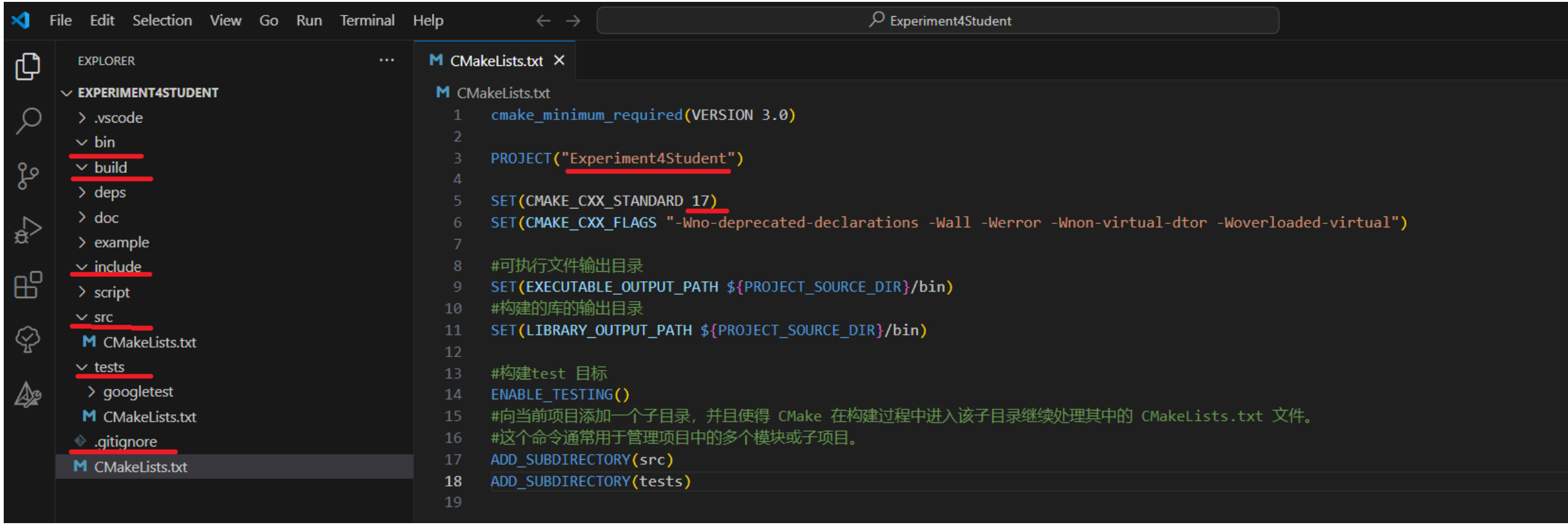
实验一StepbyStep

从工程模版创建一个空的工程，比如叫Experiment4Student



首先可以看到工程下的bin、build、include目录都是空的，src和tests目录下没有任何C++源码文件

请修改根目录下的CMakeLists.txt(红色下划线所示，C++标准改成17，工程名称改成项目根目录名)

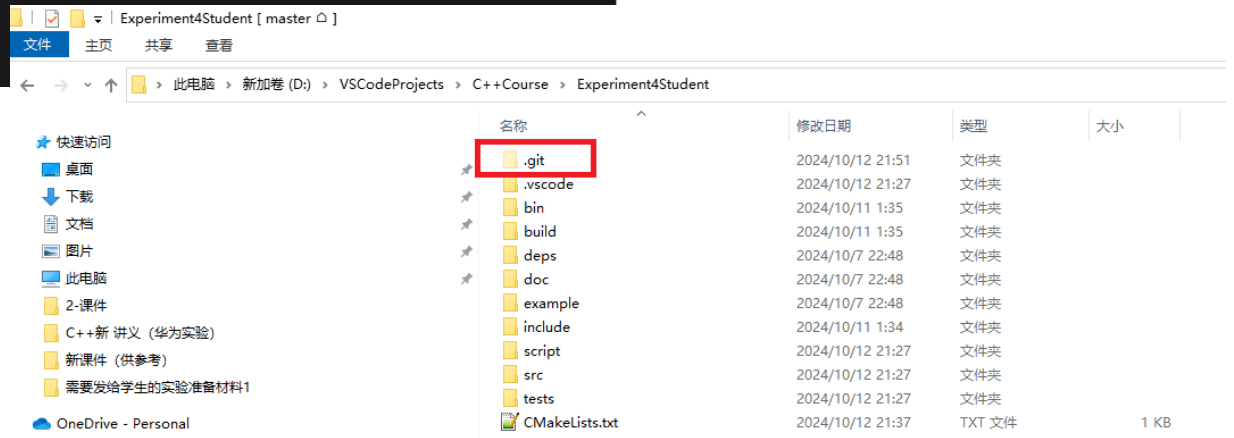
同时要看到工程根目录下有一个.gitignore文件，这个是配置当提交代码到git时，哪些文件/目录是不需要提交的。在更新的工程模板里已经添加了这个文件

将这个工程提交给本地git仓库（因为暂时没有远程服务器搭建远程仓库）

打开Terminal，输入下面命令创建本地仓库

git init　　#初始化仓库可以发现当前目录下多了一个.git的目录，这个就是本地仓库

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Active code page: 65001
PS D:\VSCodeProjects\C++Course\Experiment4Student> git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in D:/VSCodeProjects/C++Course/Experiment4Student/.git/
PS D:\VSCodeProjects\C++Course\Experiment4Student>
```

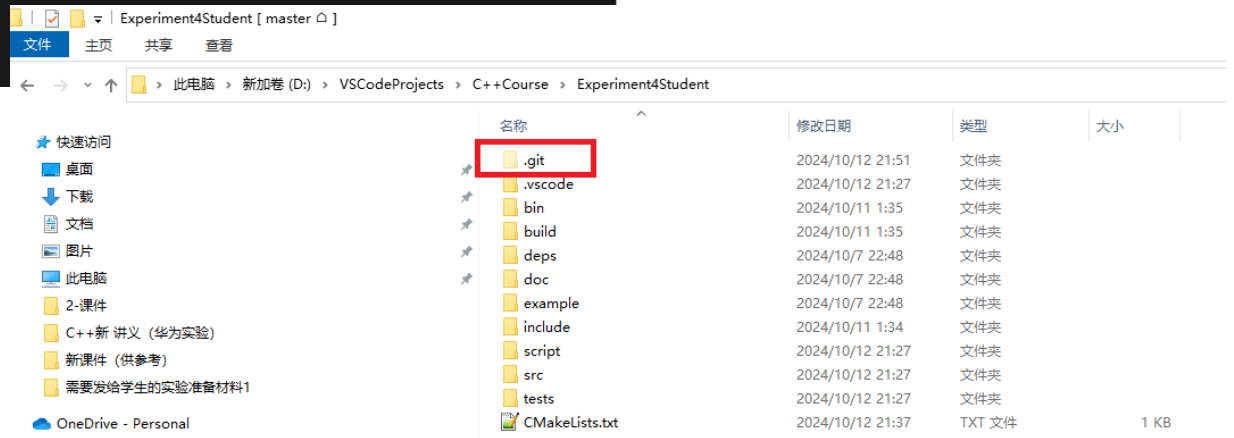


将这个工程提交给本地git仓库（因为暂时没有远程服务器搭建远程仓库）

打开Terminal，输入下面命令创建本地仓库

git init　　#初始化仓库可以发现当前目录下多了一个.git的目录，这个就是本地仓库

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Active code page: 65001
PS D:\VSCodeProjects\C++Course\Experiment4Student> git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in D:/VSCodeProjects/C++Course/Experiment4Student/.git/
PS D:\VSCodeProjects\C++Course\Experiment4Student>
```

