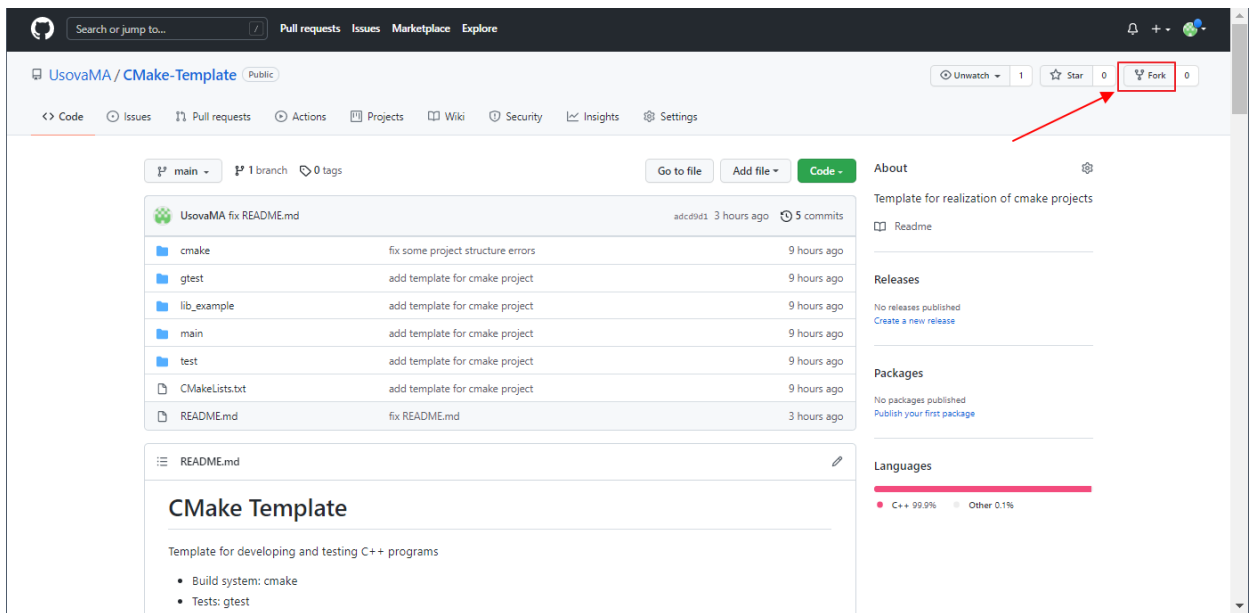


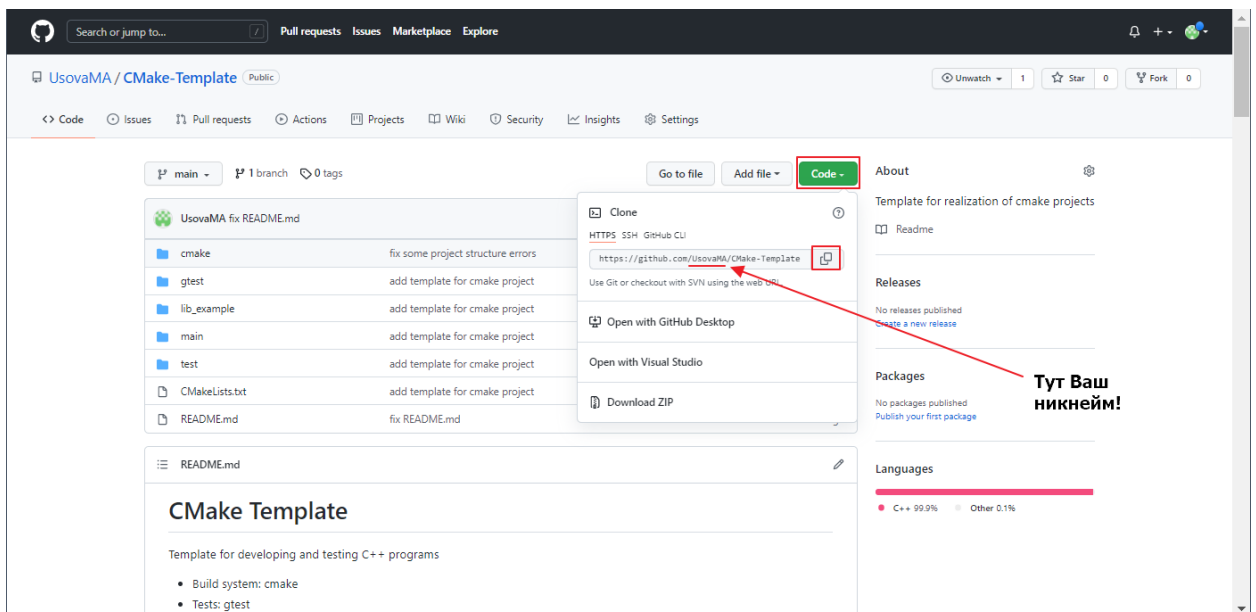
# Инструкция по CMake

- 1) Сделать fork проекта на GitHub <https://github.com/UsovaMA/CMake-Template>.



- 2) Клонировать свой fork проекта на компьютер (назовём его локальный репозиторий, а GitHub – удалённый репозиторий).

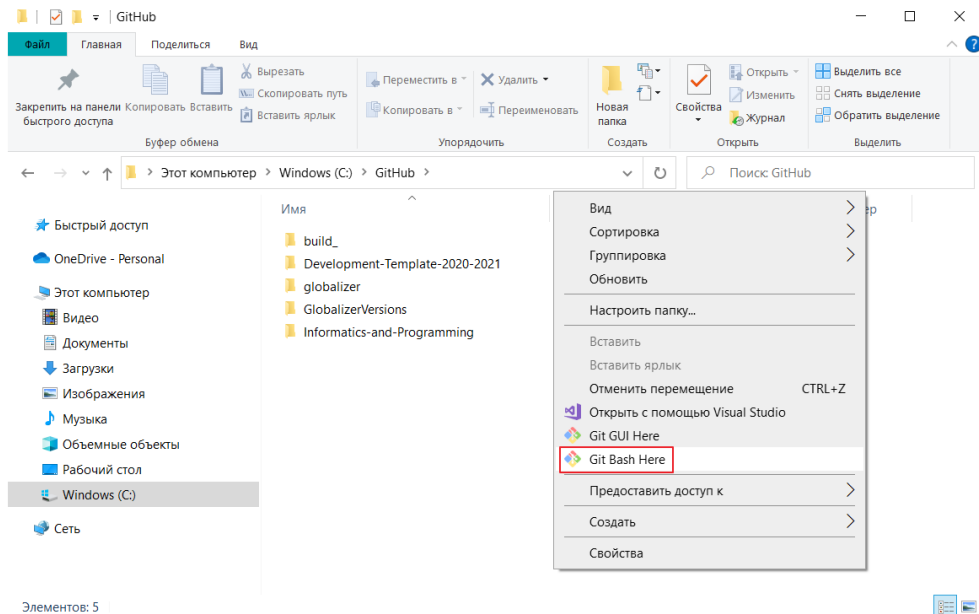
а. Получить ссылку на Ваш удалённый репозиторий



- б. С помощью командной строки клонировать удалённый репозиторий, перейти в создававшуюся папку проекта с помощью команды `cd`, сразу переключиться в ветку разработки, чтобы не забыть и не начать случайно делать изменения в ветке `main/master`. См. рисунки ниже с примером выполнения данных команд.

**Замечание.** Выберите для своих GitHub-проектов правильное место на своём компьютере. В пути к папкам не должно быть пробелов, русских букв. Старайтесь привыкнуть давать названия только на английском и с использованием знаков – и `_`. Примеры:

