Государственное образовательное учреждение высшего профессионального образования

"Московский государственный технический университет имени Н.Э.Баумана"

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №4
курса Анализ алгоритмов
на тему "Распараллеленный алгоритм Винограда умножение матриц"

Студент: Денисенко А.А., группа ИУ7-64б
Преподаватели: Волкова Л.Л., Строганов Ю.В.

Москва, 2020 г.

Содержание

Введение

Умножение матриц является основным инструментом линейной алгебры и находит широкое применение в математике, 3D-графике, алгоритмах машинного обучения, механике, экономике и многих других областях. Например, в 3D-графике матрицы используются для преобразования координат. С этой целью точка, представленная как однострочная матрица, умножается на соответствующую матрицу преобразования. Чтобы повернуть точку относительно оси X и параллельно перенести ее, нужно ее матрицу с ее координатами последовательно умножить на матрицу поворота и параллельного переноса.

Если рассмотреть результат умножения двух матриц, то видно, что каждый элемент в нём представляет собой скалярное произведение соответствующих строки и столбца исходных матриц. Можно заметить также, что такое умножение допускает предварительную обработку, позволяющую часть работы выполнить заранее. На такой оптимизации построен алгоритм Винограда, который будет рассмотрен в данной работе.

Также этот алгоритм будет оптимизирован с использованием многопоточности. Постоянное повышение мощности компьютерных систем приводит к тому, что задачи, которые еще в недалеком прошлом не могли быть решены в реальном масштабе времени, успешно решаются благодаря использованию параллельных алгоритмов, реализуемых на многопроцессорных системах, обладающих высоким быстродействием.

Целью данной работы является изучение и реализация параллельного алгоритма Винограда для умножения матриц, сравнение зависимости времени работы алгоритма от числа параллельных потоков исполнения и размера матриц и сравнение стандартного и параллельного алгоритма. Для достижения поставленной цели необходимо решить следующие задачи:

1. изучение возможности улучшения алгоритма Винограда с помощью его распараллеливания;
2. изучение организации взаимодействия параллельных потоков
3. получение практических навыков реализации указанных алгоритмов;
4. сравнительный анализ перечисленных алгоритмов умножения матриц по затрачиваемым ресурсам времени;
5. экспериментальное подтверждение различий во временно́й эффективности алгоритма в зависимости от числа потоков при помощи разработанного программного обеспечения на материале замеров процессорного времени выполнения реализации на варьирующихся размерах квадратных матриц;
6. описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

1 Аналитическая часть

Матрицы вида $A = [m \times n]$ и $B = [n \times k]$, в которых количество столбцов в первой матрице совпадает с количеством строк во второй, называются согласованными. Произведение двух согласованных матриц $A = [m \times n]$ и $B = [n \times k]$ - это новая матрица $C = [m \times k]$, элементы которой считаются по формуле: $c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + ... + a_{in} \cdot b_{nj} = \sum_{i=1}^{n} a_{it} \cdot b_{tj}$

1.1 Алгоритм Винограда

Алгоритм Винограда является более эффективным благодаря сокращению количества операций умножения. Результат умножения двух матриц представляет собой скалярное произведение соответствующих строки и столбца. Такое умножение позволяет выполнить заранее часть работы. Пусть $U = (u_1, u_2, u_3, u_4)$, $V = (v_1, v_2, v_3, v_4)$
Тогда $U \times V = u_1 \cdot v_1 + u_2 \cdot v_2 + u_3 \cdot v_3 + u_4 \cdot v_4$
Это равенство можно переписать в виде
$U \times V = (u_1 + v_2) \cdot (u_2 + v_1) + (u_3 + v_4) \cdot (u_4 + v_3) - u_1 \cdot u_2 - u_3 \cdot u_4 - v_1 \cdot v_2 - v_3 \cdot v_4$
Значения $u_1 \cdot u_2, u_3 \cdot u_4, v_1 \cdot v_2, v_3 \cdot v_4$ можно рассчитать заранее.

