# 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.

# Для чого використовується модульне тестування?

Для тестування окремих модулів на коректність. Надати приклад використання модулю.

# Що таке test-driven-developing (TDD)?

TDD – це, техніка, яка говорить, що перед тим як реалізовувати бізнес логіку, треба написати тесту на цю логіку.

# Що таке JUnit?

JUnit – це, фреймворк для тестування модулів на мові програмування Java.

# Для чого використовується JUnit?

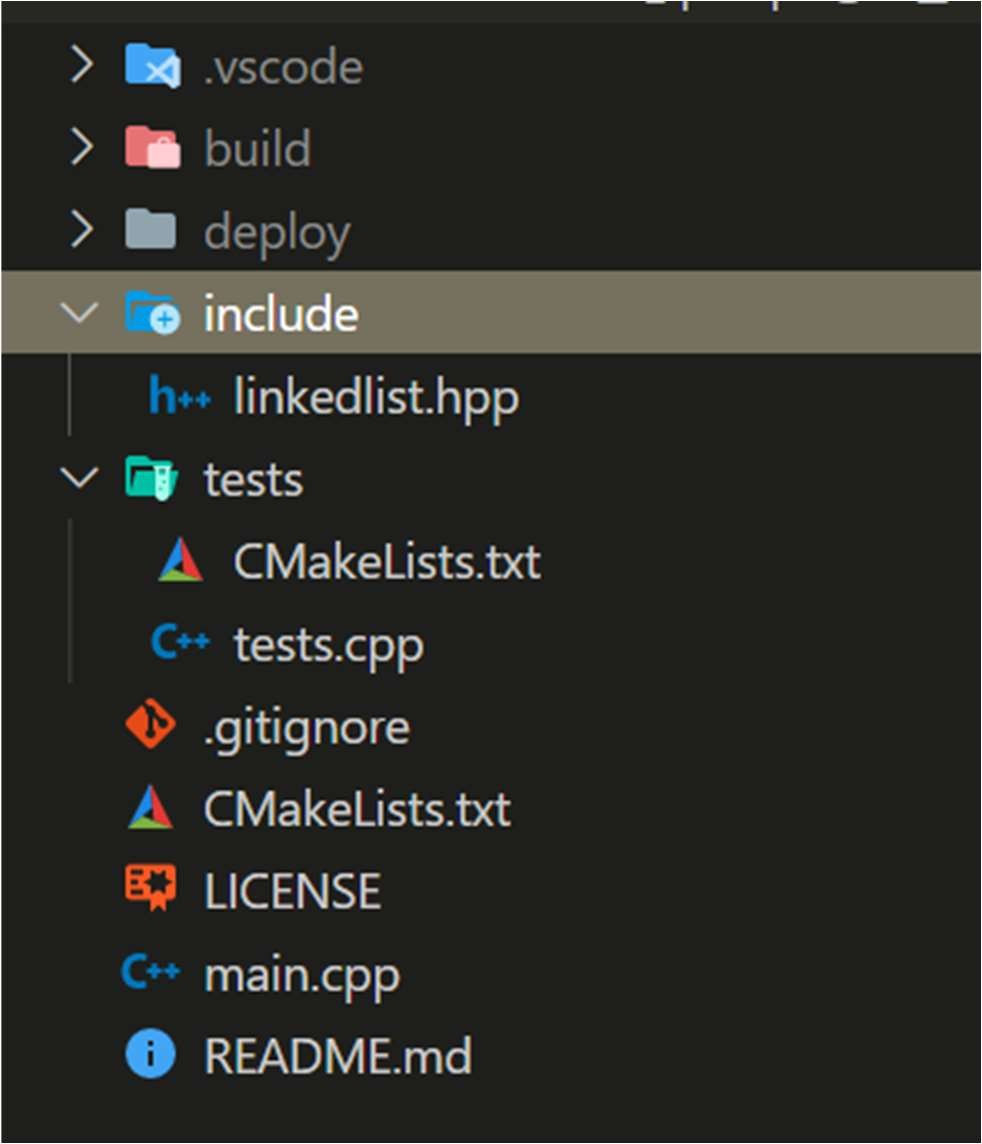
JUnit використовується для тестування модулів на мові програмування Java.