- хорошо – CMake-Template, CMakeTemplate, CMake\_Template,
- плохо – CMake Template, CMake-шаблон.



```

MINGW64:/c/GitHub/CMake-Template

oppab@LAPTOP-VB10NRU3 MINGW64 /c/GitHub (master)
$ git clone https://github.com/UsovaMA/CMake-Template.git
Cloning into 'CMake-Template'...
remote: Enumerating objects: 140, done.
remote: Counting objects: 100% (140/140), done.
remote: Compressing objects: 100% (76/76), done.
Receiving objects: 74% remote: Total 140 (delta 51), reused 132 (delta 46), pack-reused
0(104/140), 49.96 MiB | 1.33 MiB/s, done.
Receiving objects: 100% (140/140), 50.07 MiB | 1.32 MiB/s, done.
Resolving deltas: 100% (51/51), done.

oppab@LAPTOP-VB10NRU3 MINGW64 /c/GitHub (master)
$ cd CMake-Template/

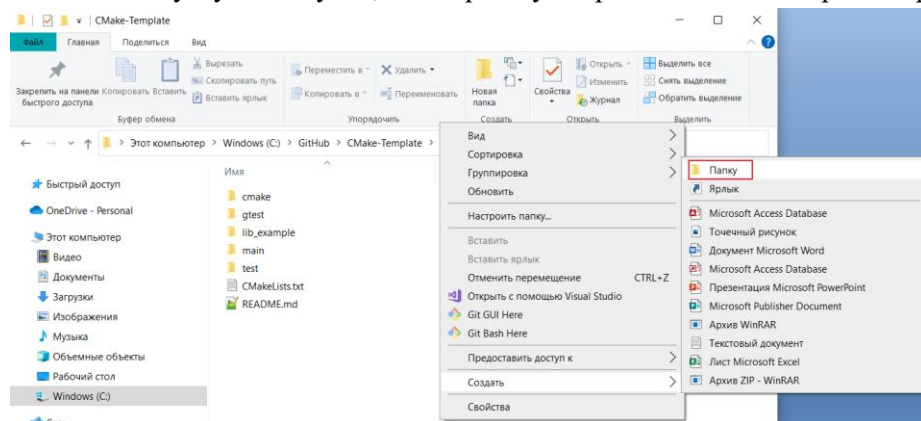
oppab@LAPTOP-VB10NRU3 MINGW64 /c/GitHub/CMake-Template (main)
$ git checkout -b development
Switched to a new branch 'development'

oppab@LAPTOP-VB10NRU3 MINGW64 /c/GitHub/CMake-Template (development)
$
  
```

Более подробно о командах, вариантах работы с системой git, часто встречаемых ошибках и проблемах читать в описании проекта (файл README.md) - <https://github.com/UsovaMA/CMake-Template/blob/main/README.md>.

3) Попробовать собрать шаблон таким, какой он есть

а. Создайте пустую папку sln, в которой будет располагаться собранное решение.



б. Из данной папки вызовете командную строку (вводим cmd вместо пути).

- с. Далее вводим команду для сборки проекта **cmake -G "Visual Studio 16 2019" ..**  
В команде указываем версию студии, которая у вас установлена.

