使用VS Code进行Qt开发的实现

Qt Creator界面不美观，而VS Code更漂亮一些。

因为Qt5支持使用CMake进行构建，而VS Code也可以支持CMake构建系统，因此是完全可以的。

测试环境

Qt 5.15.0
CMake 3.17.5
Visual Studio 2019 16.7.5（使用C++的桌面开发）
Visual Studio Code 1.49.3

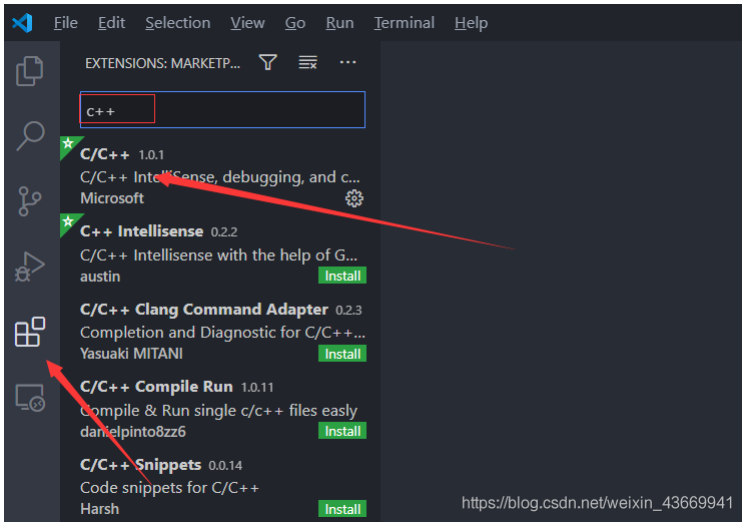
步骤

1. 将Qt的bin目录添加到环境变量

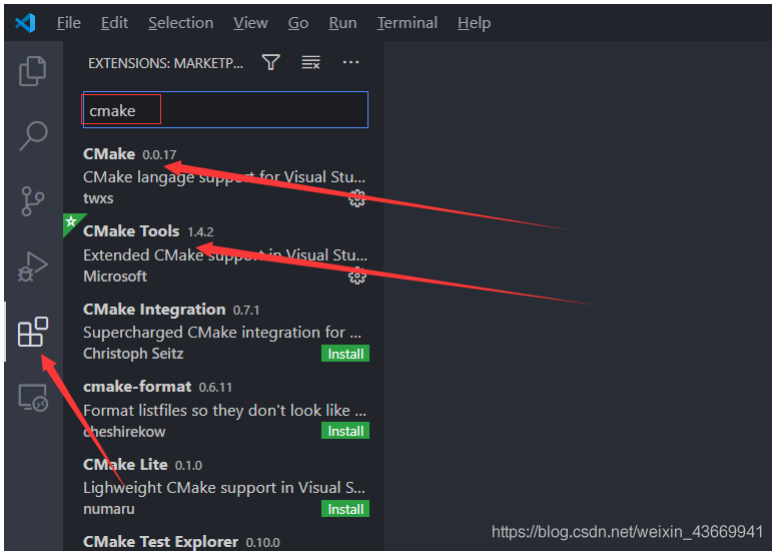
假设Qt安装在C:\Qt，那么将C:\Qt\5.15.0\msvc2019_64\bin添加到环境变量。