# 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.

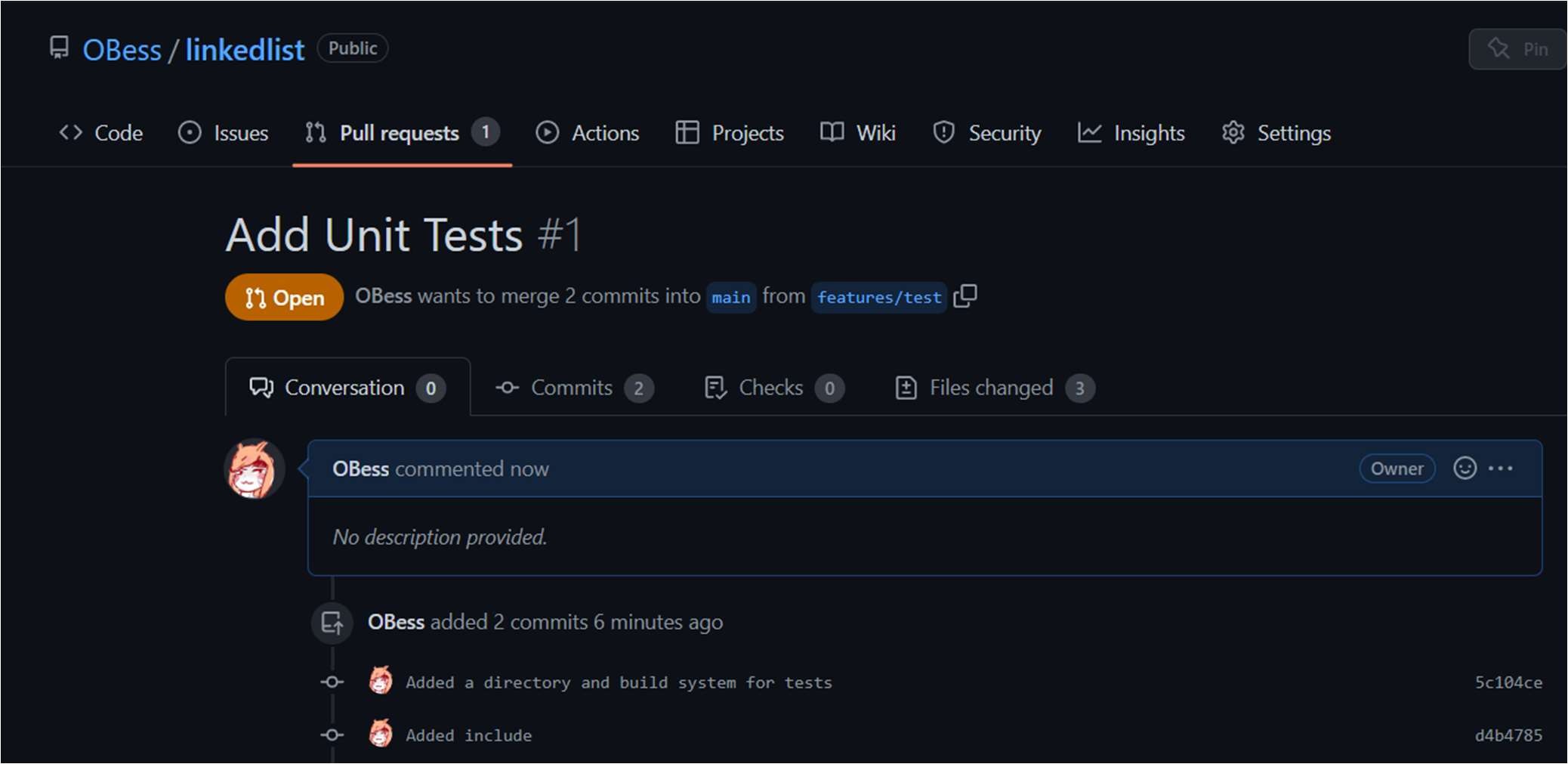
# Хід роботи:

