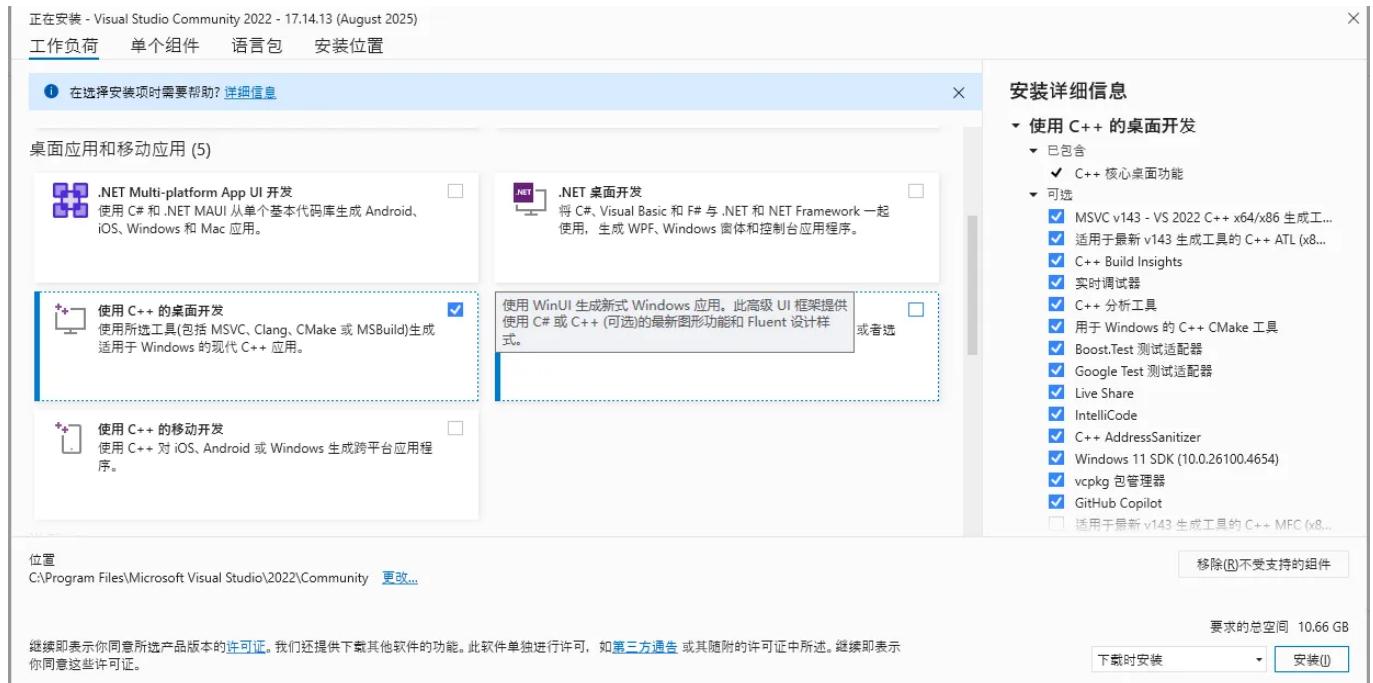
The screenshot shows the Microsoft Visual Studio download page. At the top, there's a navigation bar with links for Microsoft, Visual Studio, developer tools, download, buy, and sign in. A prominent purple banner says "Visual Studio 中心就在这里!" and "我们很高兴引入一个中心位置来完成 Visual Studio 的所有操作". Below this, a large heading "下载" is centered. To its right is a "预览版" section with a "下载" button and a link to "了解详细信息". The main area features three download options: "社区" (Community), "专业版" (Professional), and "Enterprise". The "Community" option is highlighted with a red border around its "免费下载" (Free Download) button. Below each option are links for "发行说明", "比较版本", "如何脱机安装", and "许可条款".

然后就开始安装了，安装好后运行下载好的exe文件，然后点击继续，跳到下面的界面



往下翻，找到**C++桌面开发**和**Visual Studio 扩展开发(图里没展示)**，勾选，右面会发现有一栏叫做**可选**的东西，我的评价是我基本没用到过，而且当时也不知道这些都干嘛的，直接安装了，占了电脑10.66G一言难尽。所以我是非常不推荐这个重量级选手的(安装的记得改安装路径)