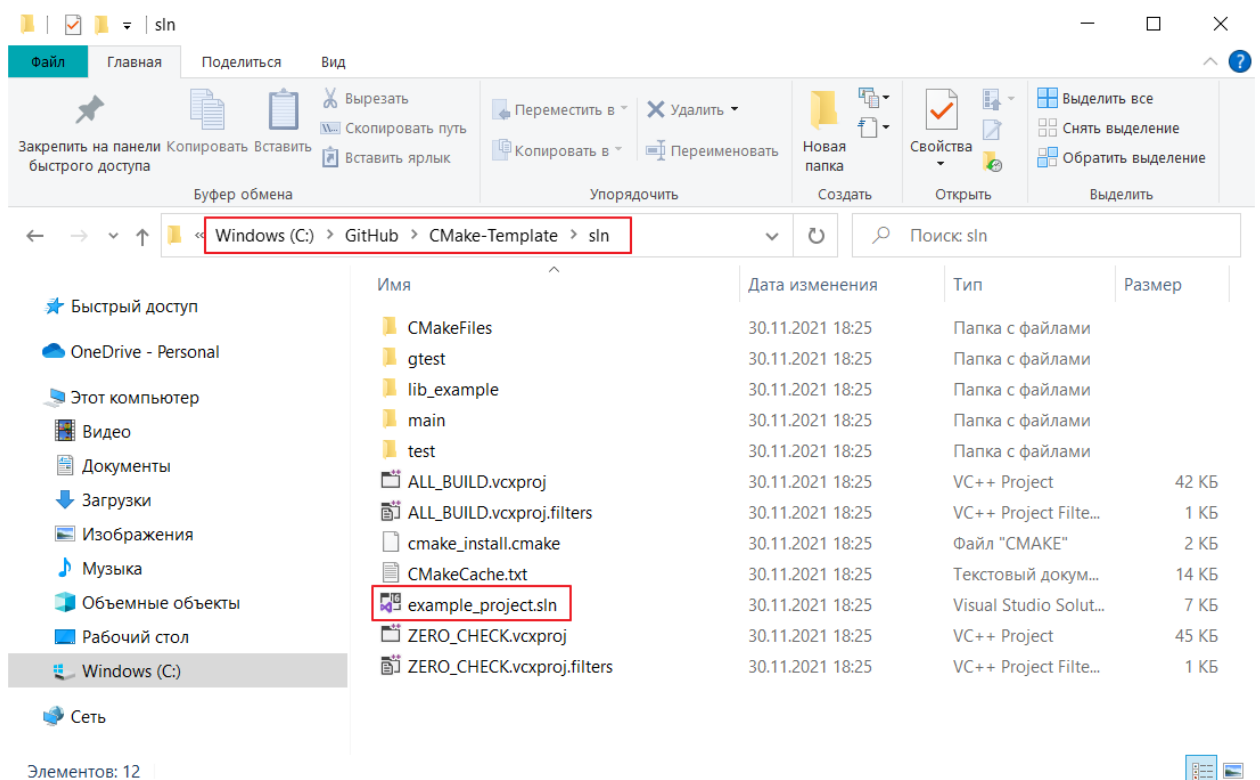
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.1348]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\GitHub\CMake-Template\sln>cmake -G "Visual Studio 16 2019" ..
-- The C compiler identification is MSVC 19.29.30038.1
-- The CXX compiler identification is MSVC 19.29.30038.1
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual Studio/2019/Community/VC/Tools/MSVC/14.29.30037/bin/Hostx64/x64/cl.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual Studio/2019/Community/VC/Tools/MSVC/14.29.30037/bin/Hostx64/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for pthread.h
-- Looking for pthread.h - not found
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: C:/GitHub/CMake-Template/sln

C:\GitHub\CMake-Template\sln>
```

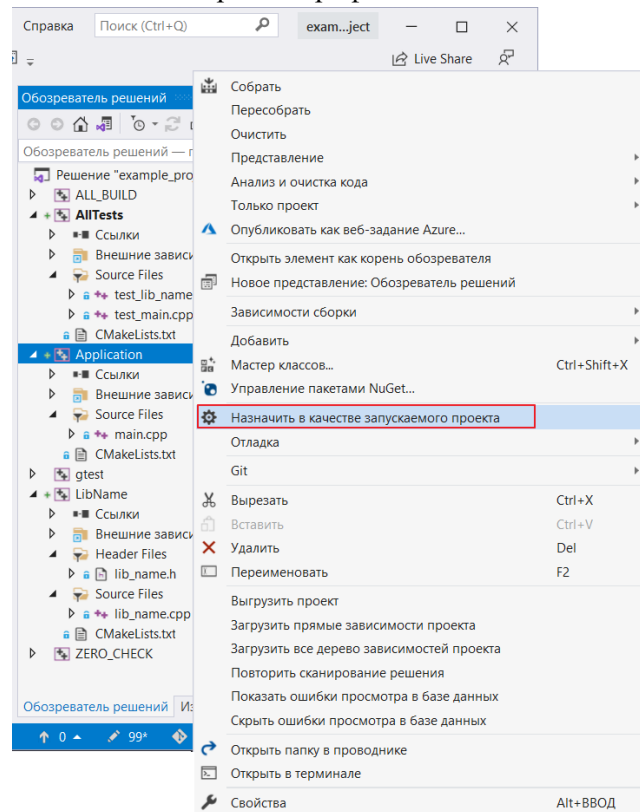
Смотрим на выведенную информацию – ошибок быть не должно.

- d. Проверяем, содержимое папки **sln**, там должен появиться собранный проект, запускаем **.sln** файл.



- е. Назначаем автозапускаемыми проектами поочередно **LibName** и **AllTests**, смотрим что всё корректно работает. Результаты удачной отработки проекта-шаблона представлены на рисунках ниже.

**Указания.** Правая кнопка мыши по проекту -> Назначить автозапускаемым проектом.  
Автозапускаемый проект выделяется жирным шрифтом.