Однако для хранения массивов строковых и столбцовых коэффициентов требуется дополнительная память, а в случае нечетного количества столбцов матрицы требуются дополнительные вычисления. В таком случае полученная формула принимает вид: $U \times V = u_1 \cdot v_1 + u_2 \cdot v_2 + u_3 \cdot v_3 + u_4 \cdot v_4 + u_5 \cdot v_5 = (u_1 + v_2) \cdot (u_2 + v_1) + (u_3 + v_4) \cdot (u_4 + v_3) - u_1 \cdot u_2 - u_3 \cdot u_4 - v_1 \cdot v_2 - v_3 \cdot v_4 + u_5 \cdot v_5$

1.2 Параллельный алгоритм Винограда

Как было показано в лабораторнои работе №2, трудоемкость алгоритма Винограда имеет сложность $O(nmk)$ для умножения матриц размеров $[nm]$ на $[mk]$. Таким образом, чтобы значительно ускорить время работы алгоритма, следует распараллелить ту его часть, которая содержит 3 вложенных цикла.

Вычисление результата для каждои строки происходит независимо от результата выполнения умножения для других строк. Поэтому потоки следует выбрать таким образом, чтобы каждый из них выполнял вычисление некоторых строк результирующии матрицы. Такой подход выбран потому, что проход по строкам матрицы является наиболее эффективным с точки зрения организации данных в памяти.

2 Конструкторская часть

2.1 IDEF0

Ниже (на рис. 1) представлена IDEF0-диаграмма алгоритма умножения матриц.

Рисунок 1 : Диаграмма алгоритма умножения матриц

2.2 Разработка алгоритмов

В данном разделе приведена схема работы алгоритма Винограда (рис. 2, 3) и схема работы распараллеленного алгоритма Винограда(рис. 4, 5)