- 扩展开发主要是帮你加一些工具，比如代码补全，代码高亮，git工具等



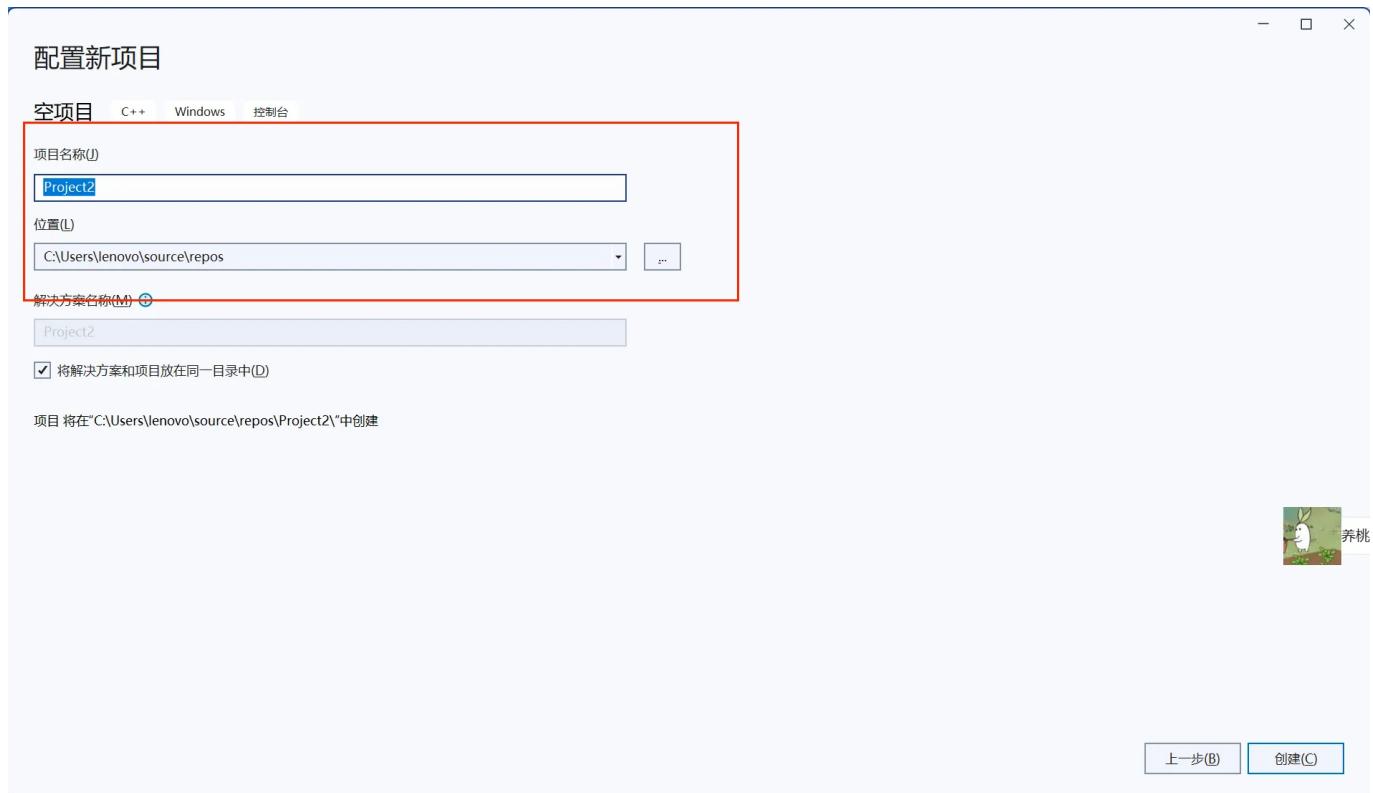
打开之后，创建新项目



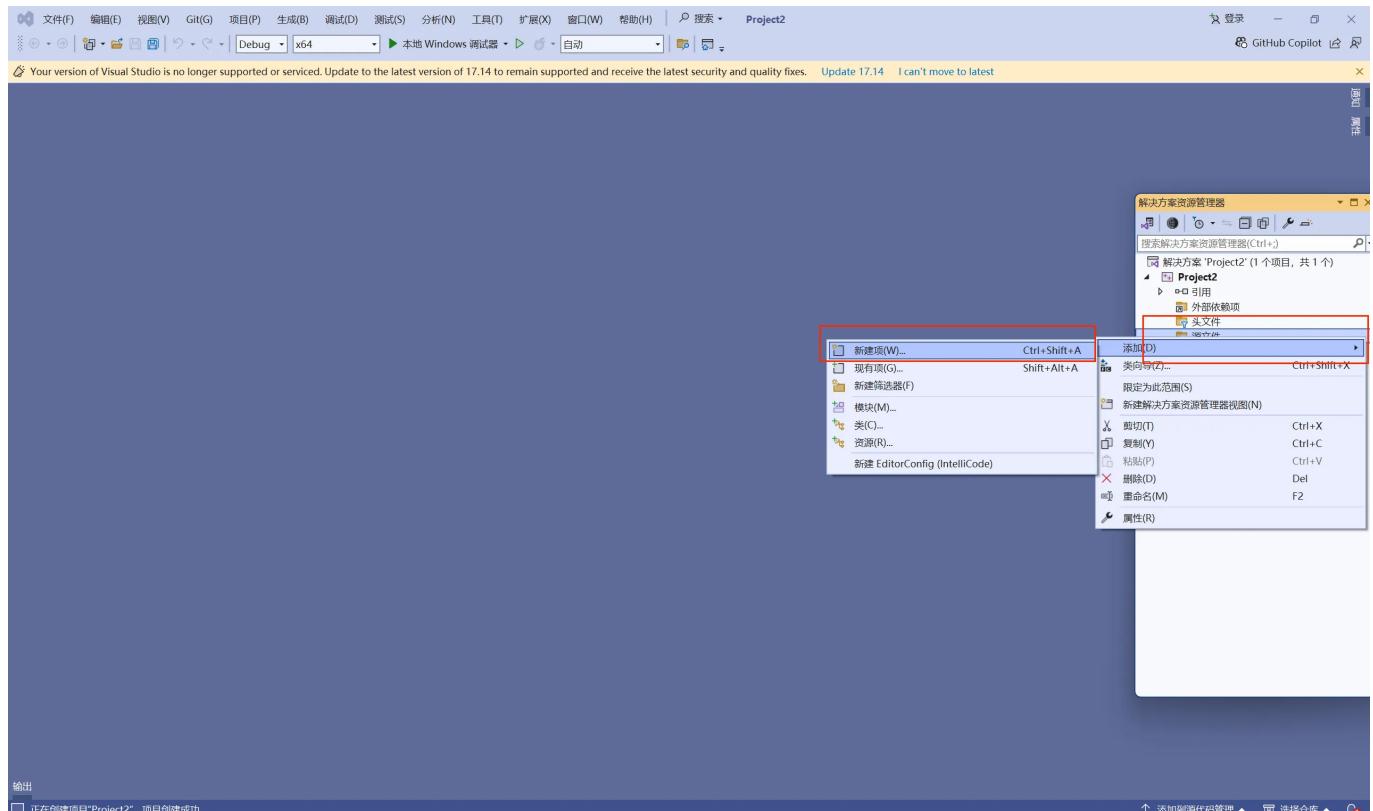
创建空项目



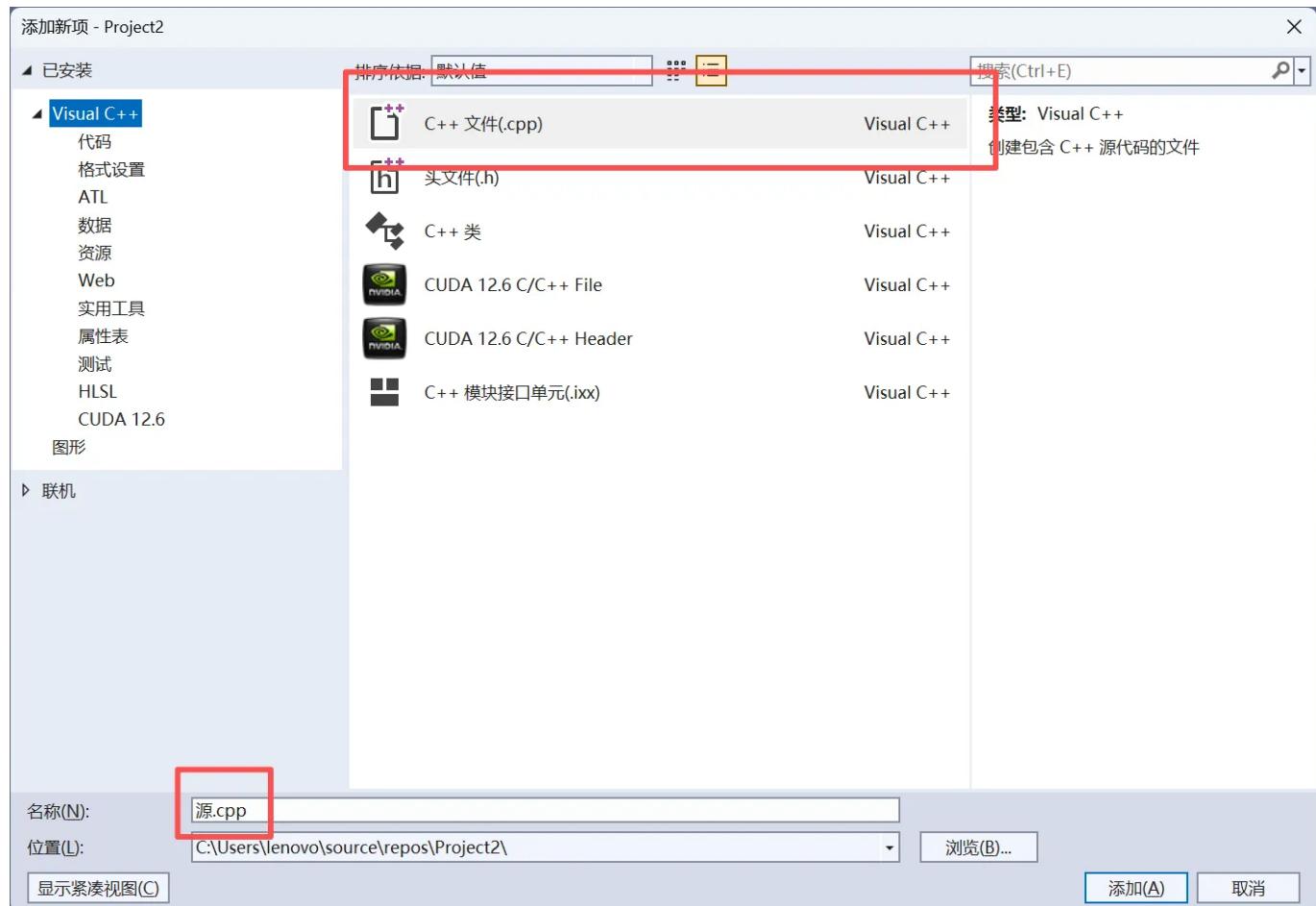
填写项目名称和存放位置这个和Clion差不多



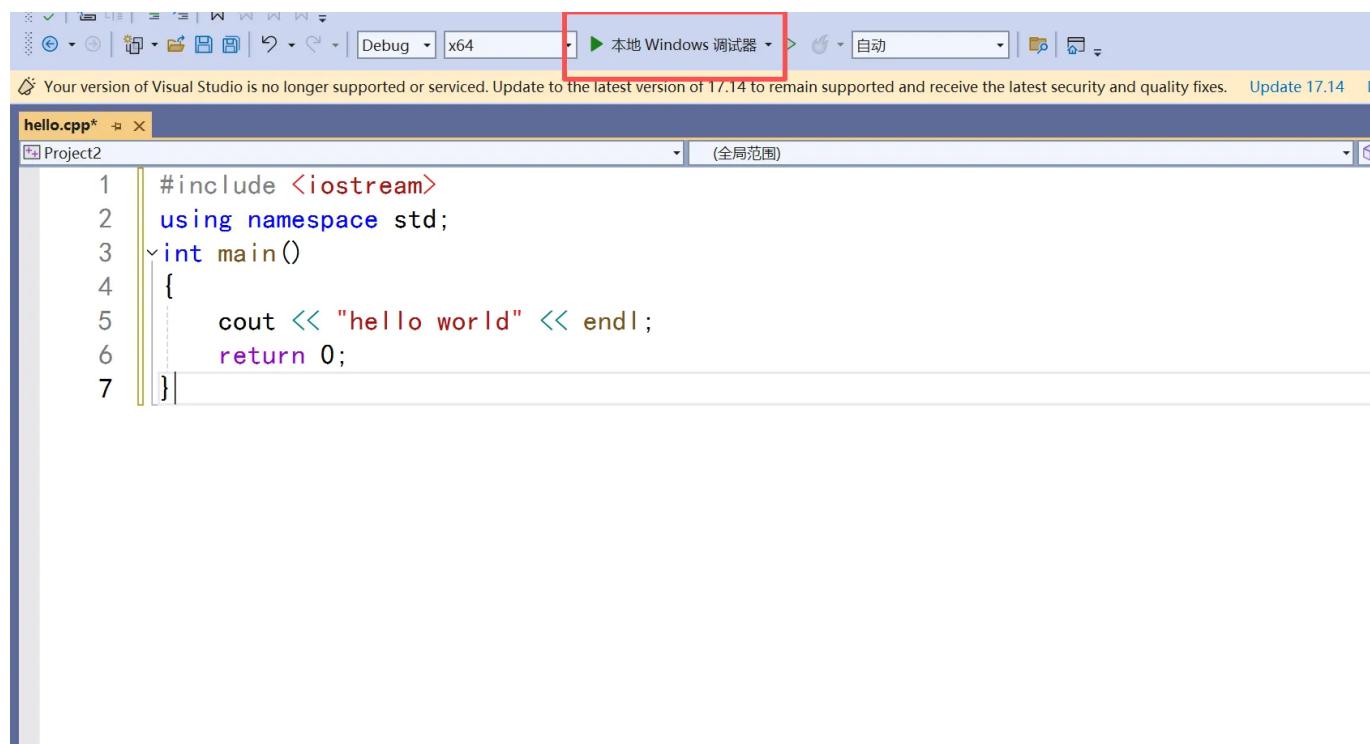
右键源文件这里就是放cpp文件,然后添加新建项



选择cpp文件, 然后改个名字



写完代码点击绿色的三角运行就可以了

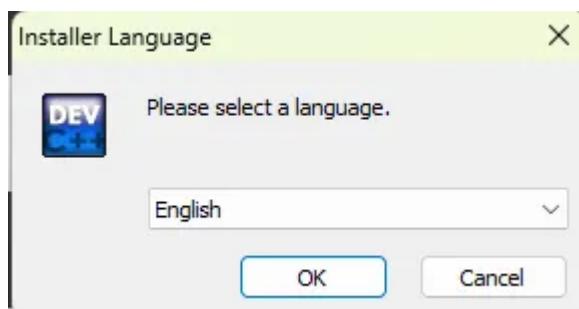


Dev C++

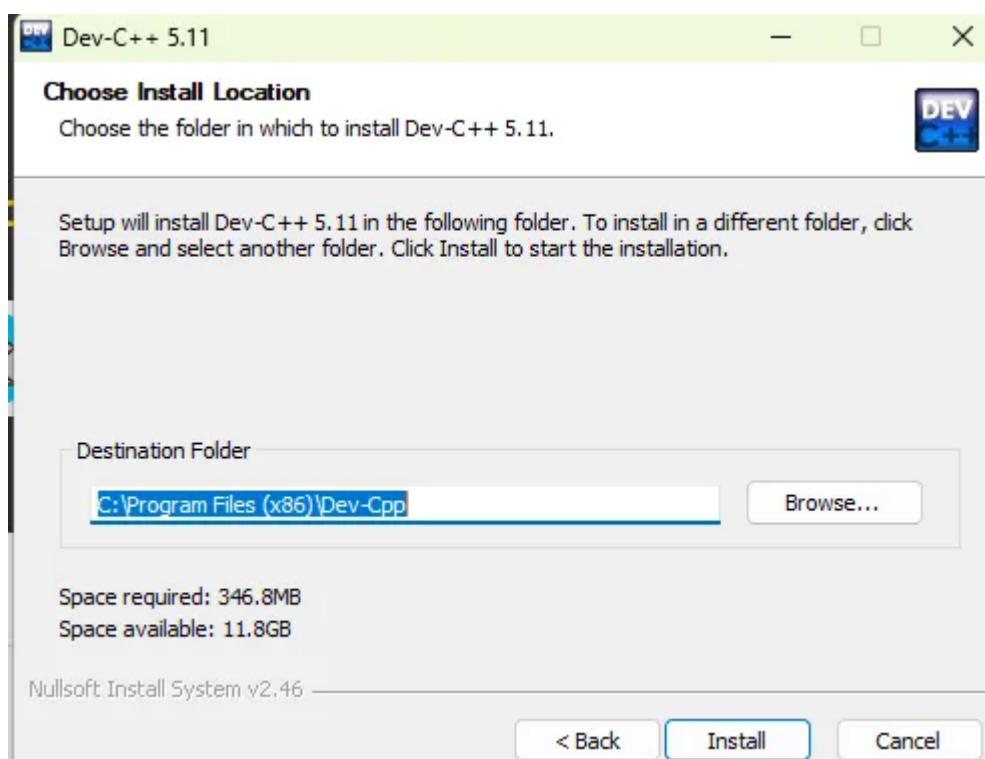
依旧打开官方链接下吧——下一个不吱声

The screenshot shows the SourceForge project page for Dev-C++. At the top, there are navigation links for Business Software, Open Source Software, SourceForge Podcast, and Resources. A banner for Shift, a browser built by you, is displayed. The main content area features the Dev-C++ logo and a brief description: "A free, portable, fast and simple C/C++ IDE. Brought to you by: orwelldevcpp". Below this, a box highlights "143 Reviews" and a large green "Download" button. To the right, there are statistics: "Downloads: 56,658 This Week" and "Last Update: 2016-11-29". A sidebar on the right contains an advertisement for "Build Your Agents" and another for "Code::Blocks". The bottom of the page includes tabs for Summary, Files, Reviews, Support, External Link, Tracker, Code, and Forums.

