

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ  
 НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу  
Кафедра системного проектування**

**Звіт**

**про виконання практичної роботи №4  
з дисципліни «Паралельні обчислення»**

Виконав:  
студент III курсу, групи ДА-22  
Жадько Микита Сергійович

Прийняв:

асистент Яременко В. С.

Київ – 2025

**ПОСТАНОВКА ЗАДАЧІ**

1. Розробити клієнт-серверний застосунок для вирішення завдання з лабораторної роботи номер 1, передавши масив даних з клієнта на сервер, а потім – отримавши результат назад на сторону клієнта. Для виконання основного завдання дозволено використовувати лише платформні (WinSock, POSIX) та вбудовані засоби роботи з сокетами.
2. Самостійно розробити протокол прикладного рівня (application protocol) для взаємодії клієнта з сервером. Для цього врахувати декілька кроків в процесі взаємодії:
3. надсилання даних та конфігурації обчислень (наприклад, вказати кількість потоків для виконання обчислень)
4. надсилання команди (та отримання відповіді на команду) для початку обчислень
5. надсилання команди для запиту статусу і результату обчислень.
6. Додати до розробленого серверу підтримку підключення декількох клієнтів одночасно. Додатковий бал можна отримати у випадку наявності двох клієнтів, один з яких буде розроблений на мові, відмінній від мови серверу та першого клієнту (другий клієнт дозволено створювати з використанням скриптових мов).
7. Застосунок повинен коректно оброблювати виняткові ситуації як на стороні клієнту, так і на стороні серверу і адекватно реагувати на них. Без завершення своєї роботи. Обов’язковим є коректна обробка порядку кодування байтів у повідомленні.
8. В протоколі роботи необхідно навести опис розробленого застосунку. До цього опису повинні входити обґрунтування вибору протоколу передачі даних, а також архітектурний опис клієнта.
9. Занести до протоколу роботи опис розробленого протоколу прикладного рівня у вигляді таблиці, що включає: перелік всіх команд, аргументи команд та їх опис, список можливих відповідей на команду.
10. Занести до протоколу роботи UML діаграму викликів взаємодії серверу та клієнту, починаючи від запуску клієнту, до завершення роботи.
11. Надати висновок, що повинен містити аналіз та опис проблем з котрими зіштовхнувся студент, або з якими може зіштовхнутися розробник при організації міжпроцесової взаємодії.

**ХІД РОБОТИ**

Програма реалізує клієнт-серверну архітектуру для обробки матриць у багатьох потоках. Клієнт передає матрицю та конфігурацію потоків на сервер, який виконує обчислення. Після завершення обчислень сервер повертає результати клієнту.

Клієнт

* Генерує матрицю заданого розміру.
* Визначає конфігурацію потоків для тестування.
* Надсилає дані на сервер.
* Перевіряє статус обчислень, натискуючи Enter.
* Отримує результати після завершення обчислень.

Сервер

* Приймає дані від клієнта.
* Обробляє матрицю у багатьох потоках відповідно до конфігурації.
* Повертає проміжні результати та фінальні дані клієнту.

Протокол прикладного рівня (Application Protocol) визначає команди, які використовуються для взаємодії між клієнтом і сервером. У таблиці нижче наведено опис команд, аргументів та можливих відповідей.

|  |  |  |  |
| --- | --- | --- | --- |
| Команда | Аргументи | Опис | Коди відповідей |
| HELLO | - | Клієнт повідомляє про своє підключення. | "CONNECTED" |
| SEND\_DATA | Заголовок (DataHeader), конфігурація потоків, матриця | Клієнт надсилає матрицю та конфігурацію потоків. | "DATA\_RECEIVED" |
| START\_  COMPUTATION | - | Клієнт запитує початок обчислень. | "COMPUTATION\_  STARTED"  "ERROR" |
| GET\_STATUS | - | Клієнт запитує поточний статус обчислень. | "STATUS: COMPLETED"  "STATUS: <тпоточний\_потік>/  <всього> |
| GET\_RESULT | - | Клієнт запитує фінальні результати обчислень. | Результати у форматі тексту (час виконання відносно к-ті потоків). |

