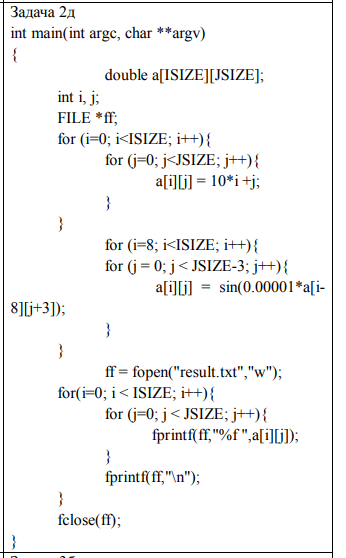
**Лабораторная работа, вариант 2 д**

**Условие**



**Идея решения**

Задача состоит из двух циклов, первый «инициализация» массива не имеет зависимости по данным, поэтому параллелится разбиением на равные части для каждого процессора.

Второй цикл имеет зависимость по данным вычисленным на предыдущей итерации цикла(конкретно [i-8]), поэтому раздать каждому потоку непрерывный набор индексов для подсчёта становится невозможным. Было принято решение что при параллельном запуске программы первый поток только пересылает другим массив размера JSIZE элементов строки i и получается назад JSIZE-3 элементов i+8 строки. Так как зависимость по данным для одного конкретного i не существует относительно вычисленных раньше элементов, то можно обойтись изменениями этого же массива, не создавая новый. После вычисления массив передается пересылается первому потоку.

**Запуск**

./build.sh

./run.sh -n <number of threads>

**Профилирование**

./test.sh

**Проверка**

Команда diff single.txt result.txt должна иметь пустой вывод

single.txt получена по результатам работы оригинальной однопоточной работы и не изменяется (эталон).

**Результаты**

Проведено исследования эффективности алгоритма от 1 до 8 потоков, при наличии **6ти** реальных. При 1 потоке исполняется только сам алгоритм, никаких пересылок не выполняется. Время 0.043 секунды. При двух потоках 0.012 секунды. Почему ускорение более чем в 2 раза? Ускорение достигается на первом цикле за счёт распараллеливания инициализации. Но подсчет второго цикла исполняется только на втором потоке и казалось бы ускорения быть не должно. Предположение в том, что этот поток когда считает максимальный сдвиг по памяти всего 3 \* sizeof(double) когда это делается в однопоточной задаче, то происходят постоянные прыжки по памяти на 8003 \* sizeof(double) что вероятно вносит основной вклад во время вычисления программы, тут этот недочет исправлен. В качестве примера можно проитерироваться по двумерному массиву сначала в формате строки{столбцы} потом в столбцы{строки} во втором случае время итерирования может быть в десятки раз больше.

Далее видно ускорение до 4х потоков, потом время не изменяется, объяснить можно тем, что в момент четырех потоков скорость подсчета трех строк из JSIZE элементов сравнивается со скоростью одной итерации приема-передачи такого массива. То есть если мы увеличим чисто потоков, то все кроме первого будут простаивать определенное время, так как уже посчитали свой массив и ждут когда первый его примет и пришлет новый.

На 7 и 8 потоках видно резкое возрастание времени работы программы, что объясняется наличием всего 6ти реальных, поэтому создаются виртуальные потоки из-за которых происходит сильное замедление работы программы.