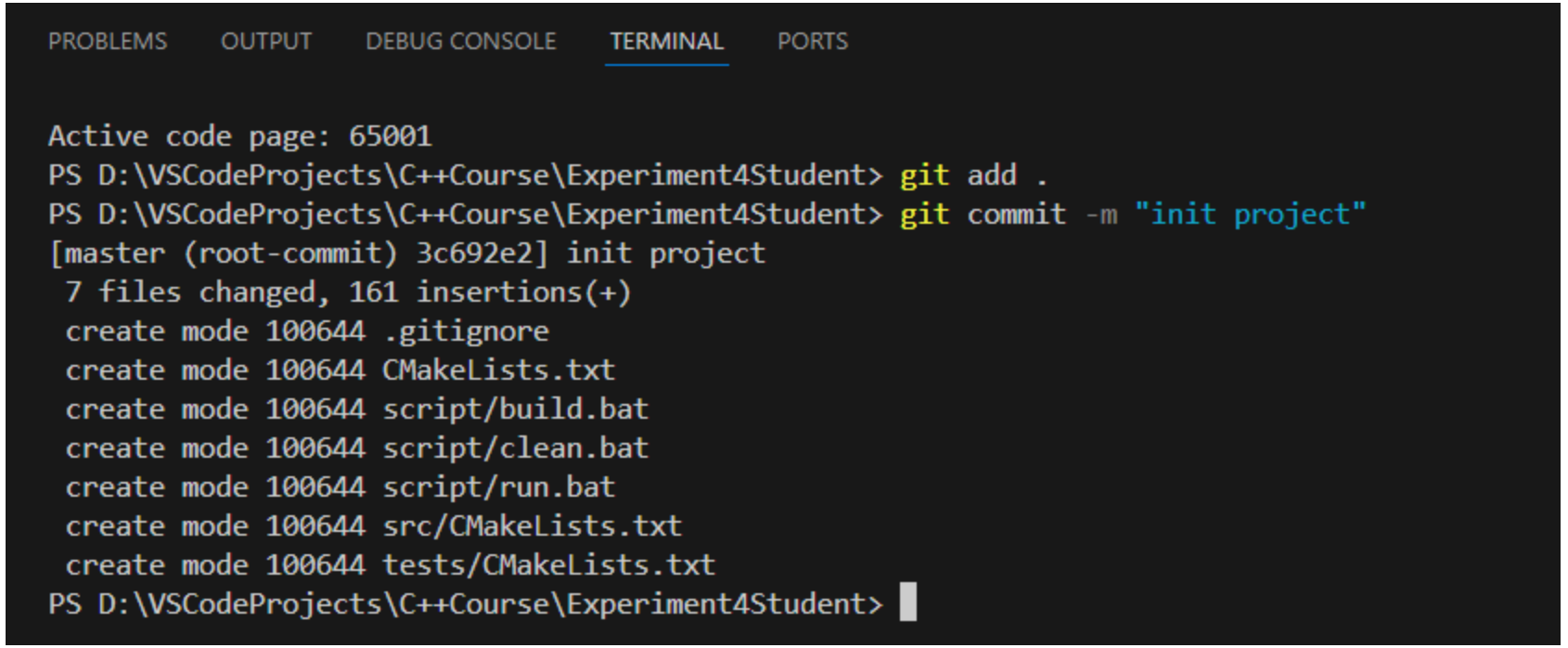


将这个工程提交给本地git仓库（因为暂时没有远程服务器搭建远程仓库）

在Terminal里接着输入下面二行命令

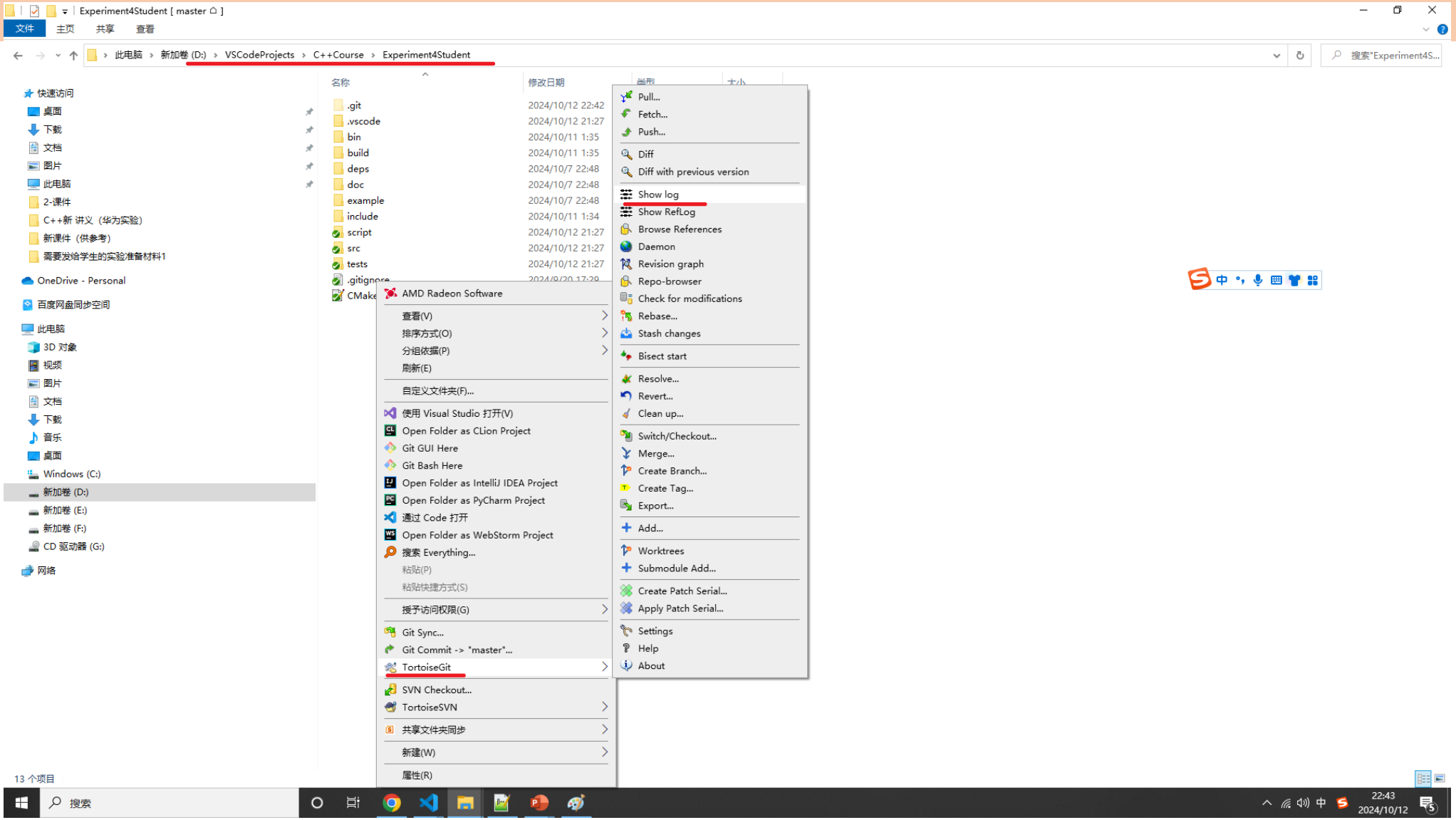
```
git add .                                  #提交修改过或新增的文件到暂存区
git commit -m "init project"               #将暂存区的文件提交到本地仓库中
```

注意：每次提交代码到本地仓库时，一定要规范-m 后面的comment信息

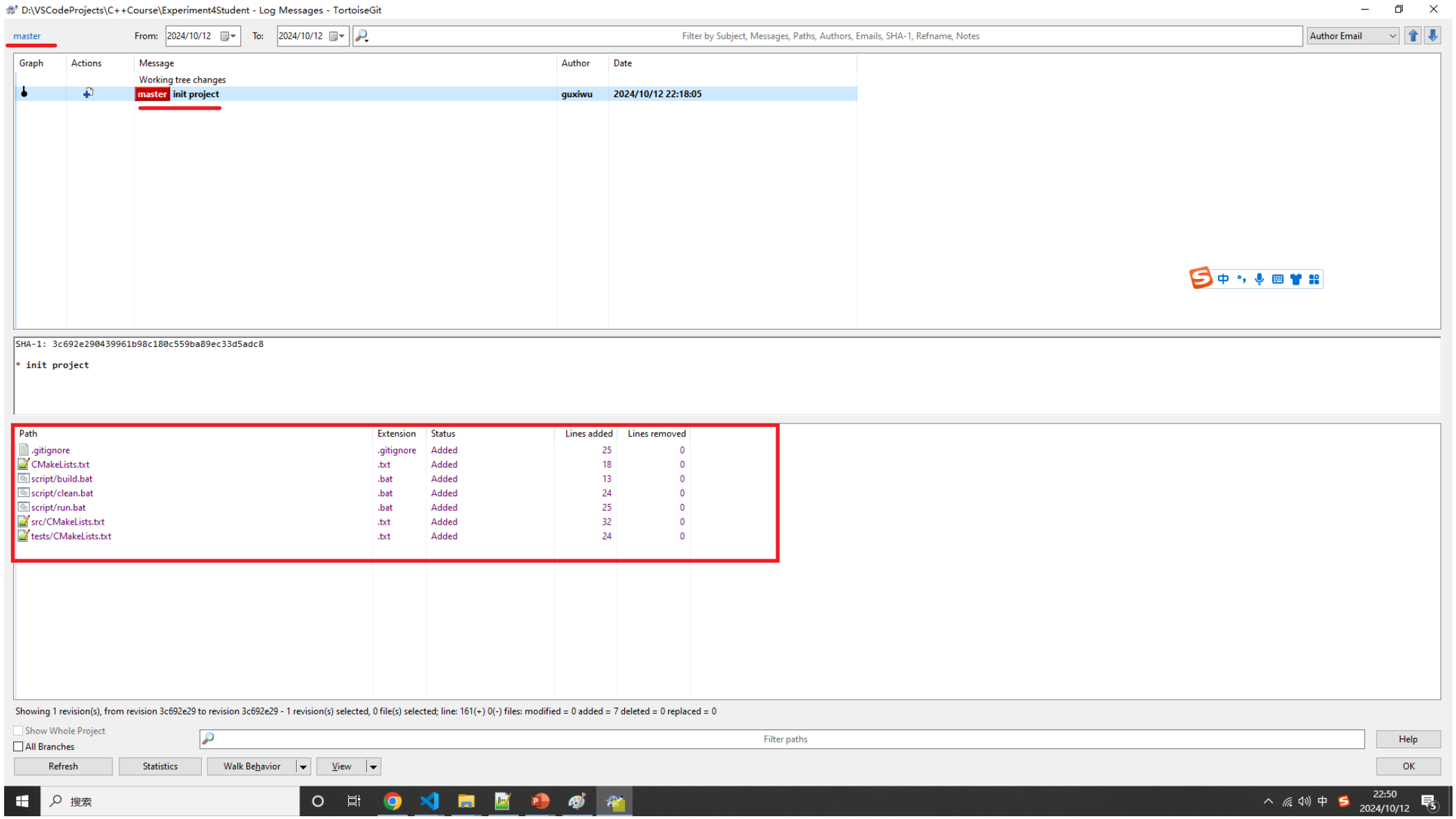


建议安装TortoiseGit（https://tortoisegit.org/download/）

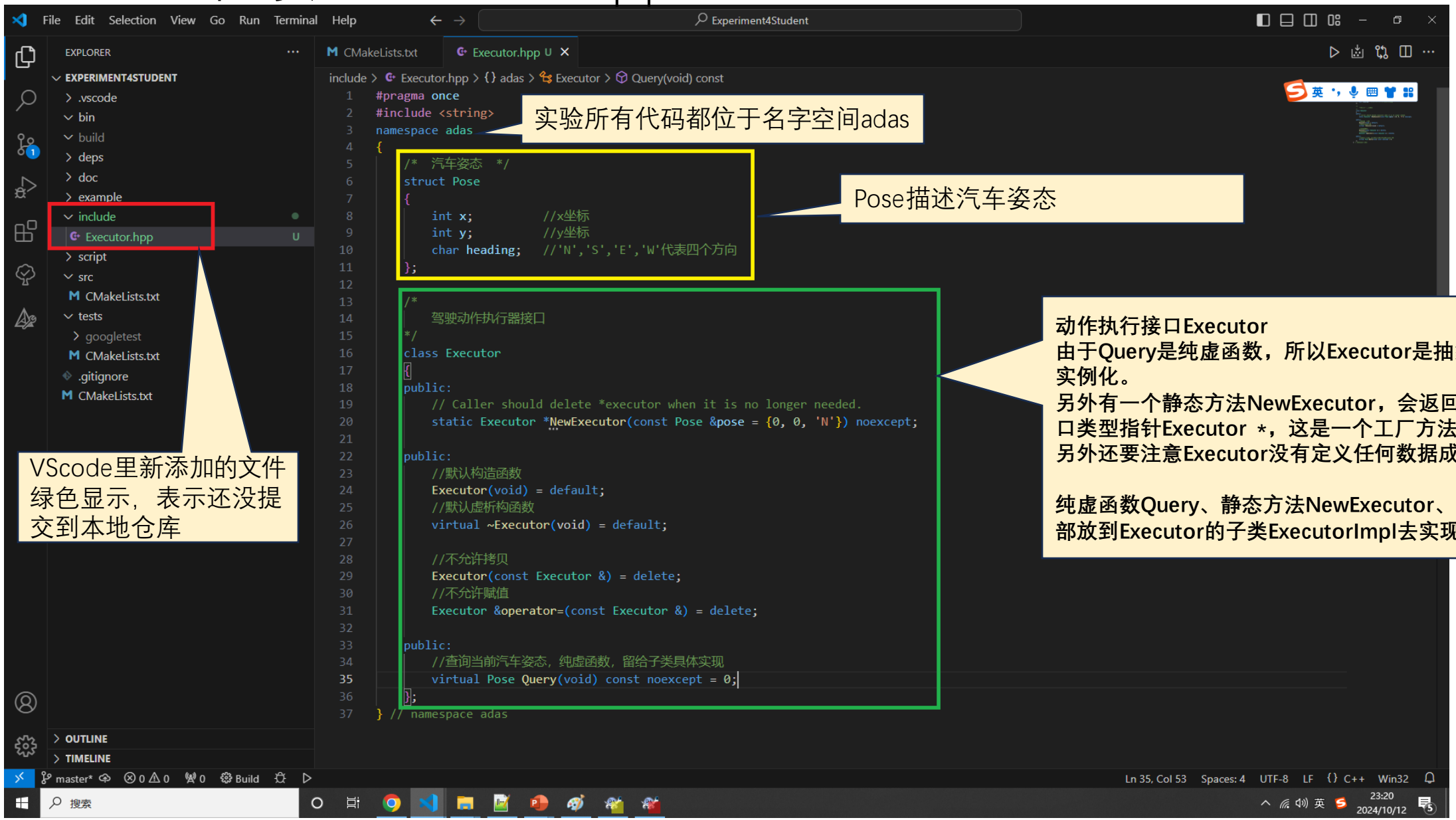
安装过程一路默认就行。安装好，打开工程目录，里面应该可以看到.git子目录。在空白处点击鼠标右键



