Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе № 2
по курсу «Анализ алгоритмов»
на тему: «Умножение матриц»

Студент ИУ7-54Б (Группа) (Подпись, дата) Булдаков М. (И. О. Фамилия)

Преподаватель (Подпись, дата) Волкова Л. Л. (И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Матрицы являются одним из основных инструментов линейной алгебры, они позволяют описывать и анализировать линейные отношения между различными объектами и явлениями. В настоящее время матрицы широко используются в науке, технике, экономике и других сферах человеческой деятельности.

Размеры матриц могут быть очень большими в зависимости от конкретной задачи, поэтому оптимизация алгоритмов обработки матриц является важной задачей программирования. Основной акцент будет сделан на оптимизации алгоритма умножения матриц.

Цель данной лабораторной работы — описать, реализовать и исследовать алгоритмы умножения матриц и их оптимизации. Для достижения поставленной цели необходимо выполнить следующие задачи.

1) Описать алгоритмы умножения матриц:

    — классический алгоритм;

    — алгоритм Винограда;

    — алгоритм Штрассена.

2) Оптимизировать перечисленные алгоритмы.

3) Разработать программное обеспечение, реализующее алгоритмы умножения.

4) Выбрать инструменты для реализации и замера процессорного времени выполнения алгоритмов.

5) Проанализировать затраты реализаций алгоритмов по времени и по памяти.

1 Аналитический раздел

Матрицей называется прямоугольная таблица чисел, вида (1.1), состоящая из m строк и n столбцов [1].

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \tag{1.1}$$

Пусть A — матрица, тогда $_{ij}$ — элемент этой матрицы, который находится на *i-ой* строке и *j-ом* столбце.

Если количество столбцов первой матрицы совпадает с количеством строк второй матрицы, то возможно выполнить их матричное умножение. В результате умножения получится матрица-произведение, количество строк в которой равно количеству строк первой матрицы, а количество столбцов равно количеству столбцов второй матрицы.

1.1 Классический алгоритм

Пусть даны две прямоугольные матрицы A и B размеров $[m \times n]$ и $[n \times k]$ соответственно. В результате произведение матриц A и B получим матрицу C размера $[m \times k]$, элементы которой вычисляются по (1.2).

$$c_{ij} = \sum_{l=1}^{n} a_{il} b_{lj} \tag{1.2}$$

Классический алгоритм умножения матриц, реализует формулу (1.2).

1.2 Алгоритм Винограда

Алгоритм Винограда — алгоритм умножения квадратных матриц с асимптотической сложностью $O(n^{2,373})$ [2].

Рассмотрим два вектора $V = (v1, v2, v3, v4)$ и $W = (w1, w2, w3, w4)$.

Их скалярное произведение равно (1.3).

$$V \cdot W = v_1 \cdot w_1 + v_2 \cdot w_2 + v_3 \cdot w_3 + v_4 \cdot w_4 \tag{1.3}$$

Равенство (1.3) можно переписать в виде (1.4)

$$V \cdot W = (v_1 + w_2) \cdot (v_2 + w_1) + (v_3 + w_4) \cdot (v_4 + w_3) - v_1 \cdot v_2 - v_3 \cdot v_4 - w_1 \cdot w_2 - w_3 \cdot w_4 \quad (1.4)$$

Теперь допустим, что у нас есть две матрицы A и B размерности $m \times n$ и $n \times p$ соответственно, и мы хотим найти их произведение $C = A \cdot B$. Тогда алгоритм будет состоять из следующих шагов:

1) Подготовительные вычисления. Сначала создаются два вспомогательных массива (1.5) и (1.6).

$$\text{rowFactor}[i] = \sum_{j=0}^{n/2-1} A[i][2j+1] \cdot A[i][2j] \quad (1.5)$$

для $0 \leq i < m$

$$\text{colFactor}[j] = \sum_{i=0}^{n/2-1} B[2i+1][j] \cdot B[2i][j] \quad (1.6)$$

для $0 \leq j < p$.

2) Умножение матриц. Вычисляем результирующую матрицу C по формуле (1.7).

$$\begin{aligned} C[i][j] = \sum_{k=0}^{n/2-1} &(A[i][2k+1] + B[2k][j]) \cdot (A[i][2k] + B[2k+1][j]) \\ &- \text{rowFactor}[i] - \text{colFactor}[j] \end{aligned} \quad (1.7)$$

для $0 \leq i < m$ и $0 \leq j < p$.

3) Коррекция. Если n нечетно, добавляем коррекцию, в соответствии с формулой (1.8).

$$C[i][j] += A[i][n] \cdot B[n][j] \quad (1.8)$$

для $0 \leq i < m$ и $0 \leq j < p$.

1.3 Алгоритм Штрассена

Алгоритм Штрассена — это алгоритм умножения квадратных матриц, который является более эффективным для больших матриц, чем классический метод умножения [3].

Если добавить к матрицам A и B одинаковые нулевые строки и столбцы, их произведение станет равно матрице C с теми же добавленными строками и столбцами. Поэтому в данном алгоритме рассматриваются матрицы порядка $m \cdot 2^{k+1}$, где $k \in \mathbb{N}$, а все остальные матрицы, сводятся к этому размеру добавлением нулевых строк и столбцов.

Алгоритм состоит из следующих шагов.

1) Разбиение матриц (1.9).

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \tag{1.9}$$

, где A_{ij} и B_{ij} – матрицы порядка $m \cdot 2^k$

2) Вычисление вспомогательных матриц (1.10).

$$\begin{aligned} M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\ M_2 &= (A_{21} + A_{22})B_{11} \\ M_3 &= A_{11}(B_{12} - B_{22}) \\ M_4 &= A_{22}(B_{21} - B_{11}) \\ M_5 &= (A_{11} + A_{12})B_{22} \\ M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\ M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned} \tag{1.10}$$

3) Вычисление результирующих подматриц (1.11).

$$\begin{aligned} C_{11} &= M_1 + M_4 - M_5 + M_7 \\ C_{12} &= M_3 + M_5 \\ C_{21} &= M_2 + M_4 \\ C_{22} &= M_1 - M_2 + M_3 + M_6 \end{aligned} \tag{1.11}$$

Результирующая матрица состоит из C_{ij} (1.12).

$$AB = C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \tag{1.12}$$

Вывод

В данном разделе были описаны алгоритмы классической матричной мультипликации, алгоритм Штрассена и алгоритм Винограда.