```
Консоль отладки Microsoft Visual Studio
1 / 4 = 0.25

C:\GitHub\CMake-Template\s\main\Debug\Application.exe (процесс 17672) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" ->
"Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

```
Консоль отладки Microsoft Visual Studio

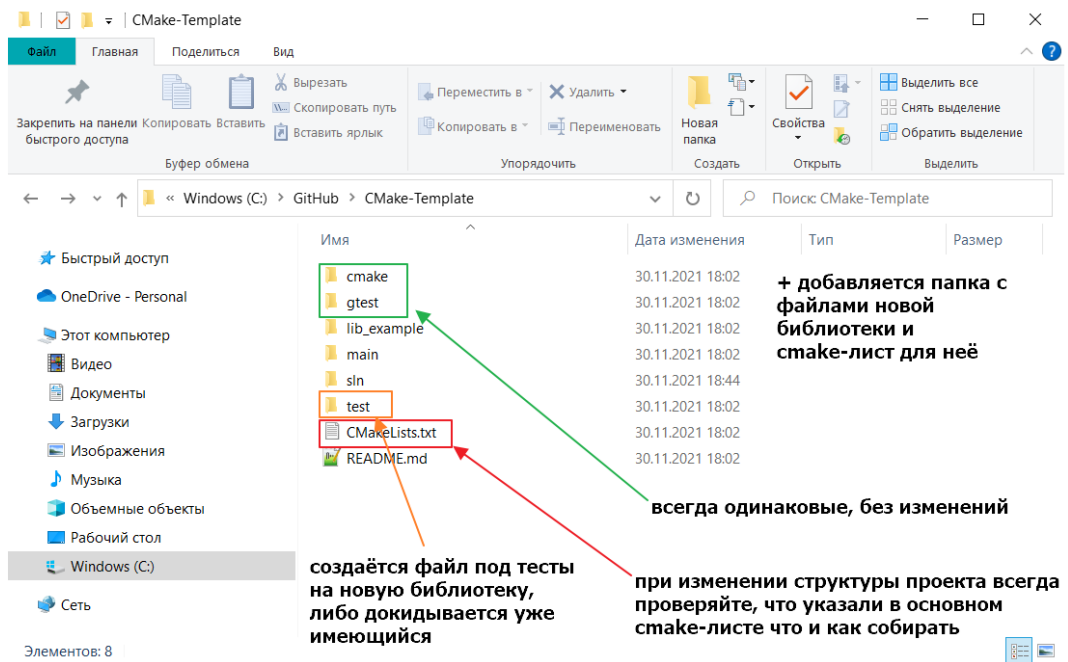
[=====] Running 4 tests from 1 test case.
[=====] Global test environment set-up.
[=====] 4 tests from test_lib
[ RUN      ] test_lib.can_div_test
[ OK       ] test_lib.can_div_test (0 ms)
[ RUN      ] test_lib.can_div_correctly_test
[ OK       ] test_lib.can_div_correctly_test (0 ms)
[ RUN      ] test_lib.can_div_correctly_with_remainder_test
[ OK       ] test_lib.can_div_correctly_with_remainder_test (0 ms)
[ RUN      ] test_lib.throw_when_try_div_by_zero_test
[ OK       ] test_lib.throw_when_try_div_by_zero_test (1 ms)
