**Лабораторная номер 1. Основы работы с потоками.**

**Цель:** Изучение работы с потоками (для С++ API операционной системы либо c++ std threads) и методов замера времени программ.

**Язык: С++** *(или любой другой императивный язык с явно создаваемыми потоками)***.**

**Задание:**

Написать простую программу, реализующую простые вычисления в одном и в нескольких потоках.

Количество потоков вычислений должно задаваться в процессе работы. Как минимум, необходимо сравнить работу в режиме одного потока и нескольких.

Результат последовательной и многопоточной версии должен выводиться при едином запуске программы (без перекомпиляции программы с измененными флагами).

Для всех реализаций вычислений должен выполняться замер времени работы и проверка корректности результатов работы (сравнение с однопоточным вычислением), вычисления должны производиться на основе одних и тех же исходных данных. Необходимо добиться, что бы распараллеленный вариант работал быстрее и выдавал идентичный последовательному результат.

Для запуска потоков использовать вызовы API операционной системы или C++ threads (или что там вашем любимом языке). Для замеров времени рекомендуется omp\_get\_wtime(). Замерять время в конфигурации Release со всеми оптимизациями!

В замере времени должно учитываться только время вычислений, время подготовки данных учитывать не нужно.

**Варианты:**

**\*** Данные можно генерировать случайными, объем входных данных подбирайте так, что бы время работы было не менее 100мс

\*Массив заполняем числами в заданном фиксированном диапазоне, например m[i] = rand() % M, где M задается (это влияет на вероятность возникновения ошибки в некоторых алгоритмах)

1. Посчитать в массиве количество нулей, отрицательных и положительных элементов (работает очень быстро даже в 1 потоке!)
2. Посчитать среднее значение элементов массива
3. Подсчитать в массиве количество цепочек из k (k=10) нулей;
4. Поиск самой длинной последовательности элементов, сумма которых равна 0;
5. Подсчет количества уникальных элементов (вывести количество значений, встречающихся в массиве ровно один раз);
6. То же что 1-5, но на фильтрованном массиве («блочный фильтр» ( или фильтр Гаусса ())
7. Сортировка подсчетом массива целых чисел.
8. Реализовать умножение матриц (нужно реализовать вариант с гонкой (т.е. неправильный) и нормальный, в котором либо гонки нет, либо доступ к данным синхронизирован)
9. Решение системы линейных уравнений (то же, с гонкой и нормальный)
10. что то свое (но лучше заранее проконсультироваться), на что фантазии хватит.

**Баллы:**

1. Выбор и реализация последовательной версии алгоритма: **2б 3б 4б**
2. Реализация корректной параллельной версии алгоритма: **2б 3б 4б** (Должна выдавать результат идентичный последовательному алгоритму но быстрее)
3. Постановка эксперимента (таблицы, графики с результатами, выводы) **1б**

Итого: 5 – 7 – 9 баллов.

**Лабораторная номер 2. Исследование методов синхронизации, эксперименты с ошибками синхронизации.**

**Цель:** Изучение методов синхронизации многопоточных программ и агрегации данных при многопоточных вычислениях.

**Язык:** то же, что в 1лаб.раб.

**Задача:**

В программе из лабораторной 1 реализовать разные методы синхронизации / агрегации вычислений (не менее 3х):

* запись в общие переменные;
* атомарные операции;
* синхронизация критическими секциями или мьютексами;
* индивидуальные переменные для вычислений в каждом потоке.

Сравнить и обосновать разницу во времени работы и ошибки в вычислениях, оценить зависимость вероятности возникновения ошибок от исходных данных.

Результат работы всех реализаций должен выводиться при едином запуске программы (без перекомпиляции программы с измененными флагами).

Для всех реализаций вычислений должен выполняться замер времени работы и проверка корректности результатов работы (сравнение с однопоточным вычислением), вычисления должны производиться на основе одних и тех же исходных данных. Необходимо добиться, что бы распараллеленный вариант работал быстрее и выдавал идентичный последовательному результат.

**Баллы:**

1. Реализация алгоритма с разными вариантами синхронизации/агрегации данных **+1б** за каждый вариант кроме тех, что реализованы в 1 лаб.раб.
2. Реализация «некорректного алгоритма» в котором состояние гонки может привести к неверным результатам вычислений **1б**
3. Воспроизведение ошибки от состояния гонки – подобрать параметры, при которых ошибка в вычислениях более-менее устойчиво воспроизводится: **1б**
4. Исследование частоты ошибок в зависимости от параметров алгоритма **1б**.
5. Постановка эксперимента (таблицы, графики с результатами, выводы) **1б**

Итого: **4+** балла.