2 Конструкторский раздел

В этом разделе будет представлено описание используемых типов данных, а также схематические изображения алгоритмов матричного умножения: стандартного, Штрассена, алгоритма Винограда и оптимизированного алгоритма Винограда.

2.1 Требования к программному обеспечению

Программа должна поддерживать два режима работы: режим массового замера времени и режим умножения матриц.

Режим массового замера времени должен обладать следующим функционалом:

— генерировать матрицы различного размер для проведения замеров;

— осуществлять массовый замер, используя сгенерированные данные;

— результаты массового замера должны быть представлены в виде таблицы и графика.

К режиму умножения матриц выдвигается ряд требований:

— возможность работать с матрицами разного размера, которые вводит пользователь;

— наличие интерфейса для выбора действий;

— проверять возможность умножения матриц.

2.2 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие структуры и типы данных:

— количество строк — целое число;

— количество столбцов — целое число;

— матрица — двумерный список вещественных чисел.

2.3 Разработка алгоритмов

На рисунке 2.1 представлена схема классического алгоритма, выполняющего умножение двух матриц. На рисунках 2.2 – 2.4 изображены схемы алгоритма Винограда без оптимизаций. На рисунках 2.5 и 2.6 изображены схемы алгоритмов умножения матриц Штрассена. На рисунках 2.7 – 2.9 изображены схемы алгоритма Винограда с оптимизациями.

