ПРОГРАМНИЙ ПРОЕКТ №4

Виконавець: Воловік Олександр Олексійович

Група: К-26

Розробити клас ThreadData зі схемою **дрібнозернистого блокування** для одночасного доступу до *різних* полів. Протестувати продуктивність на 3-х профілях навантаження (A, B, C) з 1, 2 та 3 потоками.

- **Кількість м'ютексів:** 2 (mtx_field0 та mtx_field1).
- **Принцип:** mtx_field0 захищає field0, mtx_field1 захищає field1.
- **Обґрунтування:** Схема дозволяє паралельний доступ до *різних* полів, на відміну від одного м'ютекса на весь об'єкт, що підвищує паралелізм.
- **Особливість:** read/write блокують 1 м'ютекс. to_string блокує **обидва** м'ютекси для консистентності, що є найдорожчою операцією.

========== ТАБЛИЦЯ РЕЗУЛЬТАТІВ (усереднені, мс)

Сценарій	1 Потік	2 Потоки	3 Потоки
Оптимізований (A)	32.19	104.53	312.00
Рівномірний (B)	18.68	63.16	167.84
Невідповідний (C)	10.32	34.62	86.96

==========

Результати відповідають очікуванням:

- **Сценарій A (Висока суперечність):** 45% операцій string блокують *обидва* м'ютекси. Очікується **дуже погане масштабування**.
- **Сценарій B (Рівномірний):** Найкращий випадок. 80% операцій паралельні (по 40% на поле). Очікується **помітне, але не ідеальне** прискорення.
- **Сценарій C ("Гаряча точка"):** 85% операцій на field1. Схема не дає переваг. Очікується **вкрай низьке масштабування**.

Самостійно було спроектовано та реалізовано клас ThreadData з нуля. У ньому було впроваджено схему дрібнозернистого блокування з використанням двох окремих std::mutex для незалежного захисту кожного поля даних. Також було самостійно написано логіку потокобезпечних методі read, write та to_string, які коректно блокують один або обидва м'ютекси відповідно до вимог операції.