

1 Приходько Віталій, група К-24.

кількість полів m=2

	Поле 0		Поле 1		
варіант	read	write	read	write	string
№13	10%	10%	50%	10%	20%
№14	25%	5%	5%	25%	40%
№15	10%	5%	50%	10%	25%
№16	20%	5%	20%	5%	50%
№17	5%	5%	30%	5%	55%
№18	20%	10%	25%	10%	35%
№19	5%	40%	5%	5%	45%
№1	10%	5%	50%	5%	30%
№2	40%	5%	5%	5%	45%

2

Варіант №2

3 Схема захисту даних складається з масиву з двох м'ютексів типу `std::shared_mutex`.

У моєму варіанті 90% операцій не змінюють дані. Звичайний `std::mutex` пропускав би потоки по черзі. `std::shared_mutex` дозволяє захоплювати ресурс у режимі `shared_lock`, завдяки чому багато потоків можуть читати дані одночасно, не заважаючи один одному. Повне блокування (`unique_lock`) відбувається тільки під час запису.

4

```
vit@MacBook-Air-Vitalii lab4 (main) % ./lab4
Test running with 1000000 operations per thread.
      Scenario      1 Thread      2 Threads      3 Threads
Specific      0.028124      0.48044      1.12613
Equal         0.021192      1.67032      3.20836
Random        0.017309      2.03281      3.11225
vit@MacBook-Air-Vitalii lab4 (main) %
```

5 Висновки: Результати таблиці повністю відповідають очікуванням. У сценарії "Specific" час виконання суттєво зменшується при збільшенні кількості потоків, оскільки `shared_mutex` дозволяє розподілити 90% роботи. У сценарії "Random", де відбуваються тільки записи, багатопотоковість погіршила результат, тому що `unique_lock` змушує потоки ждати свою чергу.

6 Самостійно зроблено структуру `MyData` та реалізовано методи доступу з використанням `shared_mutex`. Написано генератор тестових даних, який створює файли з командами, з імовірностями, що відповідають варіанту.