МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЛЬВІВСЬКА ПОЛІТЕХНІКА



**АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ**

**КОМП’ЮТЕРНИХ СИСТЕМ**

Завдання 5: “ Тестування гри Rock-Paper-Scissors”

Виконав:

ст. гр. КІ-404

Мох М. П.

Приняв:

Федак П.Р.

Львів – 2024

***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

**Варіант - 12**





**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Автоматизовані тести є невід'ємною частиною розробки програмного забезпечення, які виконуються автоматично для перевірки правильності роботи програм та виявлення можливих помилок. У мові програмування C++, для написання тестів, можна використовувати вбудовану бібліотеку cassert (assert.h у C), яка містить макрос assert. Цей макрос використовується для визначення виразів, які мають бути істинними. У випадку, якщо вираз виявиться хибним (FALSE), програма завершиться з повідомленням про помилку.

Для зроблення тестування більш систематичним та автоматизованим, розробники часто використовують спеціальні фреймворки для тестування, такі як Google Test або Catch. Ці фреймворки дозволяють створювати розширені тестові сценарії, вести звіти про результати тестування, та автоматизувати процес виявлення помилок.

Тести в розробці програмного забезпечення є важливою частиною процесу, оскільки вони дозволяють перевіряти правильність роботи коду, виявляти можливі помилки та забезпечувати стабільність програми. Тестування може бути використане на різних етапах розробки, від написання нового коду до внесення змін у вже існуючий.

Тестування може бути виконане як вручну (manual testing), так і автоматично (automated testing). Автоматизовані тести мають численні переваги, такі як швидкість виконання, можливість запуску тестів в будь-який момент, ідентифікація помилок на ранніх етапах розробки та можливість повторного використання тестових сценаріїв.

**ХІД РОБОТИ**

У ході роботи було проведено тестування серверної та клієнтської частини програми з використанням **OpenCppCoverage** для аналізу покриття коду. Тести підтвердили правильність роботи програми. За допомогою OpenCppCoverage було згенеровано звіт у форматі HTML, який показав, які частини коду були протестовані, а які залишаються непокритими. Інструмент дозволив візуально оцінити покриття тестами та знайти ділянки коду, що потребують додаткових тестів, що допомогло підвищити якість тестування і забезпечити ефективне виявлення помилок.

