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

# ОТЧЕТ

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

Студент гр. 0303 Парамонов В.В.

Преподаватель Сергеева Е. И.

Санкт-Петербург
2023

**Цель работы.**

Исследовать разницу между “грубой” и “тонкой” блокировкой. Воспользоваться структурами данных на основе данных блокировок для решения задачи производители-потребители.

**Постановка задачи.**

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

Обеспечить параллельное выполнение потоков:

- подготовки следующей порции данных;
- обработки готовой порции данных;
- вывода предыдущих полученных результатов.

Данную задачу выполнить со следующими условиями:

2.1. Использовать очередь с “грубой” блокировкой.

2.2. Использовать очередь с “тонкой” блокировкой.

** Использовать механизм “условных переменных”.*

*** Сравнить производительность 2.1. и 2.2 в зависимости от количества производителей и потребителей.*

**Выполнение задач.**

1. **Рассмотрим изменения структуры решения в сравнении с лабораторной 1**:
   1) Появились новые шаблонные классы baseThreadsQueue (Виртуальный базовый класс многопоточной очереди) и его потомки: roughThreadsQueue (многопоточная очередь с “грубой” блокировкой), thinThreadsQueue (многопоточная очередь с “тонкой” блокировкой). Каждый из этих классов основаны на использовании std::queue и синхронизации доступа к её методам с помощью условных переменных. В roughThreadsQueue методы push и pop блокируются одним мьютексом,

что означает, что когда один поток выполняет любую операцию, то остальные блокируются. В thinThreadsQueue используются отдельные мьютексы на push и pop, что обеспечивает блокировку работы только с одними и теми же элементами очереди.

2) Задачи из папки tasks были убраны и заменены одной, которая реализует отдельные потоки из требований лабораторной.

## 2. Исследование получаемых с использованием программы результатов:

1) Исследуем скорость работы задач 2.1 и 2.2 в зависимости от количества производителей и потребителей, результаты измерений времени работы, усредненные для 100 запусков, представлены в таблицах 1 и 2 (размеры матриц – (20, 40), (40, 20); количество генерируемых производителями и обрабатываемыми потребителями наборов данных – 300; макс. Размер очереди – 10):

Таблица 1 – Измерение времени работы программы с очередью с “грубой” блокировкой (2.1)

| Количество производителей | Количество потребителей | Время выполнения задачи(мкс) |
|---|---|---|
| 1 | 1 | 36117 |
| 1 | 20 | 24368 |
| 15 | 5 | 21369 |
| 10 | 10 | 21161 |
| 100 | 100 | **24515** |

Таблица 2 – Измерение времени работы программы с очередью с “тонкой” блокировкой (2.2)

| Количество производителей | Количество потребителей | Время выполнения задачи (мкс) |
|---|---|---|
| 1 | 1 | **35984** |
| 1 | 20 | **24192** |
| 15 | 5 | **21121** |
| 10 | 10 | **20958** |
| 100 | 100 | 24993 |

Исходя из полученных данных в таблицах 1 и 2 очередь с “тонкой” блокировкой в большинстве случаев быстрее, чем очередь с “грубой” блокировкой. Это объясняется тем, что очередь с “тонкой” блокировкой обладает меньшей гранулярностью и блокирует только потоки, выполняющие действия с одними и теми же элементами структуры данных, что обеспечивает выигрыш по времени в сравнении с блокировкой всей структуры данных при каждой операции (“грубая” блокировка). В единственном случае, когда очередь с “грубой” блокировкой оказалось быстрее, сказался фактор того, что количество потоков выполнения намного превысил количество физических потоков компьютера, из-за чего блокировка большего числа потоков оказалась даже лучше (так как потоки меньше мешали друг другу выполнять задачи).

**Заключение.**

В ходе работы была изучена разница между “грубой” и “тонкой” блокировкой. Были использованы очереди на основе данных блокировок для решения задачи производители-потребители. Было практически подтверждено, что в большинстве случаев очередь с “тонкой” блокировкой быстрее очереди с “грубой” блокировкой.