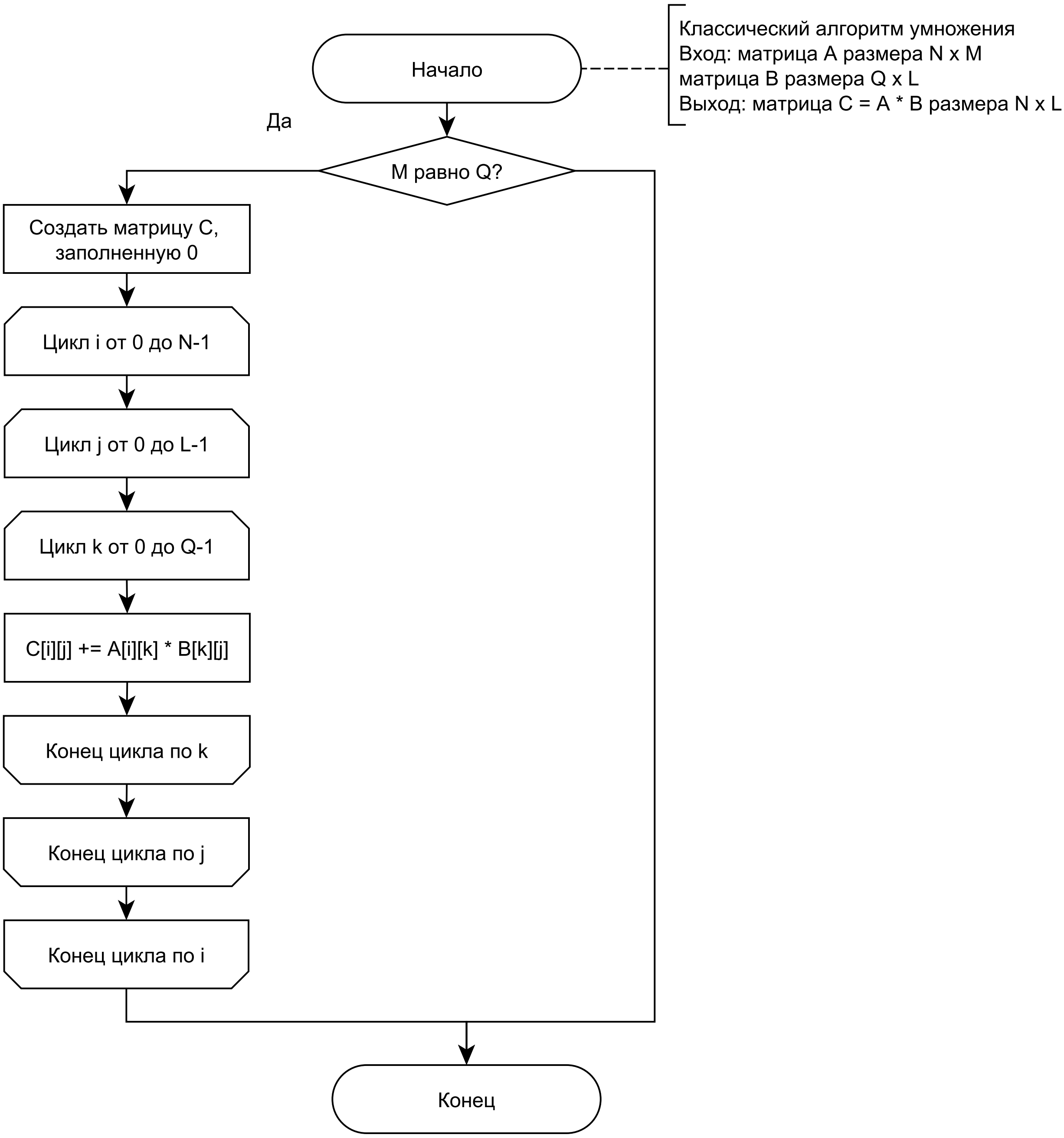

Рисунок 2.1 – Классический алгоритм умножения матриц

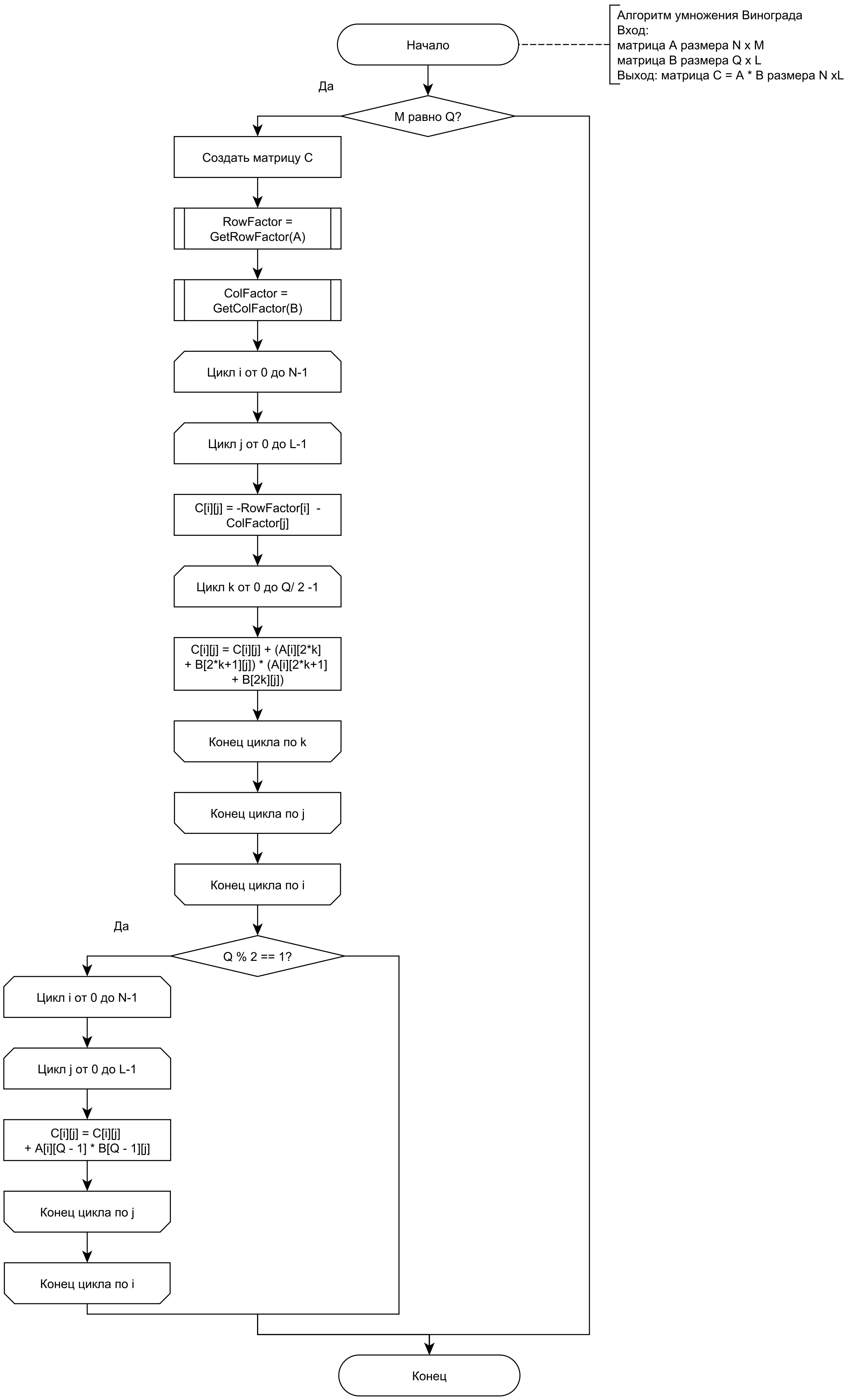

Рисунок 2.2 – Алгоритм умножения матриц Винограда

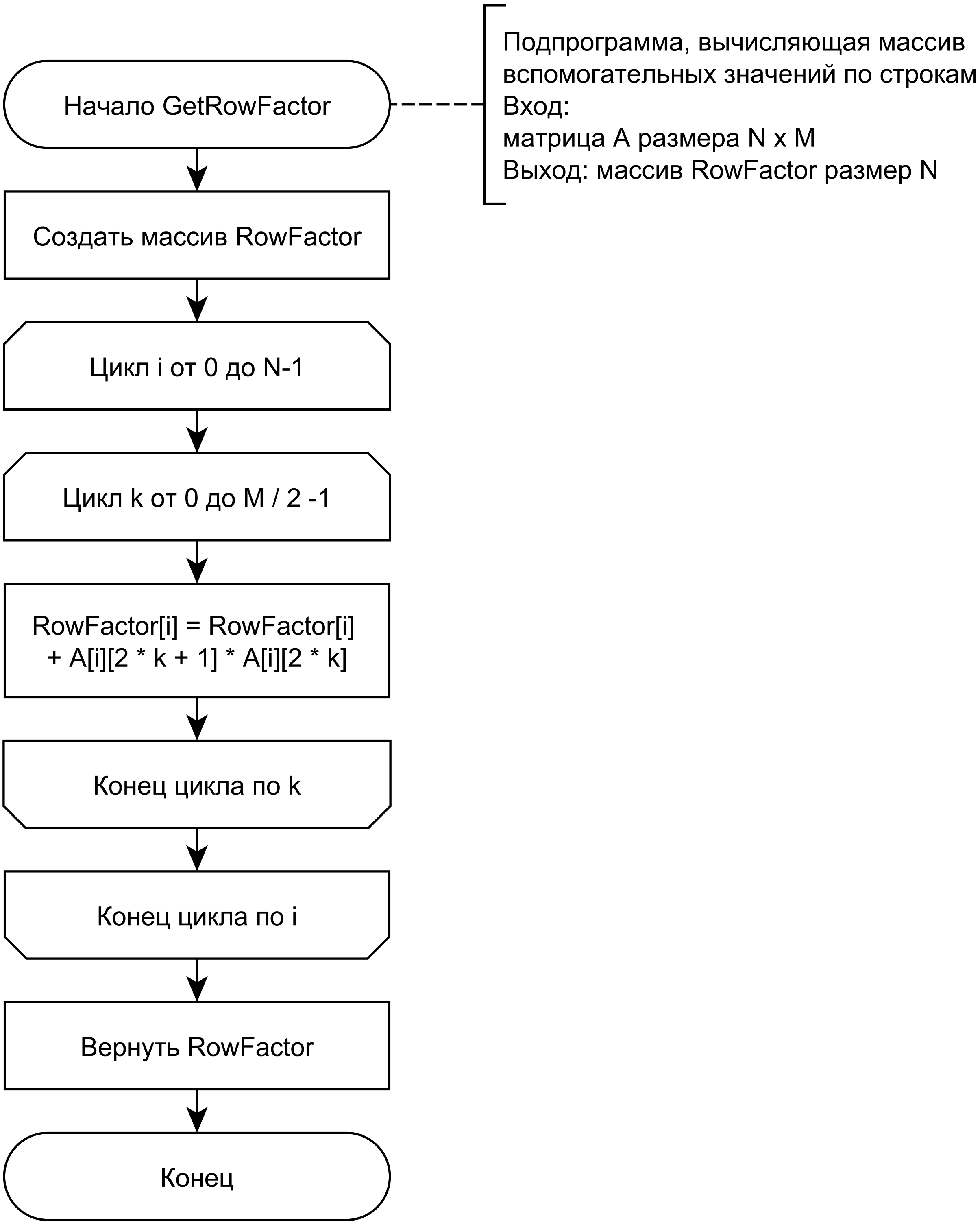