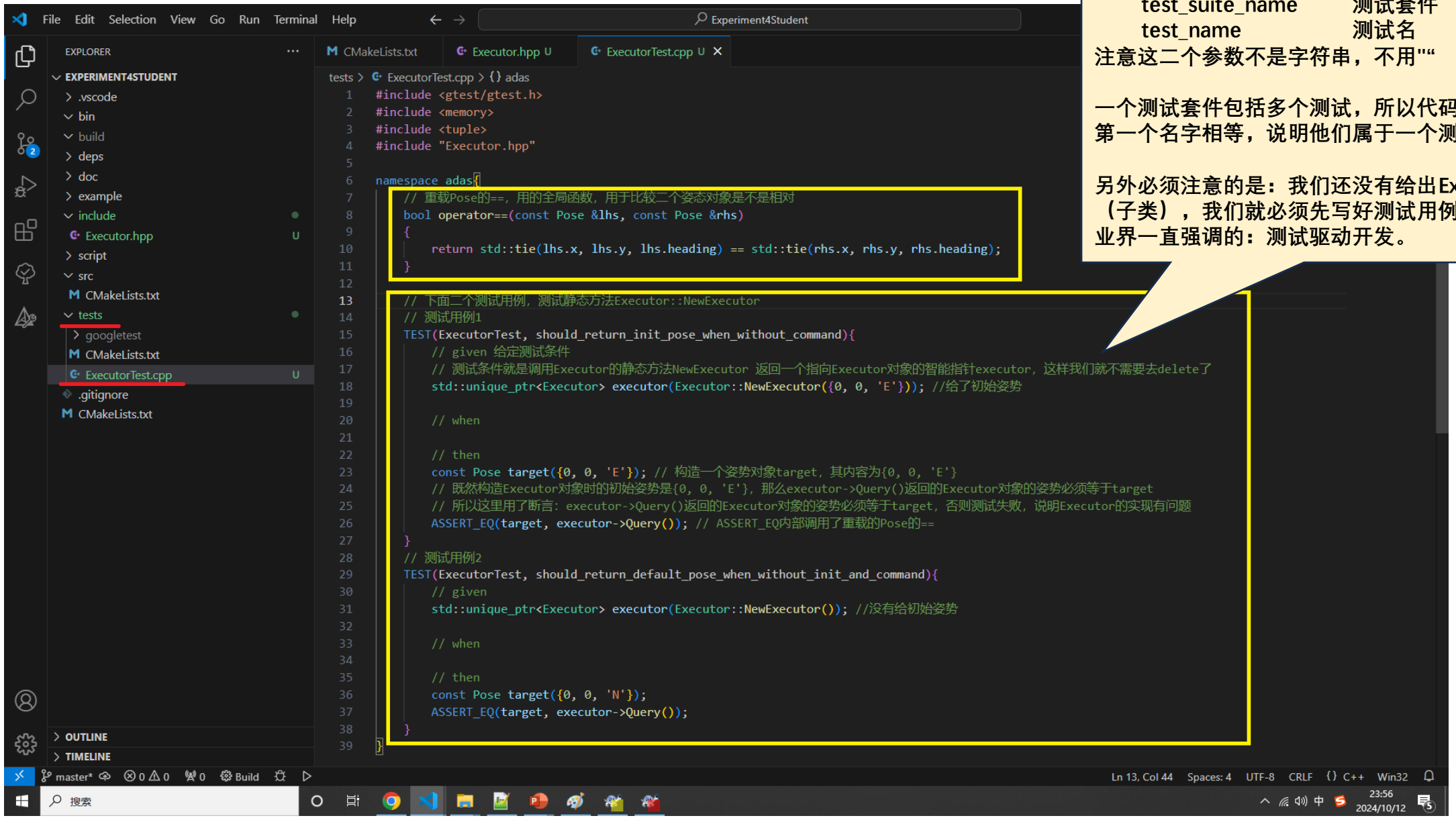
通过TortoiseGit查看提交的log日志



在include下添加Executor.hpp



在tests下添加ExecutorTest.cpp



TEST是一个宏，包括二个参数：

test_suite_name 测试套件

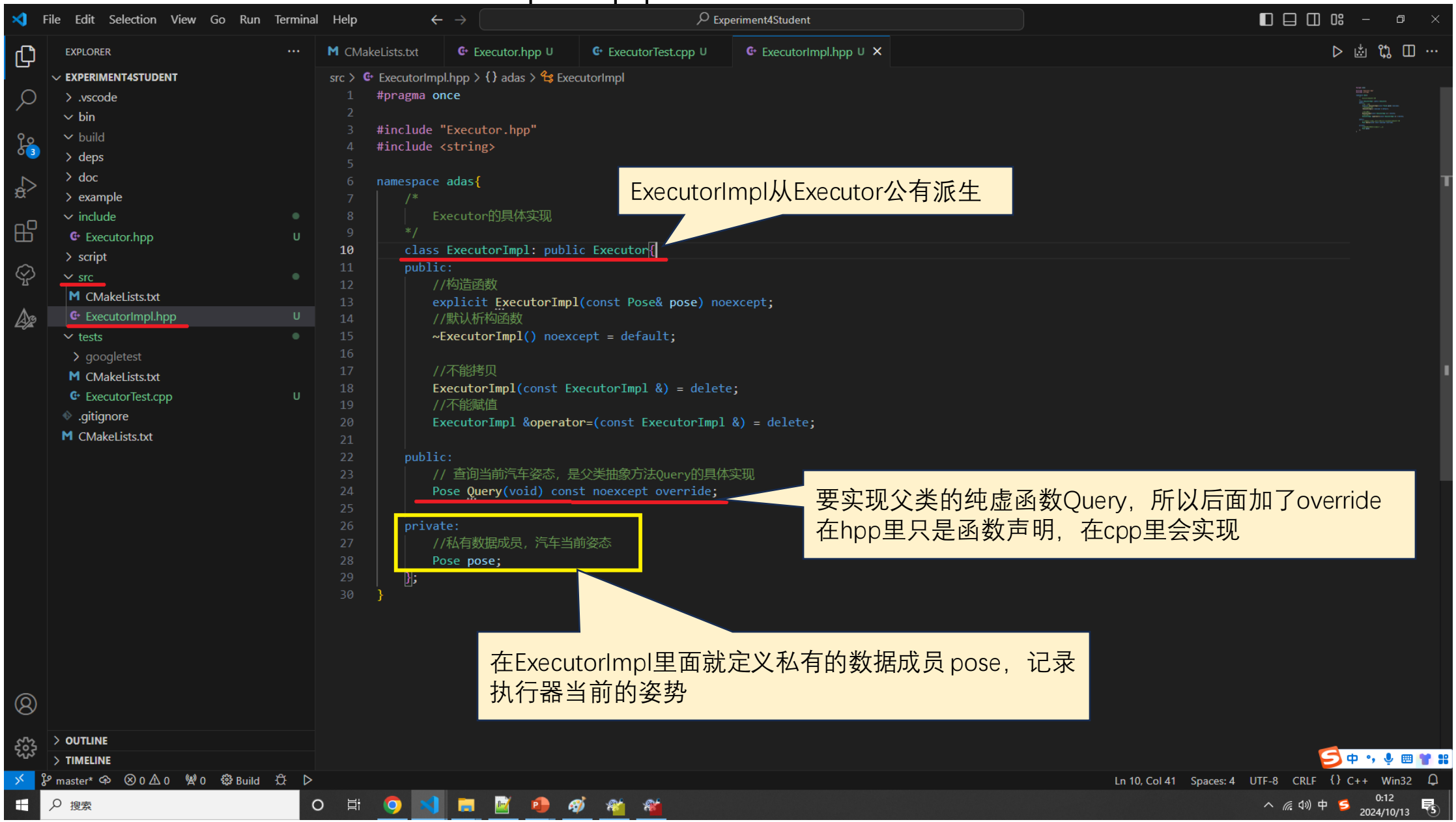
test_name 测试名

注意这二个参数不是字符串，不用""

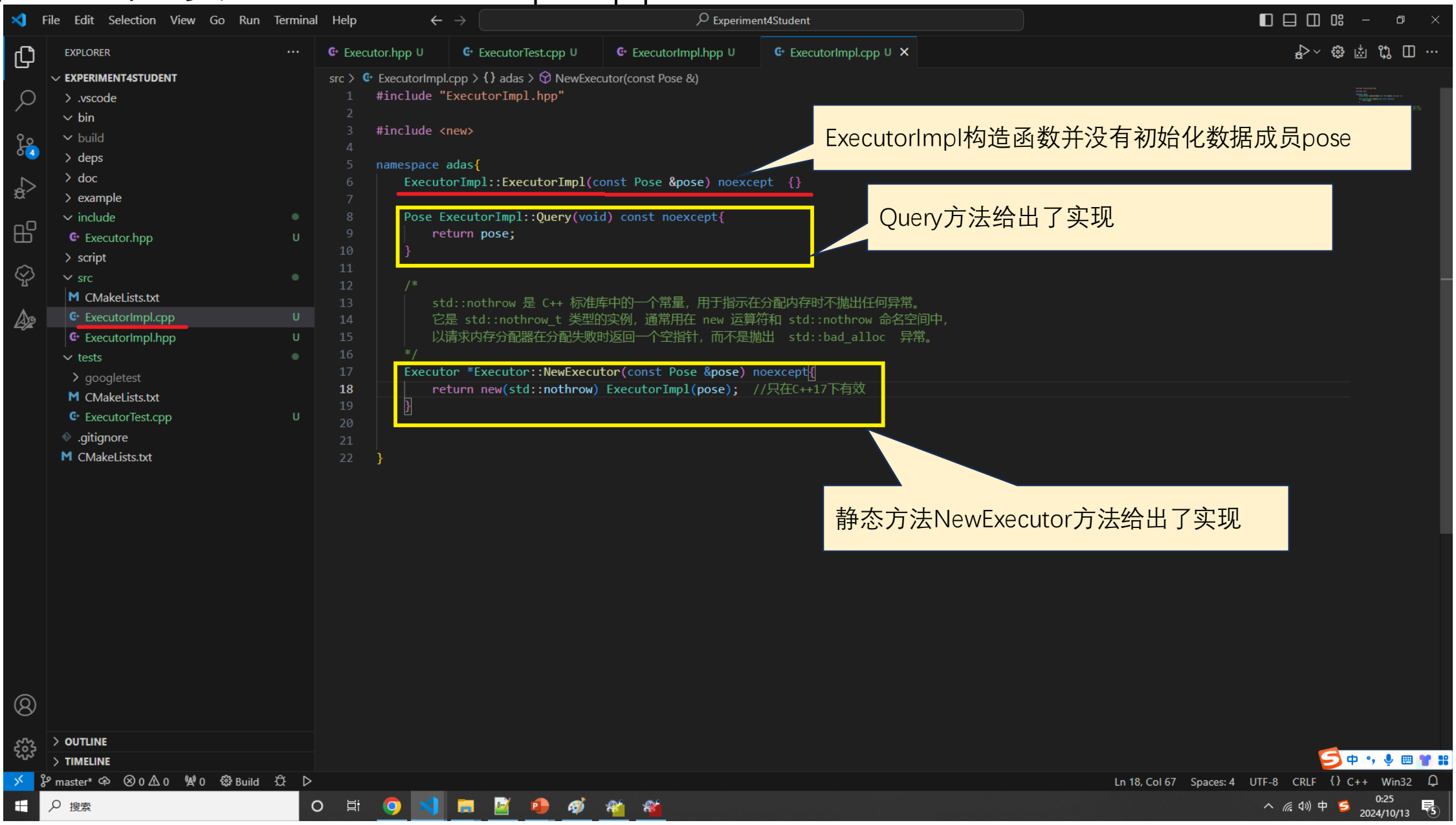
一个测试套件包括多个测试，所以代码里二个测试的第一个名字相等，说明他们属于一个测试套件