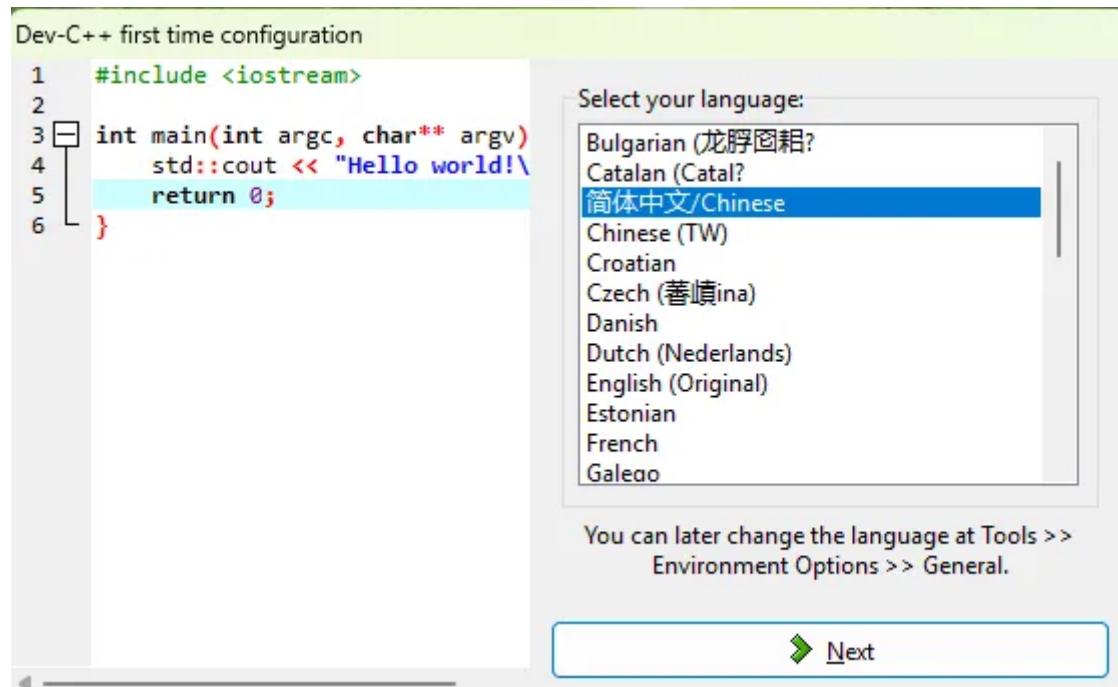
然后运行exe文件，没有中文，只能选英文了。



然后接下来就是I agree,next最后选择安装路径装完就行了。



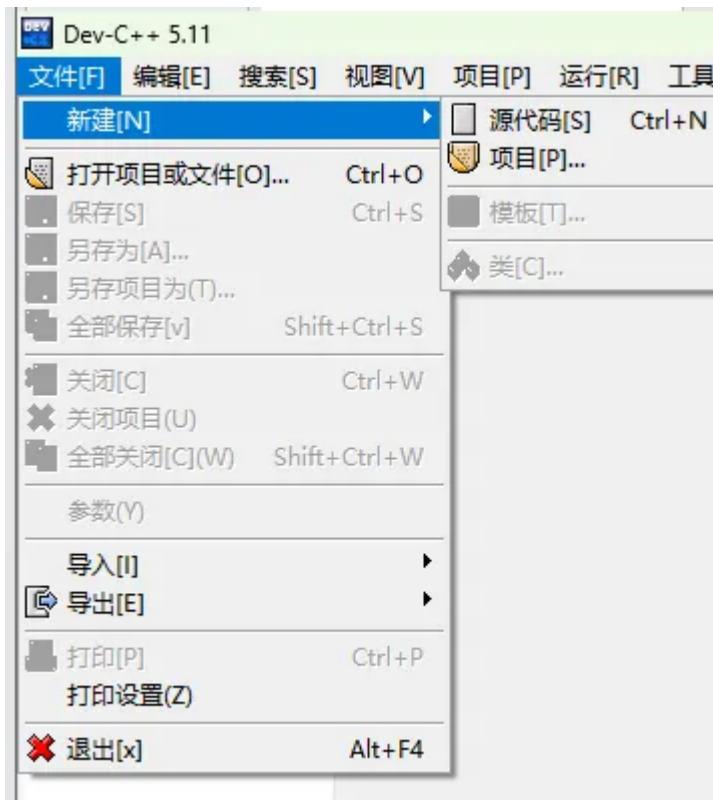
打开软件，这里能选中文了。



这里自己搭配一下颜色主题之类的，有些字体很抽象，建议不要选。



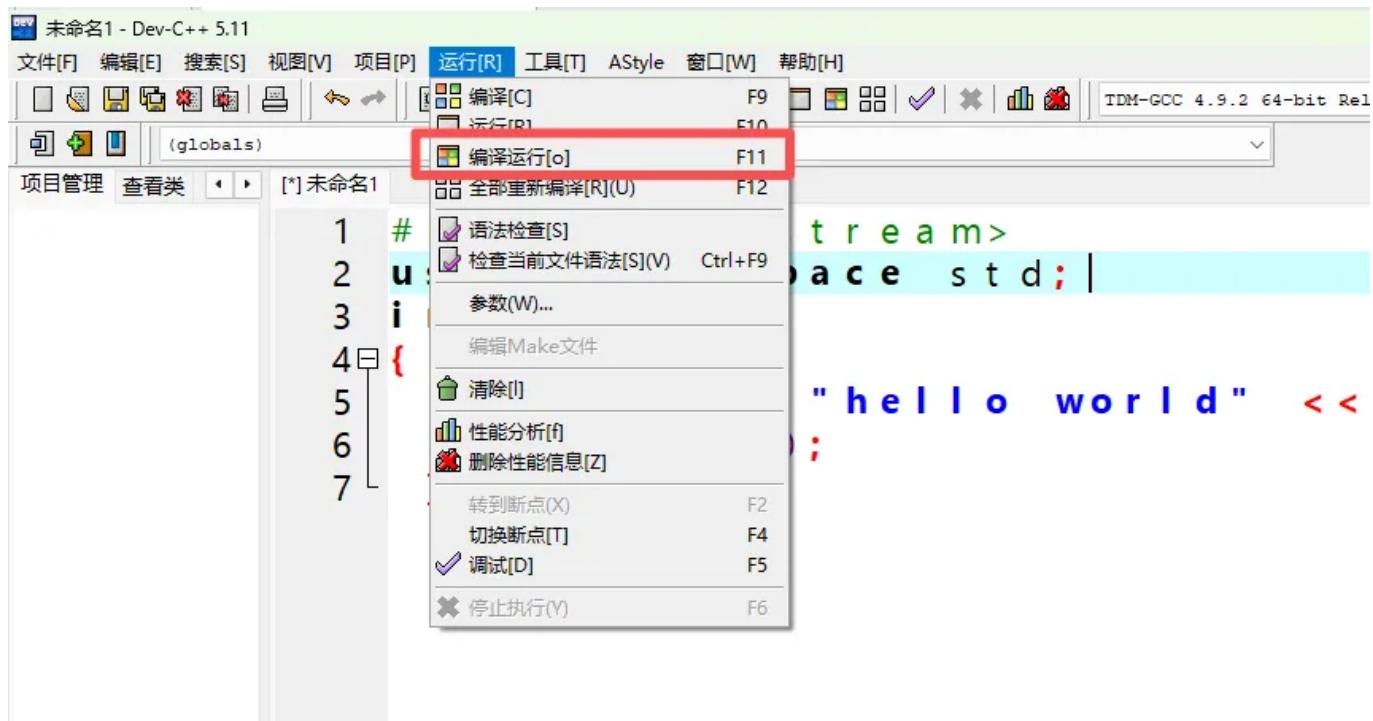
选择文件里面的新建然后有一个源代码



这里用一个很抽象的字体写了代码

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

左上角选择编译运行，会有一个保存的过程，保存到你写代码的位置。



最后代码就跑起来了

这几个就是常用的IDE了。

调试

Cliom

这是一段求1到n和的代码

```

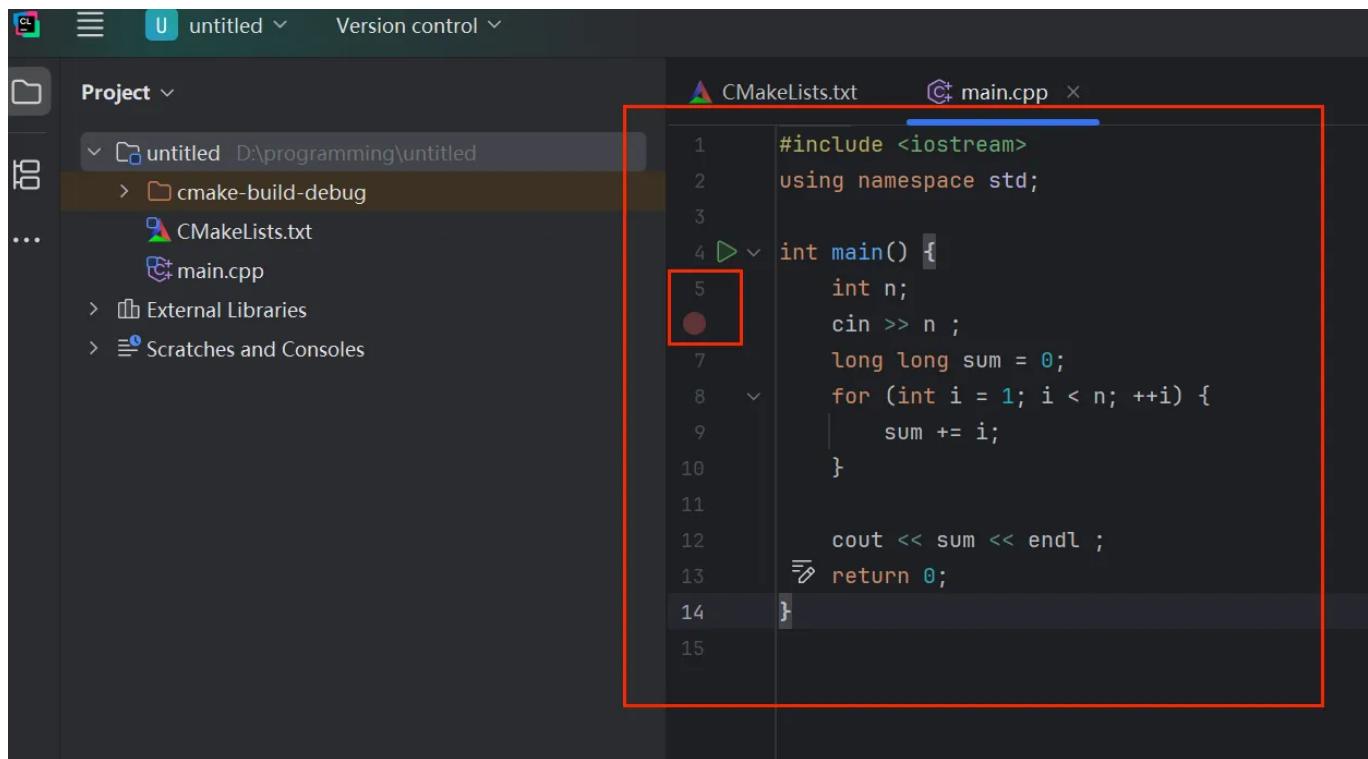
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n ;
    long long sum = 0;
    for (int i = 1; i < n; ++i) {
        sum += i;
    }
    cout << sum << endl ;
    return 0;
}

```

输入5 输出10，不知道哪里有问题

鼠标放在行号左侧，点击设置一个断点，就是指从哪里开始调试

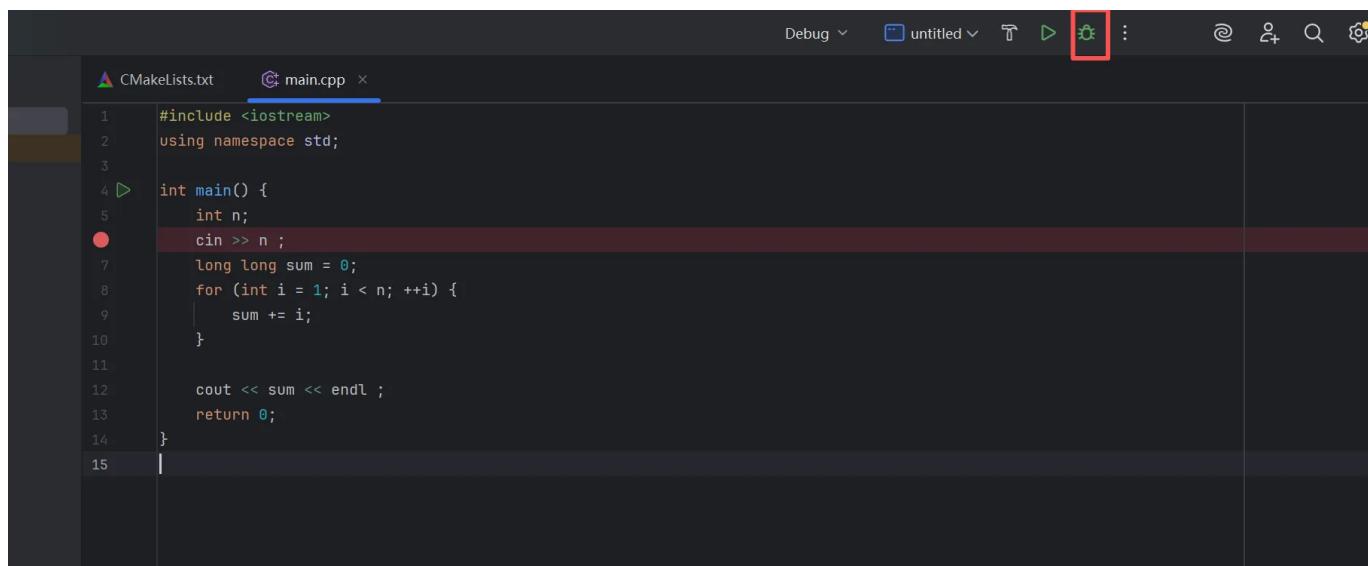


```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int n;
6     cin >> n ;
7     long long sum = 0;
8     for (int i = 1; i < n; ++i) {
9         sum += i;
10    }
11
12    cout << sum << endl ;
13    return 0;
14 }
15

```

然后点那个小虫子一样的debug,为什么设置成小虫子样子，可以参考一下bug的来源历史

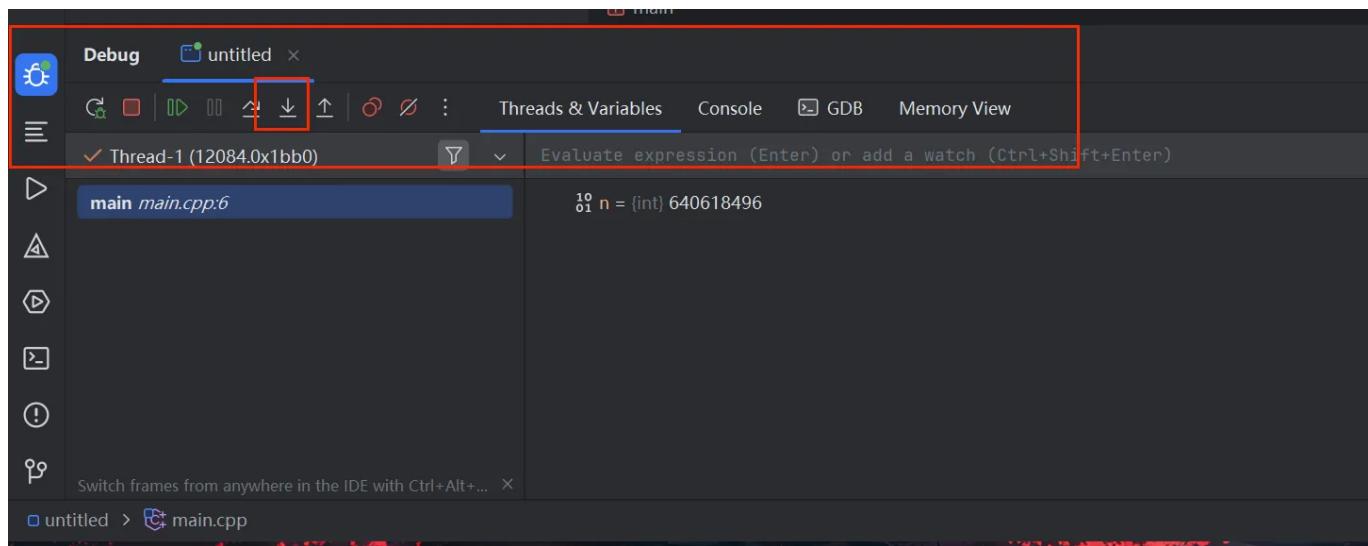


```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int n;
6     cin >> n ;
7     long long sum = 0;
8     for (int i = 1; i < n; ++i) {
9         sum += i;
10    }
11
12    cout << sum << endl ;
13    return 0;
14 }
15

```

初学者需要知道四个东西我，向下箭头是下一步，向下箭头左边是跳过整步，variables显示每个变量实时的值，console是对接输入输出的。遇到要输入东西的地方就在console，需要要cin，cout的建议跳过整步，防止进入一些奇奇怪怪的东西，然后遇到循环可以一步一步走，每步监视variables里的东西。



由于一些特殊原因调试的时候，可以用一个具体值，就不要用cin了，因为他的调试会把cin的底层也翻出来 比如我拿n = 5.