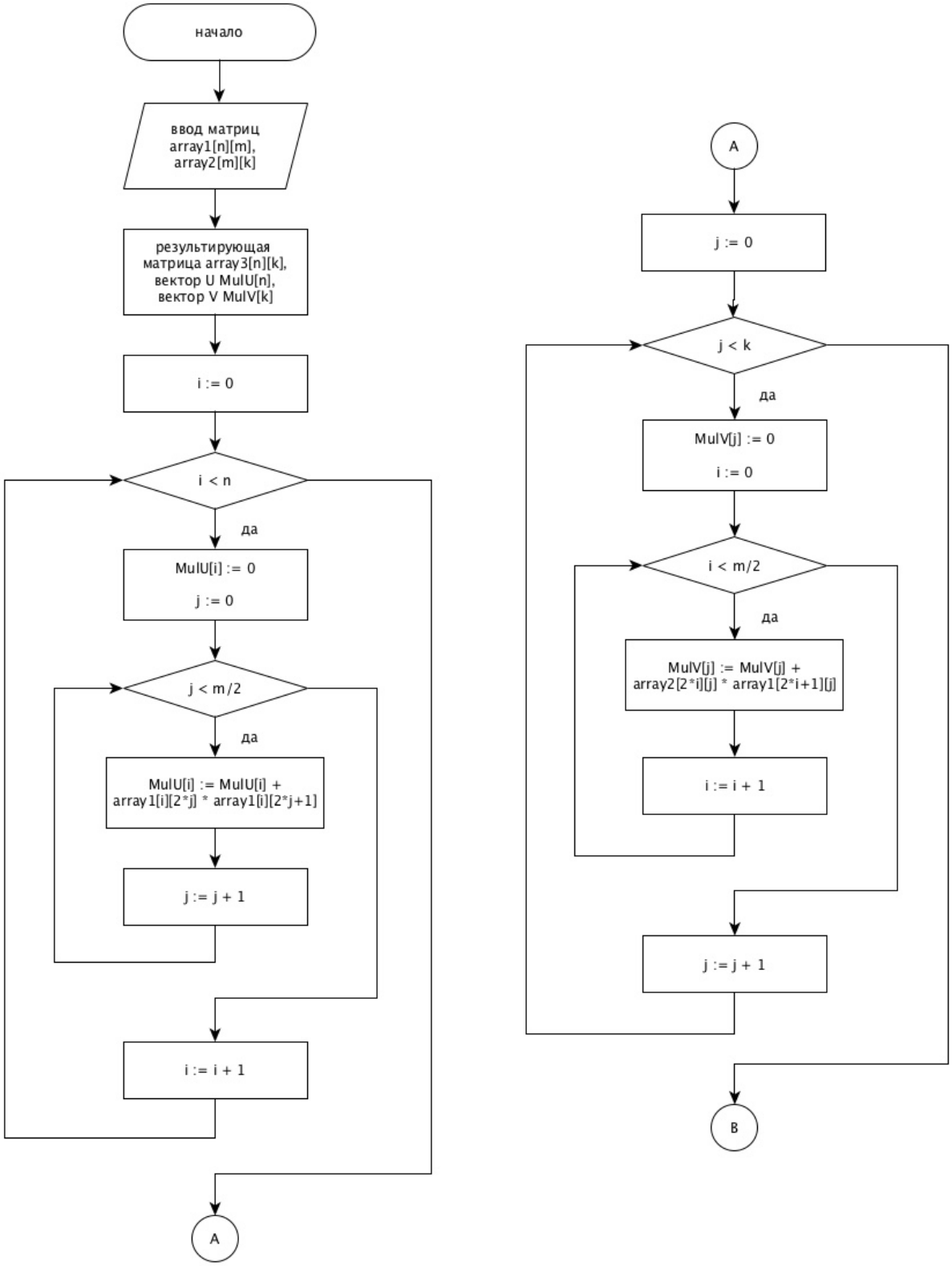

Рисунок 2 : Схема алгоритма Винограда умножения матриц (часть 1)

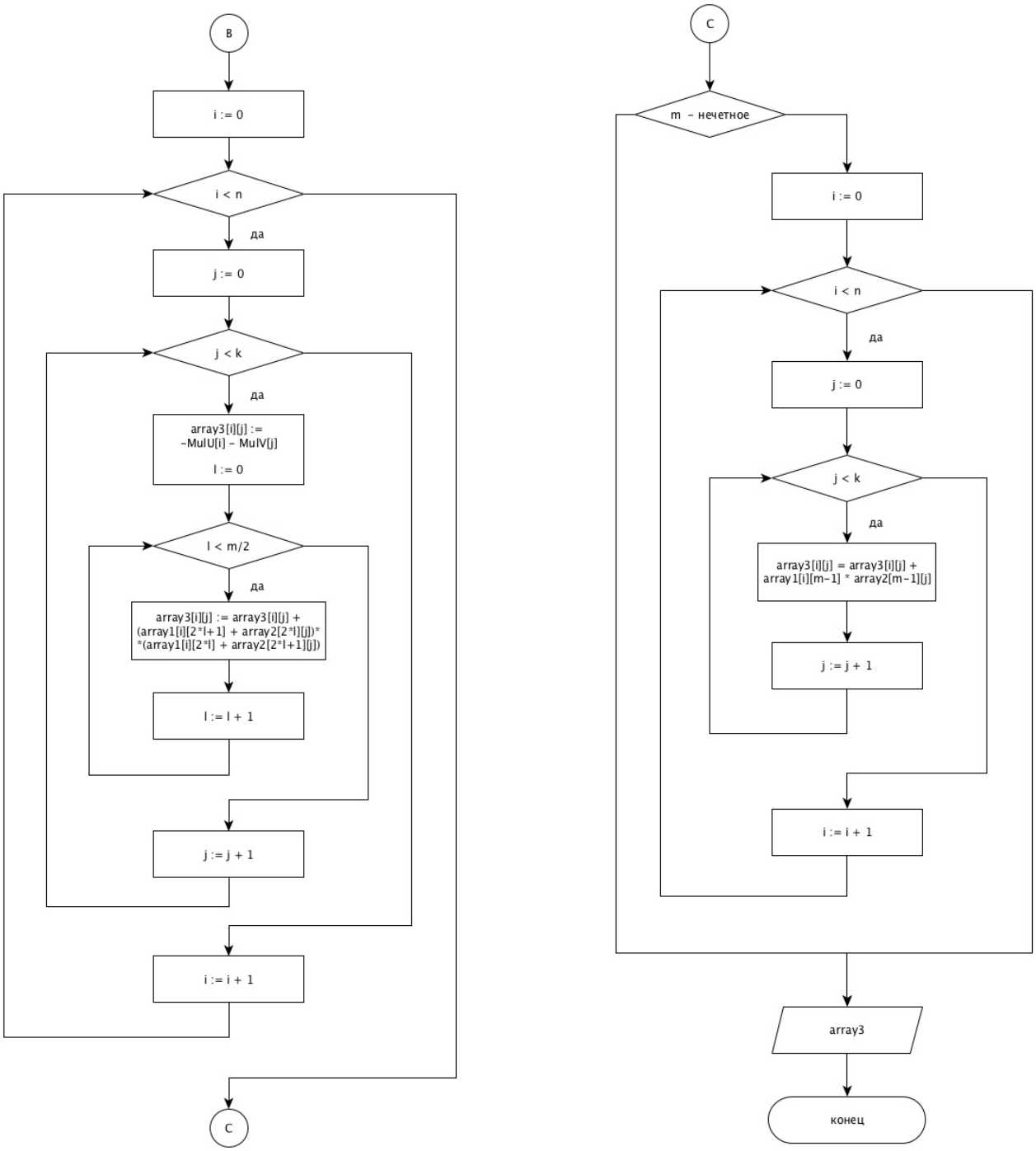

Рисунок 3 : Схема алгоритма Винограда умножения матриц (часть 2)