另外必须注意的是：我们还没有给出Executor的实现（子类），我们就必须先写好测试用例代码。这就是业界一直强调的：测试驱动开发。

在src下添加ExecutorImpl.hpp

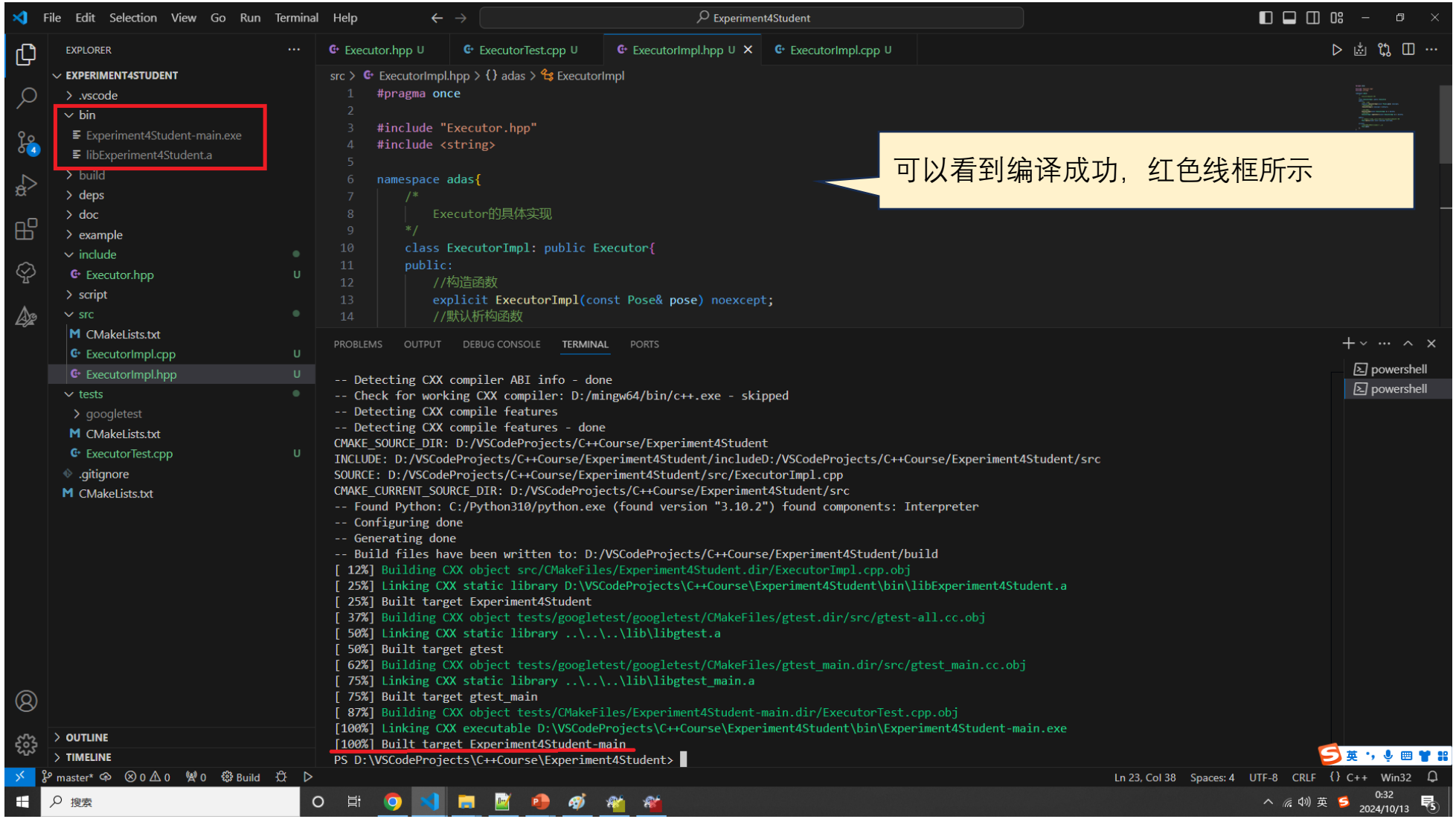


在src下添加ExecutorImpl.cpp



现在来编译工程

打开Terminal，输入下面命令
.\script\build.bat



现在来运行编译好的代码

打开Terminal，输入下面命令
.\script\run.bat

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Expected equality of these values:
  target
    Which is: 12-byte object <00-00 00-00 00-00 00-00 45-00 00-00>
  executor->Query()
    Which is: 12-byte object <50-01 AD-00 00-00 00-00 5F-63 6F-6D>
[  FAILED  ] ExecutorTest.should_return_init_pose_when_without_command (27 ms)
[ RUN      ] ExecutorTest.should_return_default_pose_when_without_init_and_command
D:\VSCodeProjects\C++Course\Experiment4Student\tests\ExecutorTest.cpp:37: Failure
Expected equality of these values:
  target
    Which is: 12-byte object <00-00 00-00 00-00 00-00 4E-00 00-00>
  executor->Query()
    Which is: 12-byte object <50-01 AD-00 00-00 00-00 5F-63 6F-6D>
[  FAILED  ] ExecutorTest.should_return_default_pose_when_without_init_and_command (33 ms)
[----------] 2 tests from ExecutorTest (95 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test suite ran. (117 ms total)
[  PASSED  ] 0 tests.
[  FAILED  ] 2 tests, listed below:
[  FAILED  ] ExecutorTest.should_return_init_pose_when_without_command
[  FAILED  ] ExecutorTest.should_return_default_pose_when_without_init_and_command

 2 FAILED TESTS
PS D:\VSCodeProjects\C++Course\Experiment4Student>
```

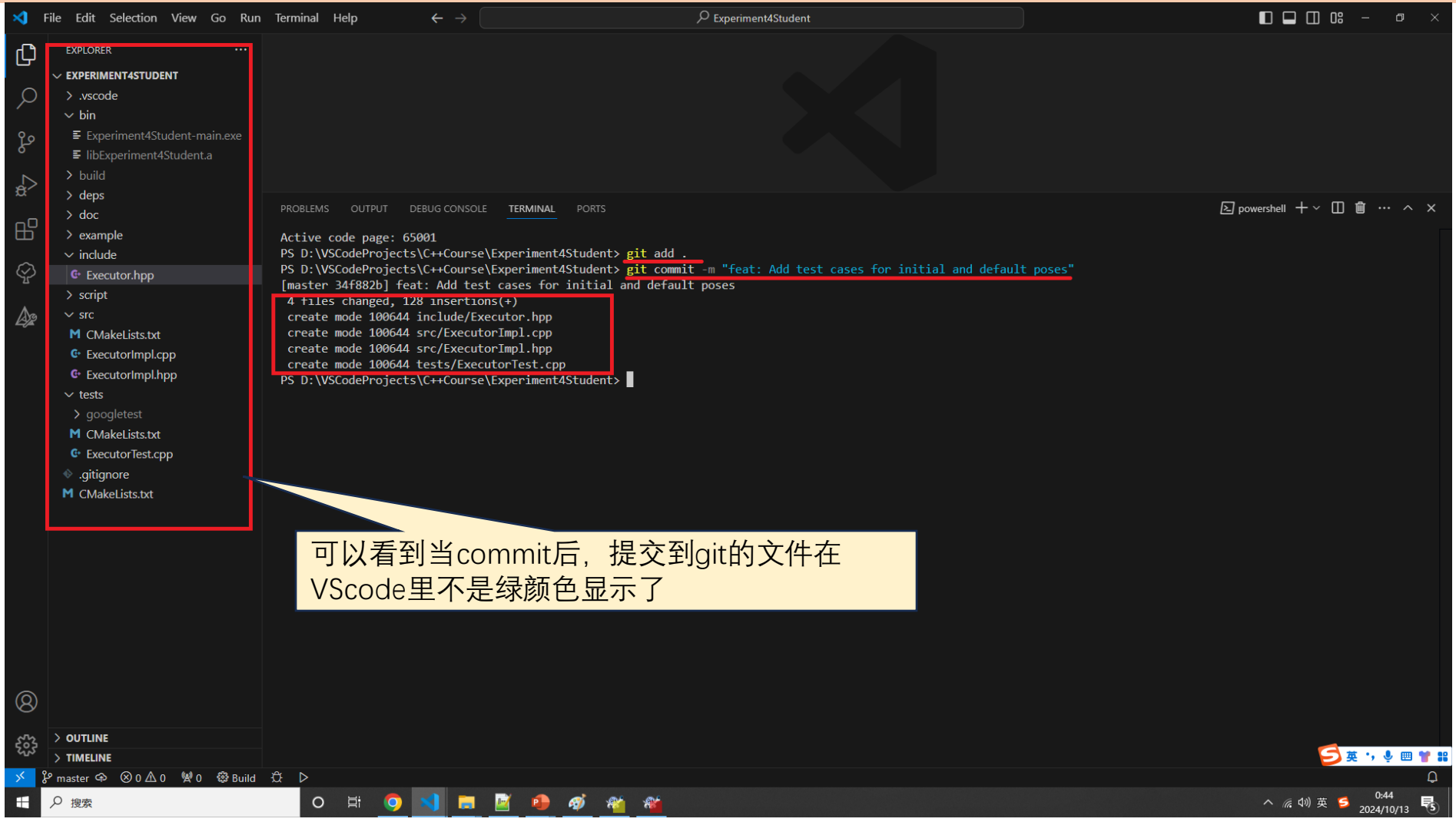
可以看到测试失败，因为ExecutorImpl构造函数并没有初始化数据成员pose

现在将目前的代码提交到Git的本地仓库

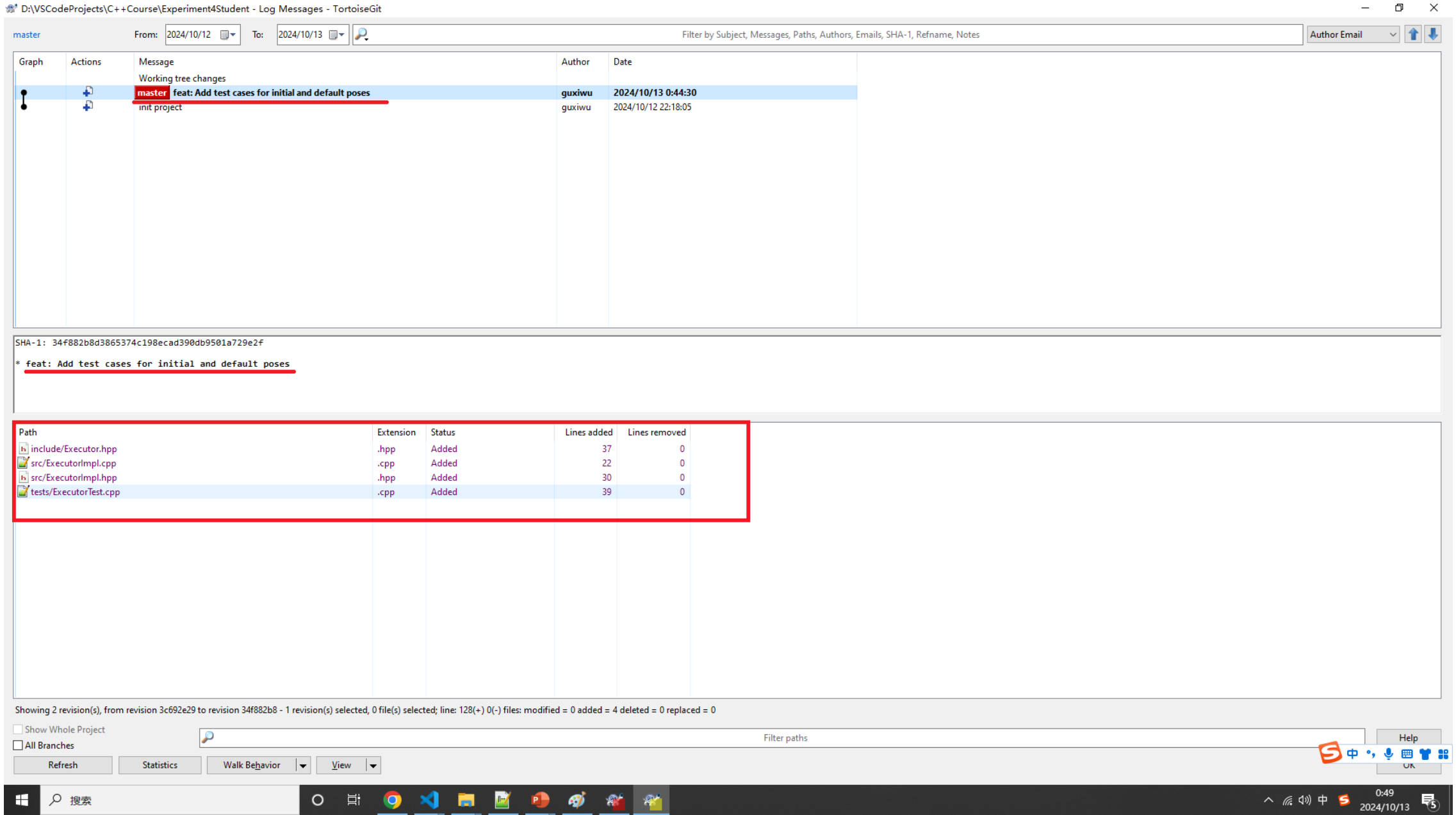
打开Terminal，输入下面命令

git add . **#提交修改过或新增的文件到暂存区**

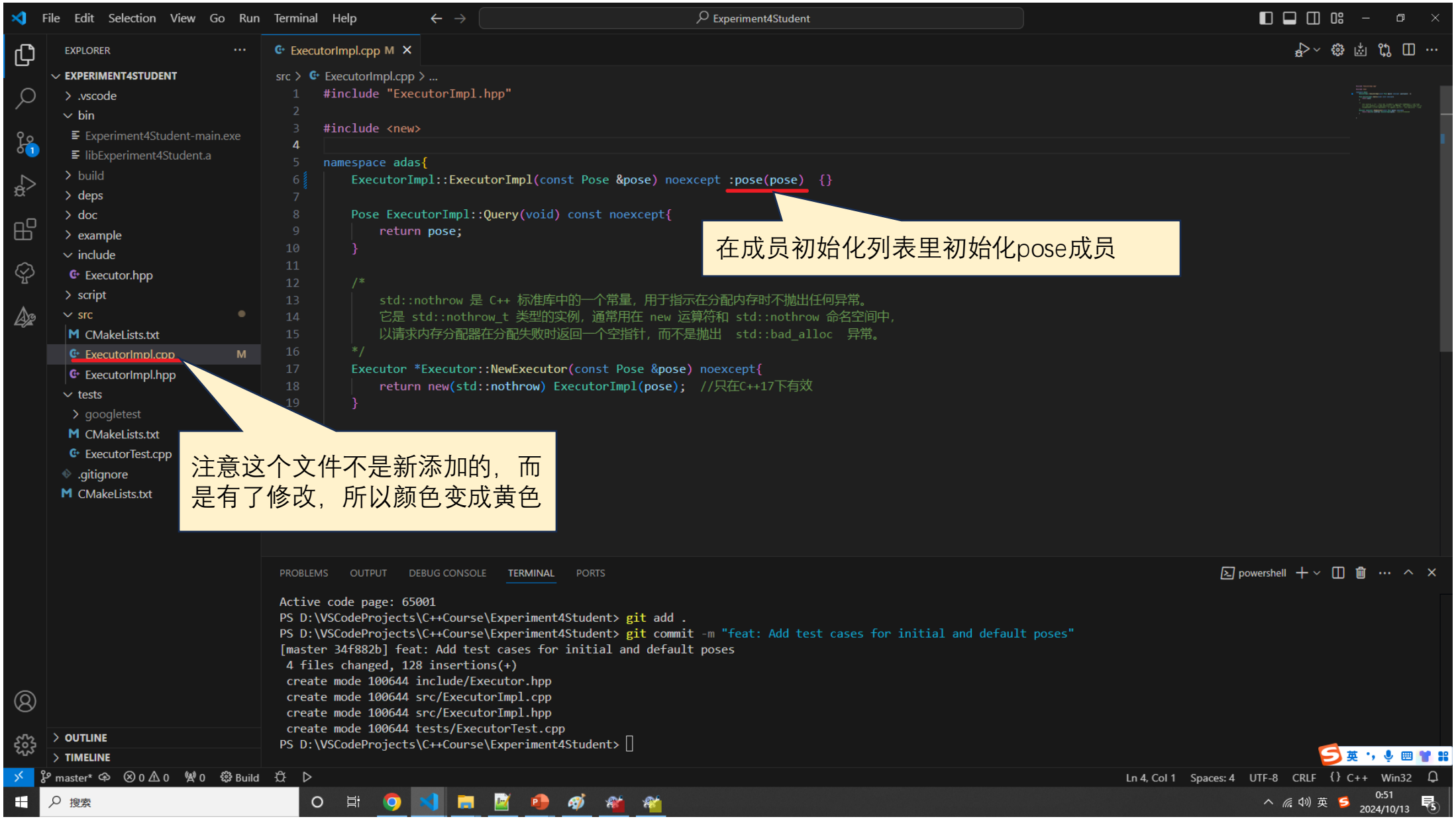
git commit -m "feat: Add test cases for initial and default poses" #将暂存区的文件提交到本地仓库中



通过TortoiseGit查看提交的log日志



现在修改ExecutorImpl.cpp



现在重新来编译和运行工程

打开Terminal，输入下面命令
.\script\build.bat
.\script\run.bat

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Active code page: 65001
PS D:\VSCodeProjects\C++Course\Experiment4Student> .\script\run.bat

D:\VSCodeProjects\C++Course\Experiment4Student>rem 自动执行bin目录下的exe文件（如果有多个exe文件，会自动执行最后找到的exe文件）
D:\VSCodeProjects\C++Course\Experiment4Student\bin\Experiment4Student-main.exe
Running main() from D:\VSCodeProjects\C++Course\Experiment4Student\tests\googletest\googletest\src\gtest_main.cc
[==========] Running 2 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 2 tests from ExecutorTest
[ RUN      ] ExecutorTest.should_return_init_pose_when_without_command
[       OK ] ExecutorTest.should_return_init_pose_when_without_command (0 ms)
[ RUN      ] ExecutorTest.should_return_default_pose_when_without_init_and_command
[       OK ] ExecutorTest.should_return_default_pose_when_without_init_and_command (0 ms)
[----------] 2 tests from ExecutorTest (31 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test suite ran. (51 ms total)
[  PASSED  ] 2 tests.
PS D:\VSCodeProjects\C++Course\Experiment4Student>
```

