Виконавець: Банасевич Вікторія Любомирівна, **група:** К-25

Лабораторна робота №4: Розробка багатопотокової структури даних, **Варіант:** №9

Умова задачі (тільки частина, специфічна для варіанту): Розробити потоко-безпечну структуру даних з m=3 цілими полями. Структура має бути оптимізована під наступний розподіл операцій:

- **Поле 0:** Читання (read) 10%, Запис (write) 10%
- **Поле 1:** Читання (read) 10%, Запис (write) 10%
- **Поле 2:** Читання (read) 40%, Запис (write) 5%
- **Операція string:** 15%

Схема захисту даних: Для захисту даних було обрано схему, що базується на прикладі guarded_data.cpp. Використовується **один глобальний м'ютекс** (std::mutex), який захищає доступ до всіх трьох полів структури одночасно. Будь-яка операція (read(i), write(i) або operator string()) перед виконанням захоплює цей єдиний м'ютекс за допомогою std::lock_guard.

Обґрунтування: Ця схема була обрана через її простоту та пряму відповідність наданому прикладу "Guarded Data". Вона гарантує потокову безпеку, оскільки в будь-який момент часу лише один потік може мати доступ до даних. Однак, ця схема не є оптимізованою під умову варіанту, оскільки вона створює "вузьке горло": всі операції, навіть ті, що не конфліктують (напр. read(0) та read(2)), змушені виконуватись послідовно, а не паралельно.

Табличка розмірів 3x3 з усередненими результатами виконання

0.5668 с	1.8582 с	2.9875 с
0.4976 с	2.1796 с	3.4613 с
1.7124 с	3.9605 с	16.1308 с

***Примітка:** Рядки (згори вниз): Сценарій (a) - згідно варіанту №9, Сценарій (b) - рівномірний, Сценарій (c) - невідповідний. Стовпці (зліва направо): Виконання на 1 потоці, на 2 потоках, на 3 потоках.*

Висновки (чи відповідає табличка з попереднього абзацу очікуванням): Результати у таблиці повністю відповідають очікуванням для обраної схеми захисту з одним м'ютексом. Оскільки всі операції (читання, запис, string) змушені чекати на **одне й те саме блокування**, реальний паралелізм відсутній. Замість прискорення, ми бачимо **значне уповільнення** при додаванні 2-го та 3-го потоків (наприклад, з 1.71с до 16.13с у сценарії (c)). Це відбувається через високу конкуренцію потоків, які "б'ються" за єдине блокування, що додає великі накладні витрати. Схема безпечна, але абсолютно не масштабується.

Що реально було зроблено самостійно

Повністю самостійно було реалізовано генератор файлів (generate_files.cpp), який використовує std::discrete_distribution для створення послідовностей операцій згідно з ймовірностями, вказаними у варіанті. А також основну програму (main.cpp), яка зчитує згенеровані файли, організовує запуск тестів (для 1, 2 та 3 потоків), проводить вимірювання часу за допомогою std::chrono та виводить фінальну таблицю.