Також зобразимо UML-діаграму нашого застосунку:

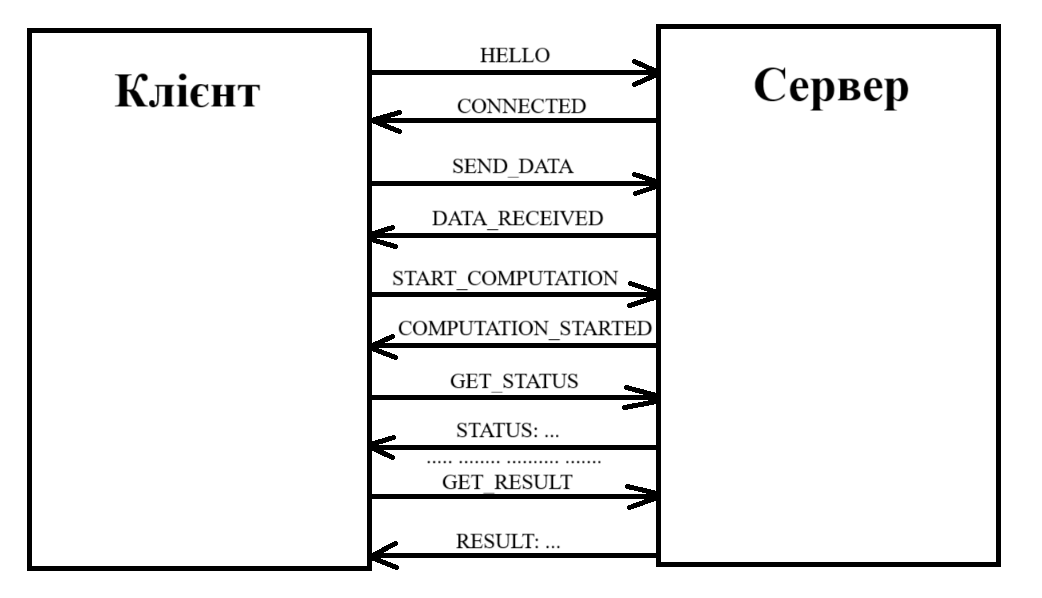


Рисунок 1 — UML-діаграма застосунку.

Далі протестуємо роботу нашого сервера, запустивши майже одночасно двох клієнтів, підключених до сервера.

