# **Отчёт по лабораторной работе №5**

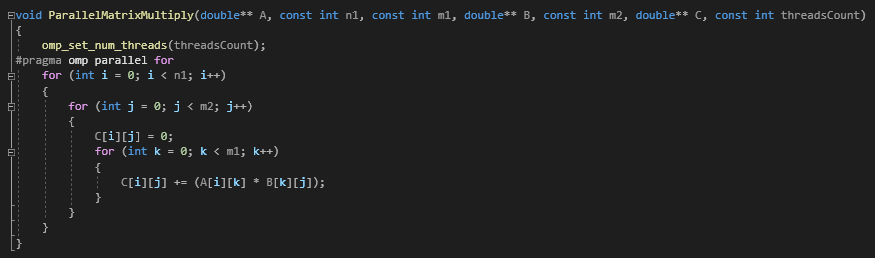
Часть 1

Цель: произвести параллельное перемножение двух матриц, сравнить результаты

**Программная реализация:**

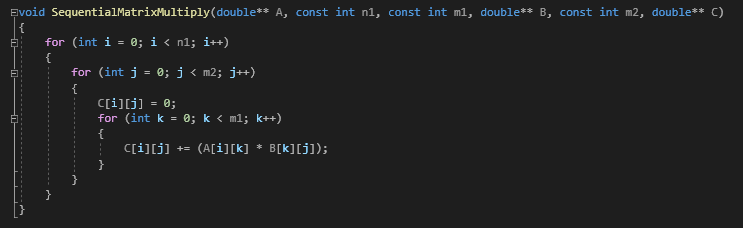
В методе ParallelMatrixMultiply производится параллельное умножение двух матриц A и B размеров n1 x m1 и n2 x m2 соответственно. Результат сохраняется в матрице C размером n1 x m2.

Распараллеливания циклов с помощью директивы "#pragma omp parallel for".



Последовательное умножение двух матриц A и B размеров n1 x m1 и n2 x m2 соответственно. Результат сохраняется в матрице C размером n1 x m2.

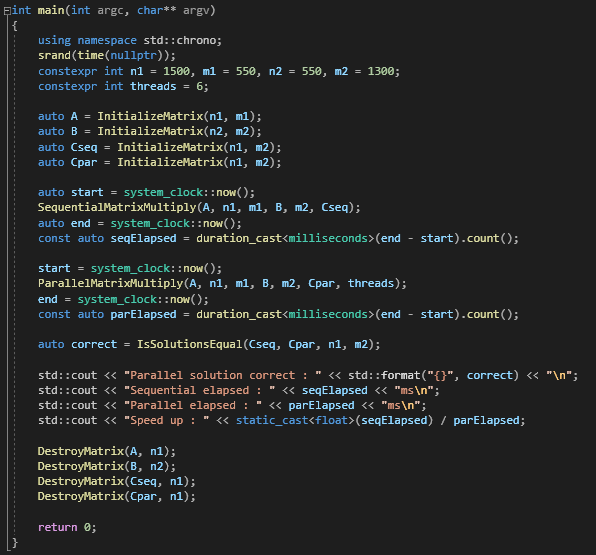
Функция использует вложенные циклы для итерации по элементам матрицы C. Каждый элемент матрицы C вычисляется как сумма произведений соответствующих элементов матриц A и B.



В начале программы определяются размеры матриц A, B и Cseq/Cpar. Затем создаются и инициализируется двумерные массивы с помощью функции InitializeMatrix.

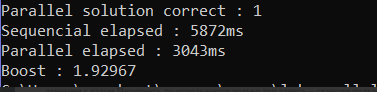
Далее выполняется последовательное умножение матриц A и B с помощью функции SequentialMatrixMultiply, а затем параллельное умножение матриц A и B с помощью функции ParallelMatrixMultiply. Засекается время выполнения каждого метода.

Затем результаты сравниваются с помощью функции IsSolutionsEqual, чтобы проверить, является ли параллельное решение правильным.

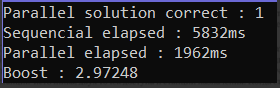


**Вывод программы:**

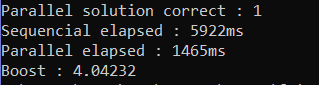
**Threads: 2**



**Threads: 4**



**Threads: 8**



Часть 2

Цель: разработать частотный анализатор текстов. k “читающих” потоков считывают данные из n файлов (формат файлов произвольный, можно использовать txt). Поток-интерфейс отвечает за взаимодействие с пользователем (командная строка или иной формат). Пользователю доступны следующие команды: вывести на экран 5 самых распространённых на данный момент букв; вывести на экран вероятность появление буквы, введённой пользователем; выдать три самые редкие буквы.

**Решение:**

В начале программы используем функцию readFiles, которая читает содержимое нескольких файлов и подсчитывает частоту каждой буквы в них. Функция использует параллельное программирование с помощью OpenMP, чтобы ускорить выполнение задачи. Входные параметры функции - количество файлов fileCount и количество потоков threadCount. Результат подсчета частот записывается в выходной вектор freqOut. Каждый поток отвечает за чтение своей порции файлов, а затем результаты суммируются в критической секции, чтобы избежать гонок данных.

