МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

«МОДУЛЬНЕ ТЕСТУВАННЯ»

Лабораторна робота №3

“Управління ІТ-проектами”

для студентів базового напрямку 6.050101 “Комп’ютерні науки”

Студент: Сущенко Д. Ю.

Група: КН-410

Варіант: 22

Кафедра: САПР

Перевірила: Климкович Т. А.

Львів – 2022

Мета роботи:

Ознайомитись з принципами модульного тестування проектів. Набути практичних навичок роботи з JUnit.

Завдання:

1. Ознайомитись з принципами модульного тестування.
2. Організувати модульне тестування проекту зробленого у попередній лабораторній роботі, інтегрувати процес модульного тестування до автоматизації збірки.

Індивідуальне завдання:

Додати unit-тести для проекту розробленого в першій лаб. роботі з використанням бібліотеки GoogleTest. Та виконати автоматизацію по збиранню та запуску тестів.

Відповіді на контрольні запитання:

1. **Що таке модульне тестування?**

   Модульне тестування – це, процес у програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми.
2. **Для чого використовується модульне тестування?**

   Для тестування окремих модулів на коректність. Надати приклад використання модулю.
3. **Що таке test-driven-developing (TDD)?**

   TDD – це, техніка, яка говорить, що перед тим як реалізовувати бізнес логіку, треба написати тесту на цю логіку.
4. **Що таке JUnit?**

   JUnit – це, фреймворк для тестування модулів на мові програмування Java.
5. **Для чого використовується JUnit?**

JUnit використовується для тестування модулів на мові програмування Java.

6. **Яка різниця між JUnit 3 та JUnit 4?**

   JUnit3 старіша версія, де використовується лише наслідування класу, після чого child – клас просто викликають. В JUnit4 додали підтримку Java 5, де є можливість використовувати анотації.

Хід роботи:

Було створено окрему директорію «tests» (рис. 1), де будуть міститися усі тести поточного проекту, щоби було легше в майбутньому їх знаходити та доповнювати.

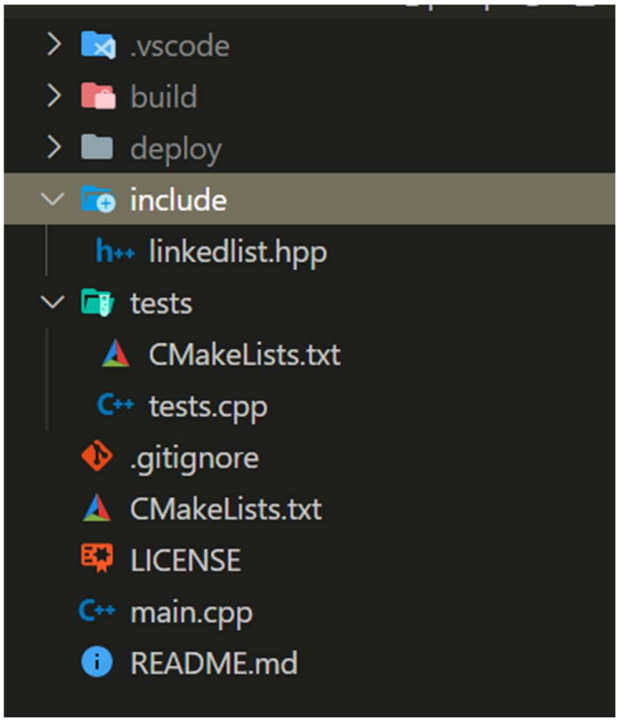

Рис. 1 Створена директорія для тестів

Також, для того, аби проект не «зламався» було вирішено створити окрему вітку на ім’я «features/tests» (рис. 2), де і будуть створені тести, після чого «змерджити» їх в подальшому до головного стовбуру.