The screenshot shows the CLion IDE with the code editor displaying 'main.cpp'. A breakpoint is set at line 8. The code defines a function 'main' that calculates the sum of integers from 1 to n. The variable 'n' is set to 5. The debugger window at the bottom shows the current state: n = {int} 5, sum = {long long} 3, and i = {int} 3. The 'Threads & Variables' tab is selected.

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int n = 5 ;    n: 5
6     long long sum = 0;    sum: 3
7     for (int i = 1; i < n; ++i) {    n: 5    i: 3
8         sum += i;    i: 3
9     }
10
11     cout << sum << endl ;
12     return 0;
13 }
14

```

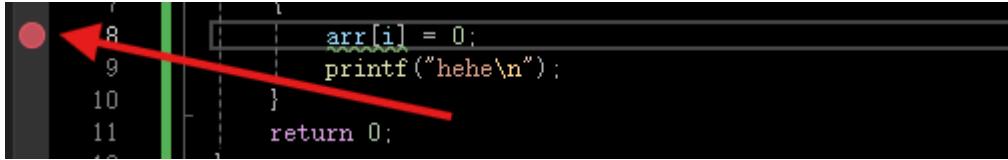
[有关CLion从安装到调试](#)这篇博客我感觉是十分优质的，但有很多计算机的术语，重点看调试部分就可以了

其他的也差不多，只要找到调试这个按钮都大同小异，这个过程就是这样，**不要直接把自己的错误代码扔给测试平台**，说来惭愧，开始写java的时候才发现调试和测试驱动开发的重要性，我觉得这个点gradescope平台就

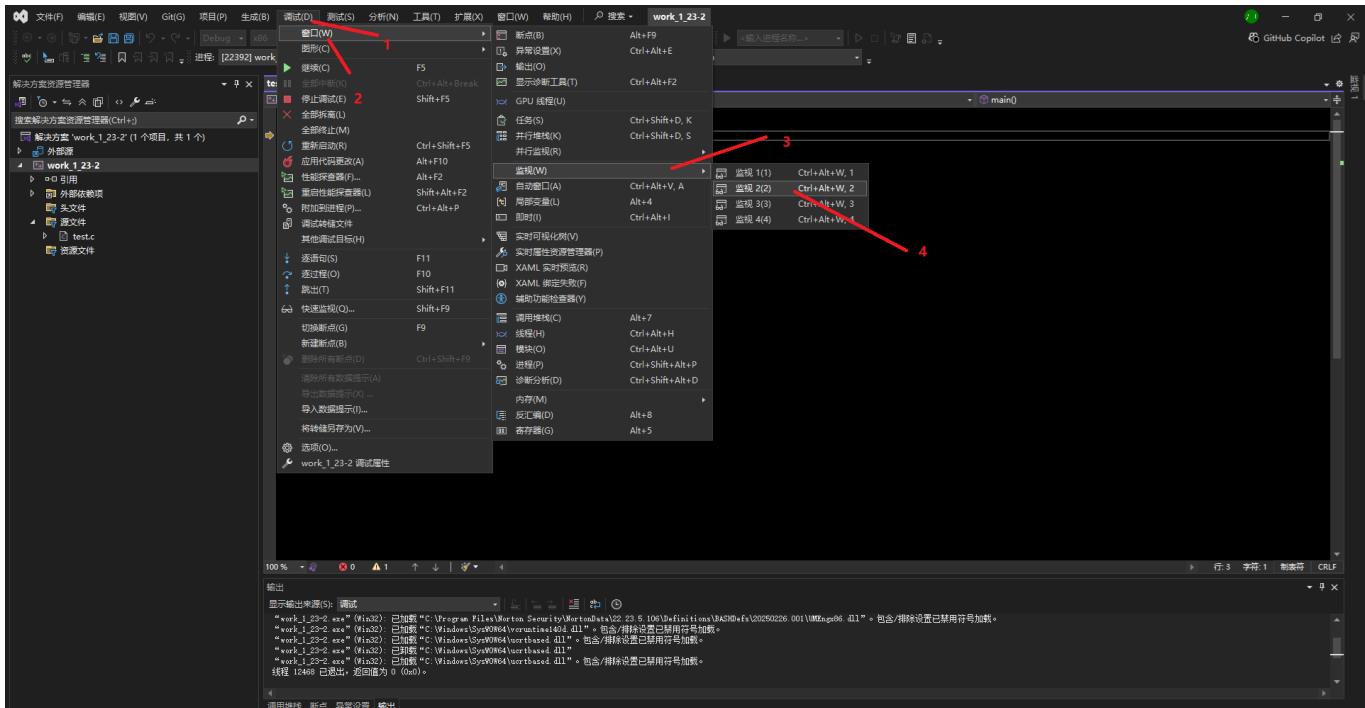
做的特好，再交就要等好几个小时，必须要自己学会调试

VS

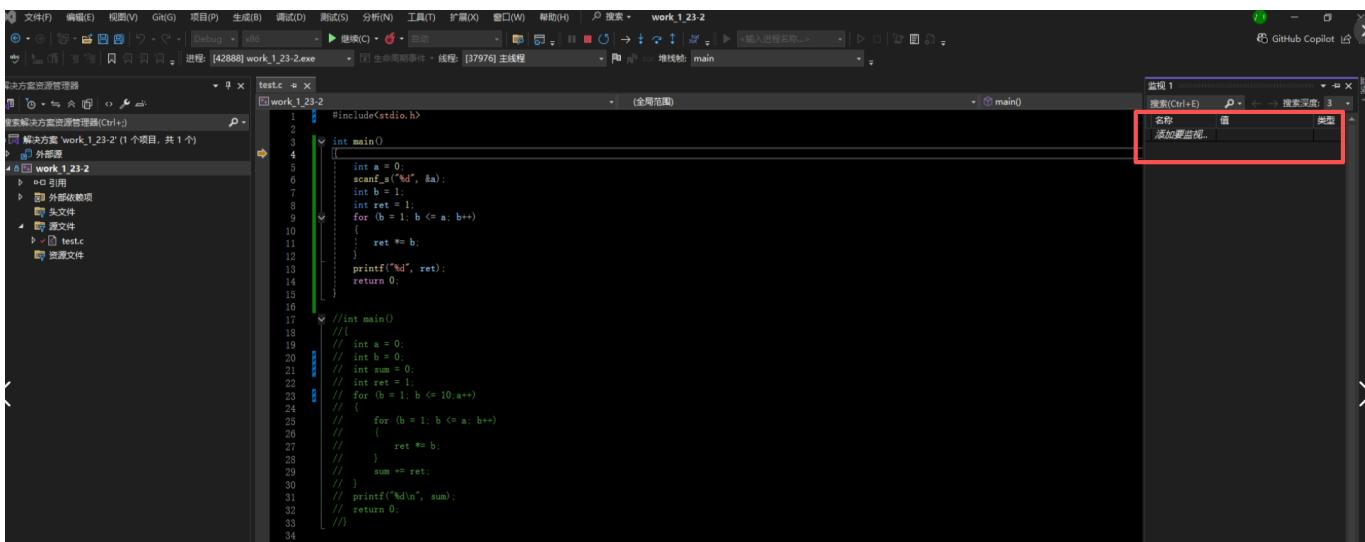
vs的也需要先设置断点



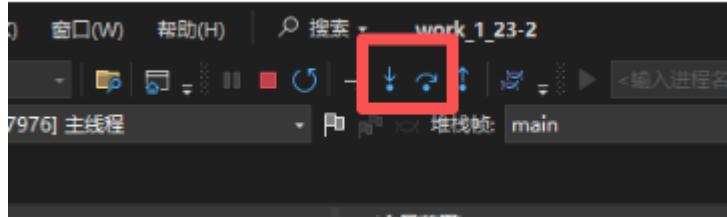
先按F10，然后找要监视的变量(Clion里会直接帮我们找全部的) 监视：操作为 调试 窗口 监视



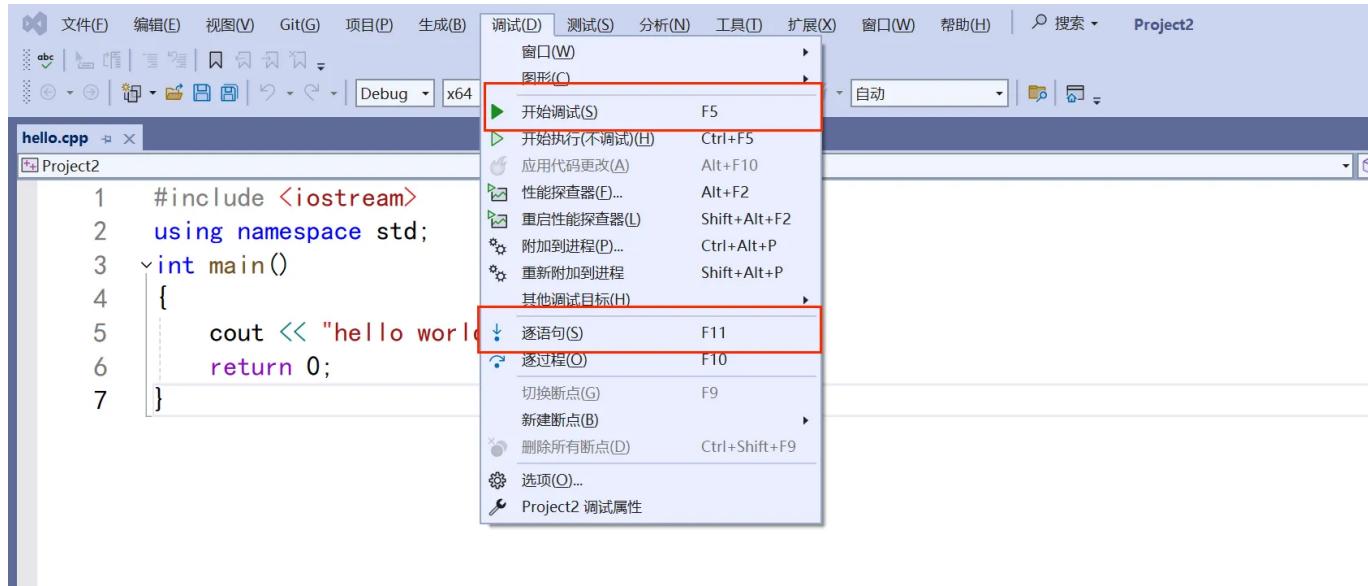
然后开始找到监视，输入你要监视的变量



vs里面是分为逐语句和逐过程，下图左面是逐语句，右面是逐过程，调试方法就同Clion了。

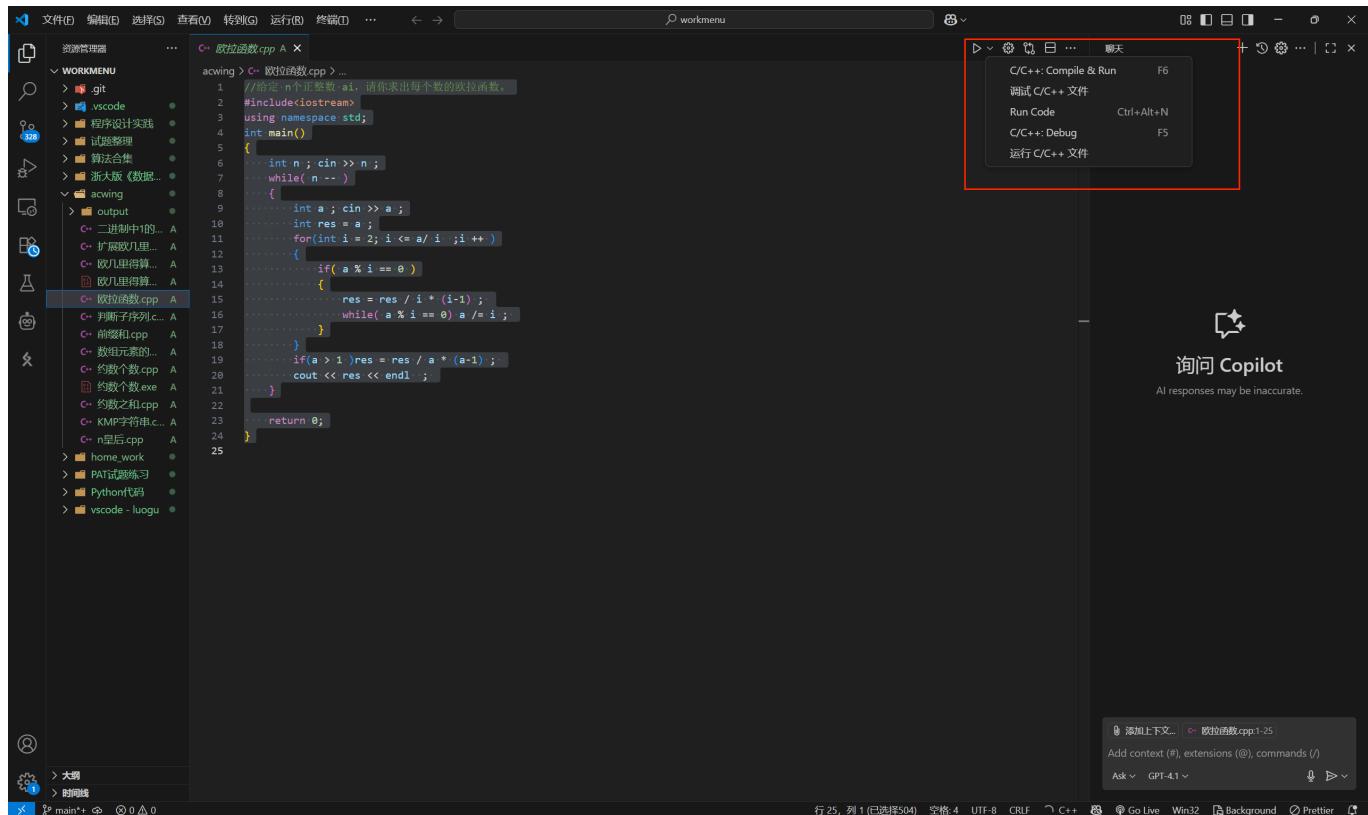


也可以直接点击开始调试或者逐语句，现在的IDE都很智能，都会知道你要干什么，所以你要干的就是学会Debug。



vscode

如果你看了上面的博客，配好了环境，大概率是可以直接按调试按钮直接调试的，就会有这样一个选择按钮，然后设置断点，点击debug就可以了，和其他IDE大同小异。



如果你选择了命令行，vscode的调试需要用到GDB，这就交给你的计算机导论实验课吧，当然网上找个教程，也可以走一下便捷的路途，但是我十分推荐学一下命令行的一些指令，这是一个十分痛苦的过程，当时学[Missing Lecture](#)这节课，崩溃到让我想放弃计算机，命令行的强大真的是我无法想的。所以计算机科学导论真的不是水课，只是我没学到东西。

dev

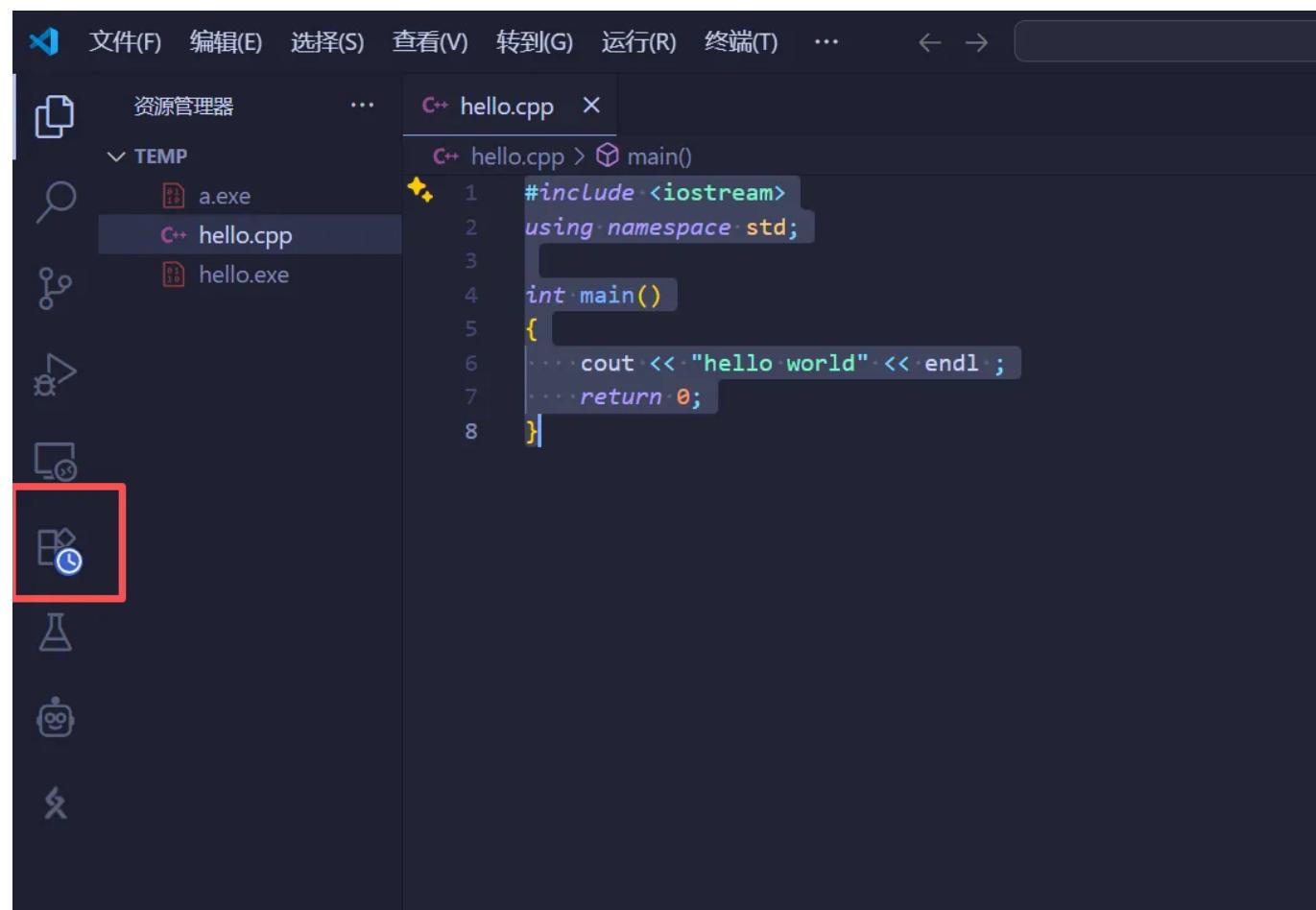
dev的调试也很简单，掌握了前面的就差不多了。

[dev的从0到1开发可以看这篇文档，绝对不是助教太嫌弃devc++，所以卸载了。](#)

插件的使用

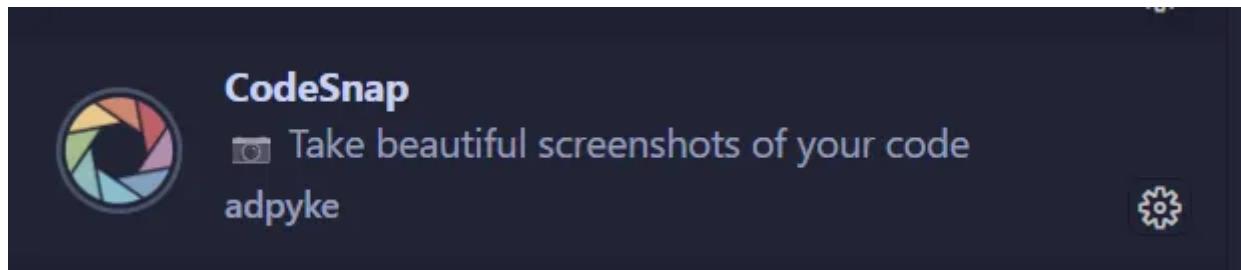
这里主要介绍的是vscode中好用的插件。

找到这个图标

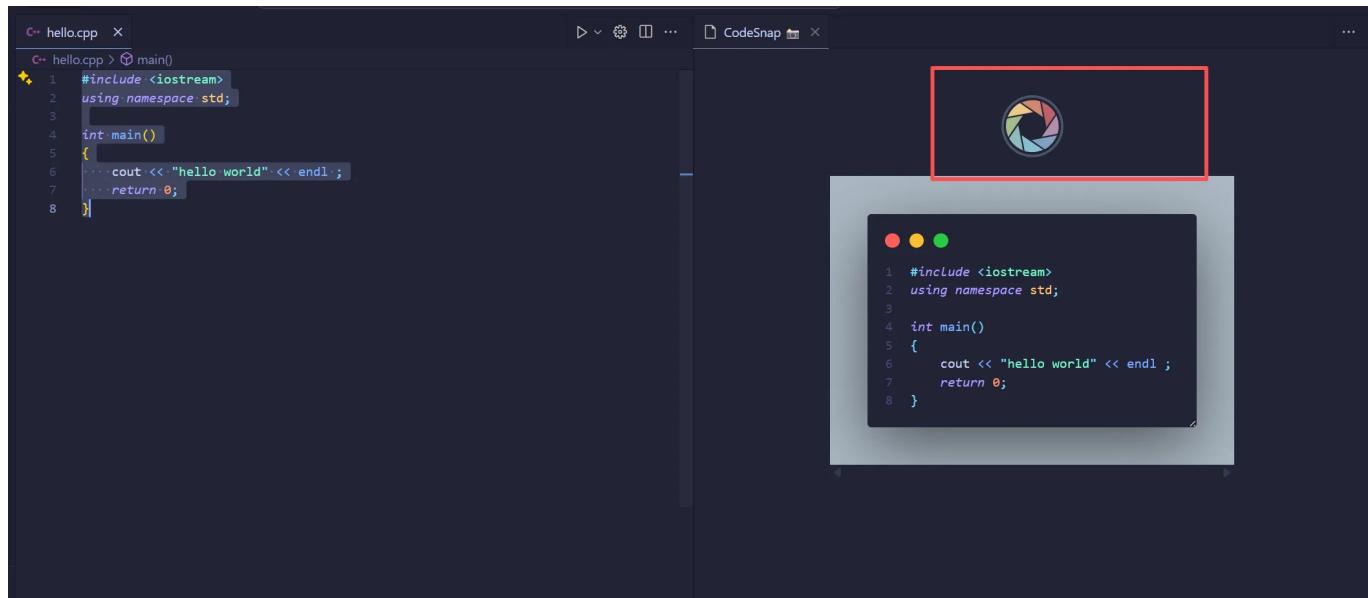


CodeSnap

代码截图软件，选中你要截图的代码右键选择CodeSnap

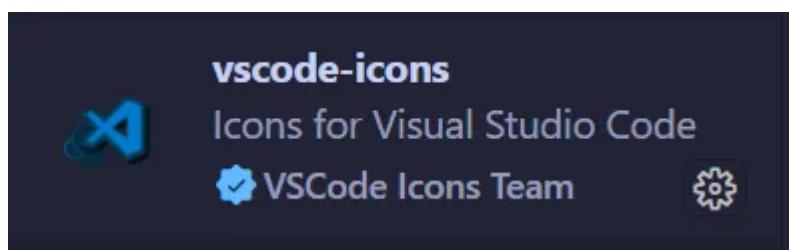


点一下这个圈圈就可以了

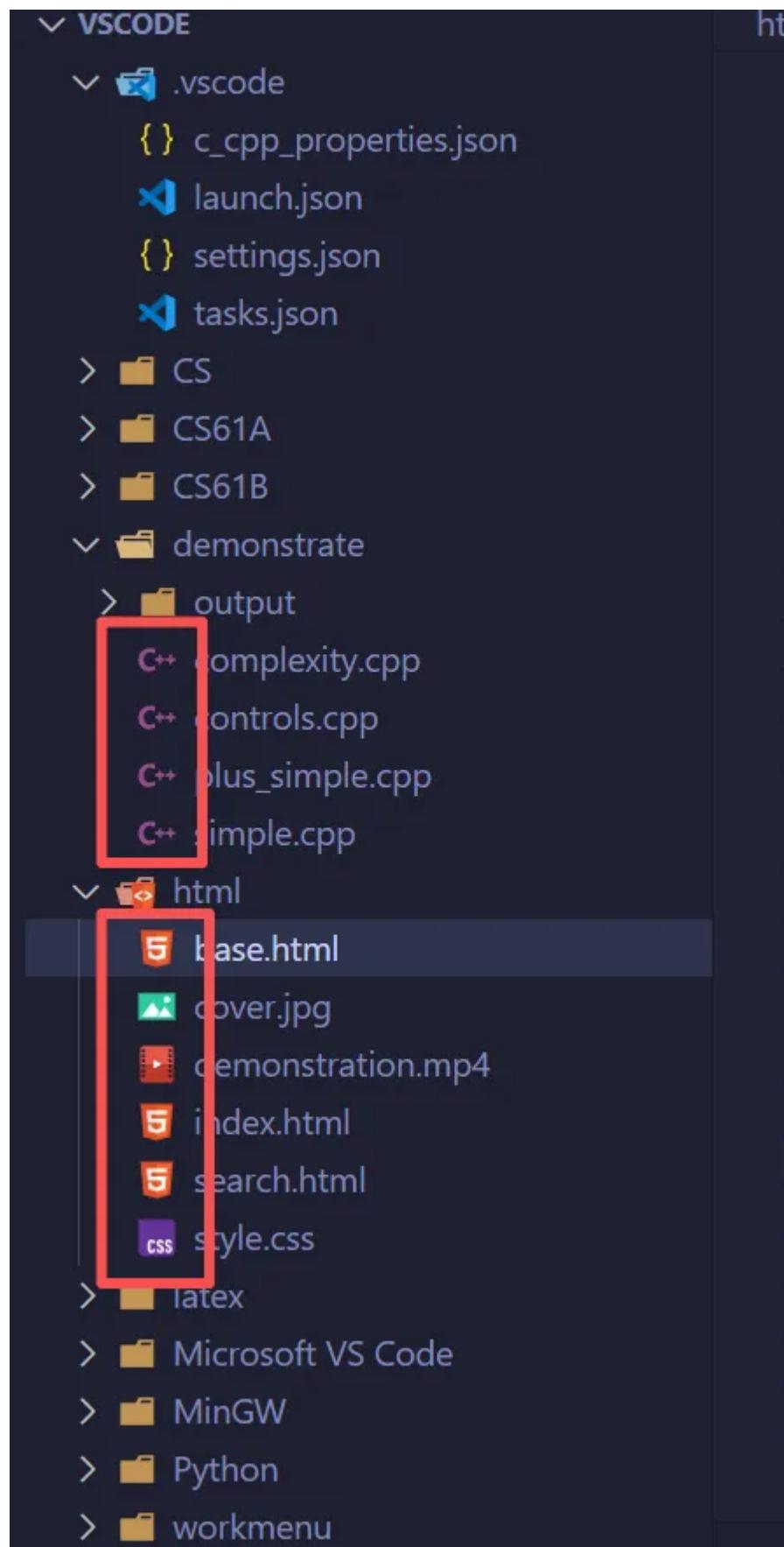