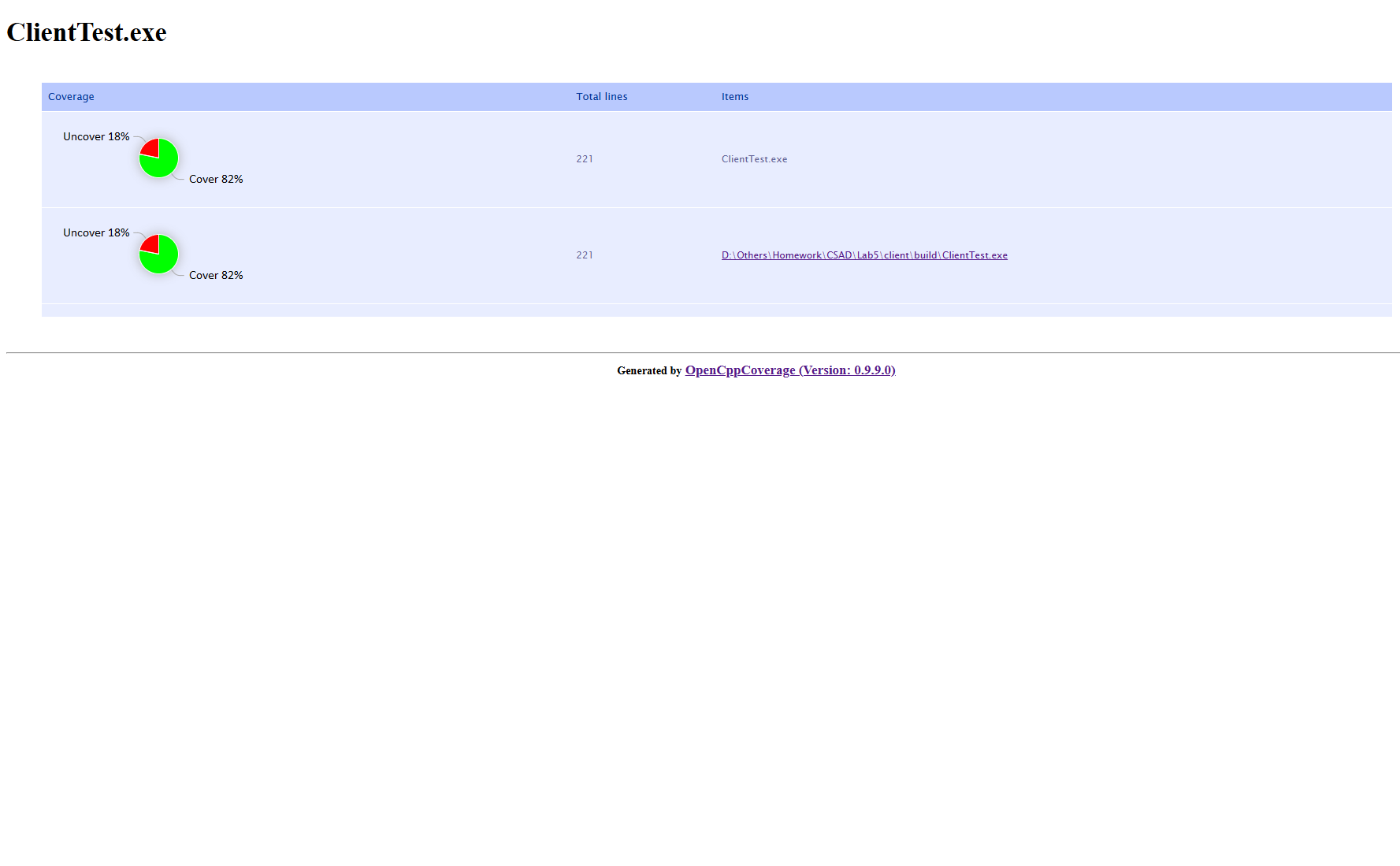


Рис.1 Тестування клієнської частини

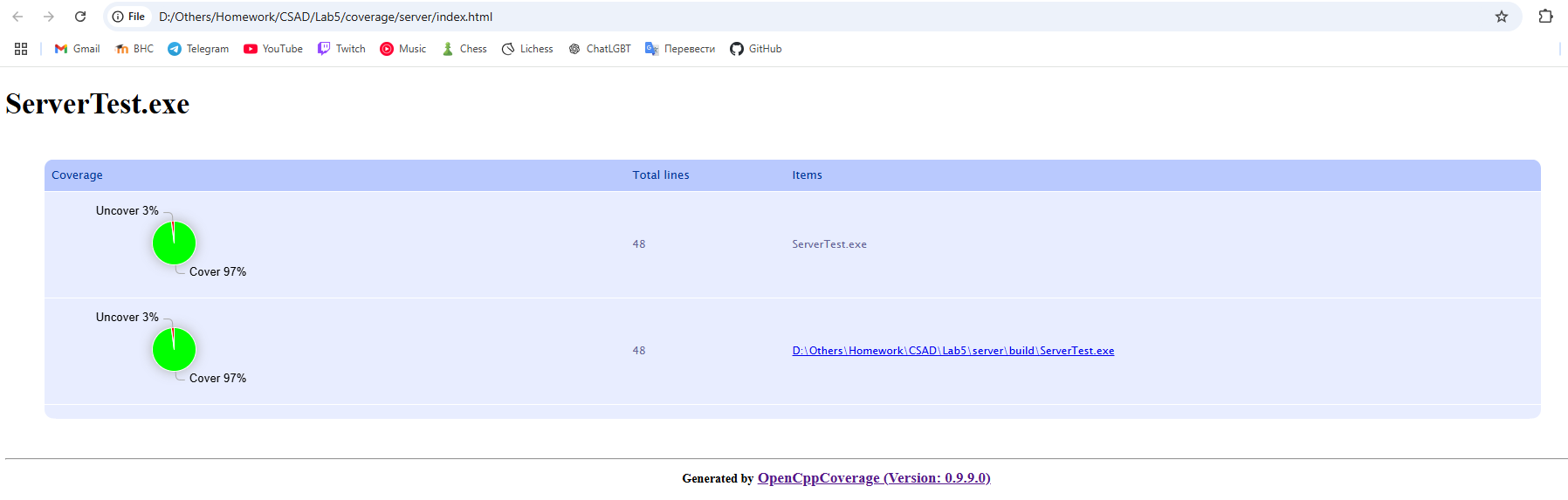


Рис.2 Тестування серверної частини

**Висновок:**

У ході виконання лабораторної роботи були розроблені тести для клієнтської та серверної частин коду, які пройшли успішно, підтвердивши правильність роботи програми.

Для оцінки якості коду було використано інструмент **OpenCppCoverage**, що дозволяє отримати звіт про покриття коду тестами. Це дозволило візуалізувати, які частини коду протестовано, а які залишились непокритими. Виключення певних файлів і бібліотек допомогло зосередитись на власному коді. Звіт у форматі HTML дав змогу детально проаналізувати покриття та визначити ділянки для подальшого тестування.

Загалом, використання **OpenCppCoverage** покращило якість тестування і допомогло забезпечити вищий рівень впевненості в коректності програми.

СПИСОК ЛІТЕРАТУРИ

1. **OpenCppCoverage Documentation**  
   Офіційна документація для OpenCppCoverage.  
   <https://github.com/OpenCppCoverage/OpenCppCoverage>
2. **Google Test Documentation**  
   Документація до популярного фреймворку для юніт-тестування на C++.  
   <https://google.github.io/googletest/>
3. **Code Coverage**  
   Роз’яснення та кращі практики щодо покриття коду та його важливості.  
   <https://www.jetbrains.com/help/idea/code-coverage.html>