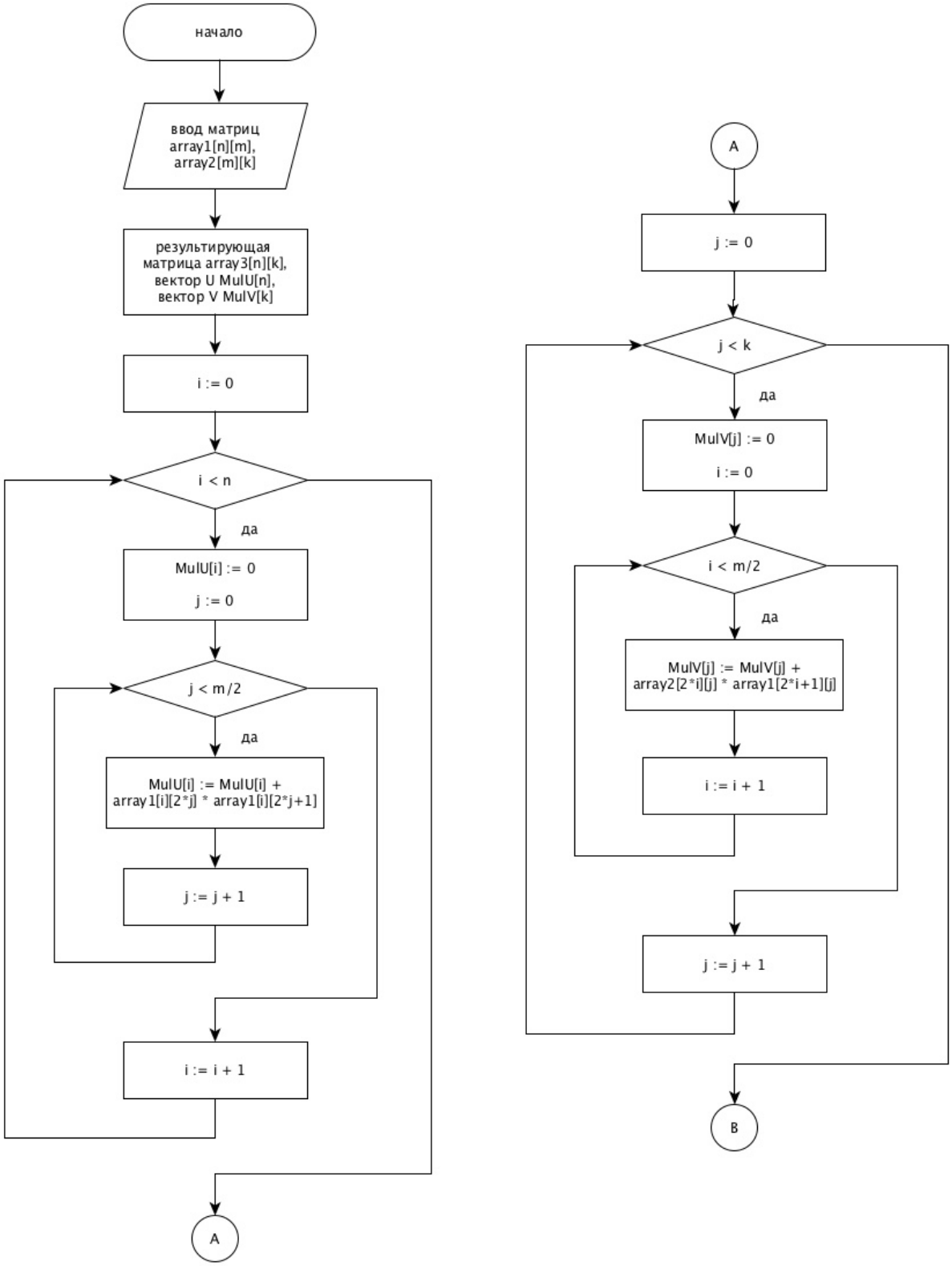

Рисунок 4 : Схема распараллеленного алгоритма Винограда умножения матриц (часть 1)