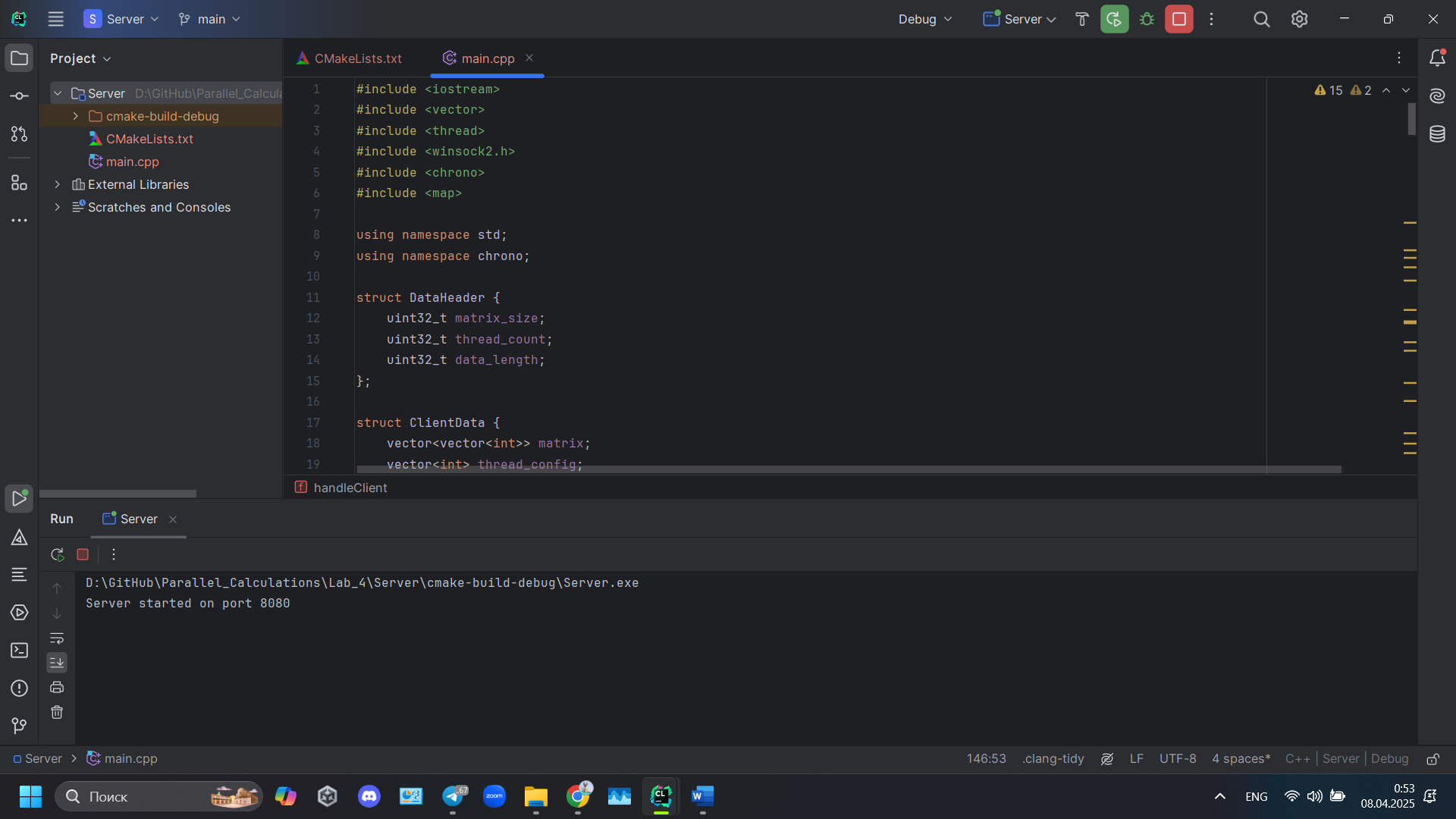


Рисунок 2 — Запуск серверу.