[=====] 4 tests from test_lib (2 ms total)

[=====] Global test environment tear-down
[=====] 4 tests from 1 test case ran. (4 ms total)
[ PASSED  ] 4 tests.

C:\GitHub\CMake-Template\s\test\Debug\AllTests.exe (процесс 16800) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" ->
"Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

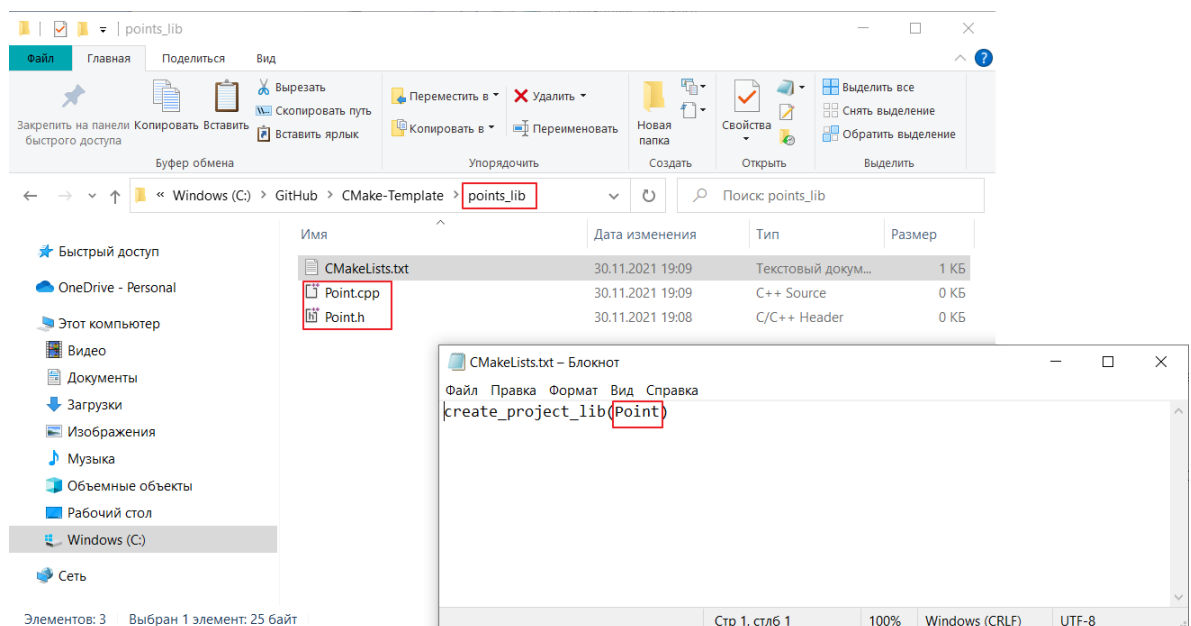
- 4) После учебной сборки поправить шаблон под свой проект. Я привожу пример с добавлением **ещё одной** библиотеки в проект. **Помимо уже имеющейся!** Шаблонную статическую библиотеку я оставляю. Убрать её из проекта можно в любое время, но пусть она будет перед глазами как пример оформления.

**Замечание.** Если вы хотите снести шаблонную библиотеку и вместо неё сделать свою, то достаточно переименовывания соответствующих файлов и названий в stake-листах, чтобы сделать рабочую заготовку.

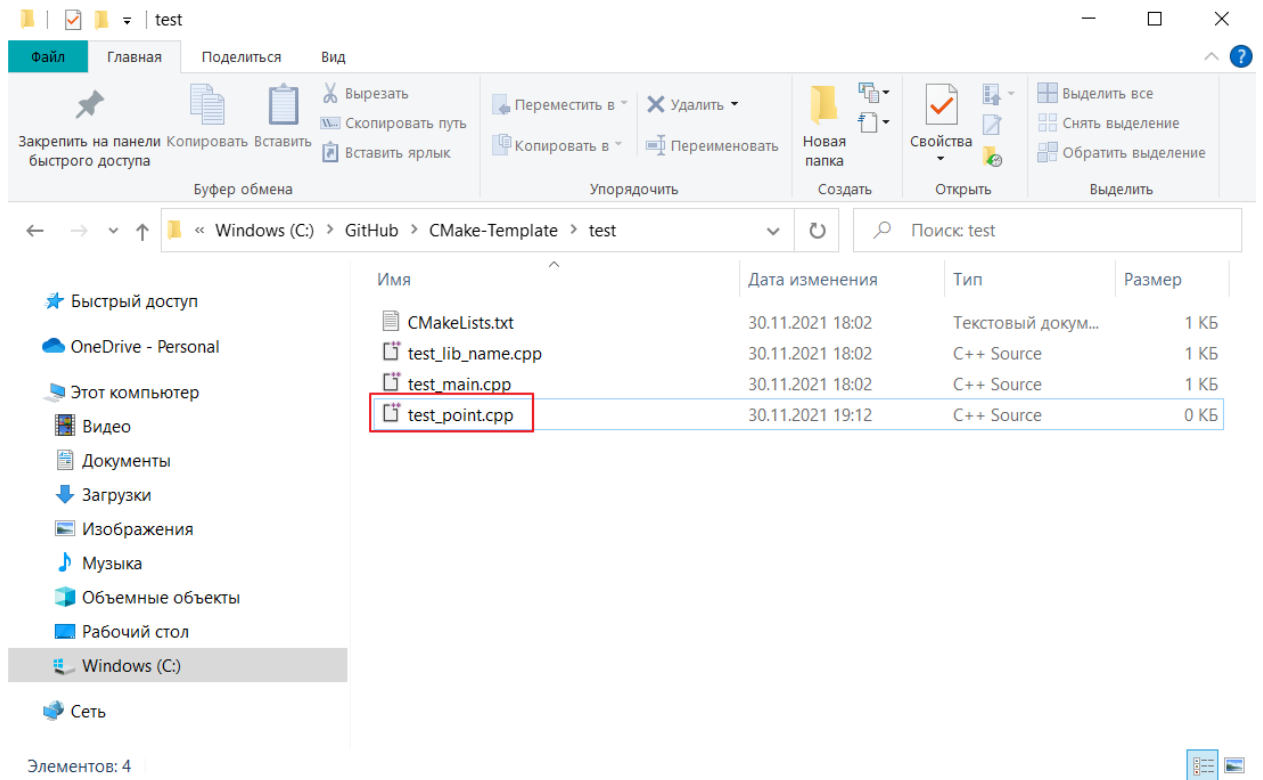