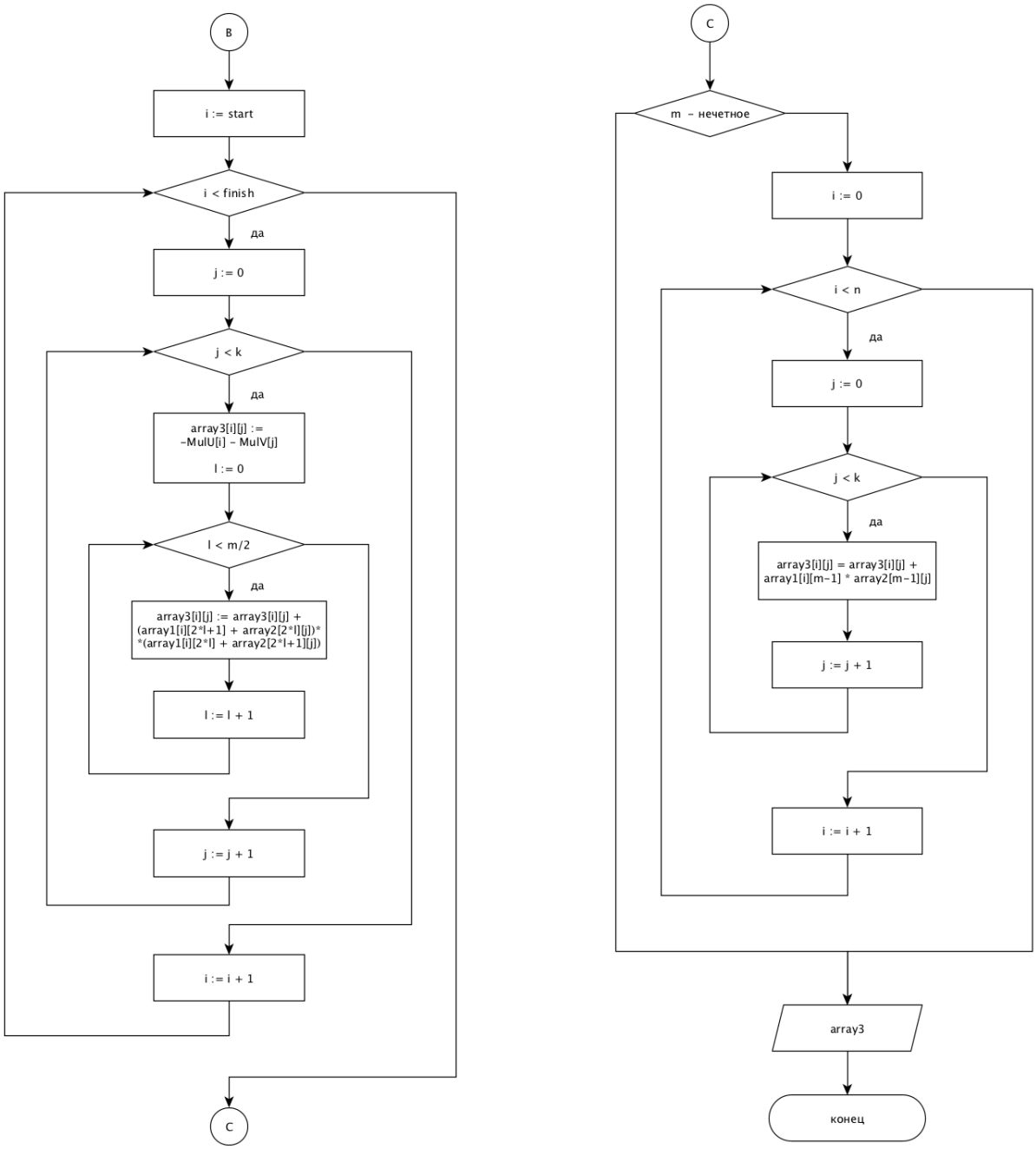

Рисунок 5 : Схема распараллеленного алгоритма Винограда умножения матриц (часть 2)

3 Технологическая часть

3.1 Требования к программному обеспечению

Программа должна умножать случайным образом сгенерированные матрицы целых чисел, размер вводится с клавиатуры. Результатом работы программы является матрица-произведение сгенерированных матриц и время работы алгоритма.