vscode-icons

更好的图标显示



纯属长得好看一点了



Chinese Language

我知道你讨厌英文。



C/c++ Extension Pack

一个c++的扩展包能够帮你去实现代码扩展功能



代码高亮以及这种tab补全好像这个包做的都挺好的

A screenshot of a code editor displaying C++ code. The code includes declarations for a double variable 'price', a main function, and some custom functions 'JF_cat' and 'cat'. A cursor is positioned inside a brace in the main function's body. A tooltip or dropdown menu is open at the cursor position, showing completion suggestions for the word 'string'. The suggestions include various string-related functions and headers: 'string', 'string_literals', 'string_view', 'string_view_literals', 'stringbuf', 'stringstream', 'to_string', '_STRING_CONVERSIONS_H', '_STRINGFWD_H', 'basic_string', 'basic_string_view', and 'basic_string...' (with a template parameter). The code editor has a dark theme.

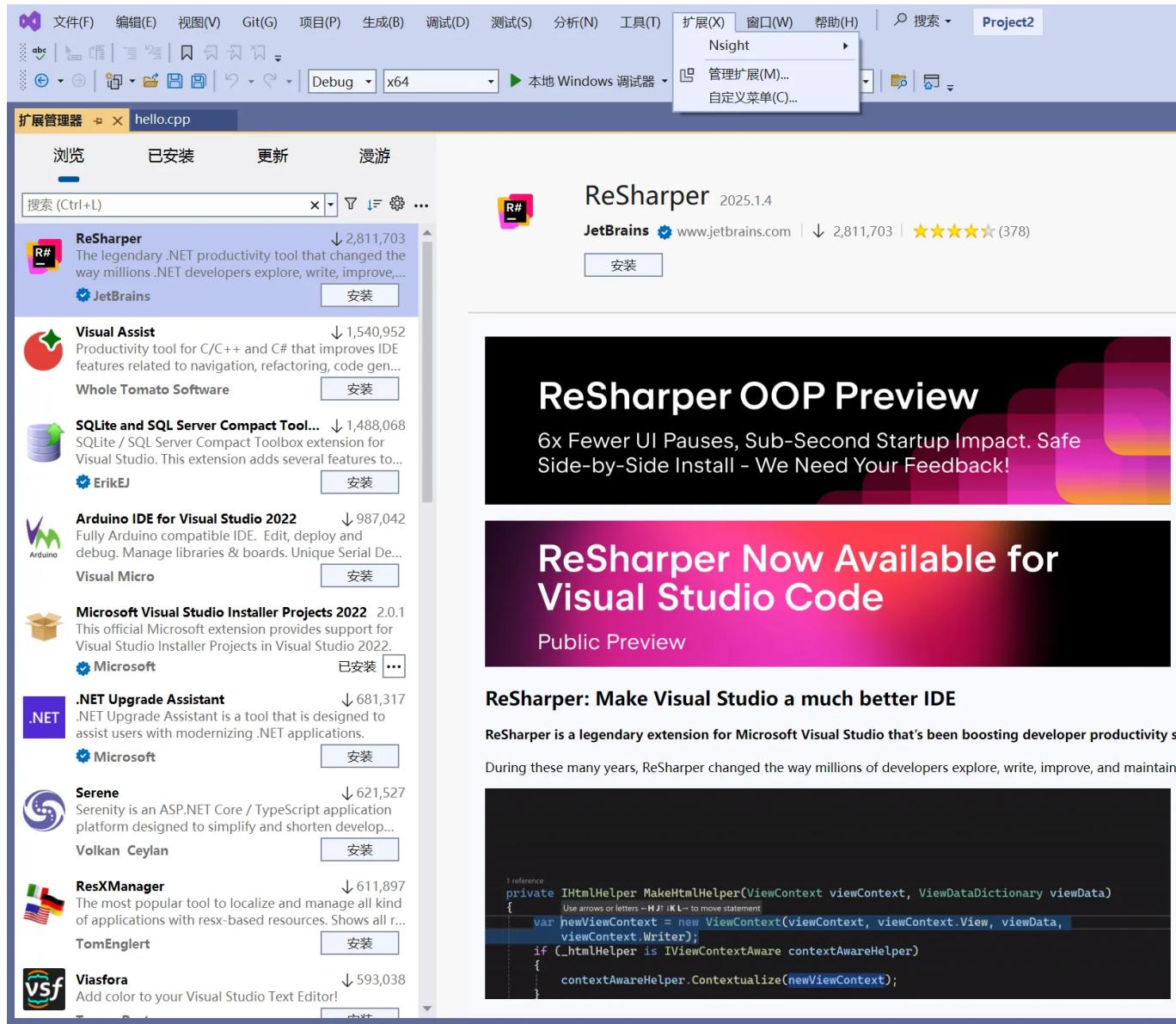
至于有的扩展包可能是用java的，可能是latex的，可能是jupyter的，这些暂时用不到c++里面实用的就那几个了。

对于CLion来说，双击shift，搜索plugins

The screenshot shows the CLion IDE interface with the 'istream.tcc' file open in the editor. A context menu is open over the code, specifically the 'Plugins' section of the 'Welcome Screen' settings. The 'Plugins' tab is selected, displaying a list of available plugins with their current status (ON or OFF). Plugins listed include: Check Suggested Plugins, Activity Monitor..., Auto-Update Plugins, Plugins: All Hallow's Eve Color Scheme, Plugins: Angular, Plugins: Backup and Sync, Plugins: Blackboard Color Scheme, Plugins: C/C++ Package Manager, Plugins: CLion Boost.Test, Plugins: CLion C/C++ File Runner, Plugins: CLion CTest, Plugins: CLion Catch, Plugins: CLion Doctest, and Plugins: CLion Google Testing and Mocking Framework. Most plugins are currently set to 'ON'. At the bottom of the dialog, there are buttons for 'Manage installed plugins and download new ones from repository' and 'Open In Right Split'.

