МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Параллельные алгоритмы»
Тема: Реализация потокобезопасных структур данных с блокировками

Студентка гр. 0304 _____ Говорющенко А.В.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2023

Цель работы.

Изучить принцип построения потокобезопасных структур данных с блокировками.

Задание.

Реализовать итерационное (потенциально бесконечное) выполнение подготовки, обработки и вывода данных по шаблону “производитель-потребитель” (на основе лаб. 1 (части 1.2.1 и 1.2.2)).

Обеспечить параллельное выполнение потоков обработки готовой порции данных, подготовки следующей порции данных и вывода предыдущих полученных результатов.

Использовать механизм “условных переменных”.

2.1

Использовать очередь с “грубой” блокировкой.

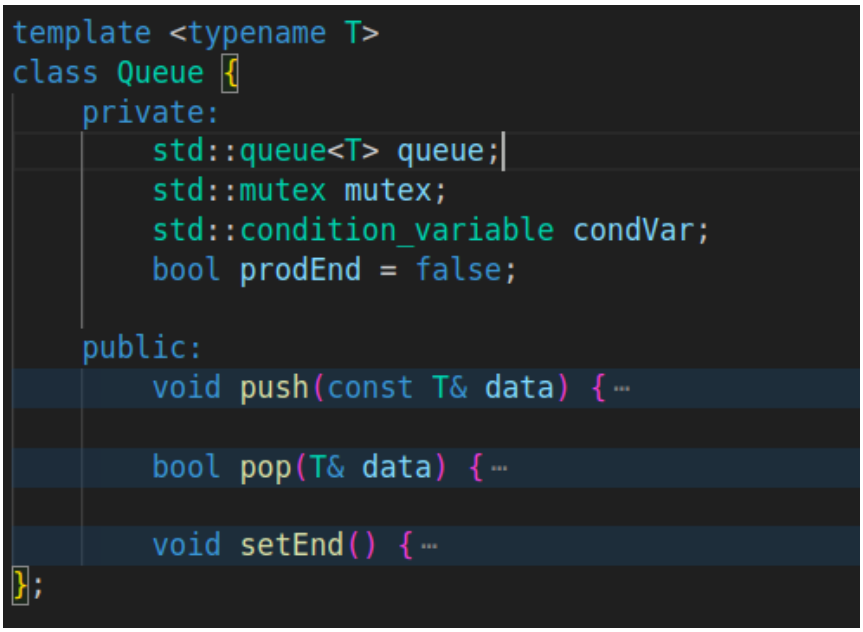
2.2

Использовать очередь с “тонкой” блокировкой.

Выполнение работы.

1. Очередь с «грубой» блокировкой

Для очереди с «грубой» блокировкой был написан шаблонный класс Queue, с методами push, pop и setEnd, которые добавляют и удаляют элемент из очереди соответственно, а также метод, устанавливающий флаг о том, что очередь пуста и больше элементов в нее добавлено не будет.



В данной задаче есть производители и потребители. Производители добавляют в очередь сгенерированные матрицы, а потребители забирают их из очереди и перемножают. Все блокировки происходят с помощью мьютекса и условной переменной. Мьютекс блокируется в начале каждого метода, а условная переменная в методе pop() на время ожидания элементов для обработки.

2. Очередь с «тонкой» блокировкой.

Для реализации очереди с «тонкой» блокировкой был написан класс QueueFine, который представляет собой односвязный список. В отличии от предыдущего класса заключается в том, что здесь есть два мьютекса, один на начало (блокируется при удалении), другой на конец очереди (при добавлении).

```
template <typename T>
class QueueFine {
    private:
        struct Node {
            T data;
            std::unique_ptr<Node> next;
        };

        std::unique_ptr<Node> head;
        Node *tail;
        std::mutex mutexHead;
        std::mutex mutexTail;
        std::condition_variable condVar;
        bool prodEnd = false;

        Node *getLockedTail() {
            std::unique_lock<std::mutex> guard(mutexTail);
            return tail;
        }

    public:
        QueueFine(): head{std::make_unique<Node>()}, tail{head.get()} {}

        void push(const T& data) {…

        bool pop(T& data) {…

        void setEnd() {…
};
```

3. Сравнение потокобезопасных очередей с блокировками.

Сравнение очередей осуществлялось при помощи измерений обработки очереди 300 задач по умножению матриц 10x10. Было рассмотрено 3 случая: одинаковое количество производителей и потребителей, производителей больше, чем потребителей, и наоборот.

В таблице 1 представлено время работы обеих очередей при 10 потребителях и 10 производителях.

Таблица 1. Сравнение очередей при 10 производителях и 10 потребителях.

Очередь	User time, мс	System time, мс	Real time, мс

«Грубые» блокировки	28.851	21.784	34.982
«Тонкие» блокировки	28.743	24.947	35.513

Таблица 2. Сравнение очередей при 4 производителях и 10 потребителях.

Очередь	User time, мс	System time, мс	Real time, мс
«Грубые» блокировки	9.448	5.646	10.475
«Тонкие» блокировки	9.681	5.292	11.412

Таблица 3. Сравнение очередей при 10 производителях и 4 потребителях.

Очередь	User time, мс	System time, мс	Real time, мс
«Грубые» блокировки	29.697	23.854	35.556
«Тонкие» блокировки	29.735	23.682	36.398

После рассмотрения трех ситуаций работы производителей и потребителей можно сделать следующие выводы. Быстрее всего выполняется умножение матриц, когда производителей меньше, чем потребителей. Большие затраты времени наблюдаются при обратном соотношении.

Также можно увидеть, что общее время работы программы (real time) больше при тонких блокировках, в то время как системное незначительно меньше, в случае разного количества производителей и потребителей.

Выводы.

В ходе работы были изучены потокобезопасные очереди с блокировками. Для этого были использовано два вида блокировок - «грубая» и «тонкая».

Было выявлено, что при «тонких» блокировках наблюдается меньшее время ожидания в блокировке (system time), чем в «грубых», однако общее время у них больше. Время ожидания в «тонких» блокировках меньше, когда производителей и потребителей неравное количество.