3.2 Средства реализации

Для реализации программы был выбран язык программирования C++. В рамках текущеи задачи данныи язык программирования имеет ряд существенных преимуществ:

— Библиотека с классом нативных потоков std::thread для распараллеливания вычислении;

— Статическая типизация;

— Близость к низкоуровневому C при наличии многих возможностеи высокоуровненных языков;

— Встроенная библиотека std::chrono, позволяющая измерять процессорное время.

3.3 Листинг кода

В данном разделе представлен распараллеленный алгоритм Винограда (листинг 1). Листинг 1

```
void alg_threads(Matrix &array3, Matrix& array1, Matrix& array2, int number_of_threads, int part)
{
    int n = array1.size();
    int m = array2.size();
    int k = array2[0].size();

    std::vector<int> MulU(n, 0);
    std::vector<int> MulV(k, 0);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m/2; j++) {
            MulU[i] += array1[i][2 * j] * array1[i][2 * j + 1];
        }
    }
    for (int j = 0; j < k; j++) {
        for (int i = 0; i < m/2; i++) {
            MulV[j] += array2[2 * i][j] * array2[2 * i + 1][j];
        }
    }
    for (int i = part * n / number_of_threads; i < (part + 1) * n / number_of_threads; i++) {
        for (int j = 0; j < k; j++) {
            array3[i][j] = - MulU[i] - MulV[j];
            for (int l = 0; l < m/2; l++) {
                array3[i][j] += (array1[i][2 * l] + array2[2 * l + 1][j]) * (array1[i][2 * l + 1] + a
            }
            if (n % 2 == 1) {
                array3[i][j] +=  array1[i][m - 1] * array2[m - 1][j];
            }
```

```
        }
    }
}

int alg(void)
{
    int size_matrix, number_of_threads;
    int part = 0;

    std::vector<std::thread> threads_list;

    chrono::high_resolution_clock::time_point t_start, t_end;
    std::srand(std::time(nullptr));

    Matrix array1{}, array2{}, array3{};
    std::cout << "Введите размер матрицы: ";
    std::cin >> size_matrix;
    std::cout << "Введите количество потоков: ";
    std::cin >> number_of_threads;



    fill_matrix_by_zero(array3, size_matrix);
    fill_matrix_by_random(array1, size_matrix);
    fill_matrix_by_random(array2, size_matrix);

    for (int i = 0; i < number_of_threads; i++)
        threads_list.push_back(std::thread(alg_threads, ref(array3), ref(array1), ref(array2), numbe

    t_start = chrono::high_resolution_clock::now();
    for (auto& thrd : threads_list)
        thrd.join();
    t_end = chrono::high_resolution_clock::now();



    std::cout << "Время: " << chrono::duration_cast<std::chrono::microseconds>(t_end-t_start).count()
    << std::endl;

    print_matrix("\nПервая матрица: ", array1, size_matrix);
    print_matrix("\nВторая матрица: ", array2, size_matrix);
    print_matrix("\nРезультат умножения: ", array3, size_matrix);
}
```

4 Экспериментальная часть

4.1 Пример работы

На рис. 6 приведен пример работы программы.