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());

}

………

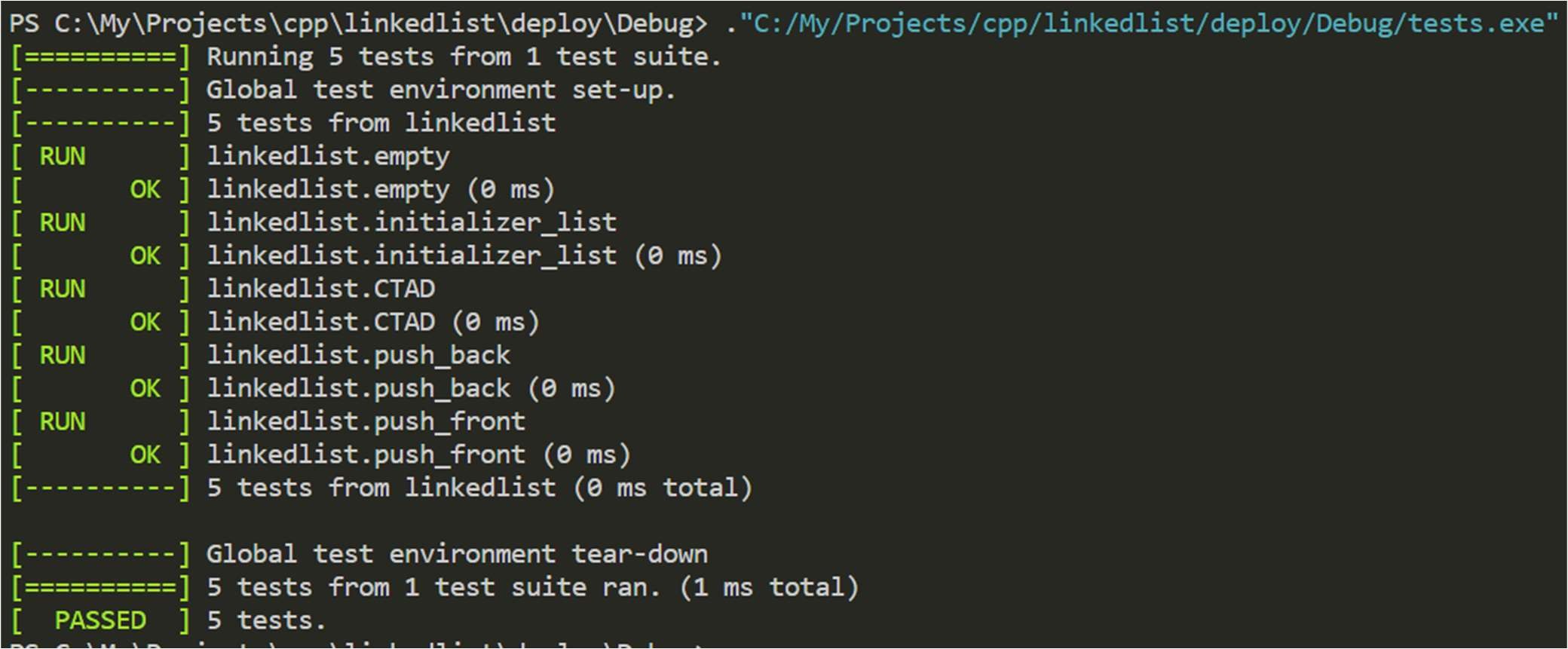
*int* main(*int argc*, *char* \**argv*[])

{

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).



**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).



# Висновок:

**Рис. 4 Злиття гілок**

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