**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 _____ Нагибин И.С.

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.

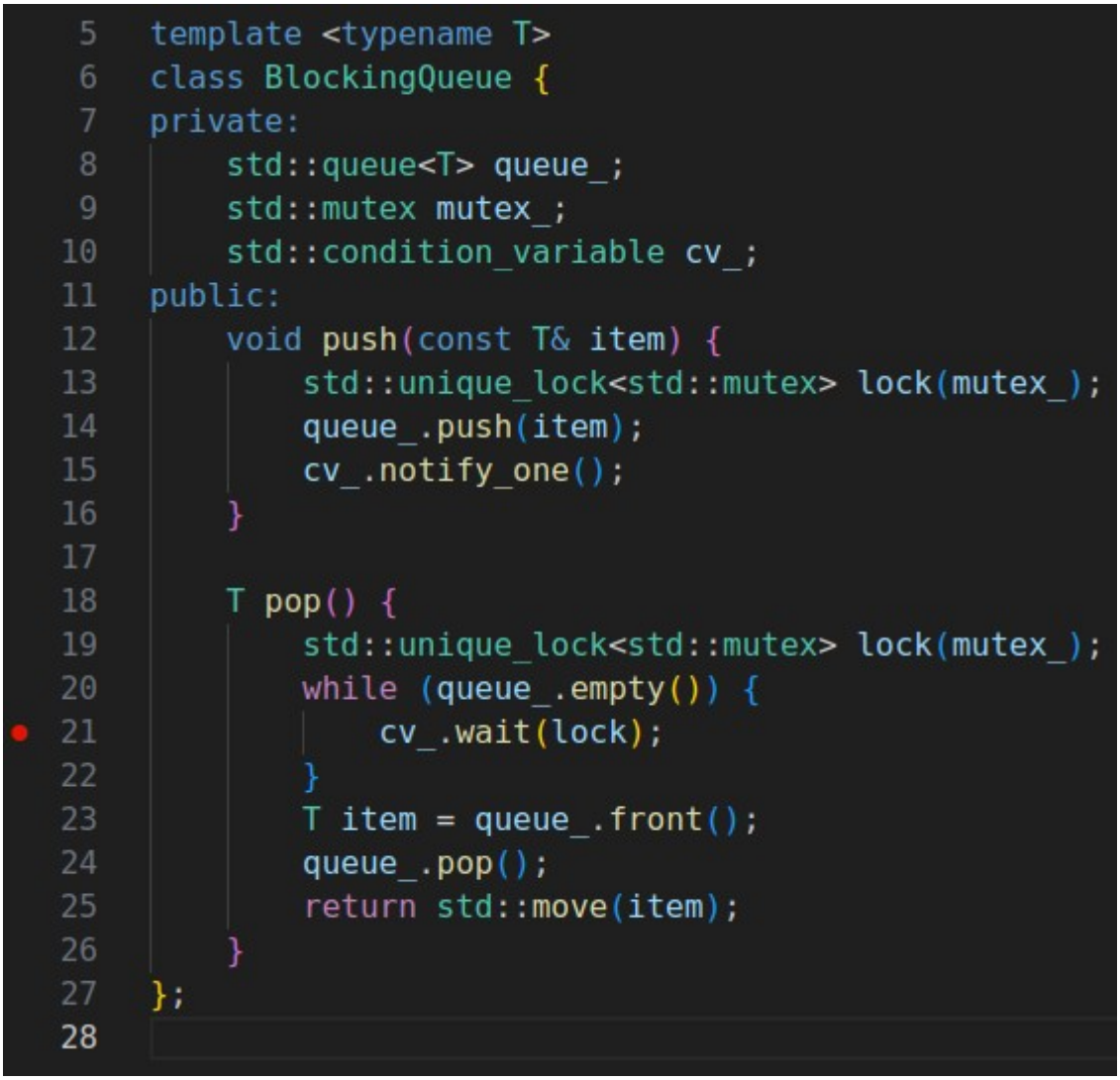
**2.2**

Использовать очередь с “тонкой” блокировкой.

**Выполнение работы.**

**1. Очередь с «грубой» блокировкой**

Для очереди с «грубой» блокировкой был написан шаблонный класс BlockingQueue, с методами push, pop, которые добавляют и удаляют элемент из очереди соответственно.