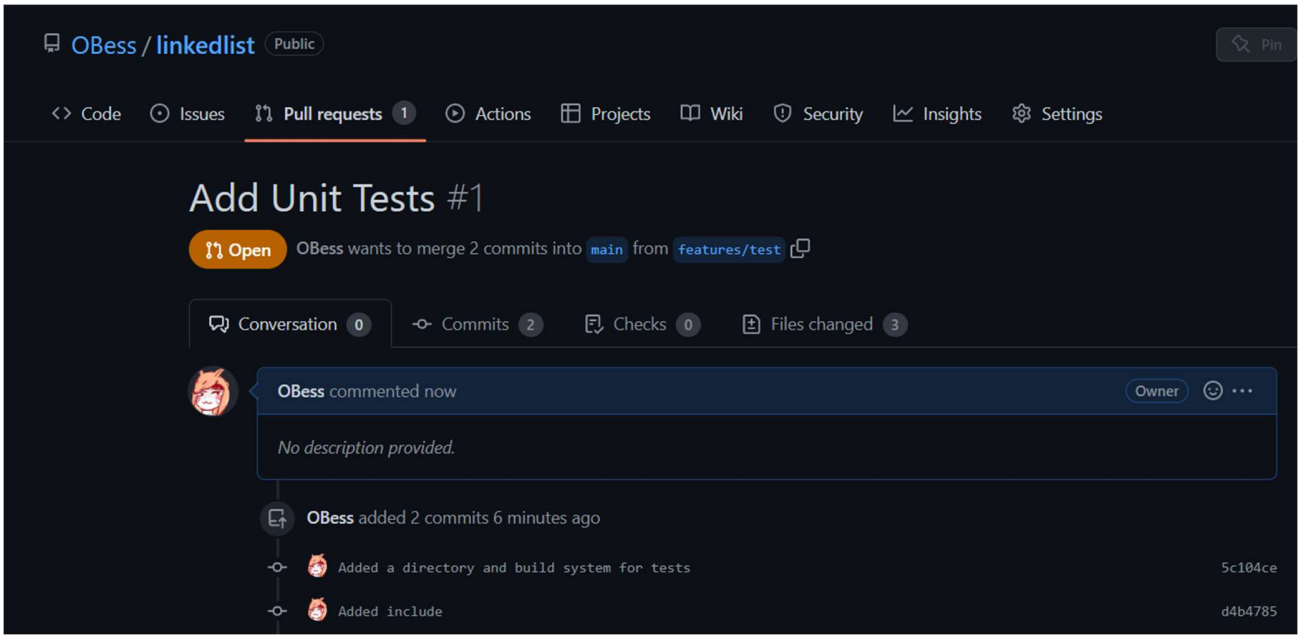

Рис. 2 Окрема вітка для тестів з pull request

Для написання тестів було використано фреймворк GoogleTest, а також, для автоматизації запуску та збірки тестів за допомогою лише однієї кнопки, було використано систему збірки CMake (лістинг 1).

Лістинг 1. Автоматизація запуску та збірки тестів.

```
project(tests VERSION 1.0)

find_package(GTest CONFIG REQUIRED)

add_executable(tests tests.cpp)
if (MSVC)
      set_property(TARGET tests PROPERTY MSVC_RUNTIME_LIBRARY
"MultiThreaded$<$<CONFIG:Debug>:Debug>")
else()
endif()

target_include_directories(tests PUBLIC PUBLIC "../include")
target_link_libraries(tests PRIVATE GTest::gmock GTest::gtest GTest::gmock_main
GTest::gtest_main)

add_test(NAME tests COMMAND tests)
```

Було написано код для тестування окремих модулів програми. В лістингу 2 продемонстровано лише тестування одного модулю, що перевіряє зв'язний список на те, чи він пустий на початку та чи не пустий, якщо додати хоча б один елемент. Всього модульних тестів було написано п'ять.

Лістинг 2. Приклад коду для тестування.

```
#include <gtest/gtest.h>

#include <linkedlist.hpp>

TEST(linkedlist, empty)
{
    container::linkedlist<int> list;

    EXPECT_TRUE(list.empty());

    list.push_back(0);

    EXPECT_FALSE(list.empty());
}

.....

int main(int argc, char *argv[])
{
    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

Після того, як тести були написані, було перевірено на їх коректність та проведено перше unit – тестування модулів (рис. 3).

```
PS C:\My\Projects\cpp\linkedlist\deploy\Debug> ."C:/My/Projects/cpp/linkedlist/deploy/Debug/tests.exe"
[==========] Running 5 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 5 tests from linkedlist
[ RUN      ] linkedlist.empty
[       OK ] linkedlist.empty (0 ms)
[ RUN      ] linkedlist.initializer_list
[       OK ] linkedlist.initializer_list (0 ms)
[ RUN      ] linkedlist.CTAD
[       OK ] linkedlist.CTAD (0 ms)
[ RUN      ] linkedlist.push_back
[       OK ] linkedlist.push_back (0 ms)
[ RUN      ] linkedlist.push_front
[       OK ] linkedlist.push_front (0 ms)
[----------] 5 tests from linkedlist (0 ms total)

[----------] Global test environment tear-down
[==========] 5 tests from 1 test suite ran. (1 ms total)
[  PASSED  ] 5 tests.
```

Рис. 3 Результати модульного тестування

Коли всі тести були вірно написані, тоді було проведено злиття двох гілок за допомого команди ***git merge <branch>*** (рис. 4).

```
C:\My\Projects\cpp\linkedlist>git merge features/test
Updating 9dbbd3b..ba26906
Fast-forward
 CMakeLists.txt          |  3 ++
 include/linkedlist.hpp  | 31 ++++++++++++++++----
 tests/CMakeLists.txt    | 14 +++++++++
 tests/tests.cpp         | 79 ++++++++++++++++++++++++++++++++++++++++++++++++++
 4 files changed, 122 insertions(+), 5 deletions(-)
 create mode 100644 tests/CMakeLists.txt
 create mode 100644 tests/tests.cpp
```

Рис. 4 Злиття гілок

Висновок:

В ході роботи було написано unit – тести на базі фреймворку GoogleTest для тестування модулів програми, які збираються та запускаються за допомогою системи збірки CMake та виводять результат на екран. Репозиторій в якому відбувається робота знаходиться з посиланням - https://github.com/OBess/linkedlist