Installation under Windows with WSL and Ubuntu 22.04 LTS

The instructions are exactly the same as for the `VegaP` project. Skip the installation part and start with the Section **Install course package**.

Please follow the instructions carefully, line-by-line. This is not the time to be creative!

Install Visual Studio Code, WSL, and Ubuntu 22.04

Follow instructions available online, for instance, Developing in WSL.

Setup Ubuntu

Type in the Ubuntu terminal the following commands, one command per line:

```
sudo apt update
sudo apt upgrade
sudo apt install g++
sudo apt install gdb
sudo apt install cmake
sudo apt install ninja-build
sudo apt install pkg-config
sudo apt install libgsl-dev
sudo apt install doxygen
sudo apt install graphviz
```

After the first `sudo`, you will have to enter the Ubuntu password created during the installation. After every command, you will be asked a permission to proceed. Reply `Y` for yes. Some code will run. Wait for it to finish and move to the next command.

To check that everything is correctly installed, type in the Ubuntu terminal:

```
g++ --version
gdb --version
cmake --version
ninja --version
doxygen --version
pkg-config --version
```

```
gsl-config --version
dot -V
```

pressing `(Enter)` after every line. If the name is not a recognized command, then check your installations.

Configure Visual Studio Code with WSL

In Ubuntu terminal type:

```
code
```

This command will start Visual Studio Code. When doing this for the first time, you should see Visual Studio Code fetching components needed to run in WSL. This should only take a short while, and is only needed once. Add `C/C++ Extension Pack` following the link:

ms-vscode.cpptools-extension-pack

Close Visual Studio Code and the Ubuntu terminal.

Install course package

1. Create a directory, where you will keep the material related to the course. Avoid spaces in the full path. For instance, get something like `C:\Vega`. Hereafter we call this directory `Vega`.

2. Place file `VegaQ.zip` in directory `Vega` and extract it. Check that the directory tree looks like `Vega:\VegaQ\test` (not as `Vega:\VegaQ\VegaQ\test`).

3. Open directory `Vega` with Command Prompt or Power Shell. Type

   ```
   dir
   ```

   You will see `VegaQ`. Type

   ```
   wsl
   ```

   This will open Ubuntu terminal. Type

   ```
   code VegaQ
   ```

   This command opens `VegaQ` project from Visual Studio Code and starts its configuration. You will be asked to select a Kit for VegaQ. Choose[1]

[1] The version number may be different. It should work just fine.

`GCC 11.3.0 Using compilers:  C = /bin/gcc, CXX=usr/bin/g++`

If everything goes well, you will see the output of the kind:

`[cmake] -- The CXX compiler identification is GNU 11.3.0`

4. In the future, to get back to your project, just open Visual Studio Code. It remembers the last state.

5. In file \VegaQ\CMakeLists.txt, around line 40, type "YOUR_ID" instead of "Vega". Later, "YOUR_ID" will be the name of your team in Quantathon. You will choose it with your teammates.

6. Build all projects with `(Shift+F7)` and then choose `(all)`. Sometimes you have to do it a couple of times to clear the errors. Help files in `.html` format will appear in \VegaQ\build\doc. You may want to bookmark some of them for a quick access from your browser.

7. Run project `quantExamples` with `(Shift+F5)`. Text file quantExamples.txt will be created in directory Vega:\VegaQ\build\output\quantExamples. "YOUR_ID" will appear on the first line.

8. Debug project `quantExamples` with `(Ctrl+F5)`. The same text file will be created, but the dialog will look different. You may have to do it a couple of times to get a result. The debug mode allows you to add breakpoints with `(F9)` and then track the values of variables.

9. Check the instructions for `CMake Tools` following the link. Skip all sections related to CMake, just learn how to configure, build, and debug.

10. Useful shortcuts:

    `(Ctrl+Shift+P)` opens Command Palette. Type `CMake` to get the commands from CMake Tools. Command Palette remembers the commands used recently. It is my preferred way to work with the Visual Studio Code.

    `(Shift+F7)` allows you to select and build a specific target.

    `(F7)` builds the active build target. You can select the build target by opening Command Palette with `(Ctrl+Shift+P)` and then typing `(CMake: Set Build Target)`.

    `(Ctrl+F5)` debugs the active launch or debug target. You can select the launch target by opening Command Palette with `(Ctrl+Shift+P)` and then typing `(CMake: Set Debug Target)`.

`(Shift+F5)` runs the active launch or debug target.

`(Ctrl+Shift+I)` formats your code to look nice.

Remark 1. Do not use the default debug command initiated by `(F5)`. Press instead `(Ctrl+F5)`. This way, CMake takes care of all the settings.

Remark 2. If `CMake` misbehaves, then do

Soft reset: `(Ctrl+Shift+P)` + `(CMake: Delete Cache and Reconfigure)` + `(CMake: Clean Rebuild)`.

Hard reset: close `Visual Studio Code`, delete directory Vega:\VegaQ\build, and restart `Visual Studio Code`.