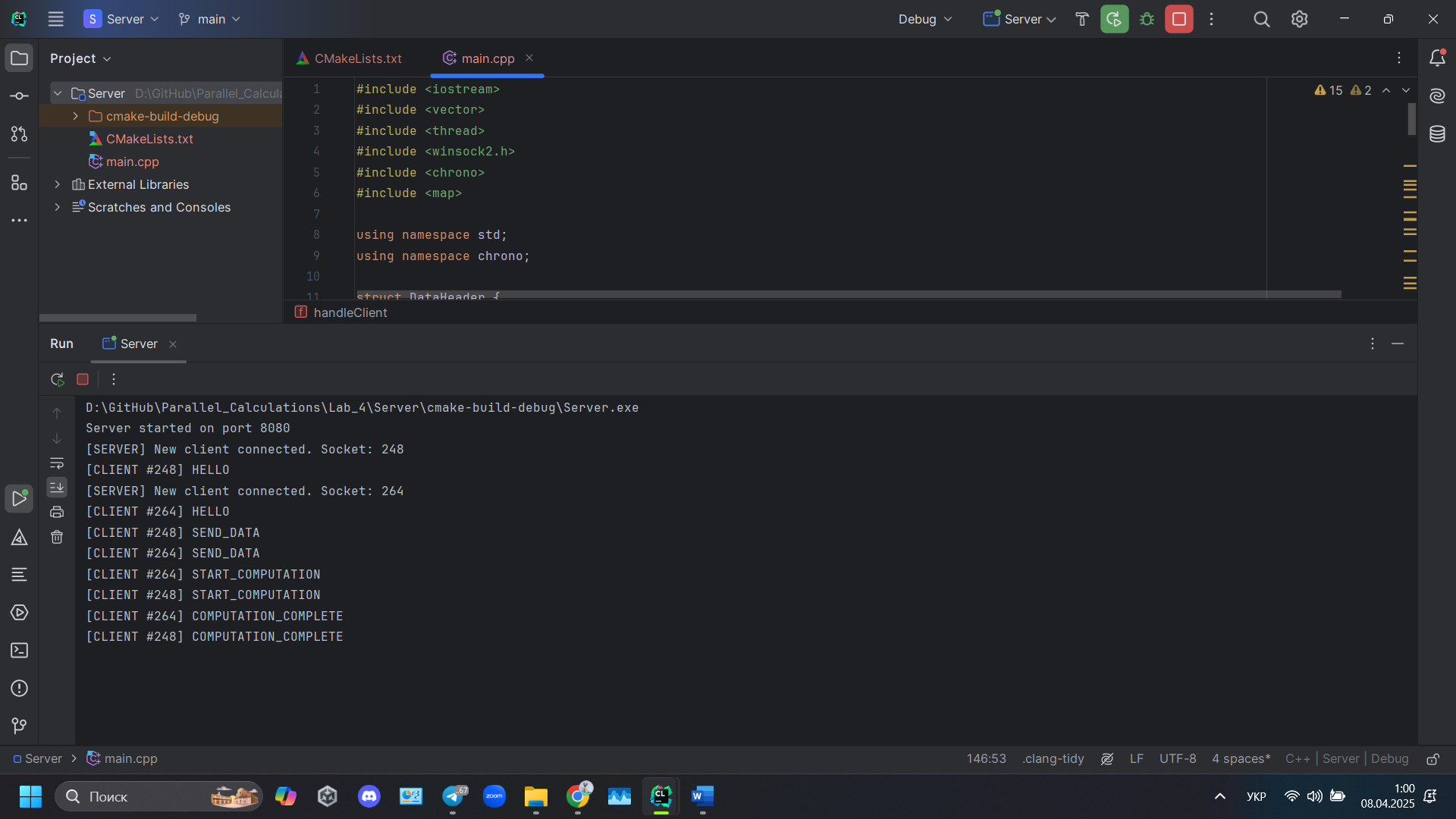


Рисунок 3 — Одночасна робота з двома клієнтами.