```

40
41     namespace std __GLIBCXX_VISIBILITY(default)
42     {
43         _GLIBCXX_BEGIN_NAMESPACE_VERSION
44
45         template<typename _CharT, typename _Traits>
46         basic_istream<_CharT, _Traits>;
47         sentry(basic_istream<_CharT, _Traits>* __sentry)
48         {
49             ios_base::iostate __err = ios_base::goodbit;
50             if (__in.good())
51             {
52                 __try
53                 {
54                     if (__in.tie())
55                         __in.tie()->flush();
56                     if (!__noskip && __pool->isopen())
57                     {
58                         const __int_type__ __eof =
59                             __streambuf_type* __sb =
60                             __int_type __c = __sb->sg
61
62                         const __ctype_type& __ct
63                         while (!traits_type::eq_int_t(
64                             && __ct.is(m, ctype_base::lower))
65                             | traits_type::isalpha(m))
66                             __c = __sb->snextc();
67
68                         // __GLIBCXX_RESOLVE_LIB_D
69                         // 195. Should basic_istr
70                         // set eofbit?
71                         if (traits_type::eq_int_t(
72                             __err |= ios_base::eofbit));
73                     }
74                 }
75             } catch(__cxxabiv1::__forced_
76             {
77                 in. M setstate( state:ios_base::badbit);
78             }
79         }
80     }
81
82     __GLIBCXX_VISIBILITY_HIDDEN void
83     basic_istream<_CharT, _Traits>::sentry::sentry()
84     {
85         ios_base::iostate __err = ios_base::goodbit;
86
87         if (!__noskip && __pool->isopen())
88         {
89             const __ctype_type& __ct
90             while (!traits_type::eq_int_t(
91                 && __ct.is(m, ctype_base::lower))
92                 | traits_type::isalpha(m))
93                 __c = __sb->snextc();
94
95             // __GLIBCXX_RESOLVE_LIB_D
96             // 195. Should basic_istr
97             // set eofbit?
98             if (traits_type::eq_int_t(
99                 __err |= ios_base::eofbit));
100         }
101     }
102
103     __GLIBCXX_VISIBILITY_HIDDEN void
104     basic_istream<_CharT, _Traits>::sentry::operator()()
105     {
106         ios_base::iostate __err = ios_base::goodbit;
107
108         if (!__noskip && __pool->isopen())
109         {
110             const __ctype_type& __ct
111             while (!traits_type::eq_int_t(
112                 && __ct.is(m, ctype_base::lower))
113                 | traits_type::isalpha(m))
114                 __c = __sb->snextc();
115
116             // __GLIBCXX_RESOLVE_LIB_D
117             // 195. Should basic_istr
118             // set eofbit?
119             if (traits_type::eq_int_t(
120                 __err |= ios_base::eofbit));
121         }
122     }
123
124     __GLIBCXX_VISIBILITY_HIDDEN void
125     basic_istream<_CharT, _Traits>::sentry::operator()()
126     {
127         ios_base::iostate __err = ios_base::goodbit;
128
129         if (!__noskip && __pool->isopen())
130         {
131             const __ctype_type& __ct
132             while (!traits_type::eq_int_t(
133                 && __ct.is(m, ctype_base::lower))
134                 | traits_type::isalpha(m))
135                 __c = __sb->snextc();
136
137             // __GLIBCXX_RESOLVE_LIB_D
138             // 195. Should basic_istr
139             // set eofbit?
140             if (traits_type::eq_int_t(
141                 __err |= ios_base::eofbit));
142         }
143     }
144
145     __GLIBCXX_VISIBILITY_HIDDEN void
146     basic_istream<_CharT, _Traits>::sentry::operator()()
147     {
148         ios_base::iostate __err = ios_base::goodbit;
149
150         if (!__noskip && __pool->isopen())
151         {
152             const __ctype_type& __ct
153             while (!traits_type::eq_int_t(
154                 && __ct.is(m, ctype_base::lower))
155                 | traits_type::isalpha(m))
156                 __c = __sb->snextc();
157
158             // __GLIBCXX_RESOLVE_LIB_D
159             // 195. Should basic_istr
160             // set eofbit?
161             if (traits_type::eq_int_t(
162                 __err |= ios_base::eofbit));
163         }
164     }
165
166     __GLIBCXX_VISIBILITY_HIDDEN void
167     basic_istream<_CharT, _Traits>::sentry::operator()()
168     {
169         ios_base::iostate __err = ios_base::goodbit;
170
171         if (!__noskip && __pool->isopen())
172         {
173             const __ctype_type& __ct
174             while (!traits_type::eq_int_t(
175                 && __ct.is(m, ctype_base::lower))
176                 | traits_type::isalpha(m))
177                 __c = __sb->snextc();
178
179             // __GLIBCXX_RESOLVE_LIB_D
180             // 195. Should basic_istr
181             // set eofbit?
182             if (traits_type::eq_int_t(
183                 __err |= ios_base::eofbit));
184         }
185     }
186
187     __GLIBCXX_VISIBILITY_HIDDEN void
188     basic_istream<_CharT, _Traits>::sentry::operator()()
189     {
190         ios_base::iostate __err = ios_base::goodbit;
191
192         if (!__noskip && __pool->isopen())
193         {
194             const __ctype_type& __ct
195             while (!traits_type::eq_int_t(
196                 && __ct.is(m, ctype_base::lower))
197                 | traits_type::isalpha(m))
198                 __c = __sb->snextc();
199
200             // __GLIBCXX_RESOLVE_LIB_D
201             // 195. Should basic_istr
202             // set eofbit?
203             if (traits_type::eq_int_t(
204                 __err |= ios_base::eofbit));
205         }
206     }
207
208     __GLIBCXX_VISIBILITY_HIDDEN void
209     basic_istream<_CharT, _Traits>::sentry::operator()()
210     {
211         ios_base::iostate __err = ios_base::goodbit;
212
213         if (!__noskip && __pool->isopen())
214         {
215             const __ctype_type& __ct
216             while (!traits_type::eq_int_t(
217                 && __ct.is(m, ctype_base::lower))
218                 | traits_type::isalpha(m))
219                 __c = __sb->snextc();
220
221             // __GLIBCXX_RESOLVE_LIB_D
222             // 195. Should basic_istr
223             // set eofbit?
224             if (traits_type::eq_int_t(
225                 __err |= ios_base::eofbit));
226         }
227     }
228
229     __GLIBCXX_VISIBILITY_HIDDEN void
230     basic_istream<_CharT, _Traits>::sentry::operator()()
231     {
232         ios_base::iostate __err = ios_base::goodbit;
233
234         if (!__noskip && __pool->isopen())
235         {
236             const __ctype_type& __ct
237             while (!traits_type::eq_int_t(
238                 && __ct.is(m, ctype_base::lower))
239                 | traits_type::isalpha(m))
240                 __c = __sb->snextc();
241
242             // __GLIBCXX_RESOLVE_LIB_D
243             // 195. Should basic_istr
244             // set eofbit?
245             if (traits_type::eq_int_t(
246                 __err |= ios_base::eofbit));
247         }
248     }
249
250     __GLIBCXX_VISIBILITY_HIDDEN void
251     basic_istream<_CharT, _Traits>::sentry::operator()()
252     {
253         ios_base::iostate __err = ios_base::goodbit;
254
255         if (!__noskip && __pool->isopen())
256         {
257             const __ctype_type& __ct
258             while (!traits_type::eq_int_t(
259                 && __ct.is(m, ctype_base::lower))
260                 | traits_type::isalpha(m))
261                 __c = __sb->snextc();
262
263             // __GLIBCXX_RESOLVE_LIB_D
264             // 195. Should basic_istr
265             // set eofbit?
266             if (traits_type::eq_int_t(
267                 __err |= ios_base::eofbit));
268         }
269     }
270
271     __GLIBCXX_VISIBILITY_HIDDEN void
272     basic_istream<_CharT, _Traits>::sentry::operator()()
273     {
274         ios_base::iostate __err = ios_base::goodbit;
275
276         if (!__noskip && __pool->isopen())
277         {
278             const __ctype_type& __ct
279             while (!traits_type::eq_int_t(
280                 && __ct.is(m, ctype_base::lower))
281                 | traits_type::isalpha(m))
282                 __c = __sb->snextc();
283
284             // __GLIBCXX_RESOLVE_LIB_D
285             // 195. Should basic_istr
286             // set eofbit?
287             if (traits_type::eq_int_t(
288                 __err |= ios_base::eofbit));
289         }
290     }
291
292     __GLIBCXX_VISIBILITY_HIDDEN void
293     basic_istream<_CharT, _Traits>::sentry::operator()()
294     {
295         ios_base::iostate __err = ios_base::goodbit;
296
297         if (!__noskip && __pool->isopen())
298         {
299             const __ctype_type& __ct
300             while (!traits_type::eq_int_t(
301                 && __ct.is(m, ctype_base::lower))
302                 | traits_type::isalpha(m))
303                 __c = __sb->snextc();
304
305             // __GLIBCXX_RESOLVE_LIB_D
306             // 195. Should basic_istr
307             // set eofbit?
308             if (traits_type::eq_int_t(
309                 __err |= ios_base::eofbit));
310         }
311     }
312
313     __GLIBCXX_VISIBILITY_HIDDEN void
314     basic_istream<_CharT, _Traits>::sentry::operator()()
315     {
316         ios_base::iostate __err = ios_base::goodbit;
317
318         if (!__noskip && __pool->isopen())
319         {
320             const __ctype_type& __ct
321             while (!traits_type::eq_int_t(
322                 && __ct.is(m, ctype_base::lower))
323                 | traits_type::isalpha(m))
324                 __c = __sb->snextc();
325
326             // __GLIBCXX_RESOLVE_LIB_D
327             // 195. Should basic_istr
328             // set eofbit?
329             if (traits_type::eq_int_t(
330                 __err |= ios_base::eofbit));
331         }
332     }
333
334     __GLIBCXX_VISIBILITY_HIDDEN void
335     basic_istream<_CharT, _Traits>::sentry::operator()()
336     {
337         ios_base::iostate __err = ios_base::goodbit;
338
339         if (!__noskip && __pool->isopen())
340         {
341             const __ctype_type& __ct
342             while (!traits_type::eq_int_t(
343                 && __ct.is(m, ctype_base::lower))
344                 | traits_type::isalpha(m))
345                 __c = __sb->snextc();
346
347             // __GLIBCXX_RESOLVE_LIB_D
348             // 195. Should basic_istr
349             // set eofbit?
350             if (traits_type::eq_int_t(
351                 __err |= ios_base::eofbit));
352         }
353     }
354
355     __GLIBCXX_VISIBILITY_HIDDEN void