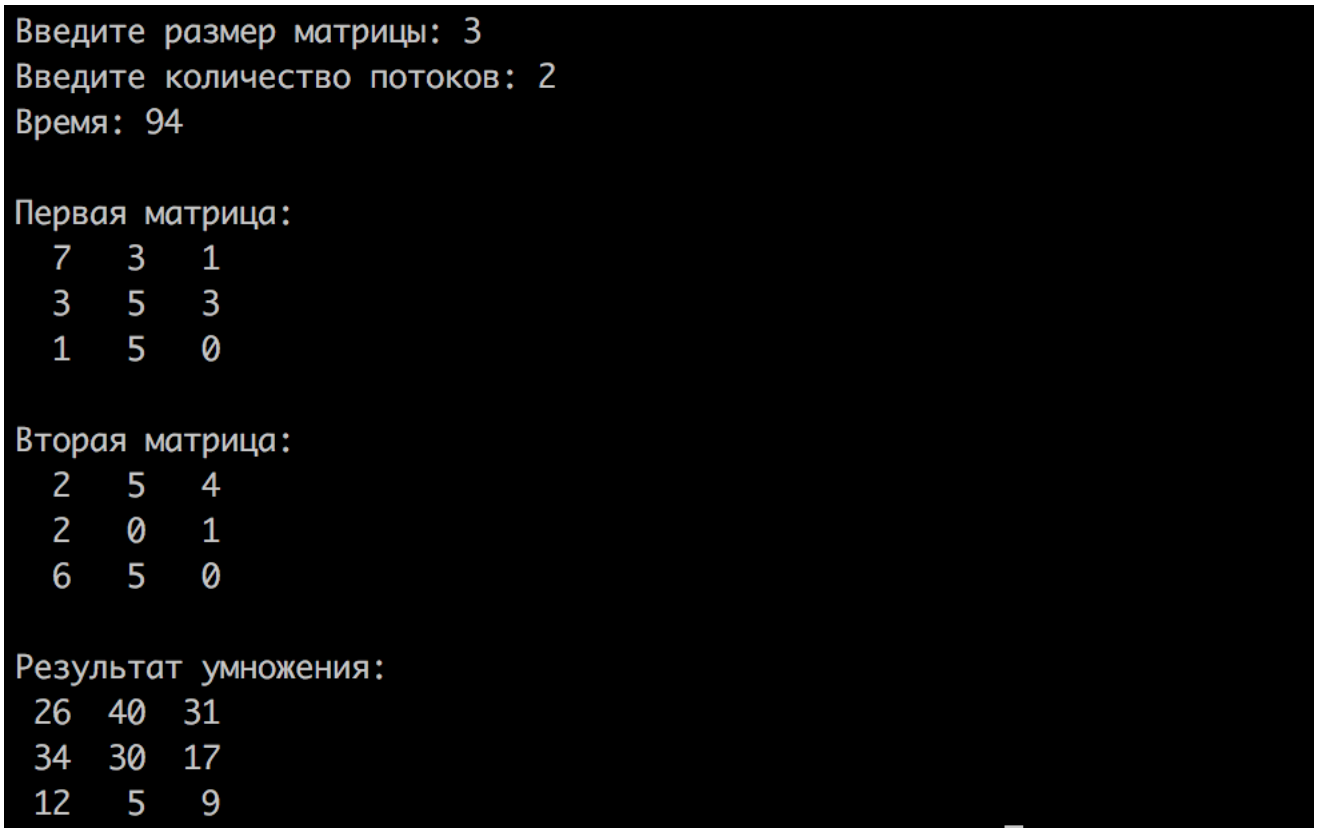

Рисунок 6 : Пример работы программы

4.2 Постановка эксперимента по замеру времени

На рис. 7, 8 представлены результаты замеров времени для основного цикла алгоритма Винограда на варьирующихся размерах квадратных матриц от 100 до 1000 и от 101 до 1001 с шагом 100. Один эксперимент ставился 100 раз, вычислялось среднее время работы. На рис.7 - график работы исследуемых алгоритмов в случае нечетного размера, рис. 8 в случае четного.

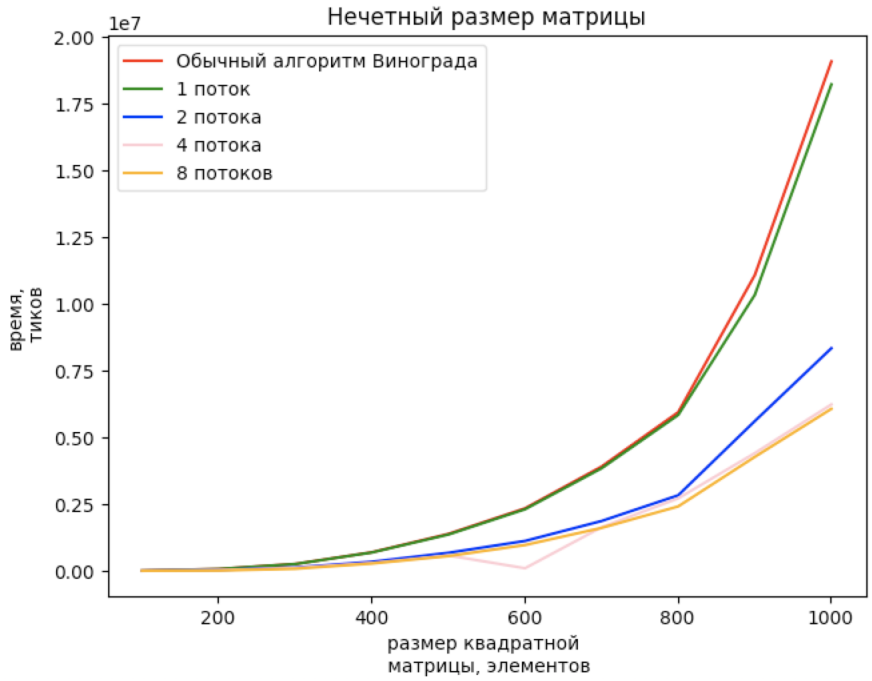

Рисунок 7 : График зависимости времени работы алгоритмов в случае нечетного размера матриц