这是一个里程碑！现在将目前的代码提交到Git的本地仓库

打开Terminal，输入下面命令

git add . **#提交修改过或新增的文件到暂存区**

git commit -m "test: Pass initial and default pose test cases" #将暂存区的文件提交到本地仓库中

Git commit 规范是为了保证团队项目中的提交信息有良好的可读性和可维护性。一条符合规范的commit message应该包含三个部分：Header，Body 和 Footer。

// type和subject必需，scope、body、footer可选

<type>(<scope>): <subject>
// 空一行
<body>
// 空一行
<footer>

因此当忽略scope、body、footer时，就是：

<type>: subject

例如我们的二次提交

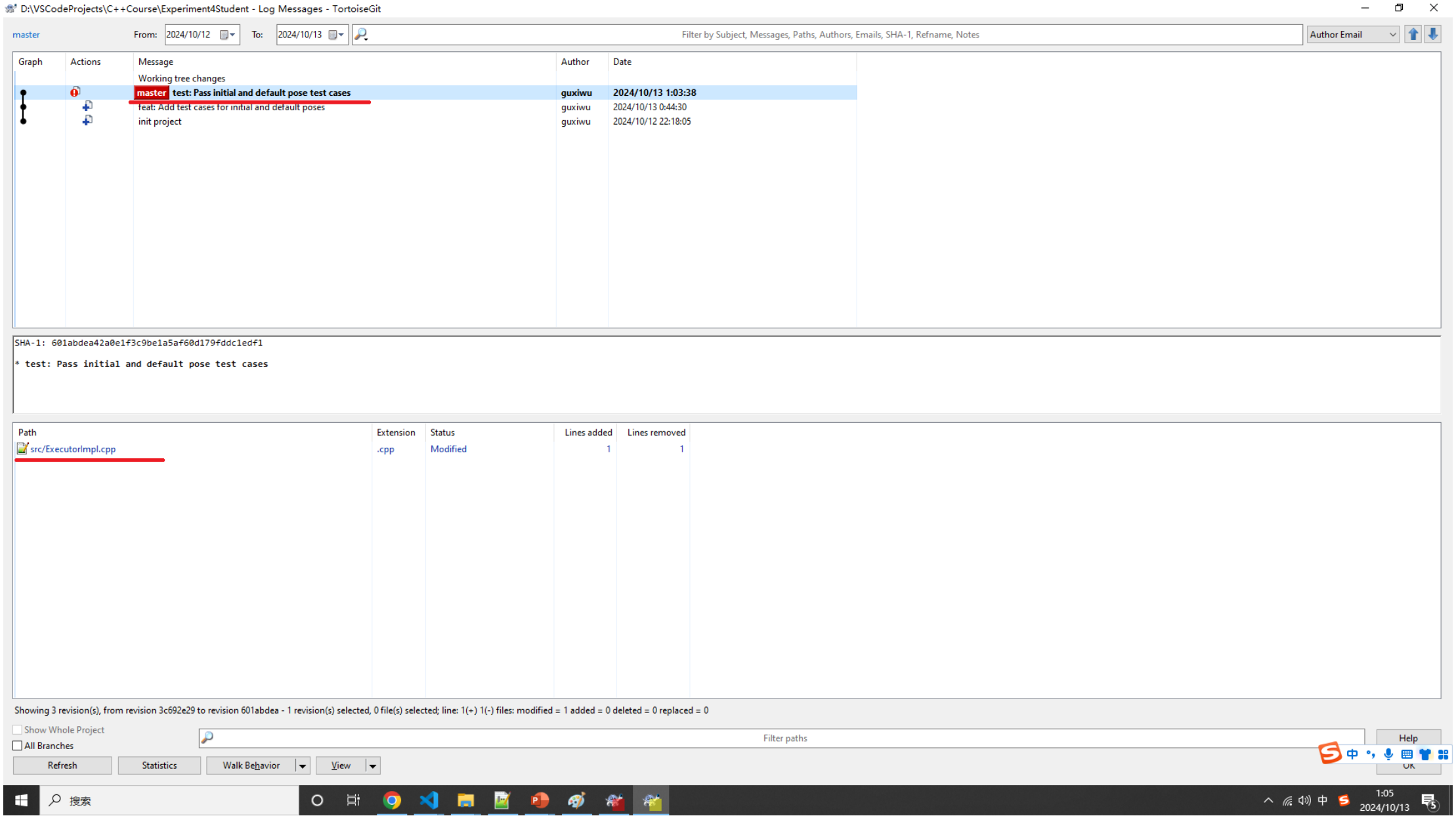
feat: Add test cases for initial and default poses
test: Pass initial and default pose test cases

小步提交：每次提交应包含一个小的、独立的功能或修复，便于回溯和管理

清晰的提交信息：提交信息应简洁明了

type	描述
feat	新增功能
fix	修复bug
ui	样式调整
refactor	重构（既不修复错误也不添加功能）
perf	优化相关，比如提升性能、体验
revert	回滚之前的commit

通过TortoiseGit查看提交的log日志



现在需要实现执行指令的功能。回顾一下需求

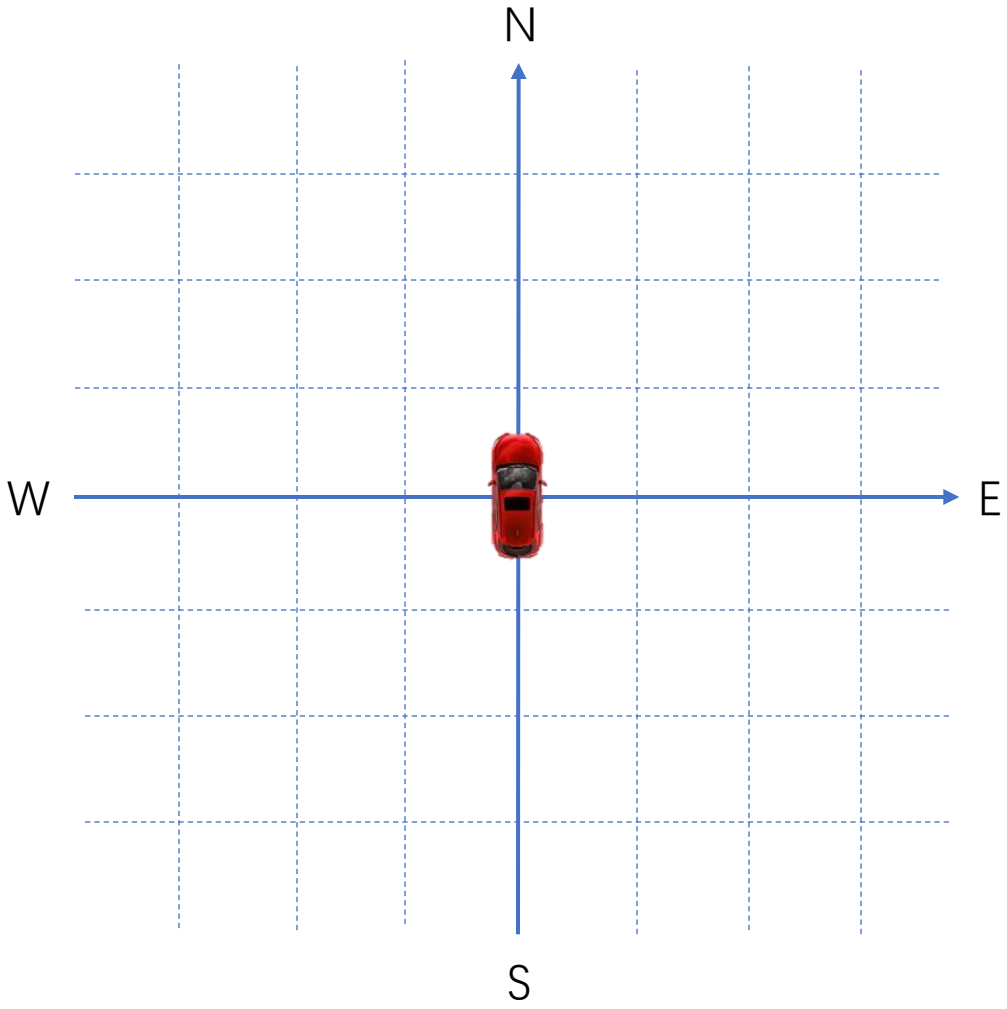
一辆车在二维坐标平面上行驶，车的位置始终为整数表示的坐标点，朝向有4种。

车的控制系统通过以字母表示的指令控制汽车。

指令序列：　M L M M R M

当前位置：　(-2,1)

