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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Параллельные алгоритмы»

Тема: Реализация потокобезопасных структур данных с блокировками

Студент гр. 0304 _____ Максимов Е.А.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург

2023

Цель работы.

Изучить основные способы работы с блокировками потоков.

Постановка задачи.

Реализовать итерационное (потенциально бесконечное) выполнение подготовки, обработки и вывода данных по шаблону «производитель-потребитель» (потоки на основе лабораторной работы №1).

Обеспечить параллельное выполнение потоков обработки готовой порции данных, подготовки следующей порции данных и вывода предыдущих полученных результатов. Использовать механизм «условных переменных».

1. Использовать очередь с «грубой» блокировкой.

2. Использовать очередь с «тонкой» блокировкой

Сравнить производительность в зависимости от количества производителей и потребителей.

Выполнение работы.

Были написаны функции для ввода/вывода данных `std::vector<int> readInputFile()`, `void writeMatrixToFile(Matrix matrix)`, `bool openOutputFile()`, `void closeOutputFile()`. Чтение параметров программы производится из файла `data/input.txt`. Также из предыдущей лабораторной работы были заимствованы функции, реализующие расчёт матрицы. К этим функциям была добавлена функция `Matrix generateMatrix()` для генерации псевдослучайных квадратных матриц.

В двух случаях для буфера данных была использована очередь `std::queue`. Также в двух случаях количество итераций на поток потребителей было таким, что количество итераций производителей равно количеству итераций потребителя. Размерность матриц составляет 20×20, значения элементов матриц — целые числа в диапазоне от 0 до 999.

1. Очередь с «грубой» блокировкой.

В реализации использовались объекты мьютекса (`std::mutex`) и условной переменной (`std::condition_variable`), общие для производителя и потребителя.

2. Очередь с «тонкой» блокировкой.

Для реализации был создан односвязаный список, которому соответствует класс списка `ThinQueue` и класс узла списка `Node`. В реализации использовались объекты мьютекса вершины и хвоста односвязного списка и условная переменная.

Для измерения времени работы программ была использована утилита `time`. Для каждого теста значение количества задач производителя составляло 100. Результаты тестирования представлены в приложении А.

Выводы.

В ходе лабораторной работы были изучены основные способы работы с блокировками потоков. По результатам анализа полученных данных в ходе тестирования выяснилось, что программа в «тонкой» блокировкой работает быстрее, чем программа с «грубой» блокировкой. Практическим результатом лабораторной работы является программный код, реализующий механизмы «грубой» и «тонкой» блокировок.

ПРИЛОЖЕНИЕ А
ТЕСТИРОВАНИЕ

Таблица А1 — Исследование зависимости времени работы программы от количества потоков для «грубой» блокировки

№	Количество производителей	Количество потребителей	Фактическое время (real), мс	Пользовательское время (user), мс	Процессорное время (sys), мс
1	1	1	8	7	2
2	1	2	12	12	5
3	1	5	16	20	6
4	10	1	47	40	37
5	10	10	46	38	46
6	10	100	90	76	144
7	100	1	345	215	283
8	100	10	387	296	512
9	100	100	402	323	461

Таблица А2 — Исследование зависимости времени работы программы от количества потоков для «тонкой» блокировки

№	Количество производителей	Количество потребителей	Фактическое время (real), мс	Пользовательское время (user), мс	Процессорное время (sys), мс
1	1	1	36	32	1
2	1	2	32	45	5
3	1	5	20	35	5
4	10	1	276	282	16
5	10	10	96	393	30
6	10	100	101	265	35
7	100	1	2676	2621	172
8	100	10	901	4655	261
9	100	100	844	4080	279