Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.04.02 Информационные технологии и системная инженерия

Профиль: «Информационные технологии и системная инженерия»

**Лабораторные работы**

**по дисциплине: «Параллельное программирование»**

**Выполнил**

студент гр. ИТСИ-24-1м

**Пеленев Денис Вячеславович**

**Принял** Преподаватель кафедры ВММБ

**Истомин Денис Андреевич**

Пермь 2025

**Оглавление**

[Лабораторная работа 1 4](#_Toc198544192)

[Задание 4](#_Toc198544193)

[Реализация 4](#_Toc198544194)

[Результат 4](#_Toc198544195)

[Лабораторная работа 2 5](#_Toc198544196)

[Задание 5](#_Toc198544197)

[Реализация 5](#_Toc198544198)

[Результат 5](#_Toc198544199)

[Лабораторная работа 3 7](#_Toc198544201)

[Задание 7](#_Toc198544202)

[Реализация 7](#_Toc198544203)

[Результат 7](#_Toc198544204)

[Лабораторная работа 4 9](#_Toc198544206)

[Задание 9](#_Toc198544207)

[Необходимо: 9](#_Toc198544208)

[Реализация 9](#_Toc198544213)

[Результат 10](#_Toc198544214)

[Лабораторная работа 5 12](#_Toc198544215)

[Задание 12](#_Toc198544216)

[Необходимо: 12](#_Toc198544217)

[Реализация 12](#_Toc198544220)

[Результат 12](#_Toc198544221)

[Лабораторная работа 6 13](#_Toc198544222)

[Задание 13](#_Toc198544223)

[Реализация 13](#_Toc198544224)

[Результат 13](#_Toc198544227)

[Лабораторная работа 7 15](#_Toc198544228)

[Задание 15](#_Toc198544229)

[Необходимо: 15](#_Toc198544230)

[Реализация 15](#_Toc198544233)

[Результат 15](#_Toc198544234)

# Лабораторная работа 1

## Задание

Необходимо:

1. При помощи SSE инструкций написать программу (или функцию), которая перемножает массив из 4х чисел размером 32 бита;
2. Написать аналогичную программу (или функцию) которая решает ту же задачу последовательно;
3. Сравнить производительность;
4. Проанализировать сгенерированный ассемблер: gcc -S sse.c.

## Реализация

Была написана программа, которая выполняет перемножение массива из 4 чисел при помощи последовательных действий, через цикл, а также при помощи SSE инструкции.

Программа, которая выполняет перемножение при помощи SSE инструкции, завершила работу в 4 раза быстрее. Это связано с тем, что SSE использует принцип SIMD, при помощи которой числа перемножаются не по одному, а 4 числа в массиве за раз.

## Результат

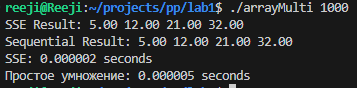


Рис. 1 Результат работы программы в Лабораторной 1

# Лабораторная работа 2

## Задание

Необходимо:

1. При помощи Pthreads написать программу (или функцию), которая создает n потоков и каждый из потоков выполняет длительную операцию;
2. Написать аналогичную программу (или функцию), которая решает ту же задачу последовательно;
3. Сравнить производительность.

## Реализация

Была написана программа, которая производит вычисление при помощи последовательных действий и используя pthreads для многопоточного вычисления в n потоках.

Программа, которая реализует многопоточное вычисление, выполнилась за 2.76 секунд, а последовательное за 9.87.

## Результат

## 

Рис. 2 Последовательное вычисление

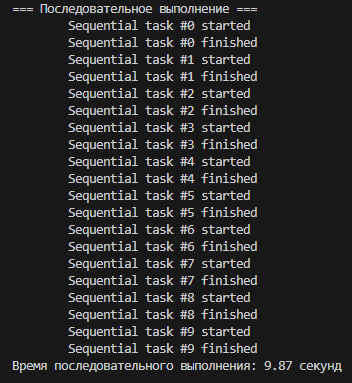
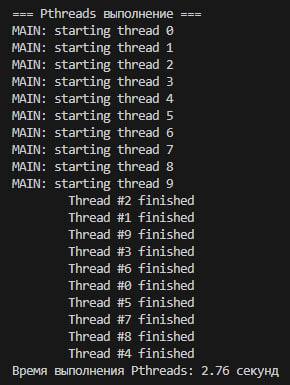


Рис. 3 Многопоточное вычисление



# Лабораторная работа 3

## Задание

Необходимо:

1. При помощи OpenMP написать программу (или функцию), которая создает n потоков и каждый из потоков выполняет длительную операцию;
2. Сравнить с последовательной программой и программой с Pthreads из предыдущей лабораторной работы.