当前朝向：　W



现在需要实现执行指令的功能。回顾一下需求

Executor组件可以执行如下的移动指令：

M: 前进，1次移动1格

Executor组件可以执行如下的转向指令：

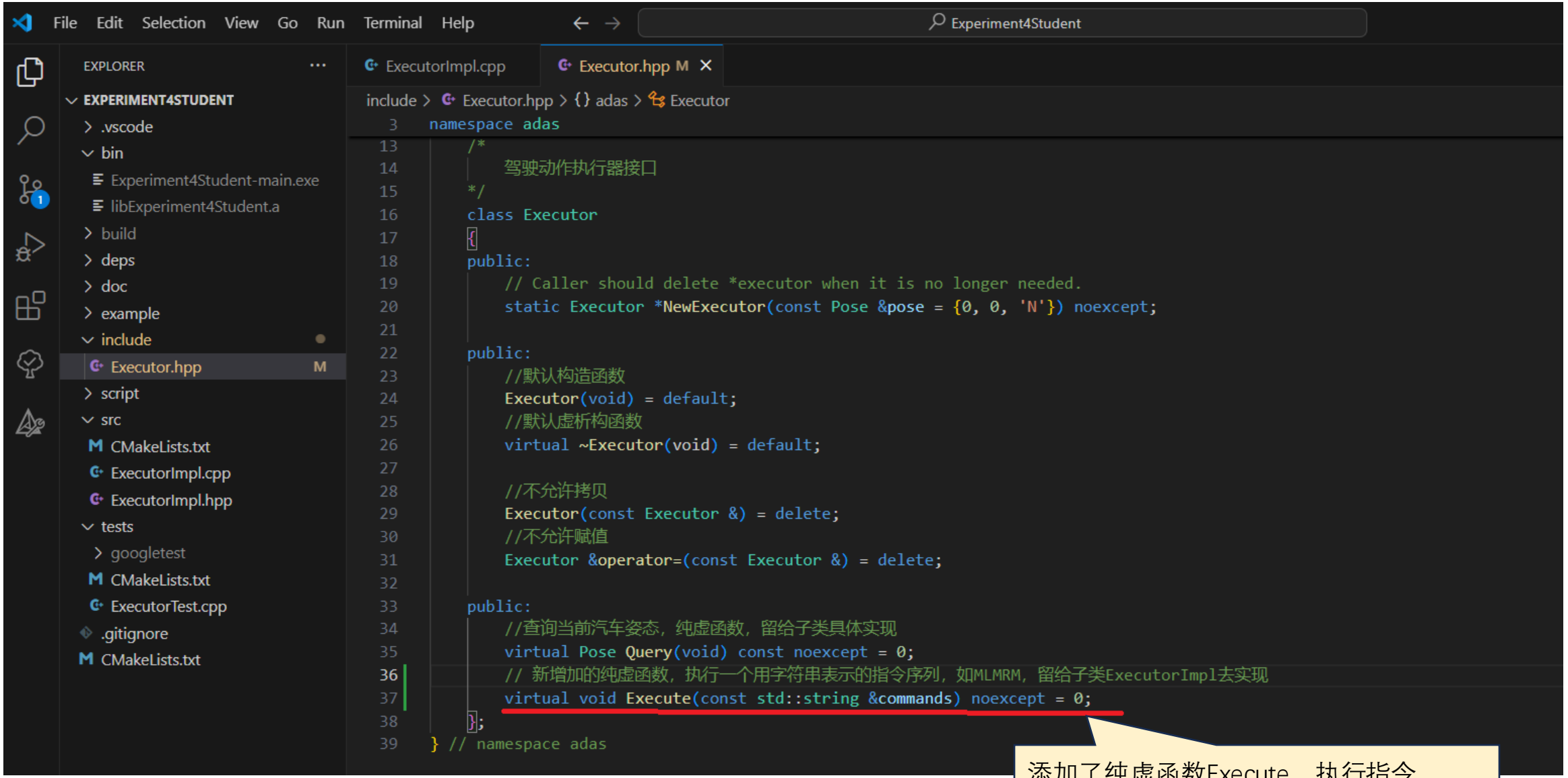
L: 左转90度，位置不变

R: 右转90度，位置不变

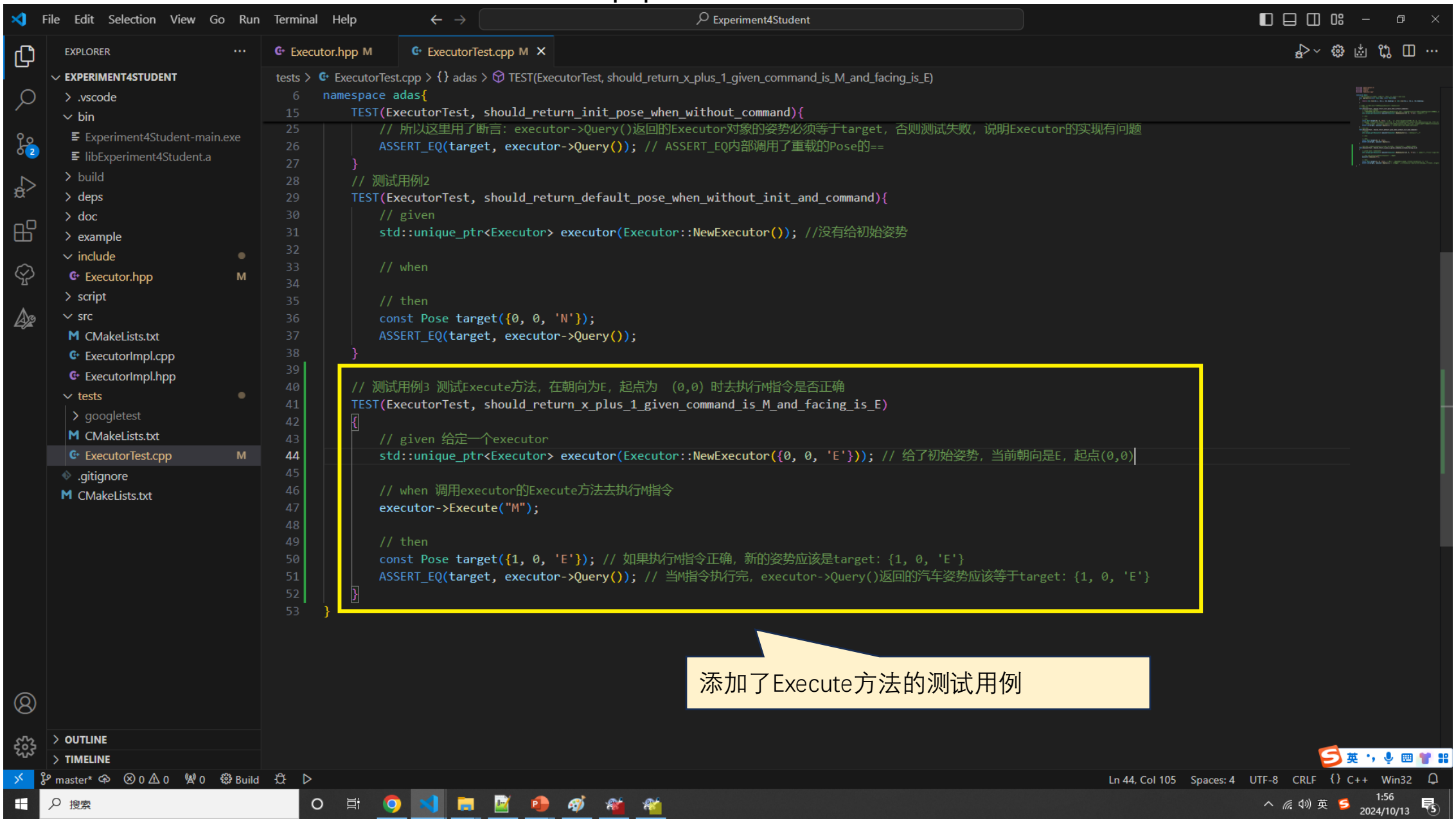
X轴移动的方向为EW方向，Y轴移动的方向为NS方向。

也可以执行这三个指令的组合序列，如MLMRM

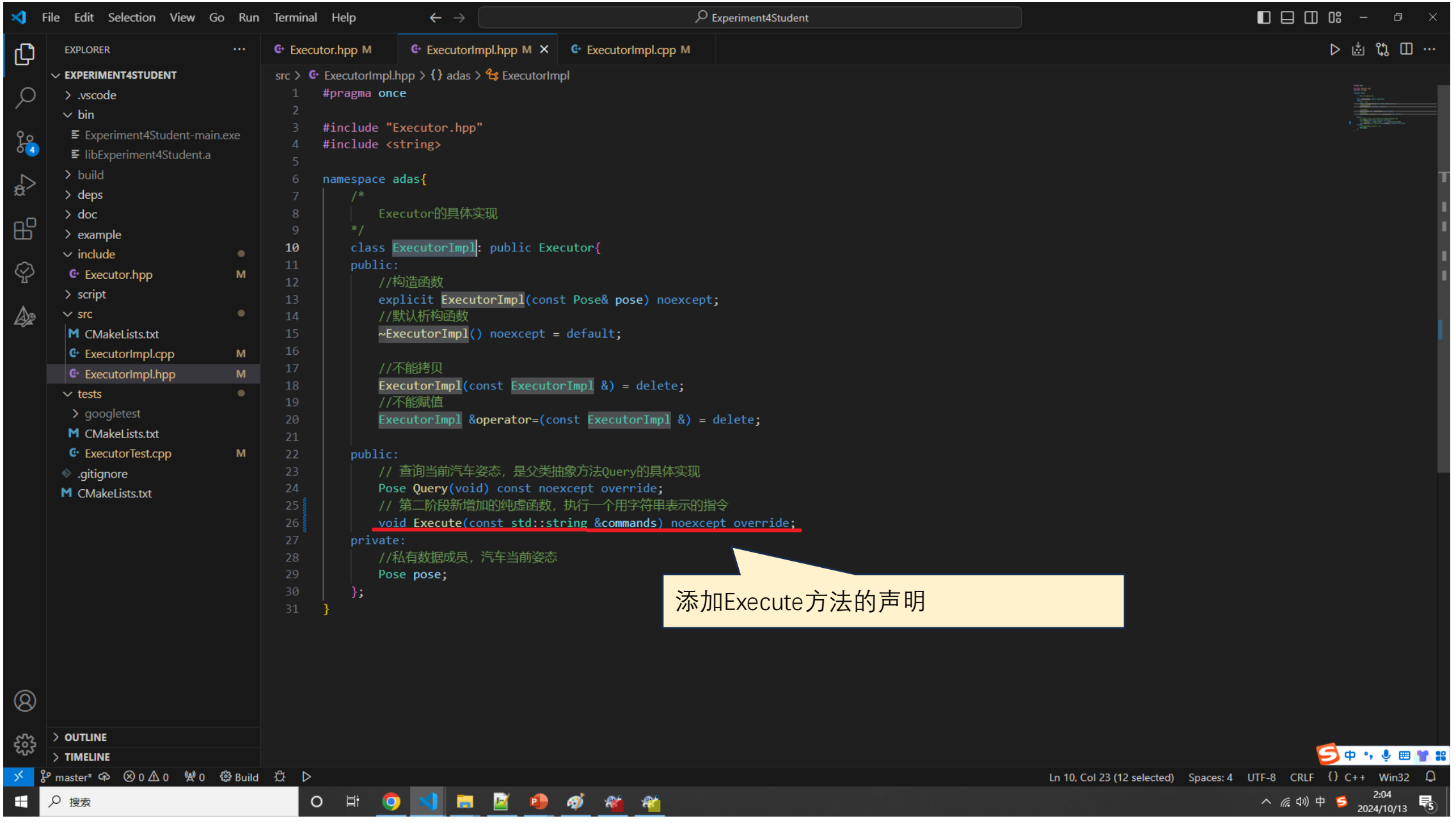
现在需要在include/Executor.hpp里，为Executor接口添加一个纯虚函数，来执行指令