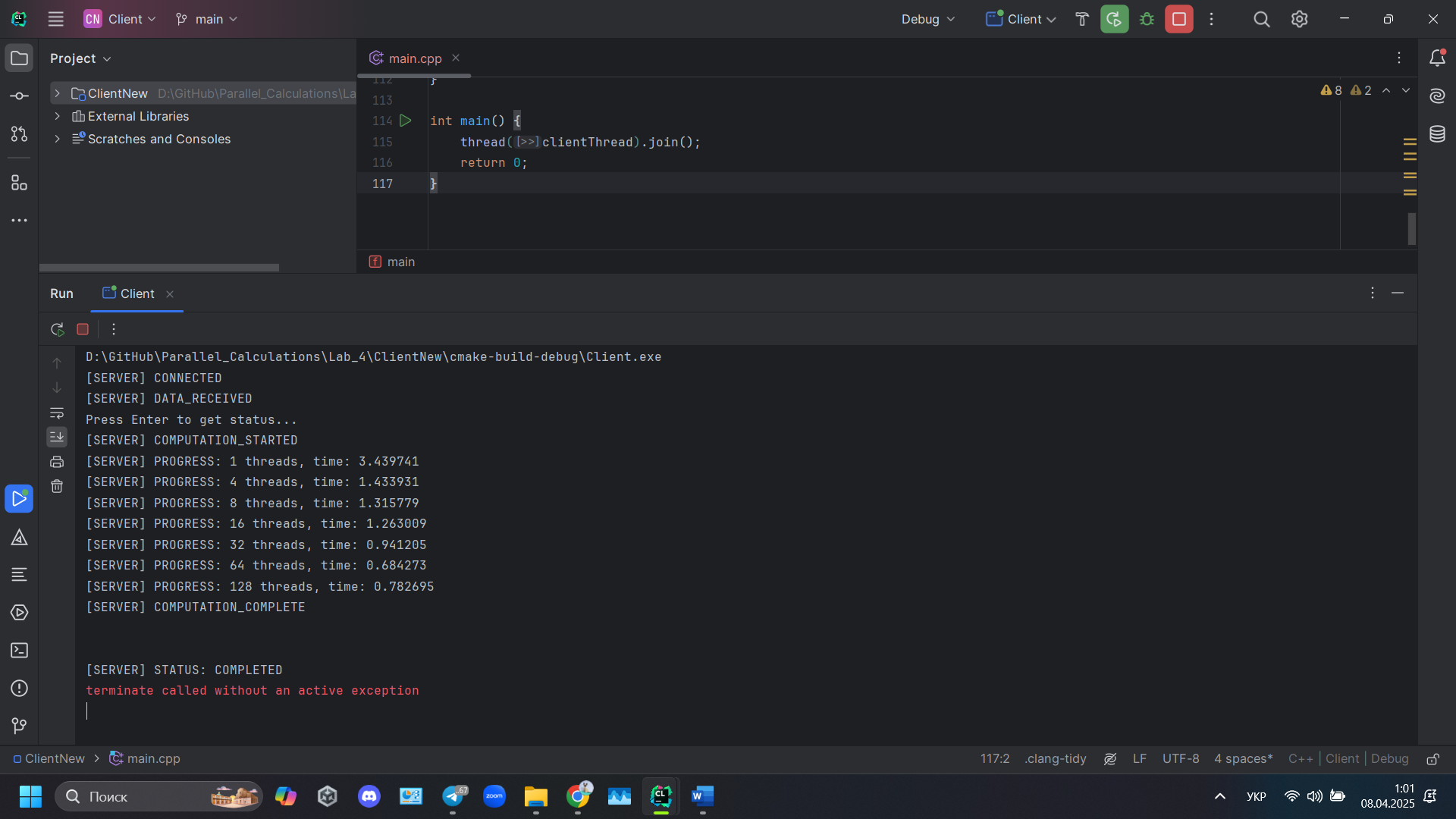


Рисунок 4 — Логи першого клієнта.