Рисунок 2.3 – Вспомогательная подпрограмма, вычисляющая массив вспомогательных значений по строкам

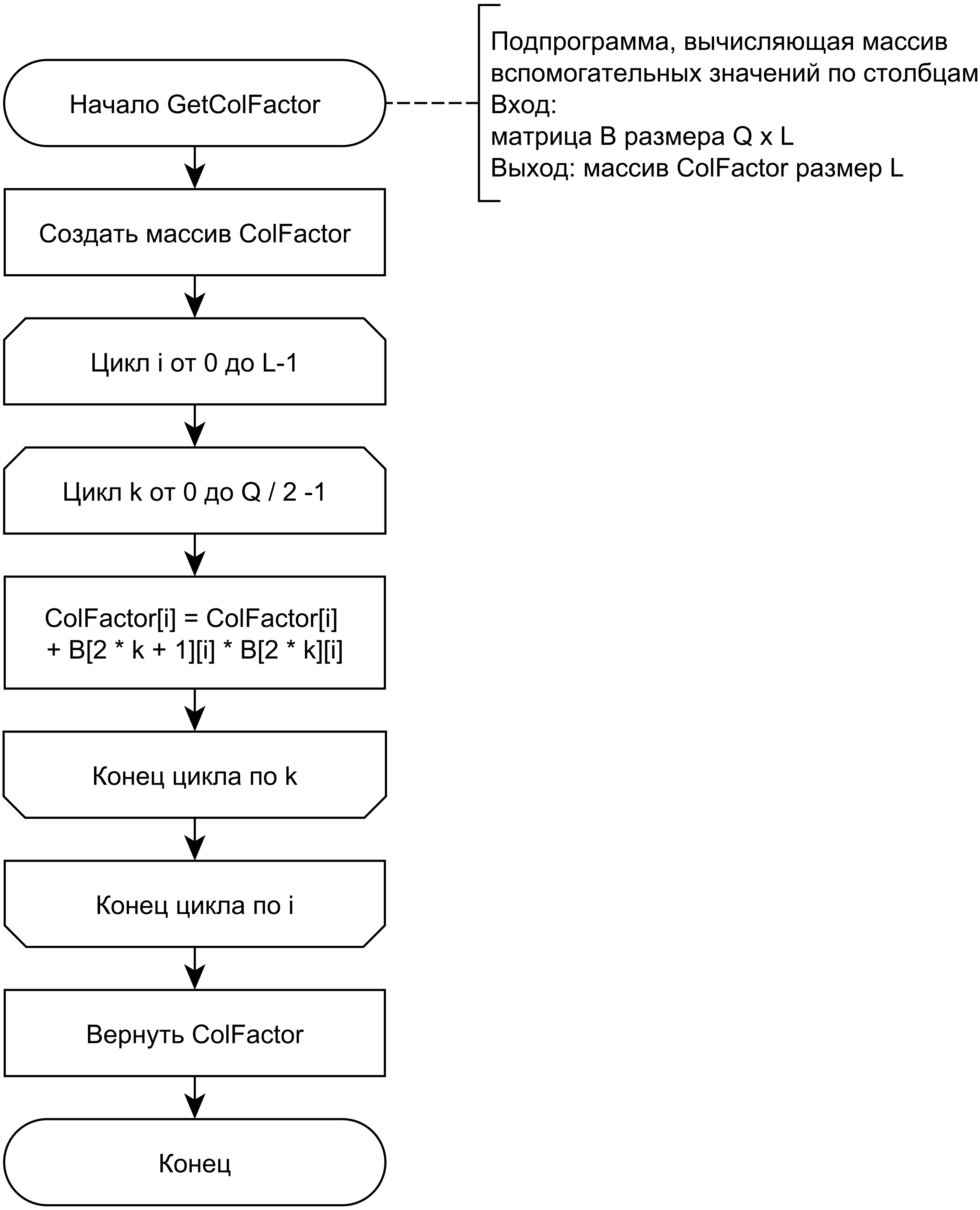

Рисунок 2.4 – Вспомогательная подпрограмма, вычисляющая массив вспомогательных значений по столбцам