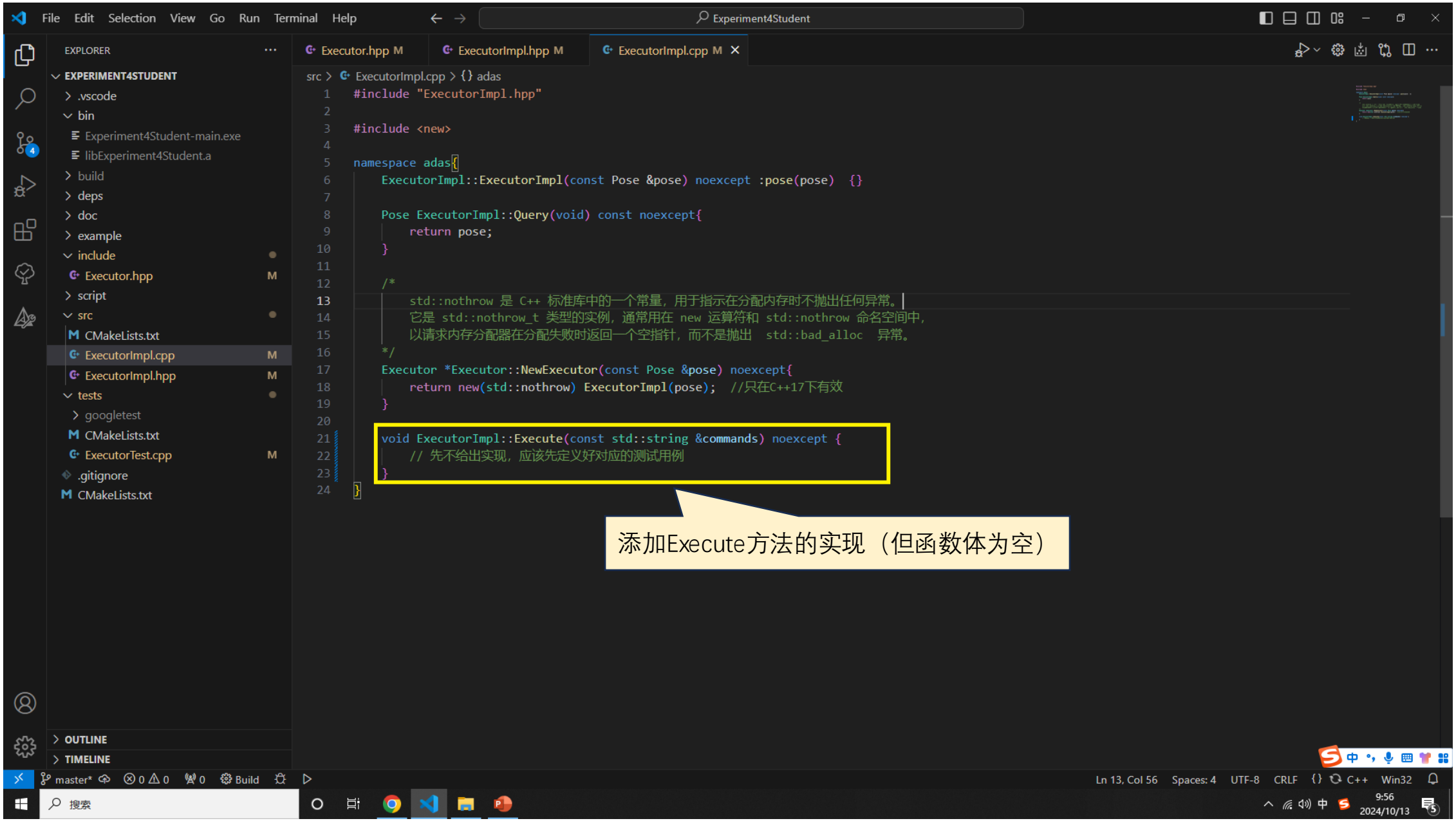
修改tests下的ExecutorTest.cpp，添加Execute方法的测试用例



修改src下的ExecutorImpl.hpp，添加Execute方法的声明



修改src下的ExecutorImpl.cpp，添加Execute方法的实现（但函数体为空）



现在来编译和运行工程

打开Terminal，输入下面命令
.\script\build.bat
.\script\run.bat

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Active code page: 65001
PS D:\VSCodeProjects\C++Course\Experiment4Student> .\script\build.bat

D:\VSCodeProjects\C++Course\Experiment4Student>rem 首先配置cmake，然后make编译程序，生成的exe文件放在bin目录下
CMAKE_SOURCE_DIR: D:/VSCodeProjects/C++Course/Experiment4Student
INCLUDE: D:/VSCodeProjects/C++Course/Experiment4Student/includeD:/VSCodeProjects/C++Course/Experiment4Student/src
SOURCE: D:/VSCodeProjects/C++Course/Experiment4Student/src/ExecutorImpl.cpp
CMAKE_CURRENT_SOURCE_DIR: D:/VSCodeProjects/C++Course/Experiment4Student/src
-- Configuring done
-- Generating done
-- Build files have been written to: D:/VSCodeProjects/C++Course/Experiment4Student/build
Consolidate compiler generated dependencies of target Experiment4Student
[ 25%] Built target Experiment4Student
Consolidate compiler generated dependencies of target gtest
[ 50%] Built target gtest
Consolidate compiler generated dependencies of target gtest_main
[ 75%] Built target gtest_main
Consolidate compiler generated dependencies of target Experiment4Student-main
[100%] Built target Experiment4Student-main
PS D:\VSCodeProjects\C++Course\Experiment4Student>
```

可以看到编译成功，但运行失败

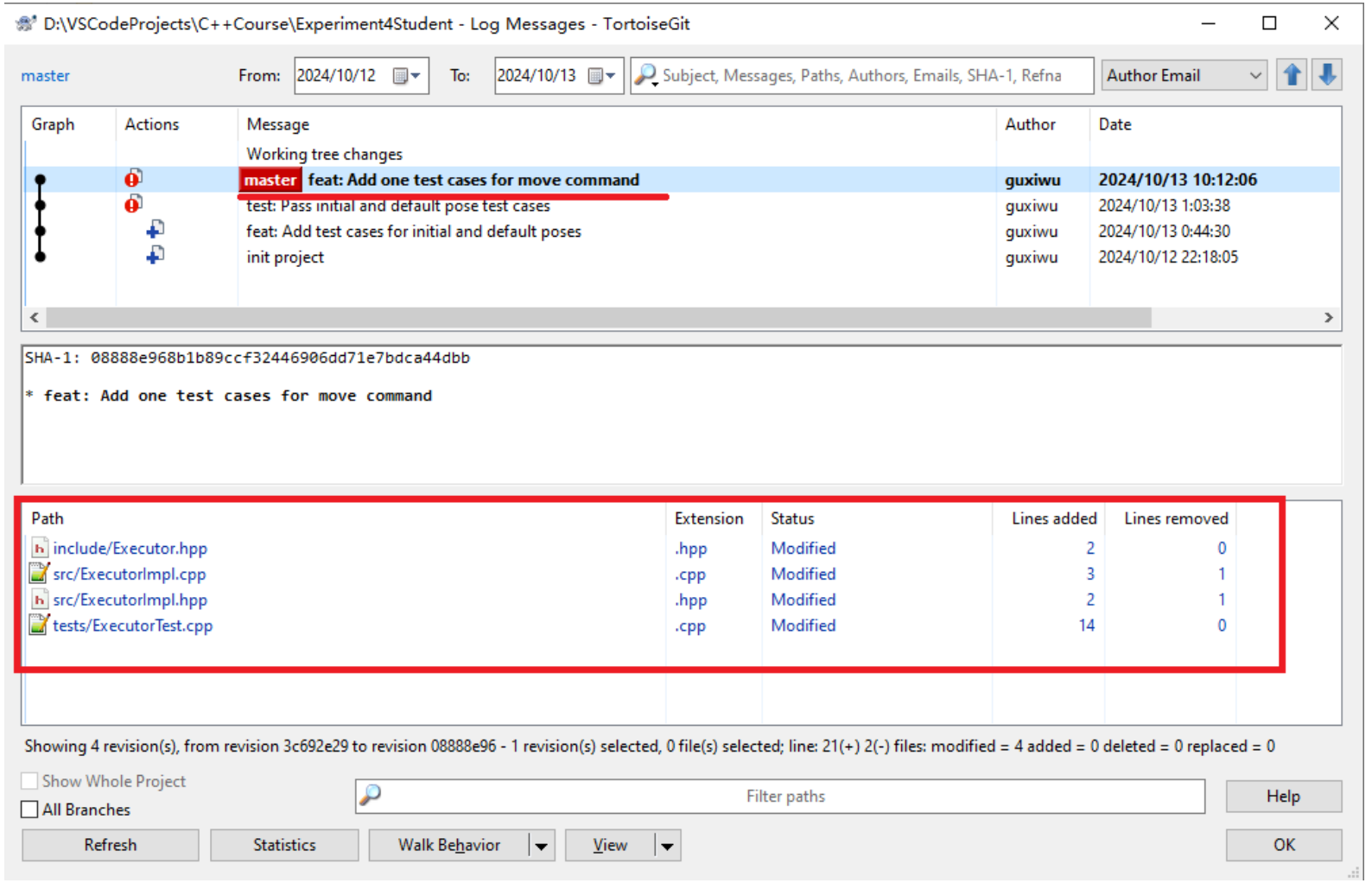
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Active code page: 65001
PS D:\VSCodeProjects\C++Course\Experiment4Student> .\script\run.bat

D:\VSCodeProjects\C++Course\Experiment4Student>rem 自动执行bin目录下的exe文件（如果有多个exe文件，会自动执行最后找到的exe文件）
D:\VSCodeProjects\C++Course\Experiment4Student\bin\Experiment4Student-main.exe
Running main() from D:\VSCodeProjects\C++Course\Experiment4Student\tests\googletest\googletest\src\gtest_main.cc
[==========] Running 3 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 3 tests from ExecutorTest
[ RUN      ] ExecutorTest.should_return_init_pose_when_without_command
[       OK ] ExecutorTest.should_return_init_pose_when_without_command (0 ms)
[ RUN      ] ExecutorTest.should_return_default_pose_when_without_init_and_command
[       OK ] ExecutorTest.should_return_default_pose_when_without_init_and_command (0 ms)
[ RUN      ] ExecutorTest.should_return_x_plus_1_given_command_is_M_and_facing_is_E
D:\VSCodeProjects\C++Course\Experiment4Student\tests\ExecutorTest.cpp:51: Failure
Expected equality of these values:
  target
    Which is: 12-byte object <01-00 00-00 00-00 00-00 45-00 00-00>
  executor->Query()
    Which is: 12-byte object <00-00 00-00 00-00 00-00 45-00 00-00>
[  FAILED  ] ExecutorTest.should_return_x_plus_1_given_command_is_M_and_facing_is_E (23 ms)
[----------] 3 tests from ExecutorTest (64 ms total)

[----------] Global test environment tear-down
[==========] 3 tests from 1 test suite ran. (84 ms total)
[  PASSED  ] 2 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] ExecutorTest.should_return_x_plus_1_given_command_is_M_and_facing_is_E

 1 FAILED TEST
PS D:\VSCodeProjects\C++Course\Experiment4Student>
```

现在将目前的代码提交到Git的本地仓库

打开Terminal，输入下面命令

git add . **#提交修改过或新增的文件到暂存区**

git commit -m "feat: Add one test cases for move command" #将暂存区的文件提交到本地仓库中



修改src下的ExecutorImpl.cpp，添加Execute方法的真正实现

我们需要实现的方法是

```
void ExecutorImpl::Execute(const std::string &commands) noexcept { }
```