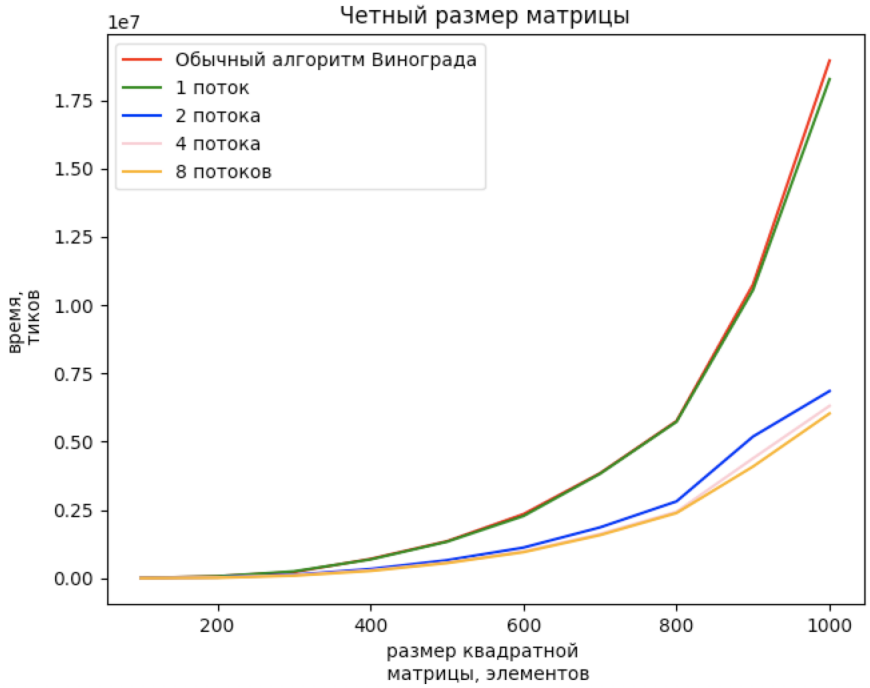

Рисунок 8 : График зависимости времени работы алгоритмов в случае четного размера матриц

Из графиков отношения размерности матрицы ко времени вычисления видно, что рассчет на одном потоке работает примерно в 2 раза медленнее вычислении на нескольких потоках. Среди распараллеленных вычислении видно, что увеличение числа потоков дает небольшои прирост к скорости примерно на 10%. Однако чем больше потоков используется, тем этот прирост меньше.

Заключение

В ходе выполнения работы было изучено распараллеливание вычислении и работа с потоками; реализованы распараллеленные вычисления; проведено экспериментальное сравнение работы алгоритма на разном количестве потоков(1, 2, 4 и 8). В результате были сделаны следующие выводы:

• распараллеленные вычисления эффективнеи в 2 раза;

• использование числа потоков больше, чем число потоков процессора не дает выигрыша по времени и может даже работать медленнее.

Список литературы

1. Kakaradov B. Ultra-Fast Matrix Multiplication: An Empirical Analysis of Highly Optimized Vector Algorithms [Электронныи ресурс] // cs.stanford.edu: [саит]. [2004]. URL: https://cs.stanford.edu/people/boyko/pu Mult_SURJ_2004.pdf

2. Stothers A.J. On the Complexity of Matrix [Электронныи ресурс] // era.lib.ed.ac.uk: [саит]. [2010]. URL: https://www.era.lib.ed.ac.uk/bitstream/handle/184 2/4734/Stothers2010.pdf

3. Williams V.V. Multiplying matrices [Электронныи ресурс] // http://theory.stanford.edu: [саит]. [2014]. URL: http://theory.stanford.edu/ virgi/matrixmult-f.pdf

4. Алгоритм Копперсмита — Винограда [Электронныи ресурс] // ru.math.wikia.com/: [саит]. URL: http://ru.math.wikia.com/wiki/Алгоритм_Копперсмита_—_Винограда (дата обращения: 4.05.2020).

5. Документация по chrono [Электронныи ресурс]. - Режим доступа http://www.cplusplus.com/reference/chrono/ Дата обращения: 4.05.2020

6. Документация по thread [Электронныи ресурс]. - Режим доступа https://ru.cppreference.com/w/cpp/thread/thr Дата обращения: 4.05.2020