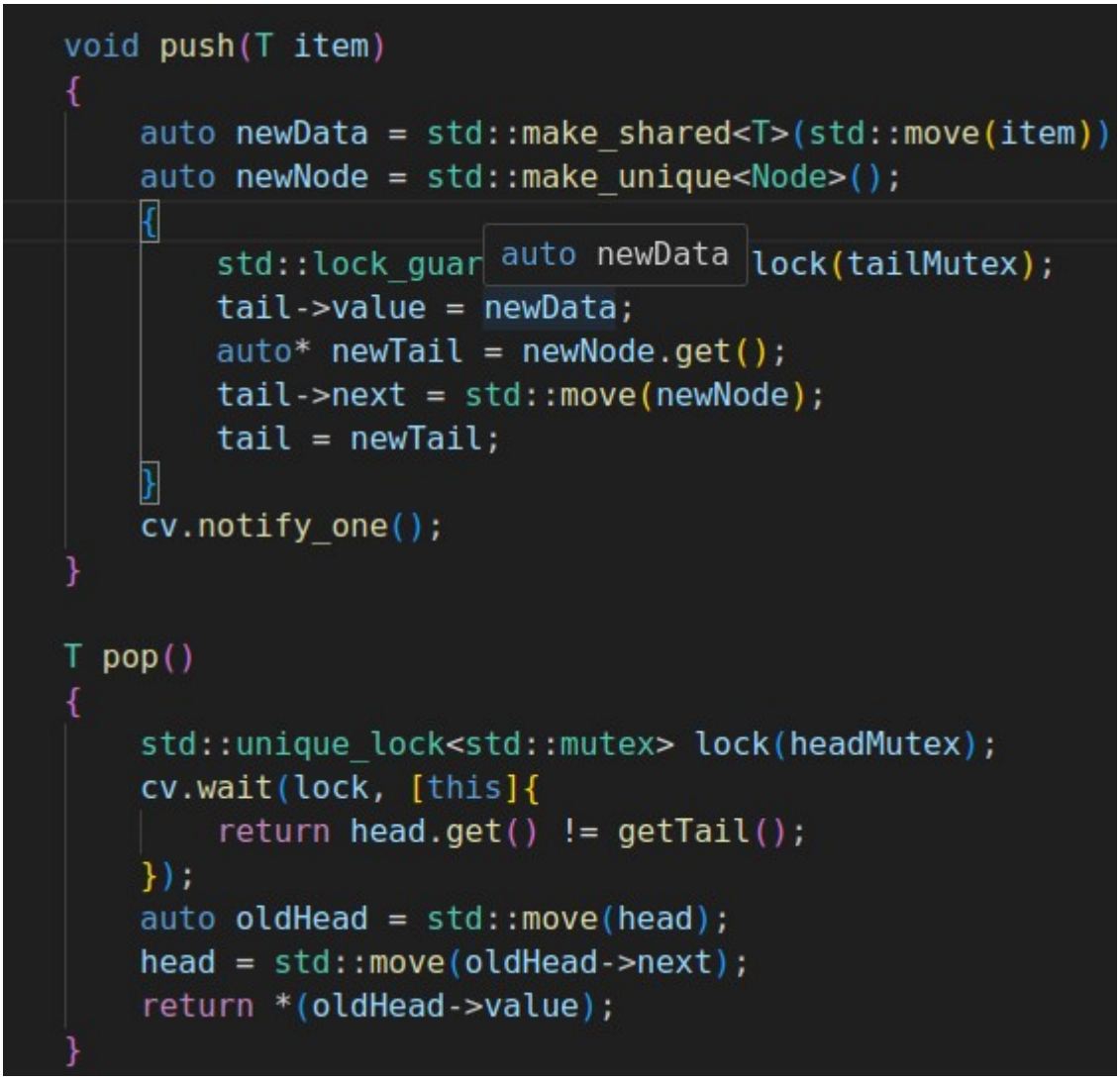


```cpp
template <typename T>
class BlockingQueue {
private:
    std::queue<T> queue_;
    std::mutex mutex_;
    std::condition_variable cv_;
public:
    void push(const T& item) {
        std::unique_lock<std::mutex> lock(mutex_);
        queue_.push(item);
        cv_.notify_one();
    }

    T pop() {
        std::unique_lock<std::mutex> lock(mutex_);
        while (queue_.empty()) {
            cv_.wait(lock);
        }
        T item = queue_.front();
        queue_.pop();
        return std::move(item);
    }
};

```

В данной задаче есть производители и потребители. Производители добавляют в очередь пары индексов для просчёта результирующей матрицы, а потребители забирают их из очереди и перемножают матрицы для получения значения.  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 FineBlockingQueue, 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 (блокируется при добавлении).




**3. Сравнение потокобезопасных очередей с блокировками.**

Сравнение очередей осуществлялось при помощи измерений обработки очереди 300^2 задач по умножению матриц 300x300. Было рассмотрено множество случаев перебором возможных комбинаций числа производителей и потребителей от 1 до 10.

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 | время, с |
|---|---|
| «Грубые» блокировки | 0.254547 |
| «Тонкие» блокировки | 0.238982 |

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

| Очередь | время, с |
|---|---|
| «Грубые» блокировки | 0.269132 |
| «Тонкие» блокировки | 0.24891 |

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

| Очередь | время, с |
|---|---|
| «Грубые» блокировки | 0.279978 |
| «Тонкие» блокировки | 0.261675 |

Таблица 4. Сравнение пяти лучших результатов очередей.

| FineBlockingQueue | BlockingQueue |
|---|---|
| Время 0.231492 s | Время 0.240651 s |
| Время 0.232194 s | Время 0.25159 s |
| Время 0.232829 s | Время 0.2524 s |
| Время 0.232873 s | Время 0.252752 s |
| Время 0.233944 s | Время 0.253376 s |

Также получено среднее время для обработки FineBlokingQueue: 0.306635 с. И для BlockingQueue 0.309384 с.

После рассмотрения полученных данных можно сделать следующие выводы. В лучшем случае очередь с «тонкой» блокировкой работает быстрее, чем очередь с «грубой». В среднем случае разница составляет тысячные доли секунды. Также можно отметить, что при равном количестве потребителей и производителей очереди показывают результаты лучше, чем при разном.

**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 обработки данных, чем при «грубых». Объясняется это частичной блокировкой структуры данных для «тонкой» очереди, в отличие от «грубой» очереди, в которой при обращении блокируется любая другая операция.