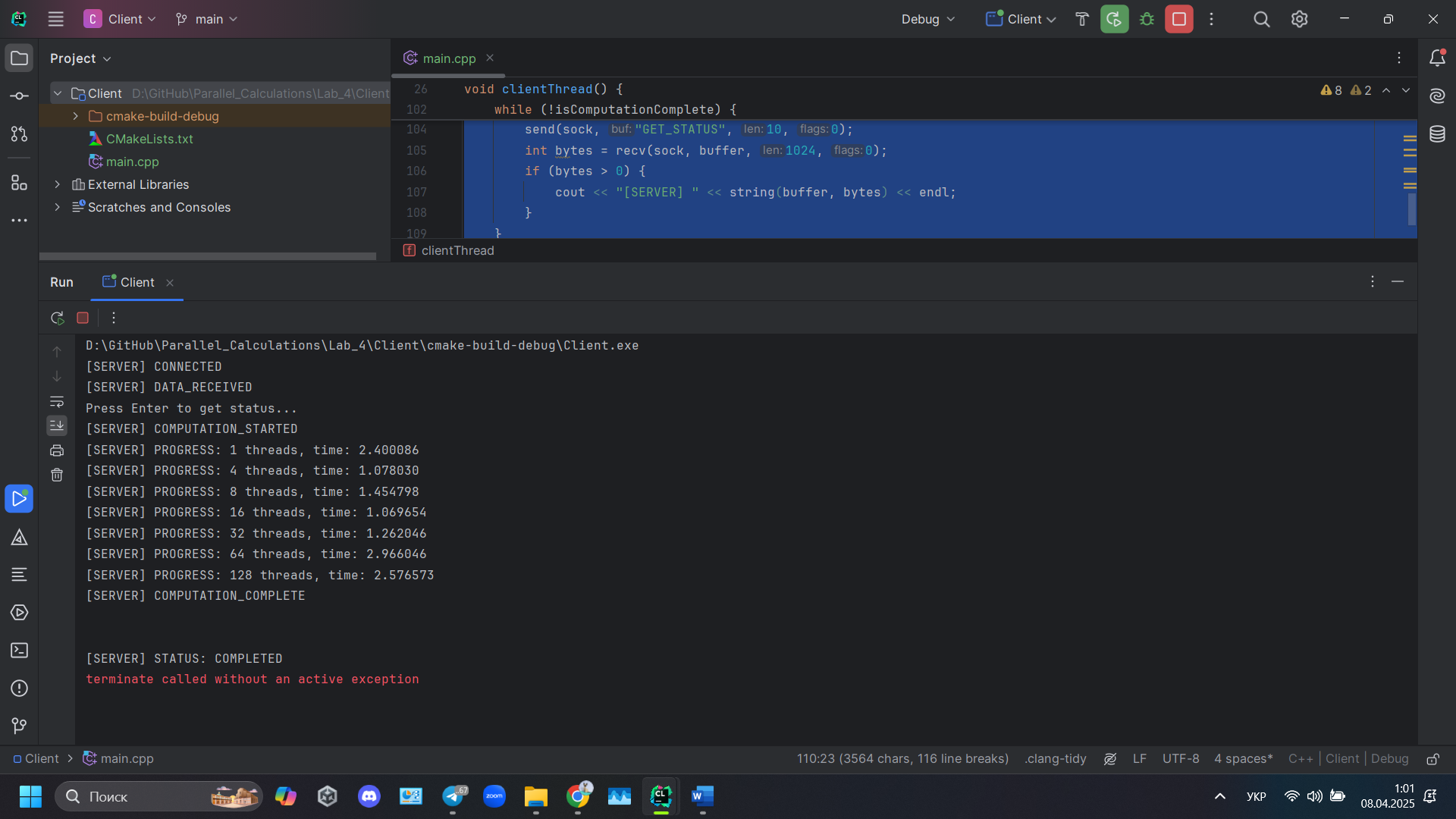


Рисунок 5 — Логи другого клієнта.