2. 安装VS Code扩展

在扩展商店搜索c++，安装微软发行的C/C++扩展。

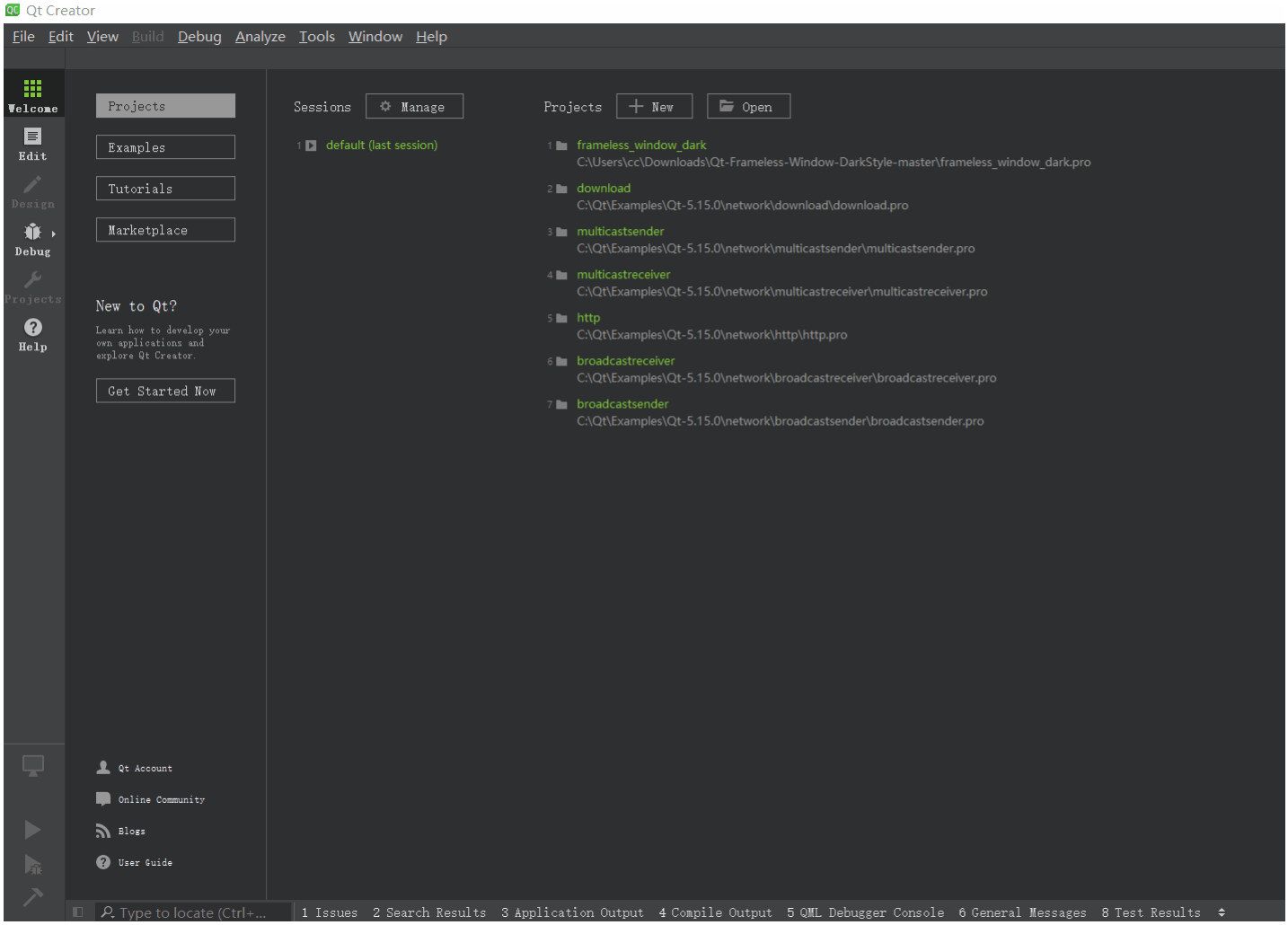


在扩展商店搜索cmake，安装前两个扩展，分别为CMake、CMake Tools。



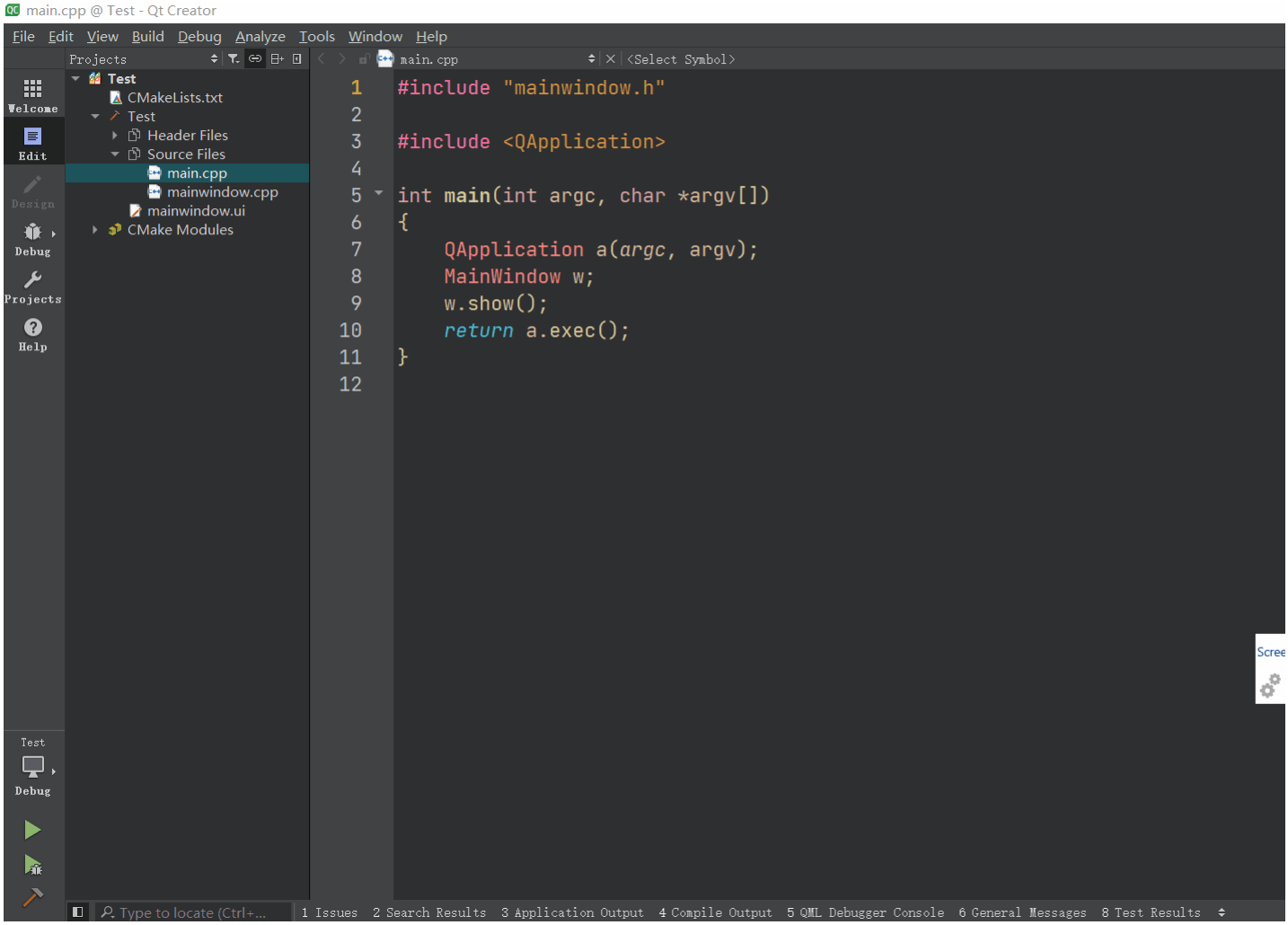
3. 使用Qt Creator创建CMake项目

使用Qt Creator创建一个测试项目，如图所示：



4. 在VS Code中打开项目

用VS Code打开CMakeLists.txt文件所在目录，如图所示：