## Реализация

Была написана программа, которая производит вычисление при помощи последовательных действий и используя OpenMP для многопоточного вычисления в n потоках.

Программа, которая реализует многопоточное вычисление, выполнилась за 2.66 секунд, а последовательное за 9.87. По скорости вычислений OpenMP обгоняет последовательное вычисление и pthreads.

## Результат

## 

Рис. 4 Последовательное вычисление

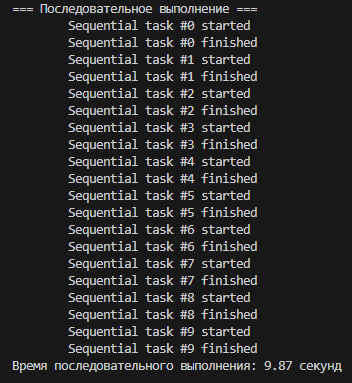
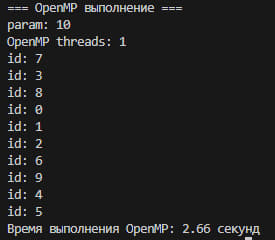


Рис. 5 Многопоточное вычисление



# Лабораторная работа 4

## Задание

## Необходимо:

## Написать программу, которая запускает несколько потоков;

## В каждом потоке считывает и записывает данные в HashMap, Hashtable, synchronized HashMap, ConcurrentHashMap;

## Модифицировать функцию чтения и записи элементов по индексу так, чтобы в многопоточном режиме использование непотокобесопасной коллекции приводило к ошибке;

## Сравнить производительность.

## Реализация

В данной лабораторной работе были рассмотрены 4 вида коллекции в многопоточном приложении из 50 потоков. В данных коллекциях были использованы функции записи и чтения из коллекции, что позволило проверить потокобезопасность.

1. HashMap – непотокобезопасная коллекция. При работе с ней возникали ошибки связанные с попыткой конкурентного доступа к общим данным. Что в свою очередь так же сказалось и на финальном результате.
2. Hashtable – потокобезопасная коллекция. В данной коллекции уже не возникало гонки данных, что обеспечило корректный результат работы.
3. Collections.synchronizedMap – потокобезопасная коллекция. В данной коллекции уже не возникало гонки данных, что обеспечило корректный результат работы.
4. ConcurrentHashMap – потокобезопасная коллекция. В данной коллекции уже не возникало гонки данных, что обеспечило корректный результат работы.

Как видно из результатов, HashMap не стоит использовать, когда необходима коллекция, к которой будет конкурентный доступ. Наиболее быстрой и при этом безопасной коллекцией является ConcurrentHashMap. Связано это с тем, что в отличие от HashTable и synchronizedMap, которые блокируются полностью при доступе к ним, в ConcurrentHashMap блокируются лишь отдельные бакеты, что позволяет другим потокам в это время вести работу с другими бакетами.

## Результат

Рис. 6 Ошибки при работе HashMap

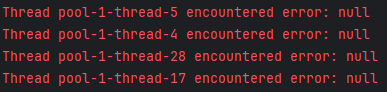


Рис. 7 Результат работы коллекций

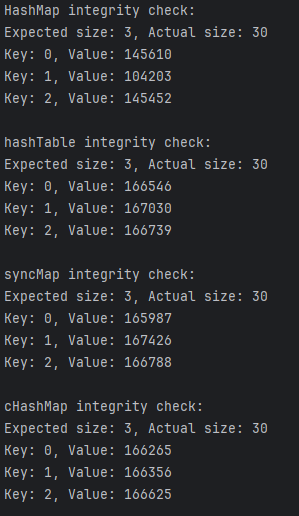
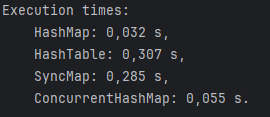


Рис. 8 Сравнение скорости работы коллекций



# Лабораторная работа 5

## Задание

## Необходимо:

## Написать программу, которая демонстрирует работу считающего семафора;

## Написать собственную реализацию семаформа (наследование от стандартного с переопределением функций) и использовать его.