### Подробно действия, которые можно/нужно проделать:

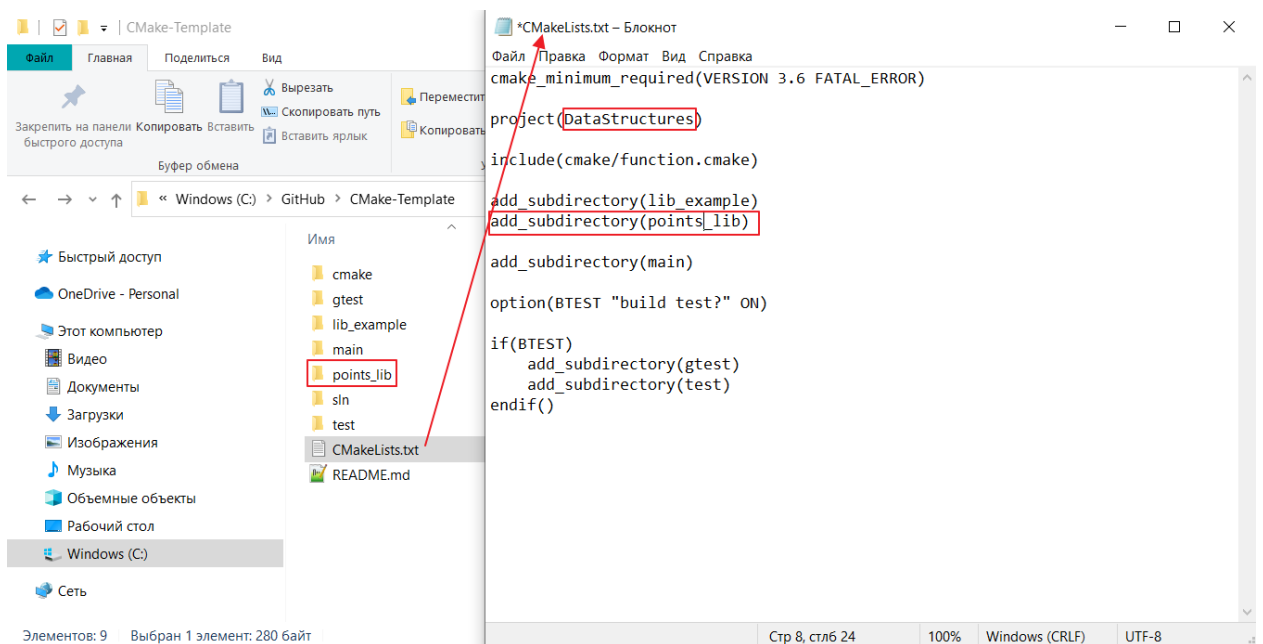
- По необходимости меняем названия в stake-листах, названия вы определяете сами, сделайте свой наиболее понятный и удобный лично Вам шаблон;
- Добавляем папку под новую библиотеку (points\_lib), размещаем в ней файлы (уже существующие или пустые/новые), создаём stake-лист (вызываем функцию создания статической библиотеки)



- Создать файл для тестов:

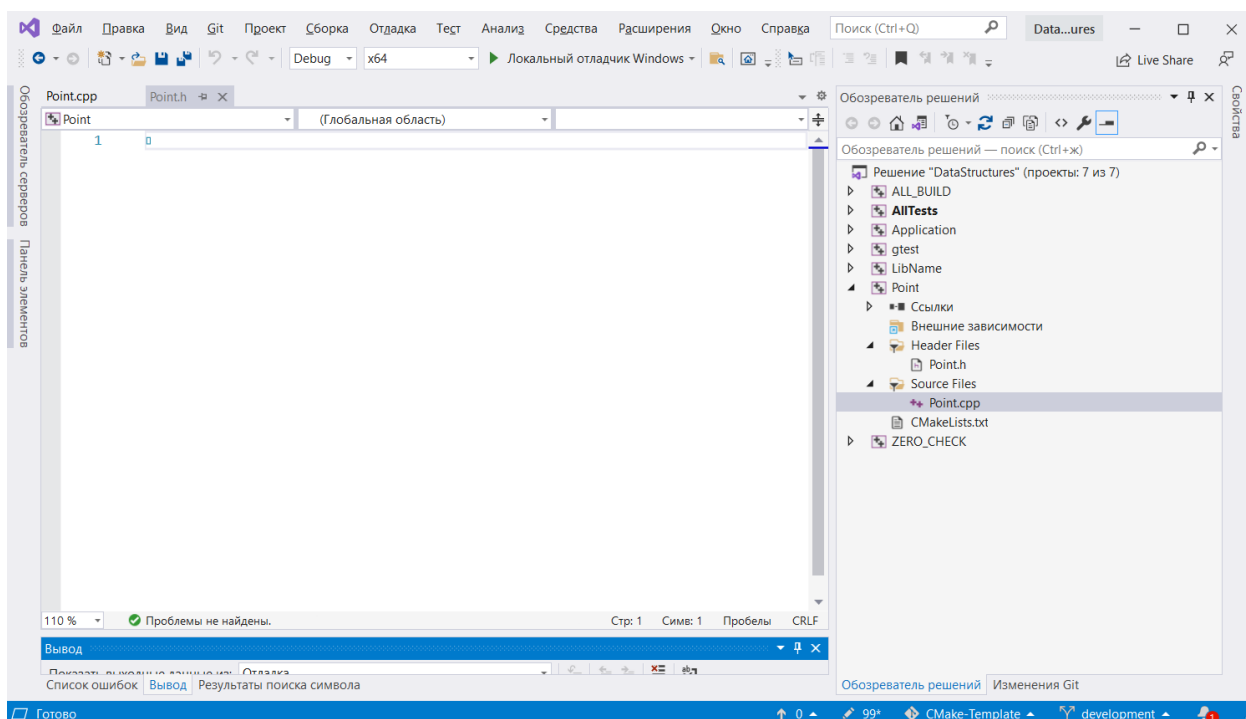


- Изменить основной cmake-лист:



- Под каждую библиотеку можно свой main-проект, а можно использовать всегда один и всё в нём дописывать по мере наращивания числа библиотек в проекте, либо адаптировать его только под последнюю задачу (то есть содержимое main всегда демонстрирует работу только с одной из библиотек, например, последней разрабатываемой). Рекомендовано попытаться сделать самостоятельно вариант с отдельным приложением для каждой отдельной статической библиотеки.

Далее очищаем папку `sln`, пересобираем проект, запускаем файл `.sln`, проверяем работоспособность. Должна получиться следующая структура.



Затем необходимо реализовать поставленную задачу и залить изменения на GitHub. Инструкции об этом также есть в `README.md` к данному проекту-шаблону.

Коротко: добавить файлы для коммита, сформировать коммит, сделать пуш в ваш репозиторий в рабочую ветку. **ВАЖНОЕ ЗАМЕЧАНИЕ:** папку `sln` не заливаем!!! В реальности эта папка может весить очень много, ваш проект с `stake` любой другой пользователь вашего кода соберёт за 10 секунд, поэтому данная сборочная информация лишняя к выкладыванию.