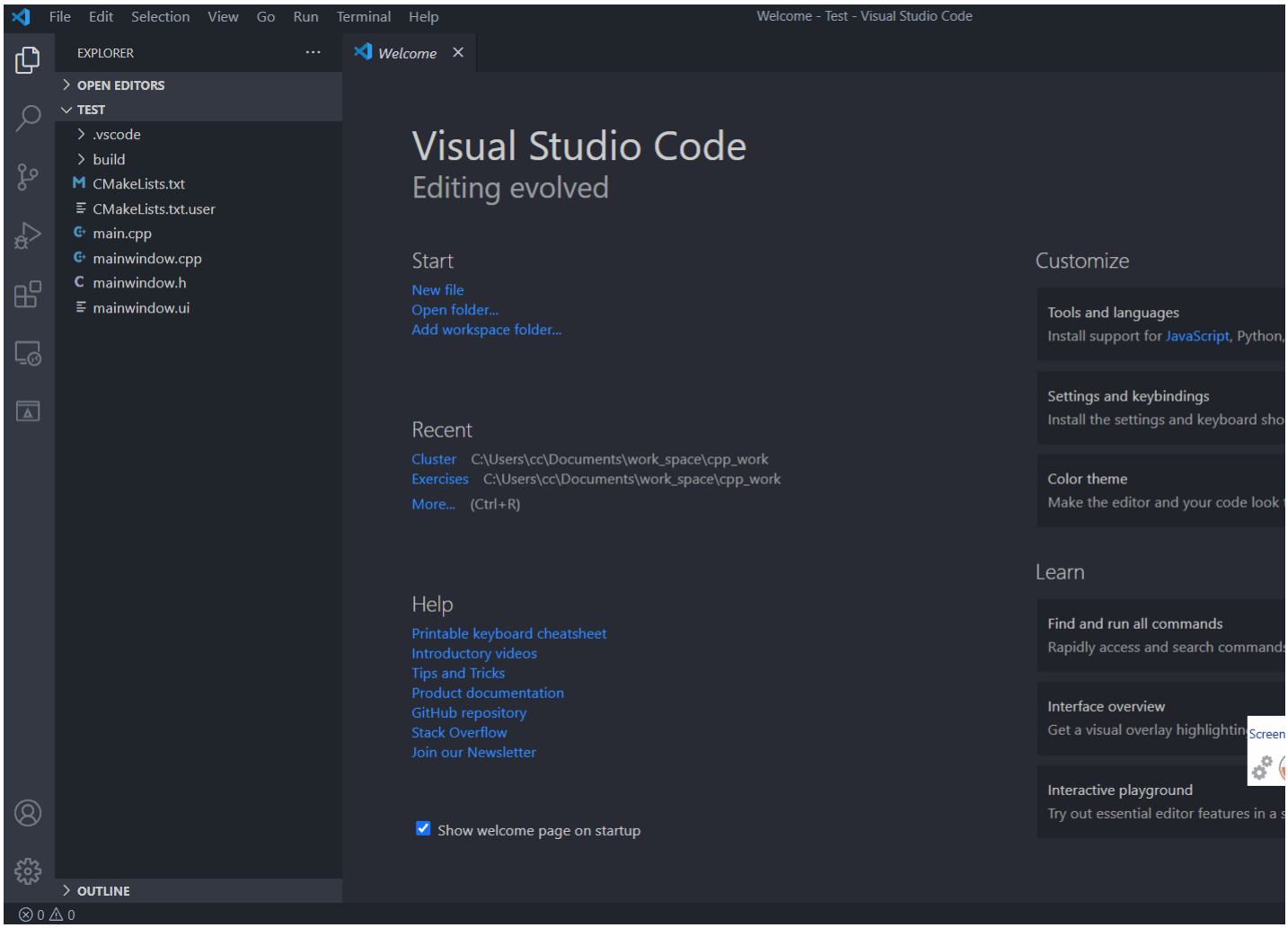


需要选择一个kit，我是用的是VS 2019的工具包，也可以使用MinGW，具体请参考CMake扩展的官方文档。

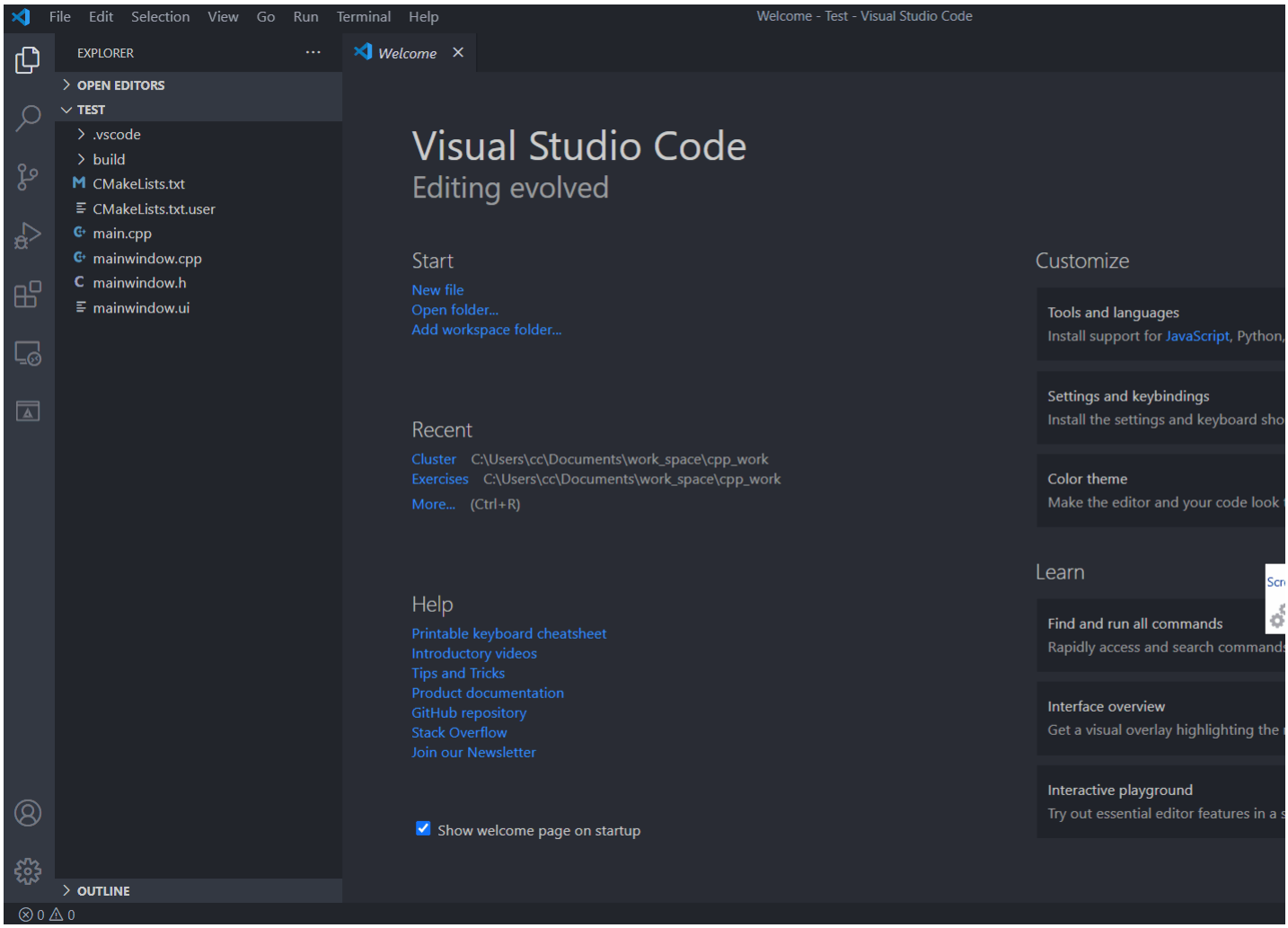
提示是否配置智能感知，选择是。

5. 构建并运行项目

按F7进行构建：

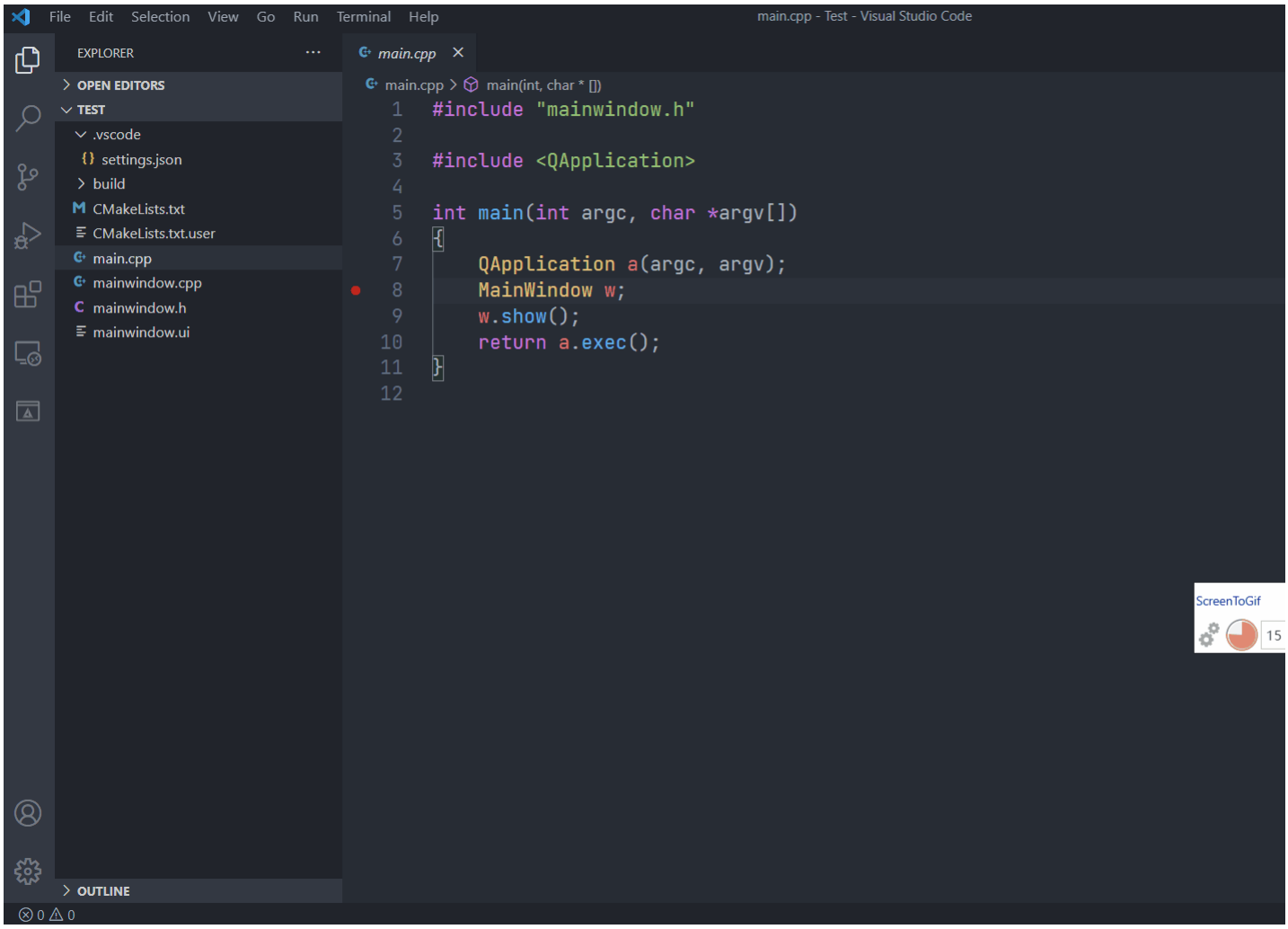

File Edit Selection View Go Run Terminal Help
Welcome - Test - Visual Studio Code
EXPLORER
OPEN EDITORS
TEST
.vscode
build
CMakeLists.txt
CMakeLists.txt.user
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
OUTLINE
Welcome
Visual Studio Code
Editing evolved
Start
New file
Open folder...
Add workspace folder...
Recent
Cluster C:\Users\cc\Documents\work_space\cpp_work
Exercises C:\Users\cc\Documents\work_space\cpp_work
More... (Ctrl+R)
Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Join our Newsletter
Show welcome page on startup
Customize
Tools and languages
Install support for JavaScript, Python,
Settings and keybindings
Color theme
Make the editor and your code look
Learn
Find and run all commands
Rapidly access and search commands
Interface overview
Interactive playground
Try out essential editor features in a
0 0