356     basic_istream<_CharT, _Traits>::sentry::operator()()
357     {
358         ios_base::iostate __err = ios_base::goodbit;
359
360         if (!__noskip && __pool->isopen())
361         {
362             const __ctype_type& __ct
363             while (!traits_type::eq_int_t(
364                 && __ct.is(m, ctype_base::lower))
365                 | traits_type::isalpha(m))
366                 __c = __sb->snextc();
367
368             // __GLIBCXX_RESOLVE_LIB_D
369             // 195. Should basic_istr
370             // set eofbit?
371             if (traits_type::eq_int_t(
372                 __err |= ios_base::eofbit));
373         }
374     }
375
376     __GLIBCXX_VISIBILITY_HIDDEN void
377     basic_istream<_CharT, _Traits>::sentry::operator()()
378     {
379         ios_base::iostate __err = ios_base::goodbit;
380
381         if (!__noskip && __pool->isopen())
382         {
383             const __ctype_type& __ct
384             while (!traits_type::eq_int_t(
385                 && __ct.is(m, ctype_base::lower))
386                 | traits_type::isalpha(m))
387                 __c = __sb->snextc();
388
389             // __GLIBCXX_RESOLVE_LIB_D
390             // 195. Should basic_istr
391             // set eofbit?
392             if (traits_type::eq_int_t(
393                 __err |= ios_base::eofbit));
394         }
395     }
396
397     __GLIBCXX_VISIBILITY_HIDDEN void
398     basic_istream<_CharT, _Traits>::sentry::operator()()
399     {
400         ios_base::iostate __err = ios_base::goodbit;
401
402         if (!__noskip && __pool->isopen())
403         {
404             const __ctype_type& __ct
405             while (!traits_type::eq_int_t(
406                 && __ct.is(m, ctype_base::lower))
407                 | traits_type::isalpha(m))
408                 __c = __sb->snextc();
409
410             // __GLIBCXX_RESOLVE_LIB_D
411             // 195. Should basic_istr
412             // set eofbit?
413             if (traits_type::eq_int_t(
414                 __err |= ios_base::eofbit));
415         }
416     }
417
418     __GLIBCXX_VISIBILITY_HIDDEN void
419     basic_istream<_CharT, _Traits>::sentry::operator()()
420     {
421         ios_base::iostate __err = ios_base::goodbit;
422
423         if (!__noskip && __pool->isopen())
424         {
425             const __ctype_type& __ct
426             while (!traits_type::eq_int_t(
427                 && __ct.is(m, ctype_base::lower))
428                 | traits_type::isalpha(m))
429                 __c = __sb->snextc();
430
431             // __GLIBCXX_RESOLVE_LIB_D
432             // 195. Should basic_istr
433             // set eofbit?
434             if (traits_type::eq_int_t(
435                 __err |= ios_base::eofbit));
436         }
437     }
438
439     __GLIBCXX_VISIBILITY_HIDDEN void
440     basic_istream<_CharT, _Traits>::sentry::operator()()
441     {
442         ios_base::iostate __err = ios_base::goodbit;
443
444         if (!__noskip && __pool->isopen())
445         {
446             const __ctype_type& __ct
447             while (!traits_type::eq_int_t(
448                 && __ct.is(m, ctype_base::lower))
449                 | traits_type::isalpha(m))
450                 __c = __sb->snextc();
451
452             // __GLIBCXX_RESOLVE_LIB_D
453             // 195. Should basic_istr
454             // set eofbit?
455             if (traits_type::eq_int_t(
456                 __err |= ios_base::eofbit));
457         }
458     }
459
460     __GLIBCXX_VISIBILITY_HIDDEN void
461     basic_istream<_CharT, _Traits>::sentry::operator()()
462     {
463         ios_base::iostate __err = ios_base::goodbit;
464
465         if (!__noskip && __pool->isopen())
466         {
467             const __ctype_type& __ct
468             while (!traits_type::eq_int_t(
469                 && __ct.is(m, ctype_base::lower))
470                 | traits_type::isalpha(m))
471                 __c = __sb->snextc();
472
473             // __GLIBCXX_RESOLVE_LIB_D
474             // 195. Should basic_istr
475             // set eofbit?
476             if (traits_type::eq_int_t(
477                 __err |= ios_base::eofbit));
478         }
479     }
480
481     __GLIBCXX_VISIBILITY_HIDDEN void
482     basic_istream<_CharT, _Traits>::sentry::operator()()
483     {
484         ios_base::iostate __err = ios_base::goodbit;
485
486         if (!__noskip && __pool->isopen())
487         {
488             const __ctype_type& __ct
489             while (!traits_type::eq_int_t(
490                 && __ct.is(m, ctype_base::lower))
491                 | traits_type::isalpha(m))
492                 __c = __sb->snextc();
493
494             // __GLIBCXX_RESOLVE_LIB_D
495             // 195. Should basic_istr
496             // set eofbit?
497             if (traits_type::eq_int_t(
498                 __err |= ios_base::eofbit));
499         }
500     }
501
502     __GLIBCXX_VISIBILITY_HIDDEN void
503     basic_istream<_CharT, _Traits>::sentry::operator()()
504     {
505         ios_base::iostate __err = ios_base::goodbit;
506
507         if (!__noskip && __pool->isopen())
508         {
509             const __ctype_type& __ct
510             while (!traits_type::eq_int_t(
511                 && __ct.is(m, ctype_base::lower))
512                 | traits_type::isalpha(m))
513                 __c = __sb->snextc();
514
515             // __GLIBCXX_RESOLVE_LIB_D
516             // 195. Should basic_istr
517             // set eofbit?
518             if (traits_type::eq_int_t(
519                 __err |= ios_base::eofbit));
520         }
521     }
522
523     __GLIBCXX_VISIBILITY_HIDDEN void
524     basic_istream<_CharT, _Traits>::sentry::operator()()
525     {
526         ios_base::iostate __err = ios_base::goodbit;
527
528         if (!__noskip && __pool->isopen())
529         {
530             const __ctype_type& __ct
531             while (!traits_type::eq_int_t(
532                 && __ct.is(m, ctype_base::lower))
533                 | traits_type::isalpha(m))
534                 __c = __sb->snextc();
535
536             // __GLIBCXX_RESOLVE_LIB_D
537             // 195. Should basic_istr
538             // set eofbit?
539             if (traits_type::eq_int_t(
540                 __err |= ios_base::eofbit));
541         }
542     }
543
544     __GLIBCXX_VISIBILITY_HIDDEN void
545     basic_istream<_CharT, _Traits>::sentry::operator()()
546     {
547         ios_base::iostate __err = ios_base::goodbit;
548
549         if (!__noskip && __pool->isopen())
550         {
551             const __ctype_type& __ct
552             while (!traits_type::eq_int_t(
553                 && __ct.is(m, ctype_base::lower))
554                 | traits_type::isalpha(m))
555                 __c = __sb->snextc();
556
557             // __GLIBCXX_RESOLVE_LIB_D
558             // 195. Should basic_istr
559             // set eofbit?
560             if (traits_type::eq_int_t(
561                 __err |= ios_base::eofbit));
562         }
563     }
564
565     __GLIBCXX_VISIBILITY_HIDDEN void
566     basic_istream<_CharT, _Traits>::sentry::operator()()
567     {
568         ios_base::iostate __err = ios_base::goodbit;
569
570         if (!__noskip && __pool->isopen())
571         {
572             const __ctype_type& __ct
573             while (!traits_type::eq_int_t(
574                 && __ct.is(m, ctype_base::lower))
575                 | traits_type::isalpha(m))
576                 __c = __sb->snextc();
577
578             // __GLIBCXX_RESOLVE_LIB_D
579             // 195. Should basic_istr
580             // set eofbit?
581             if (traits_type::eq_int_t(
582                 __err |= ios_base::eofbit));
583         }
584     }
585
586     __GLIBCXX_VISIBILITY_HIDDEN void
587     basic_istream<_CharT, _Traits>::sentry::operator()()
588     {
589         ios_base::iostate __err = ios_base::goodbit;
590
591         if (!__noskip && __pool->isopen())
592         {
593             const __ctype_type& __ct
594             while (!traits_type::eq_int_t(
595                 && __ct.is(m, ctype_base::lower))
596                 | traits_type::isalpha(m))
597                 __c = __sb->snextc();
598
599             // __GLIBCXX_RESOLVE_LIB_D
600             // 195. Should basic_istr
601             // set eofbit?
602             if (traits_type::eq_int_t(
603                 __err |= ios_base::eofbit));
604         }
605     }
606
607     __GLIBCXX_VISIBILITY_HIDDEN void
608     basic_istream<_CharT, _Traits>::sentry::operator()()
609     {
610         ios_base::iostate __err = ios_base::goodbit;
611
612         if (!__noskip && __pool->isopen())
613         {
614             const __ctype_type& __ct
615             while (!traits_type::eq_int_t(
616                 && __ct.is(m, ctype_base::lower))
617                 | traits_type::isalpha(m))
618                 __c = __sb->snextc();
619
620             // __GLIBCXX_RESOLVE_LIB_D
621             // 195. Should basic_istr
622             // set eofbit?
623             if (traits_type::eq_int_t(
624                 __err |= ios_base::eofbit));
625         }
626     }
627
628     __GLIBCXX_VISIBILITY_HIDDEN void
629     basic_istream<_CharT, _Traits>::sentry::operator()()
630     {
631         ios_base::iostate __err = ios_base::goodbit;
632
633         if (!__noskip && __pool->isopen())
634         {
635             const __ctype_type& __ct
636             while (!traits_type::eq_int_t(
637                 && __ct.is(m, ctype_base::lower))
638                 | traits_type::isalpha(m))
639                 __c = __sb->snextc();
640
641             // __GLIBCXX_RESOLVE_LIB_D
642             // 195. Should basic_istr
643             // set eofbit?
644             if (traits_type::eq_int_t(
645                 __err |= ios_base::eofbit));
646         }
647     }
648
649     __GLIBCXX_VISIBILITY_HIDDEN void
650     basic_istream<_CharT, _Traits>::sentry::operator()()
651     {
652         ios_base::iostate __err = ios_base::goodbit;
653
654         if (!__noskip && __pool->isopen())
655         {
656             const __ctype_type& __ct
657             while (!traits_type::eq_int_t(
658                 && __ct.is(m, ctype_base::lower))
659                 | traits_type::isalpha(m))
660                 __c = __sb->snextc();
661
662             // __GLIBCXX_RESOLVE_LIB_D
663             // 195. Should basic_istr
664             // set eofbit?
665             if (traits_type::eq_int_t(
666                 __err |= ios_base::eofbit));
667         }
668     }
669
670     __GLIBCXX_VISIBILITY_HIDDEN void
671     basic_istream<_CharT, _Traits>::sentry::operator()()
672     {
673         ios_base::iostate __err = ios_base::goodbit;
674