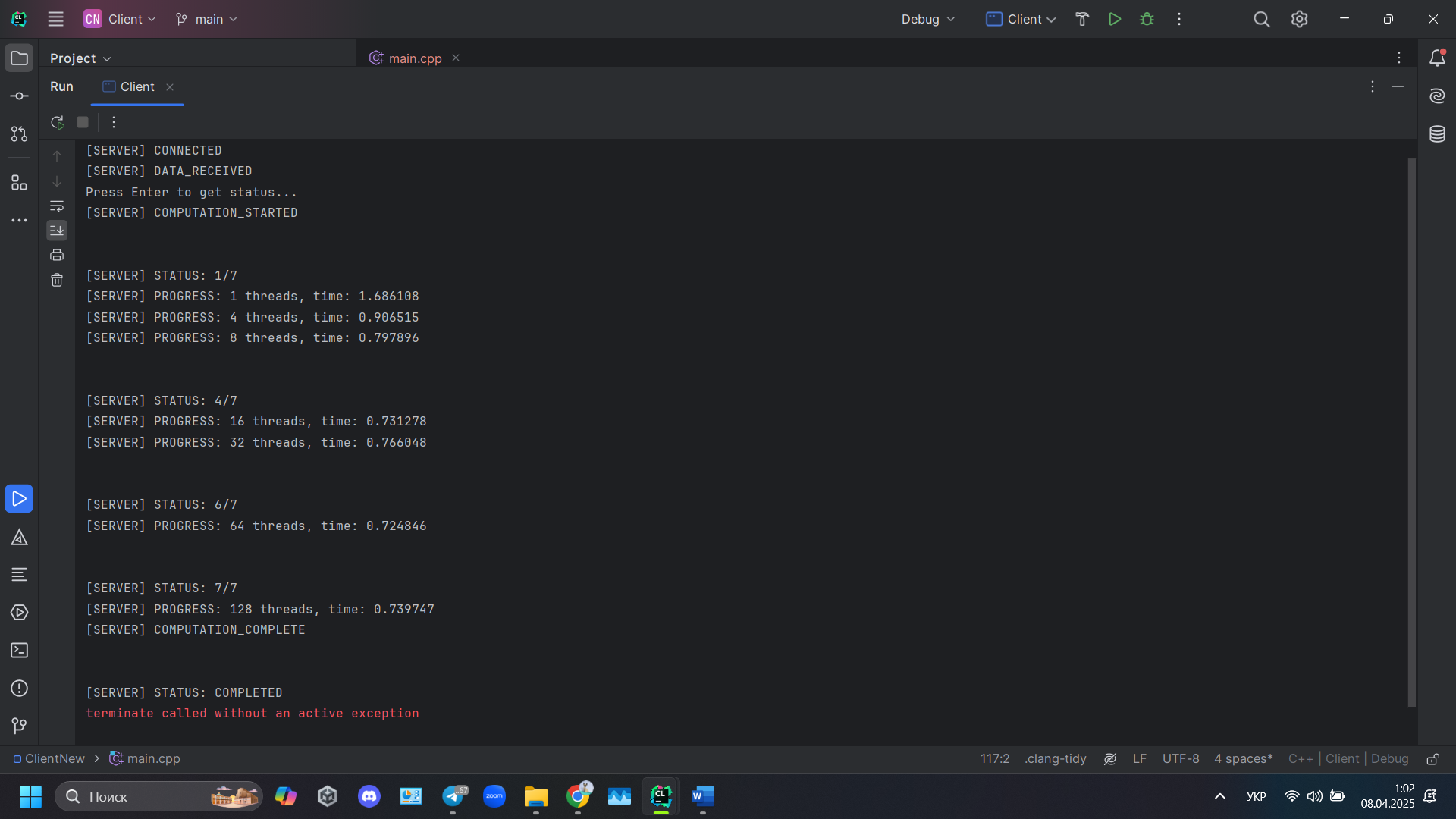


Рисунок 6 — Логи з перевіркою поточного статусу виконання завдання.

**ВИСНОВКИ**

У результаті виконання лабораторної роботи було розроблено клієнт-серверний застосунок для обробки матриць у багатьох потоках. Клієнт генерує матрицю та надсилає її на сервер разом із конфігурацією потоків, які будуть використовуватися для обчислень. Сервер обробляє дані, розподіляючи завдання між потоками, і повертає результати клієнту. Для забезпечення одночасного підключення декількох клієнтів сервер був реалізований з використанням асинхронних потоків, що дозволяє обробляти запити незалежно.

Протягом розробки було створено протокол прикладного рівня для взаємодії між клієнтом і сервером, який включає команди для передачі даних, запуску обчислень, перевірки статусу та отримання результатів. У програмі враховано проблеми кодування даних у мережевий порядок байтів, що забезпечує коректну передачу даних через сокети. Також було додано обробку виняткових ситуацій, таких як розрив з'єднання або некоректні дані, що дозволяє програмі працювати стабільно.

Під час виконання завдання виникли проблеми з синхронізацією потоків на стороні сервера та оптимізацією розподілу завдань між потоками. Для покращення продуктивності можна додатково оптимізувати алгоритми обробки матриць та використовувати більш ефективні методи передачі даних. У цілому, розроблений застосунок демонструє ефективність використання сокетів для міжпроцесової взаємодії та можливість розподілу обчислювальних задач між процесами.

Також [посилання на репозиторій](https://github.com/nikk0308/Parallel_Calculations).