

## Звіт з Лабораторної Роботи №4

**Інформація про виконавця:** Падалко Анна Романівна К-25

**Умова задачі (Варіант №7):** Завдання полягало у розробці та тестуванні багатопотокової структури даних з 3-х цілих полів. Структура має бути оптимізована під заданий набір ймовірностей операцій (варіант №7). Поле 0: read (5%), write (5%) Поле 1: read (30%), write (5%) Поле 2: read (24%), write (1%) operator string: 30%

**Схема захисту даних та її обґрунтування:** М'ютекси: 3 шт.(масив `std::mutex mtx[M]`). Що захищають: кожен `mtx[i]` захищає свій елемент `data[i]`. Перевага: забезпечує паралельні читання/записи різних полів без блокування всієї структури → краща продуктивність. Цілісність стану: `operator string` блокує всі м'ютекси через `std::scoped_lock`, щоб отримати узгоджений знімок даних.

**Таблиця результатів виконання:** Вимірювання часу (у мілісекундах) проводилося для трьох варіантів розподілу ймовірностей на 200,000 операцій.

Варіант	1 потік (ms)	2 потоки (ms)	3 потоки (ms)
А (заданий, №7)	0.07	0.40	1.03
В (рівномірний)	0.07	0.30	0.57
С (сильно не-рівн.)	0.07	0.40	0.58

**Висновки:** При 1 потоці всі варіанти працюють однаково (0.07 мс), бо немає конкуренції. При 2–3 потоках найкраще масштабується Варіант В, оскільки навантаження рівномірне, а глобальні блокування трапляються рідко. Варіанти А і С працюють гірше через високу конкуренцію: А: два гарячі місця — часті операції з полем 1 та виклики `string()`. С: надмірне навантаження на поле 0 + багато глобальних блокувань. Підсумок: блокування на рівні окремих полів дає найбільший ефект при рівномірному доступі до даних, але втрачає перевагу при появі “гарячих точок” або частих глобальних блокувань.

**Що реально було зроблено самостійно:** Самостійно була реалізована структура даних `ConcurrentData` з використанням масиву м'ютексів для блокування на рівні окремих полів кожної комірки даних. Також було написано логіку для генерації тестових файлів (`generateFile`) відповідно до заданих ймовірностей та функцію `runExperiment` для запуску експериментів з різною кількістю потоків (`jthread`) та вимірюванням часу виконання (`std::chrono`).