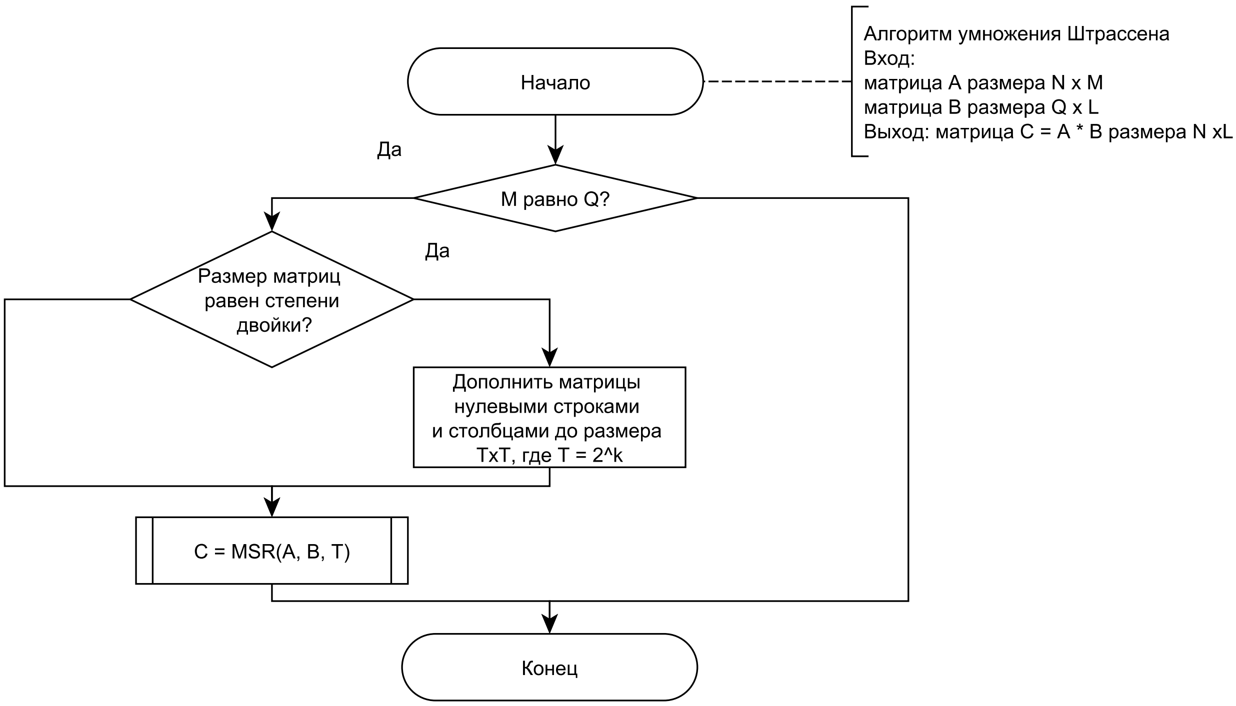

Рисунок 2.5 – Алгоритм умножения матриц Штрассена

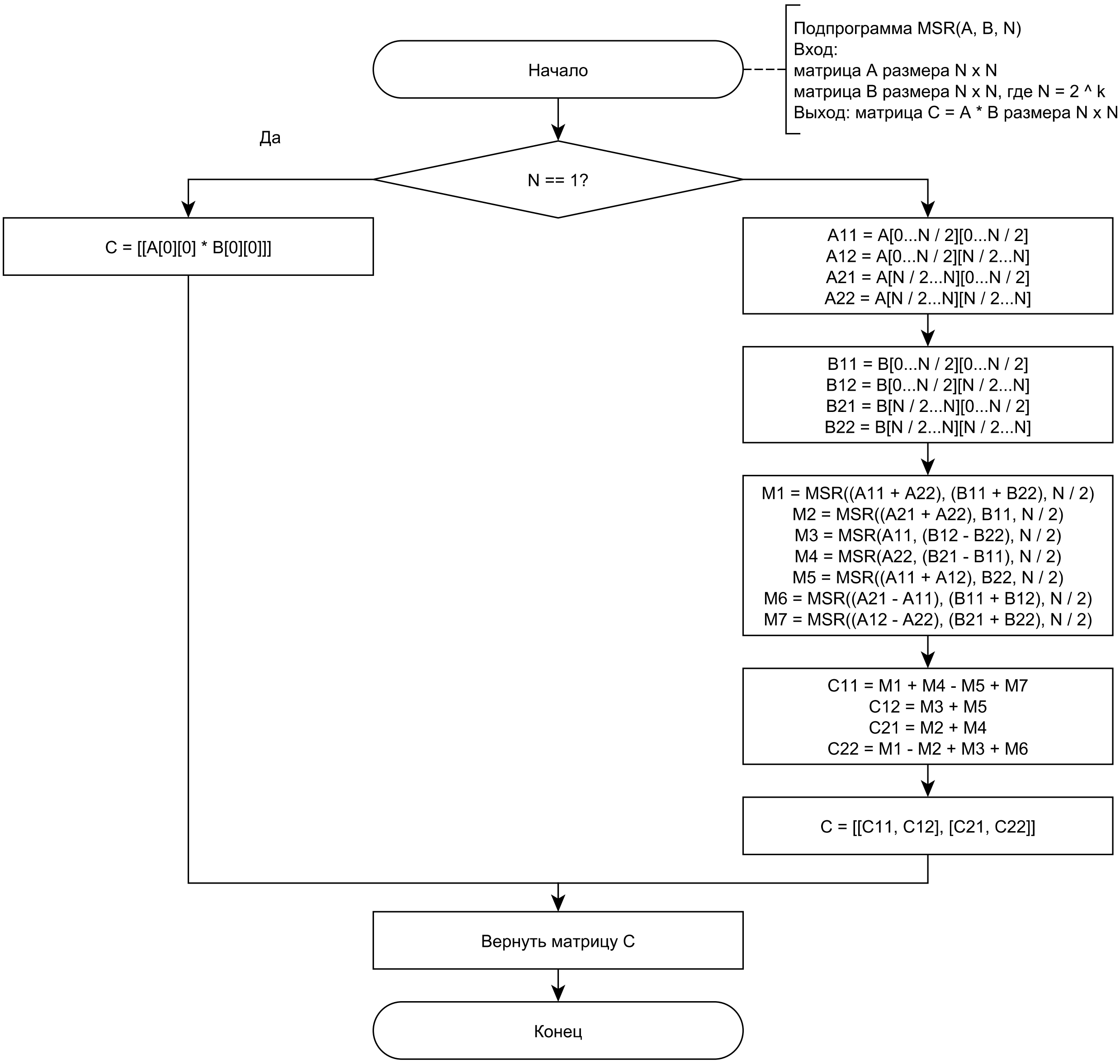

Рисунок 2.6 – Подпрограмма MSR, вычисляющая результат умножения матриц по алгоритму Штрассена

