Installation of "Financial Computing with C++" course package under Windows 10 with WSL and Ubuntu 20.04 LTS

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

Install Visual Studio Code, WSL, and Ubuntu 20.04

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 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.

You can save a little bit of time by installing all the packages simultaneously:

```
sudo apt instal g++ gdb cmake pkg-config libgsl_dev doxygen graphviz
```

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

```
g++ --version
gdb --version
cmake --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 the extensions following the links:

1. C/C++ (ms-vscode.cpptools),
2. CMake (twxs.cmake),
3. CMake Tools (ms-vscode.cmake-tools).

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:\FC`. Hereafter we call this directory `FC`.
2. Place file `CPP.zip` in directory `FC` and extract it. Check that the directory tree looks like `FC:\CPP\cfl` (not as `FC:\CPP\CPP\cfl`).
3. Open directory `FC` with Command Prompt or Power Shell. Type

   ```
   dir
   ```

   You will see `\CPP`. Type

   ```
   wsl
   ```

   This will open Ubuntu terminal. Type

```
code CPP
```

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

```
GCC for c99 9.3.0 Using compilers: C = /bin/c99-gcc
```

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

```
[cmake] -- The CXX compiler identification is GNU 9.3.0
```

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

5. In file \CPP\CMakeLists.txt, around line 32, type "YOUR_ID" instead of "kramkov".

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 \CPP\build\doc. You may want to bookmark some of them for a quick access from your browser.

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

8. Debug project `Examples` 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)` builds a specific project.

`(Ctrl+F5)` debugs the active project.

`(Shift+F5)` runs the active project without debugging.

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 FC:\CPP\build, and restart `Visual Studio Code`.