

Демченко Валерія, К-26

Необхідно створити потокобезпечну структуру даних, яка містить два незалежні цілі поля `field0` та `field1`. Доступ до них мають виконувати одночасно декілька потоків. Для кожної операції читання (`read`), запису (`write`) та виведення всіх полів (`string`) потрібно забезпечити коректну синхронізацію.

Для варіанта **14** задані частоти операцій, за якими формуються тестові файли: `read0` – **25%**, `write0` – **5%**, `read1` – **5%**, `write1` – **25%**, `string` – **40%**.

Необхідно виконати тести у режимах 1, 2 та 3 потоків для трьох наборів даних:

A – частоти відповідають умові варіанта;

B – усі частоти рівні;

C – частоти сильно відрізняються від варіанта.

У реалізації використано два м'ютекси типу `shared_mutex`: `m_mutex0` захищає поле `field0`, а `m_mutex1` — поле `field1`. Читання виконується під `shared_lock`, запис — під `unique_lock`. Операція `string` блокує обидва м'ютекси одночасно у режимі читання з використанням `std::lock`, щоб уникнути взаємного блокування. Така схема дозволяє потокам працювати паралельно з різними полями й мінімізує затримки, оскільки блокування відбувається лише при доступі до одного й того ж поля.

Результати вимірювань часу (мс):

Тип тесту	1 потік	2 потоки	3 потоки
A	1424.0	3473.4	5532.6
B	766.1	2067.5	3790.0
C	139.1	650.1	783.3

Отримані результати відповідають очікуванням: у тестах A час найбільший через велику частку операцій `string`, які блокують обидва поля. У тестах B час зростає плавно, оскільки всі операції рівноймовірні. У тестах C час найменший у однопоточковому режимі через переважання однієї операції, але збільшується з появою конкуренції між потоками. У всіх випадках додавання потоків приводить до збільшення часу, що закономірно через зростання кількості блокувань.

Я самостійно зробила основну логіку, реалізацію класу `DataStructure`, налаштувала синхронізацію, зібрала проєкт і створила тестові файли. За допомогою матеріалів та підказок я уточнила окремі технічні деталі (структуру коду, оптимізацію мейну, оформлення звіту, підказка для написання генератора, але адаптувала). Усі файли, тестування, запуск програм та отримання результатів виконано власноруч.