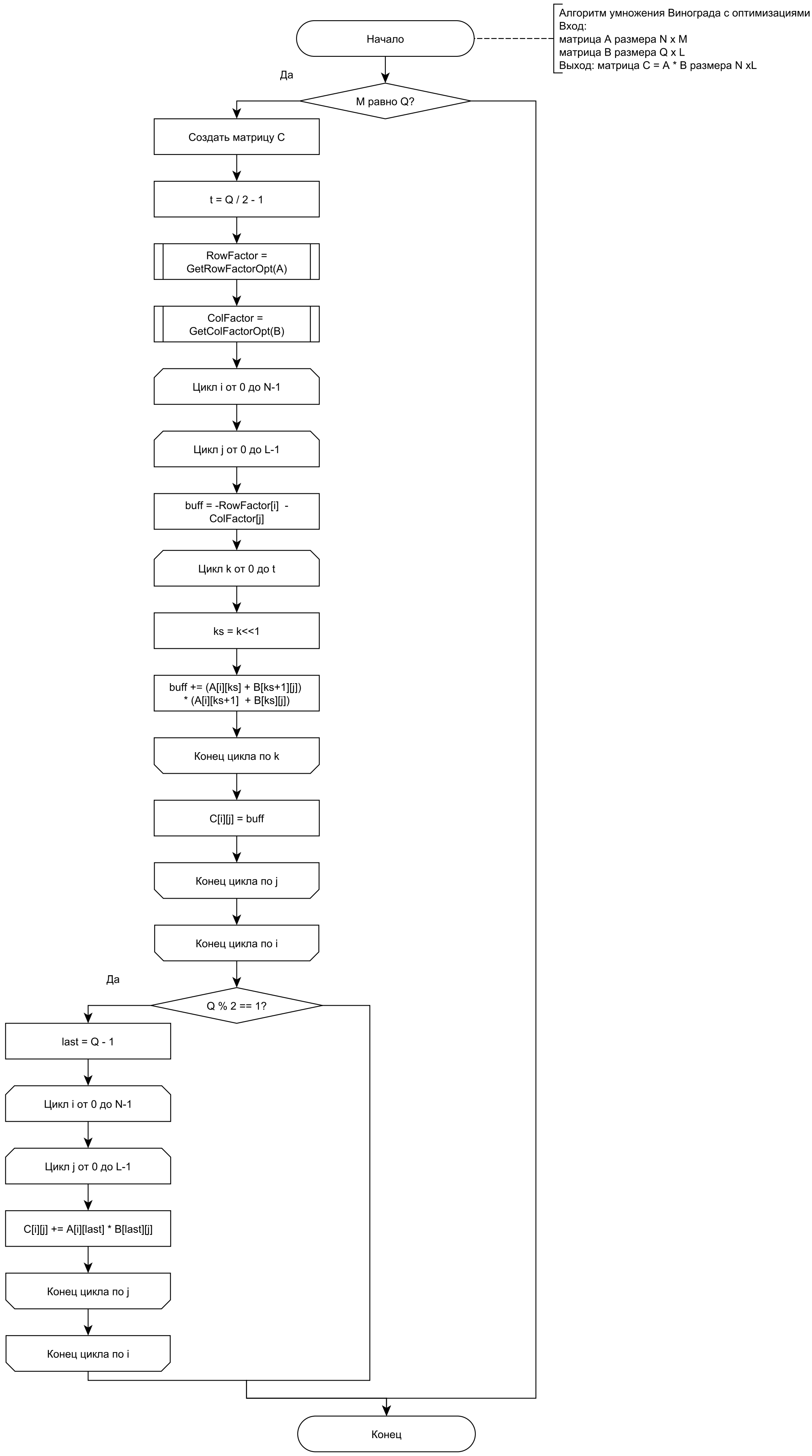

Рисунок 2.7 – Алгоритм умножения матриц Винограда

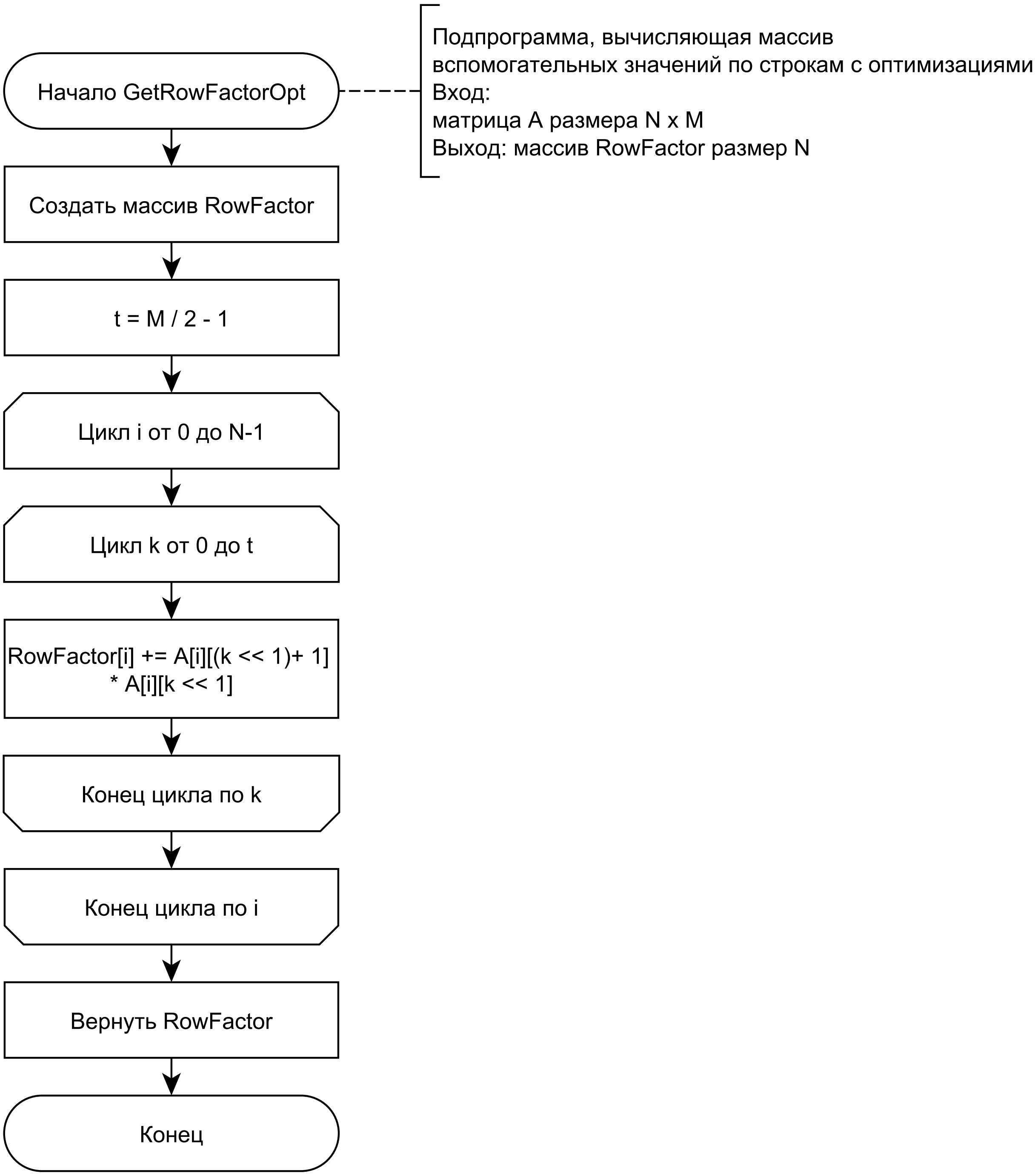

Рисунок 2.8 – Вспомогательная подпрограмма, вычисляющая массив вспомогательных значений по строкам

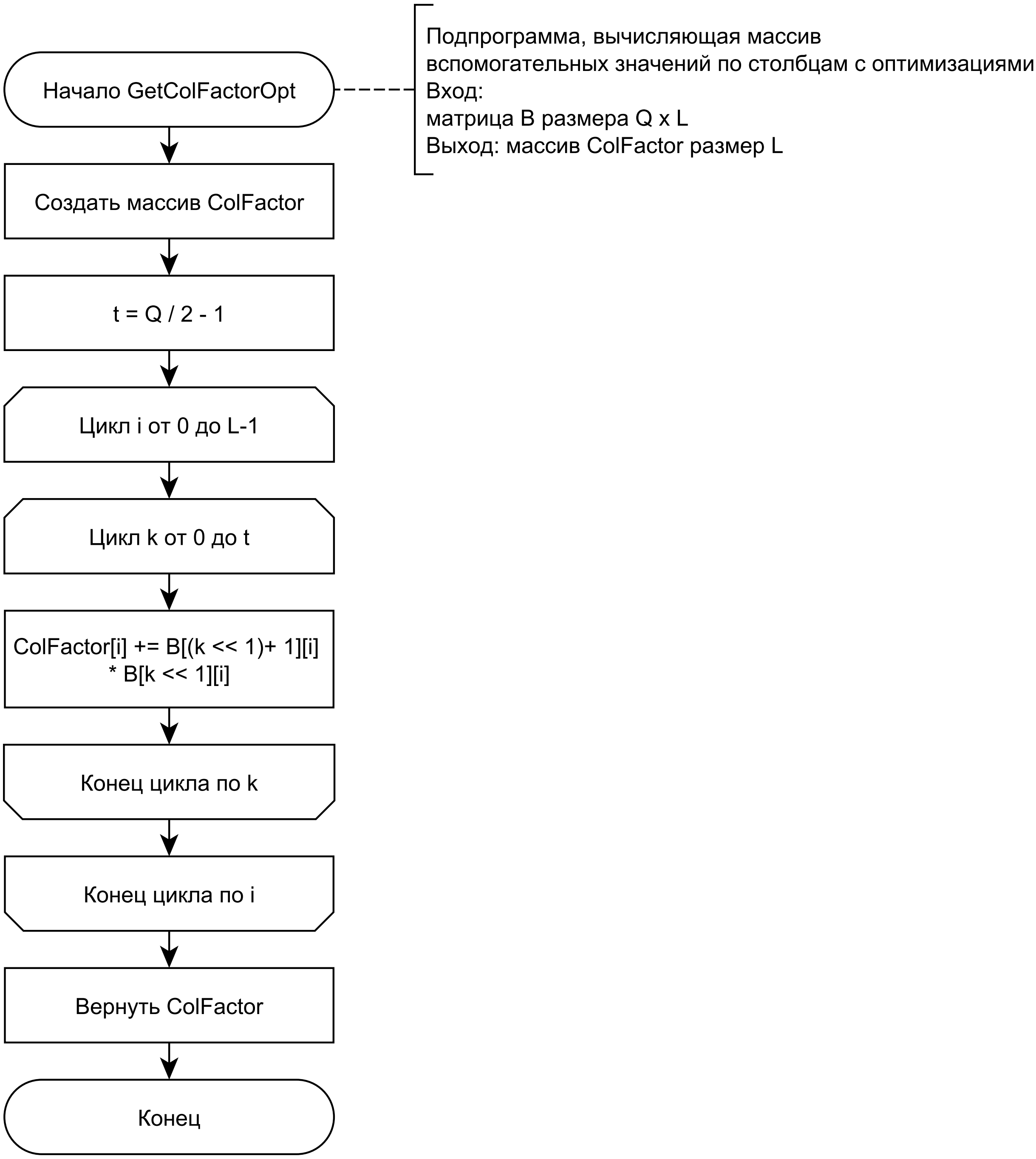

Рисунок 2.9 – Вспомогательная подпрограмма, вычисляющая массив вспомогательных значений по столбцам

2.4 Оценка трудоемкости алгоритмов

Введем модель для оценки трудоемкости алгоритмов.

1) $+, -, =, +=, -=, ==, ||, \&\&, <, >, <=, >=, <<, >>, []$ — считаем, что эти операции обладают трудоемкостью в 1 единицу.

