МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ

ЗВІТ

ЛАБОРАТОРНА РОБОТА № 5

З дисципліни «Автоматизоване проєктування комп`ютерних систем»

Виконав: ст. гр. КІ-401
Ларіонов А.О
Перевірив:
Федак П.Р.

Львів 2024

ЗМІСТ

РОЗДІЛ 1. МЕТА РОБОТИ.

Автоматизація тестування.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ВІДОМОСТІ.

Для тестування було обрано фрейморк Google Test для C++

Google Test (gtest) — це популярний фреймворк для модульного тестування в C++. Він надає потужні можливості для автоматизованого тестування, дозволяючи розробникам писати, організовувати та виконувати тести для своїх програм. Ось деякі ключові аспекти gtest та його використання: Основні можливості Google Test:

- Модульне тестування: Google Test дозволяє тестувати окремі функції або методи, що робить його корисним для перевірки поведінки компонентів програмного забезпечення.
- Підтримка асерцій: gtest пропонує багатий набір асерцій, які дозволяють перевіряти, чи виконуються певні умови (наприклад, чи дорівнює значення певному результату, чи є значення ненульовим, і так далі).
- Автоматичний запуск тестів: Тести можуть бути автоматично запущені через головну функцію, і gtest забезпечує звіти про результати кожного тесту.
- Виявлення помилок: Якщо тест не проходить, gtest надає детальні звіти про невдачі, включаючи рядок коду, де сталася помилка.
- Тестування на основі множинних параметрів: За допомогою Google Mock (підсистема gtest для створення мок-об'єктів) можна тестувати поведінку компонентів, залежних від зовнішніх об'єктів або API.
- Швидкість і масштабованість: Google Test дозволяє тестувати великі проекти та забезпечує швидке виконання тестів.

Автоматизовані тести — це метод перевірки працездатності програмного забезпечення за допомогою спеціальних інструментів і скриптів, які автоматично виконують тестові сценарії без необхідності втручання людини. Основна мета автоматизованих тестів — забезпечити високу якість коду, швидке виявлення помилок та зменшити ручну роботу тестувальників.

Модульне тестування — це метод тестування програмного забезпечення, який передбачає перевірку окремих модулів (компонентів) програми, таких як функції, класи або методи, ізольовано від решти системи. Головна мета

модульного тестування — переконатися, що кожен компонент працює відповідно до вимог.

Асерції — це твердження, які перевіряють, чи відповідає результат виконання програми очікуваному значенню. Якщо асерція невірна, тест вважається проваленим. Наприклад:

Mock-об’єкти — це об’єкти, які імітують поведінку реальних компонентів системи. Вони використовуються для тестування ізольованих частин коду без необхідності підключення повної інфраструктури

РОЗДІЛ 3. ЗАВДАННЯ.

Task 5. Implement automated tests:

1. Implement or use existing test framework;
2. Create a set of automated tests;
3. Test report should contain number of all tests, passed tests, failed tests, coverage;
4. Coverage must be more than 80%
5. Required steps

РОЗДІЛ 4. ХІД РОБОТИ.

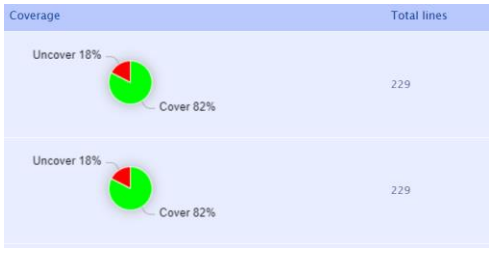
Для забезпечення належного тестування проекту, а саме клієнтської частини частини було створено окремий тестовий проект, який включає тести для основних функцій сервера. Створив окремі тести для кожного виконавчого фалу у директорії tests. Написав конфігурацію для CMake в CMakeLists з інтегруванням gtests . Забілдив проект тестування

test_game.cpp	C++ Source	2 КБ	Hi	7 КБ	79%	24.11.2024 2:01
test_game_logic.cpp	C++ Source	2 КБ	Hi	12 КБ	86%	24.11.2024 2:01
test_menu.cpp	C++ Source	1 КБ	Hi	2 КБ	70%	24.11.2024 2:01
test_serial.cpp	C++ Source	1 КБ	Hi	4 КБ	71%	24.11.2024 2:01

Результат тестування:

```
Microsoft Visual Studio Debug
[ RUN      ] ReceiveMessageFailureTest.InvalidHandle
[       OK ] ReceiveMessageFailureTest.InvalidHandle (0 ms)
[----------] 1 test from ReceiveMessageFailureTest (1 ms total)

[----------] 1 test from CloseSerialPortSuccessTest
[ RUN      ] CloseSerialPortSuccessTest.ValidHandle
[       OK ] CloseSerialPortSuccessTest.ValidHandle (1 ms)
[----------] 1 test from CloseSerialPortSuccessTest (1 ms total)

[----------] 1 test from CloseSerialPortFailureTest
[ RUN      ] CloseSerialPortFailureTest.InvalidHandle
[       OK ] CloseSerialPortFailureTest.InvalidHandle (0 ms)
[----------] 1 test from CloseSerialPortFailureTest (1 ms total)

[----------] 1 test from SerialIntegrationTest
[ RUN      ] SerialIntegrationTest.FullWorkflow
Serial port configured.
Sent by PC: Integration Test
Serial port closed.
[       OK ] SerialIntegrationTest.FullWorkflow (986 ms)
[----------] 1 test from SerialIntegrationTest (987 ms total)

[----------] Global test environment tear-down
[==========] 35 tests from 13 test cases ran. (5235 ms total)
[  PASSED  ] 35 tests.
```

Використовав lcov для збирання покриття та його виведення у вигляді HTML-звіту, де продемонстровано що покриття тестування більше 80%



ВИСНОВКИ

У результаті виконання даної роботи було успішно впроваджено автоматизоване тестування для клієнтської та серверної частин програми, що значно покращило якість і стабільність системи. Покриття тестами більше 80%, що забезпечує високий рівень перевірки функціональності програми та її стабільності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *GitHub Docs - https://docs.github.com/en*
2. *CSAD Instructions for practical tasks and coursework from "Computer systems automated design"*
3. *https://github.com/skizze83/csad2425ki401larionovao10*
4. *https://cmake.org/download/*