

У даній роботі реалізовано багатопотокову структуру даних, що складається з трьох цілих полів. Для кожного поля реалізовані операції: readField (читання значення), writeField (запис значення) та перевантажена операція toString для отримання рядкового представлення всієї структури. Частоти операцій для варіанту: Поле 0: read 20%, write 5%, Поле 1: read 20%, write 5%, Поле 2: read 10%, write 20%, string 40%. Для демонстрації використано три типи файлів з послідовністю дій: частоти відповідні умові, рівні частоти та хаотичні частоти.

Для синхронізації доступу до полів структури використано один shared_mutex на всю структуру. Читання (readField та toString) здійснюється під shared_lock, що дозволяє кільком потокам читати одночасно. Запис (writeField) здійснюється під unique_lock, що блокує одночасний доступ інших потоків до структури. Ця схема обрана, оскільки число операцій читання перевищує число записів у деяких полях, і shared_mutex дозволяє підвищити продуктивність у багатопотоковому режимі, одночасно гарантує цілісність даних при записі.

Таблиця результатів

Execution Time Table (us):			
Threads\File	A	B	C
1	313203	369372	280524
2	5480257	5793285	5132701
3	7851952	8508591	7682462

Час виконання у 1 потік найвищий для файлів із високою частотою операцій string і записів. При 2 та 3 потоках час виконання зменшується для файлів з переважанням читань, проте не пропорційно $1/N$ через наявність одного shared_mutex, який блокує одночасний запис. Для файлів з рівними частотами та хаотичними частотами спостерігаються аналогічні закономірності. Висновок: захист даних через shared_mutex ефективний для частих читань, але обмежує масштабування при великій кількості записів.

Усі частини роботи реалізовані самостійно: Створено клас DataStructure з багатопотоковою синхронізацією. Написані функції генерації тестових файлів з різними частотами дій. Реалізовано виконання файлів у 1, 2 та 3 потоках з виміром часу. Проведено усереднення результатів та складання таблиці 3×3 .