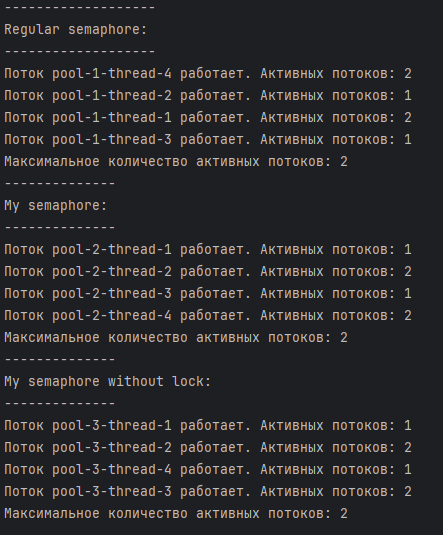
## Реализация

В данной лабораторной работе был реализован семафор с применением ReentrantLock, а так же без него. ReentrantLock обеспечивает одному и тому же потоку возможность несколько раз захватывать разрешение на работу с общими данными.

В лабораторной работе использовалось 4 потока и 2 разрешения на работу с общими данными. Как видно из результатов, программа успешно справилась с работой.

## Результат

Рис. 9 Результат работы программы в Лабораторной 5



# Лабораторная работа 6

## Задание

Необходимо создать клиент-серверное приложение:

1. Несколько клиентов, каждый клиент - отдельный процесс**;**
2. Серверное приложение - отдельный процесс**;**
3. Клиенты и сервер общаются с использоваени Socket**.**

Необходимо релизовать функционал:

1. Клиент подключается к серверу**;**
2. Сервер запоминает каждого клиента в java.util.concurrent.CopyOnWriteArrayList**;**
3. Сервер читает ввод из консоли и отправляет сообщение всем подключенным клиентам**.**

## Реализация

## В рамках задачи выполнялось ознакомление и работа с Java IPC. Созданы клиент и сервер, которые обмениваются между собой сообщениями через socket. Сервер - смотрит кто подключился к определенному ip:port и добавляет новых пользователей, а затем отслеживает их действия. Клиент – отправляет сообщения на сервер.

## Как видно из результатов, процессы-клиенты и сервер взаимодействуют корректно, выводя сообщения.

## Результат

Рис. 10 Результат работы сервера

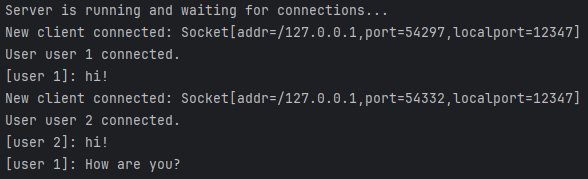


Рис. 12 Результат работы первого пользователя

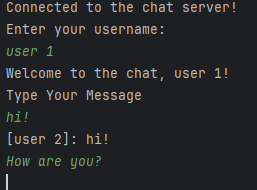
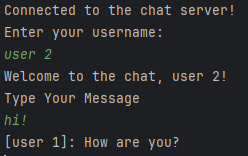


Рис. 12 Результат работы второго пользователя



# Лабораторная работа 7

## Задание

## Необходимо:

## Изучить библиотеку mappedbus;

## Запустить готовые тестовые примеры.

## Реализация

В данной программе был рассмотрен пример для межпроцессорного взаимодействия. ObjectWriter пишет данные в memory-mapped file, а ObjectReader – читает данные из этого файла.

Как видно из результатов, у нас есть 2 ObjectWriter которые пишут в общий файл, а ObjectReader читает эти данные.

## Результат

Рис. 13 Результат работы программы