该方法执行字符串commands给出的指令序列，指令序列是由三个基本指令M、L、R组成的序列

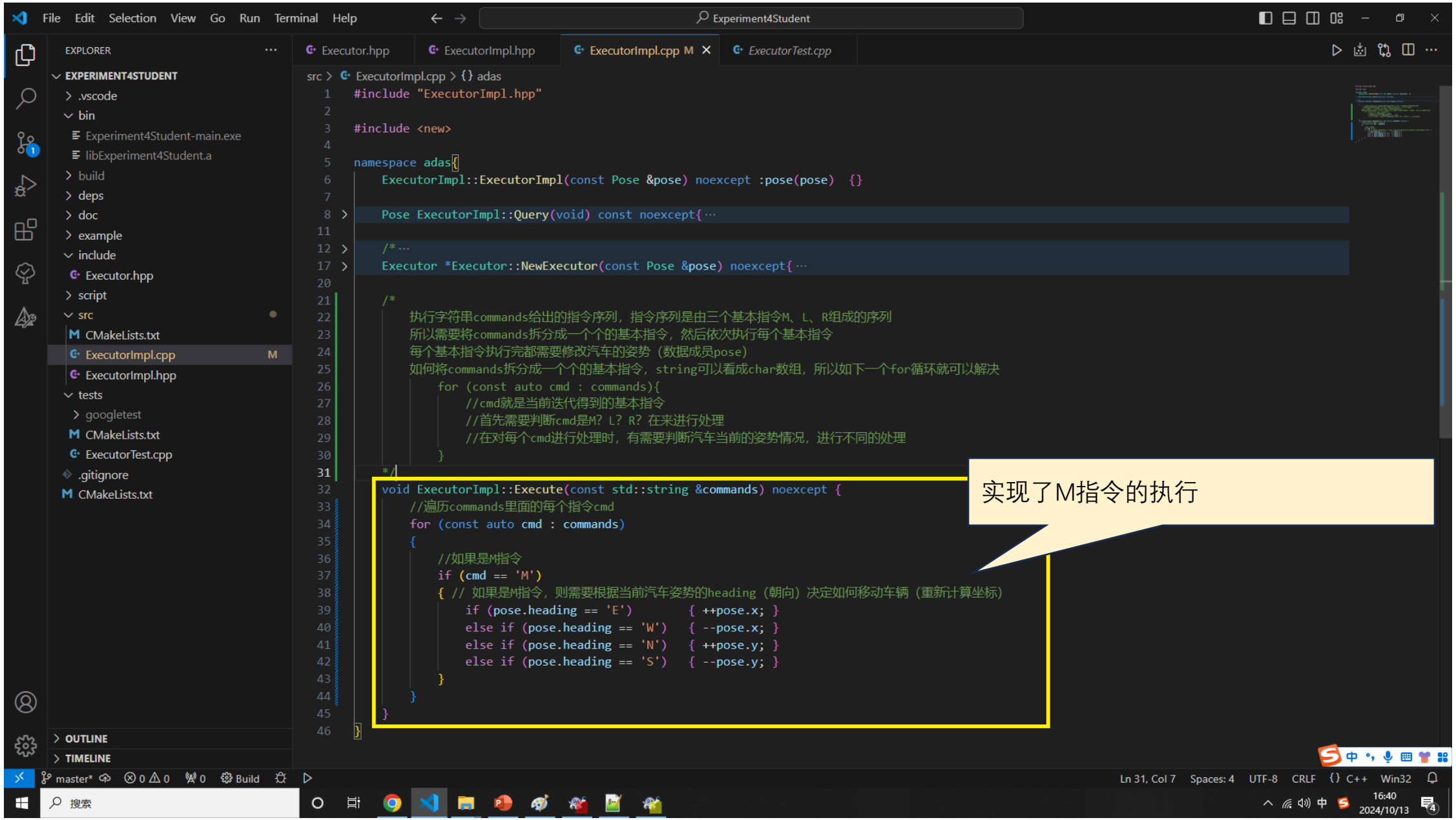
所以需要将commands拆分成一个个的基本指令，然后依次执行每个基本指令

每个基本指令执行完都需要修改汽车的姿势（数据成员pose）

如何将commands拆分成一个个的基本指令，string可以看成char数组，所以如下一个for循环就可以解决

```
for (const auto cmd : commands){
        //cmd就是当前迭代得到的基本指令
        //首先需要判断cmd是M？ L？ R？ 在来进行处理
        //在对每个cmd进行处理时，需要判断汽车当前的姿势情况，进行不同的处理
}
```

修改src下的ExecutorImpl.cpp，添加Execute方法的真正实现



现在来编译和运行工程

打开Terminal，输入下面命令
.\script\build.bat
.\script\run.bat

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Active code page: 65001
PS D:\VSCodeProjects\C++Course\Experiment4Student> .\script\build.bat

D:\VSCodeProjects\C++Course\Experiment4Student>rem 首先配置cmake，然后make编译程序，生成的exe文件放在bin目录下
CMAKE_SOURCE_DIR: D:/VSCodeProjects/C++Course/Experiment4Student
INCLUDE: D:/VSCodeProjects/C++Course/Experiment4Student/includeD:/VSCodeProjects/C++Course/Experiment4Student/src
SOURCE: D:/VSCodeProjects/C++Course/Experiment4Student/src/ExecutorImpl.cpp
CMAKE_CURRENT_SOURCE_DIR: D:/VSCodeProjects/C++Course/Experiment4Student/src
-- Configuring done
-- Generating done
-- Build files have been written to: D:/VSCodeProjects/C++Course/Experiment4Student/build
Consolidate compiler generated dependencies of target Experiment4Student
[ 12%] Building CXX object src/CMakeFiles/Experiment4Student.dir/ExecutorImpl.cpp.obj
[ 25%] Linking CXX static library D:\VSCodeProjects\C++Course\Experiment4Student\bin\libE
[ 25%] Built target Experiment4Student
Consolidate compiler generated dependencies of target gtest
[ 50%] Built target gtest
Consolidate compiler generated dependencies of target gtest_main
[ 75%] Built target gtest_main
Consolidate compiler generated dependencies of target Experiment4Student-main
[ 87%] Linking CXX executable D:\VSCodeProjects\C++Course\Experiment4Student\bin\Experime
[100%] Built target Experiment4Student-main
PS D:\VSCodeProjects\C++Course\Experiment4Student>
```

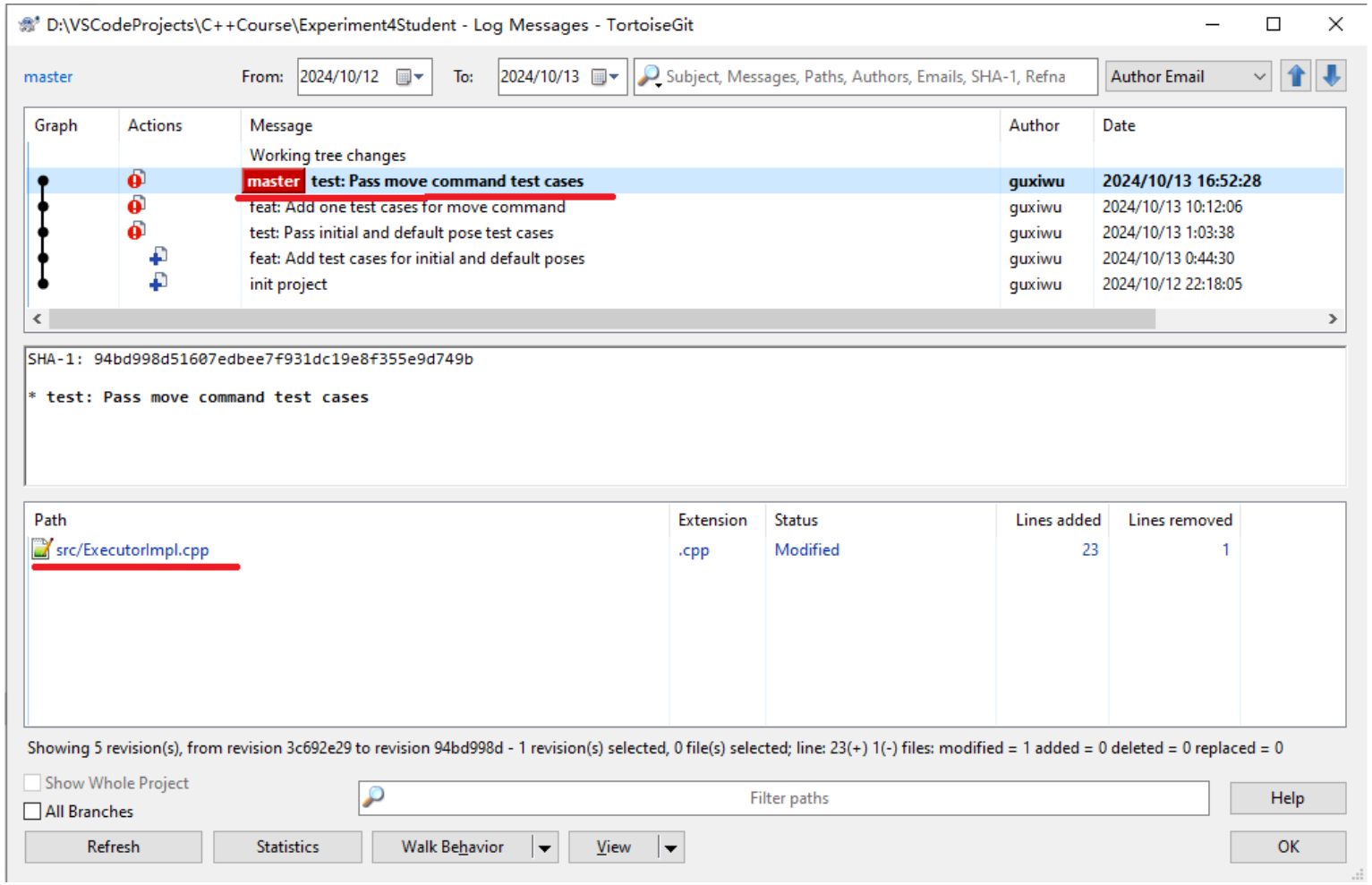
可以三个测试都通过

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Active code page: 65001
PS D:\VSCodeProjects\C++Course\Experiment4Student> .\script\run.bat

D:\VSCodeProjects\C++Course\Experiment4Student>rem 自动执行bin目录下的exe文件（如果有多个exe文件，会自动执行最后找到的exe文件）
D:\VSCodeProjects\C++Course\Experiment4Student\bin\Experiment4Student-main.exe
Running main() from D:\VSCodeProjects\C++Course\Experiment4Student\tests\googletest\googletest\src\gtest_main.cc
[==========] Running 3 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 3 tests from ExecutorTest
[ RUN      ] ExecutorTest.should_return_init_pose_when_without_command
[       OK ] ExecutorTest.should_return_init_pose_when_without_command (0 ms)
[ RUN      ] ExecutorTest.should_return_default_pose_when_without_init_and_command
[       OK ] ExecutorTest.should_return_default_pose_when_without_init_and_command (0 ms)
[ RUN      ] ExecutorTest.should_return_x_plus_1_given_command_is_M_and_facing_is_E
[       OK ] ExecutorTest.should_return_x_plus_1_given_command_is_M_and_facing_is_E (0 ms)
[----------] 3 tests from ExecutorTest (43 ms total)

[----------] Global test environment tear-down
[==========] 3 tests from 1 test suite ran. (64 ms total)
[  PASSED  ] 3 tests.
PS D:\VSCodeProjects\C++Course\Experiment4Student>
```

现在将目前的代码提交到Git的本地仓库

打开Terminal，输入下面命令

```
git add .                                                    #提交修改过或新增的文件到暂存区
git commit -m "test: Pass move command test cases"  #将暂存区的文件提交到本地仓库中
```



剩下需要自己完成的工作

方向	E	W	N	S
M	当前朝向E 执行M X+1	当前朝向W 执行M X-1	当前朝向N 执行M Y+1	当前朝向S 执行M Y-1
L	当前朝向E 执行L 朝向N	当前朝向W 执行L 朝向S	当前朝向N 执行L 朝向W	当前朝向S 执行L 朝向E
R	当前朝向E 执行R 朝向S	当前朝向W 执行R 朝向N	当前朝向N 执行R 朝向E	当前朝向s 执行R 朝向W

三个指令，每个指令四种情况，共12个测试
目前已经完成了M指令的一个测试用例及实现了M指令的执行

请完成M指令另外三个测试用例
请完成L指令四个测试用例，以及实现L指令的执行
请完成R指令四个测试用例，以及实现L指令的执行

每个测试用例的test_name这里都已经给出

移动指令
- should_return_x_plus_1_given_command_is_M_and_facing_is_E
- should_return_x_minus_1_given_command_is_M_and_facing_is_W
- should_return_y_plus_1_given_command_is_M_and_facing_is_N
- should_return_y_minus_1_given_command_is_M_and_facing_is_S

左转指令
- should_return_facing_N_given_command_is_L_and_facing_is_E
- should_return_facing_W_given_command_is_L_and_facing_is_N
- should_return_facing_S_given_command_is_L_and_facing_is_W
- should_return_facing_E_given_command_is_L_and_facing_is_S

右转指令
- should_return_facing_S_given_command_is_R_and_facing_is_E
- should_return_facing_W_given_command_is_R_and_facing_is_S
- should_return_facing_N_given_command_is_R_and_facing_is_W
- should_return_facing_E_given_command_is_R_and_facing_is_N

剩下工作的git提交

剩下工作需要进行如下提交

1） 完成了M指令的另外三个测试用例，运行成功以后

```
git add .
git commit -m "test: Add and Pass all move command test cases"
```

2)完成L指令四个测试用例，但还没有实现L指令的执行。这时编译成功，测试失败

```
git add .
git commit -m "feat: Add test cases for turn left command"
```

3）实现L指令的执行， L指令四个测试用例测试通过

```
git add .
git commit -m "test: Pass turn left command test cases"
```

4）完成R指令四个测试用例，但还没有实现R指令的执行。这时编译成功，测试失败

```
git add .
git commit -m "feat: Add test cases for turn right command"
```

5）实现R指令的执行， L指令四个测试用例测试通过

```
git add .
git commit -m " test: Pass turn right command test cases"
```

实验检查

因为每个同学的实验工程下都有git目录，因此每次实验完成后，需要提交工程（打包）到学习平台。老师和助教会解压每个同学的工程，首先会测试运行所有的测试用例，其次会检查代码的编写质量，最后会用TortoiseGit去检查每个同学的提交日志，看是否符合要求

实验报告和实验评测

由于四次实验是逐次递进，所以最后实验完成后，针对最终完成的实验内容撰写实验报告。

实验分数最终由：实验考勤(10%)+实验完成情况（60%）+实验报告（30%）组成

其中实验完成情况的分数：根据检查每次实验提交的工程（包括运行测试程序，检查代码质量，检查代码提交日志）的情况来评定。每次实验都会检查一次，四次实验完成情况的平均分作为实验完成情况的分数。