按Shift + F5运行：



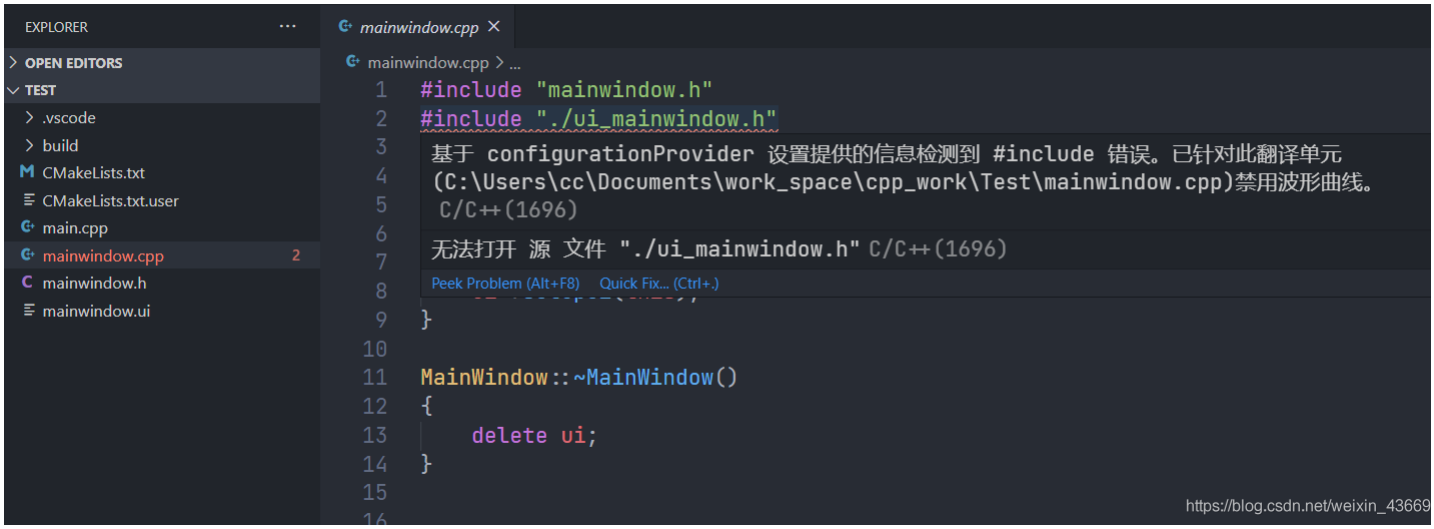
7. 调试项目

添加断点，按下Ctrl + F5进行调试：



一些小问题

上述所有步骤完成后，已经可以正常编码和运行，但智能感知有一点问题，如图：



原因在于Qt生成的ui文件没有被包含到智能感知的include目录中。

查询了CMake文档后发现ui文件所在目录会被添加到目标属性的include目录属性中：

AUTOUIC

The AUTOUIC target property controls whether cmake(1) inspects the C++ files in the target to determine if they require uic to be run, and to create rules to execute uic at the appropriate time.

If a preprocessor #include directive is found which matches <path>ui_<basename>.h, and a <basename>.ui file exists, then uic will be executed to generate the appropriate file. The <basename>.ui file is searched for in the following places

1. <source_dir>/<basename>.ui
2. <source_dir>/<path><basename>.ui
3. <AUTOUIC_SEARCH_PATHS>/<basename>.ui
4. <AUTOUIC_SEARCH_PATHS>/<path><basename>.ui

where <source_dir> is the directory of the C++ file and AUTOUIC_SEARCH_PATHS is a list of additional search paths.

The generated generated ui_*.h files are placed in the <AUTOGEN_BUILD_DIR>/include directory which is automatically added to the target's INCLUDE_DIRECTORIES.

- This differs from CMake 3.7 and below; see their documentation for details.
- For multi configuration generators, the include directory is <AUTOGEN_BUILD_DIR>/include_<CONFIG>.
- See AUTOGEN_BUILD_DIR.

https://blog.csdn.net/weixin_43669941

但实际验证发现并没有，所以我们还需要手动添加这个属性。

假设生成的目标为Test，在CMakeLists.txt文件的最后一行添加：

```
target_include_directories(Test PRIVATE "${CMAKE_BINARY_DIR}/Test_autogen/include_Debug")
```

最终的CMakeLists.txt文件内容为：

```
cmake_minimum_required(VERSION 3.5)

project(Test LANGUAGES CXX)

set(CMAKE_INCLUDE_CURRENT_DIR ON)

set(CMAKE_AUTOUIC ON)
set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTORCC ON)

set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

# QtCreator supports the following variables for Android, which are identical to qmake Android variables.
# Check http://doc.qt.io/qt-5/deployment-android.html for more information.
# They need to be set before the find_package(Qt5 ...) call.

#if(ANDROID)
# set(ANDROID_PACKAGE_SOURCE_DIR "${CMAKE_CURRENT_SOURCE_DIR}/android")
# if (ANDROID_ABI STREQUAL "armeabi-v7a")
# set(ANDROID_EXTRA_LIBS
# ${CMAKE_CURRENT_SOURCE_DIR}/path/to/libcrypto.so
# ${CMAKE_CURRENT_SOURCE_DIR}/path/to/libssl.so)
# endif()
#endif()

find_package(QT NAMES Qt6 Qt5 COMPONENTS Widgets REQUIRED)
find_package(Qt${QT_VERSION_MAJOR} COMPONENTS Widgets REQUIRED)

if(ANDROID)
add_library(Test SHARED
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
)
else()
add_executable(Test
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
)
endif()

target_link_libraries(Test PRIVATE Qt${QT_VERSION_MAJOR}::Widgets)

target_include_directories(Test PRIVATE "${CMAKE_BINARY_DIR}/Test_autogen/include_Debug")
```

智能感知正常工作：

您可能感兴趣的文章:vscode+PyQt5安装详解步骤PYQT5 vscode联合操作qtdesigner的方法