675         if (!__noskip && __pool->isopen())
676         {
677             const __ctype_type& __ct
678             while (!traits_type::eq_int_t(
679                 && __ct.is(m, ctype_base::lower))
680                 | traits_type::isalpha(m))
681                 __c = __sb->snextc();
682
683             // __GLIBCXX_RESOLVE_LIB_D
684             // 195. Should basic_istr
685             // set eofbit?
686             if (traits_type::eq_int_t(
687                 __err |= ios_base::eofbit));
688         }
689     }
690
691     __GLIBCXX_VISIBILITY_HIDDEN void
692     basic_istream<_CharT, _Traits>::sentry::operator()()
693     {
694         ios_base::iostate __err = ios_base::goodbit;
695
696         if (!__noskip && __pool->isopen())
697         {
698             const __ctype_type& __ct
699             while (!traits_type::eq_int_t(
700                 && __ct.is(m, ctype_base::lower))
701                 | traits_type::isalpha(m))
702                 __c = __sb->snextc();
703
704             // __GLIBCXX_RESOLVE_LIB_D
705             // 195. Should basic_istr
706             // set eofbit?
707             if (traits_type::eq_int_t(
708                 __err |= ios_base::eofbit));
709         }
710     }
711
712     __GLIBCXX_VISIBILITY_HIDDEN void
713     basic_istream<_CharT, _Traits>::sentry::operator()()
714     {
715         ios_base::iostate __err = ios_base::goodbit;
716
717         if (!__noskip && __pool->isopen())
718         {
719             const __ctype_type& __ct
720             while (!traits_type::eq_int_t(
721                 && __ct.is(m, ctype_base::lower))
722                 | traits_type::isalpha(m))
723                 __c = __sb->snextc();
724
725             // __GLIBCXX_RESOLVE_LIB_D
726             // 195. Should basic_istr
727             // set eofbit?
728             if (traits_type::eq_int_t(
729                 __err |= ios_base::eofbit));
730         }
731     }
732
733     __GLIBCXX_VISIBILITY_HIDDEN void
734     basic_istream<_CharT, _Traits>::sentry::operator()()
735     {
736         ios_base::iostate __err = ios_base::goodbit;
737
738         if (!__noskip && __pool->isopen())
739         {
740             const __ctype_type& __ct
741             while (!traits_type::eq_int_t(
742                 && __ct.is(m, ctype_base::lower))
743                 | traits_type::isalpha(m))
744                 __c = __sb->snextc();
745
746             // __GLIBCXX_RESOLVE_LIB_D
747             // 195. Should basic_istr
748             // set eofbit?
749             if (traits_type::eq_int_t(
750                 __err |= ios_base::eofbit));
751         }
752     }
753
754     __GLIBCXX_VISIBILITY_HIDDEN void
755     basic_istream<_CharT, _Traits>::sentry::operator()()
756     {
757         ios_base::iostate __err = ios_base::goodbit;
758
759         if (!__noskip && __pool->isopen())
760         {
761             const __ctype_type& __ct
762             while (!traits_type::eq_int_t(
763                 && __ct.is(m, ctype_base::lower))
764                 | traits_type::isalpha(m))
765                 __c = __sb->snextc();
766
767             // __GLIBCXX_RESOLVE_LIB_D
768             // 195. Should basic_istr
769             // set eofbit?
770             if (traits_type::eq_int_t(
771                 __err |= ios_base::eofbit));
772         }
773     }
774
775     __GLIBCXX_VISIBILITY_HIDDEN void
776     basic_istream<_CharT, _Traits>::sentry::operator()()
777     {
778         ios_base::iostate __err = ios_base::goodbit;
779
780         if (!__noskip && __pool->isopen())
781         {
782             const __ctype_type& __ct
783             while (!traits_type::eq_int_t(
784                 && __ct.is(m, ctype_base::lower))
785                 | traits_type::isalpha(m))
786                 __c = __sb->snextc();
787
788             // __GLIBCXX_RESOLVE_LIB_D
789             // 195. Should basic_istr
790             // set eofbit?
791             if (traits_type::eq_int_t(
792                 __err |= ios_base::eofbit));
793         }
794     }
795
796     __GLIBCXX_VISIBILITY_HIDDEN void
797     basic_istream<_CharT, _Traits>::sentry::operator()()
798     {
799         ios_base::iostate __err = ios_base::goodbit;
800
801         if (!__noskip && __pool->isopen())
802         {
803             const __ctype_type& __ct
804             while (!traits_type::eq_int_t(
805                 && __ct.is(m, ctype_base::lower))
806                 | traits_type::isalpha(m))
807                 __c = __sb->snextc();
808
809             // __GLIBCXX_RESOLVE_LIB_D
810             // 195. Should basic_istr
811             // set eofbit?
812             if (traits_type::eq_int_t(
813                 __err |= ios_base::eofbit));
814         }
815     }
816
817     __GLIBCXX_VISIBILITY_HIDDEN void
818     basic_istream<_CharT, _Traits>::sentry::operator()()
819     {
820         ios_base::iostate __err = ios_base::goodbit;
821
822         if (!__noskip && __pool->isopen())
823         {
824             const __ctype_type& __ct
825             while (!traits_type::eq_int_t(
826                 && __ct.is(m, ctype_base::lower))
827                 | traits_type::isalpha(m))
828                 __c = __sb->snextc();
829
830             // __GLIBCXX_RESOLVE_LIB_D
831             // 195. Should basic_istr
832             // set eofbit?
833             if (traits_type::eq_int_t(
834                 __err |= ios_base::eofbit));
835         }
836     }
837
838     __GLIBCXX_VISIBILITY_HIDDEN void
839     basic_istream<_CharT, _Traits>::sentry::operator()()
840     {
841         ios_base::iostate __err = ios_base::goodbit;
842
843         if (!__noskip && __pool->isopen())
844         {
845             const __ctype_type& __ct
846             while (!traits_type::eq_int_t(
847                 && __ct.is(m, ctype_base::lower))
848                 | traits_type::isalpha(m))
849                 __c = __sb->snextc();
850
851             // __GLIBCXX_RESOLVE_LIB_D
852             // 195. Should basic_istr
853             // set eofbit?
854             if (traits_type::eq_int_t(
855                 __err |= ios_base::eofbit));
856         }
857     }
858
859     __GLIBCXX_VISIBILITY_HIDDEN void
860     basic_istream<_CharT, _Traits>::sentry::operator()()
861     {
862         ios_base::iostate __err = ios_base::goodbit;
863
864         if (!__noskip && __pool->isopen())
865         {
866             const __ctype_type& __ct
867             while (!traits_type::eq_int_t(
868                 && __ct.is(m, ctype_base::lower))
869                 | traits_type::isalpha(m))
870                 __c = __sb->snextc();
871
872             // __GLIBCXX_RESOLVE_LIB_D
873             // 195. Should basic_istr
874             // set eofbit?
875             if (traits_type::eq_int_t(
876                 __err |= ios_base::eofbit));
877         }
878     }
879
880     __GLIBCXX_VISIBILITY_HIDDEN void
881     basic_istream<_CharT, _Traits>::sentry::operator()()
882     {
883         ios_base::iostate __err = ios_base::goodbit;
884
885         if (!__noskip && __pool->isopen())
886         {
887             const __ctype_type& __ct
888             while (!traits_type::eq_int_t(
889                 && __ct.is(m, ctype_base::lower))
890                 | traits_type::isalpha(m))
891                 __c = __sb->snextc();
892
893             // __GLIBCXX_RESOLVE_LIB_D
894             // 195. Should basic_istr
895             // set eofbit?
896             if (traits_type::eq_int_t(
897                 __err |= ios_base::eofbit));
898         }
899     }
900
901     __GLIBCXX_VISIBILITY_HIDDEN void
902     basic_istream<_CharT, _Traits>::sentry::operator()()
903     {
904         ios_base::iostate __err = ios_base::goodbit;
905
906         if (!__noskip && __pool->isopen())
907         {
908             const __ctype_type& __ct
909             while (!traits_type::eq_int_t(
910                 && __ct.is(m, ctype_base::lower))
911                 | traits_type::isalpha(m))
912                 __c = __sb->snextc();
913
914             // __GLIBCXX_RESOLVE_LIB_D
915             // 195. Should basic_istr
916             // set eofbit?
917             if (traits_type::eq_int_t(
918                 __err |= ios_base::eofbit));
919         }
920     }
921
922     __GLIBCXX_VISIBILITY_HIDDEN void
923     basic_istream<_CharT, _Traits>::sentry::operator()()
924     {
925         ios_base::iostate __err = ios_base::goodbit;
926
927         if (!__noskip && __pool->isopen())
928         {
929             const __ctype_type& __ct
930             while (!traits_type::eq_int_t(
931                 && __ct.is(m, ctype_base::lower))
932                 | traits_type::isalpha(m))
933                 __c = __sb->snextc();
934
935             // __GLIBCXX_RESOLVE_LIB_D
936             // 195. Should basic_istr
937             // set eofbit?
938             if (traits_type::eq_int_t(
939                 __err |= ios_base::eofbit));
940         }
941     }
942
943     __GLIBCXX_VISIBILITY_HIDDEN void
944     basic_istream<_CharT, _Traits>::sentry::operator()()
945     {
946         ios_base::iostate __err = ios_base::goodbit;
947
948         if (!__noskip && __pool->isopen())
949         {
950             const __ctype_type& __ct
951             while (!traits_type::eq_int_t(
952                 && __ct.is(m, ctype_base::lower))
953                 | traits_type::isalpha(m))
954                 __c = __sb->snextc();
955
956             // __GLIBCXX_RESOLVE_LIB_D
957             // 195. Should basic_istr
958             // set eofbit?
959             if (traits_type::eq_int_t(
960                 __err |= ios_base::eofbit));
961         }
962     }
963
964     __GLIBCXX_VISIBILITY_HIDDEN void
965     basic_istream<_CharT, _Traits>::sentry::operator()()
966     {
967         ios_base::iostate __err = ios_base::goodbit;
968
969         if (!__noskip && __pool->isopen())
970         {
971             const __ctype_type& __ct
972             while (!traits_type::eq_int_t(
973                 && __ct.is(m, ctype_base::lower))
9
```



同样这些扩展对于现阶段而言没有太重要的。

我推荐的使用顺序时，刚开始学c++可以用vscode，然后之后去搜搜有关配vscode环境的博客，在学完c++之后要学各种多文件编写，比如你要做个游戏之类的，可以使用CLion。当然所有的IDE不分高下，也可以选择vs和dev这里没有强制性要求。

IDE代码技巧

很多时候你会发现你的代码编程速度很慢，我这部分补充的是最基本的一些操作，很遗憾这部分大部分人都是在看别人编码的过程中学到的，我非常推荐的建议是

- "抠"掉自己的小键盘区的数字，用主键盘区的数字以及加减乘除符号，这里对有外设键盘的人就很棒，练习自己的敲代码速度，我知道这个过程真的很痛苦，因为我就是这样过来的。
- 程序员可能最初知道的就是Ctrl+A, Ctrl+C/V，但是在vscode和CLion里我认为Ctrl+X剪切掉整行挺好用的，不知道算不算一个小技巧。
- 按住shift然后按上下左右可以选中某块区域，这个技巧可以说是我看别人写代码然后查阅才知道的，因为当时选中代码只会用鼠标，这样减慢了删代码的速度。

- 选中某块代码tab可以整体缩进，但是很多人不知道shift+tab是向左整体缩进。
- 当你敲了左括号[的时候，IDE一般会自动补全右括号]，这时候光标在括号里，如果按了右括号]，会自动跳出括号，这个技巧很重要，防止多敲括号，而且真正提高了编程速度，以后一旦要出括号直接按一下]就可以，不用按方向键。{}也同理，按住shift+右花括号可以直接出括号。