2) $*, /, *=, /=, \%$ — считаем, что эти операции обладают трудоемкостью в 2 единицы.

3) Трудоемкость условного перехода примем за 0.

4) Трудоемкость условного оператора рассчитывается по формуле (2.1).

$$f_{if} = f_{\text{условия}} + \begin{cases} min(f_1, f_2), & \text{лучший случай} \\ max(f_1, f_2), & \text{худший случай} \end{cases} \tag{2.1}$$

5) Трудоемкость цикла рассчитывается по формуле (2.2).

$$f_{for} = f_{\text{инициализация}} + f_{\text{сравнения}} + M_{\text{итераций}} \cdot (f_{\text{тело}} + \\ + f_{\text{инкремент}} + f_{\text{сравнения}}) \tag{2.2}$$

6) Вызов подпрограмм и передачу параметров примем за 0.

2.4.1 Трудоемкость классического алгоритма

Трудоемкость классического алгоритма складывается из следующих слагаемых.

1) Внешний цикл по $i \in [0 \ldots N)$, трудоёмкость которого: $f_i = 2+N\cdot(2+f_j)$.

2) Цикла по $j \in [0 \ldots L)$, трудоёмкость которого: $f_j = 2 + L \cdot (2 + f_k)$.

3) Цикла по $k \in [0 \ldots Q)$, трудоёмкость которого: $f_k = 2 + Q \cdot (2 + 2 + 1 + 2 + 2 + 2) = 2 + 11Q$.

Тогда в итоге, трудоемкость классического алгоритма рассчитывается по формуле (2.3).

$$f_{classic} = 2 + N \cdot (2 + 2 + L \cdot (2 + 2 + 11Q)) = 11NLQ + 4NL + 4N + 2 \quad (2.3)$$

2.4.2 Трудоемкость алгоритма Винограда

Трудоемкость алгоритма Винограда формируется из следующих частей.

1) Трудоемкости вычисления массива $RowFactor$, которая рассчитывается по формуле (2.4).

$$f_r = 2 + N \cdot (2 + 4 + \frac{M}{2} \cdot (3 + 1 + 15)) = \frac{19}{2}NM + 6N + 2 \qquad (2.4)$$

2) Трудоемкости вычисления массива $ColFactor$, которая рассчитывается, аналогично пункту 1, по формуле (2.5).

$$f_c = \frac{19}{2}LQ + 6L + 2 \qquad (2.5)$$

3) Трудоемкости основной части алгоритма, которая вычисляется по формуле (2.6).

$$\begin{gathered} f_{main} = 2 + N \cdot (2 + 2 + L \cdot (2 + 7 + 4 + \frac{Q}{2} \cdot (4 + 28))) = \\ = \frac{32}{2}NLQ + 13NL + 4N + 2 \end{gathered} \qquad (2.6)$$

4) Трудоемкости поправки, вносимой для нечетного Q, рассчитываемой по формуле (2.7).

$$f_{err} = 3 + \begin{cases} 0, & \text{лучший случай} \\ 16LN + 4N + 2, & \text{худший случай} \end{cases} \qquad (2.7)$$

В итоге, трудоемкость алгоритма Винограда, вычисляется по формуле (2.8).

$$\begin{gathered}f_{vinograd} = f_r + f_c + f_{main} + f_{err} = \frac{32}{2}NLQ + \frac{19}{2}LQ + \frac{19}{2}NM + \\ +13NL + 10N + 6L + 9 + \begin{cases} 0, & \text{л. с.} \\ 16LN + 4N + 2, & \text{х. с.} \end{cases}\end{gathered} \tag{2.8}$$

2.4.3 Трудоемкость оптимизированного алгоритма Винограда

Произведены следующие оптимизации алгоритма Винограда:

— заменим операцию $x = x + k$ на $x += k$;

— заменим умножение на 2 на побитовый сдвиг;

— будем предвычислять некоторые слагаемые для алгоритма.

Трудоемкость оптимизированного алгоритма Винограда формируется из следующих частей.

1) Трудоемкости вычисления массива $RowFactor$, которая рассчитывается по формуле (2.9).

$$f_r = 3 + 2 + N \cdot (2 + 2 + \frac{M}{2} \cdot (2 + 11)) = \frac{13}{2}NM + 4N + 5 \tag{2.9}$$

2) Трудоемкости вычисления массива $ColFactor$, которая рассчитывается, аналогично пункту 1, по формуле (2.10).

$$f_c = \frac{13}{2}LQ + 4L + 5 \tag{2.10}$$

3) Трудоемкости основной части алгоритма, которая вычисляется по формуле (2.11).

$$f_{main} = 3 + 2 + N \cdot (2 + 2 + 3 + L \cdot (2 + 5 + 2 + \frac{Q}{2} \cdot (2 + 17))) = \\ = \frac{21}{2}NLQ + 9NL + 7N + 5 \quad (2.11)$$

4) Трудоемкости поправки, вносимой для нечетного Q, рассчитываемой по формуле (2.12).

$$f_{err} = 3 + \begin{cases} 0, & \text{лучший случай} \\ 11LN + 4N + 4, & \text{худший случай} \end{cases} \quad (2.12)$$

В итоге, трудоемкость оптимизированного алгоритма Винограда, вычисляется по формуле (2.13).

$$f_{vinograd} = f_r + f_c + f_{main} + f_{err} = \frac{21}{2}NLQ + \frac{13}{2}LQ + \frac{13}{2}NM + \\ +9NL + 11N + 4L + 15 + \begin{cases} 0, & \text{л. с.} \\ 11LN + 4N + 4, & \text{х. с.} \end{cases} \quad (2.13)$$

2.4.4 Трудоемкость алгоритма Штрассена

Рассчитаем трудоемкость подпрограммы MSR, изображенной на рисунке 2.6. Будем считать затраты на расширение размера матрицы незначительными. Также будем считать, что стоимость срезов равна 1.

Трудоемкость сложения или вычитания двух матриц размера $N \times N$, рассчитывается по формуле (2.14).

$$f_{sum} = 2 + N \cdot (2 + 2 + N \cdot (2 + 8)) = 10N^2 + 4N + 2 \quad (2.14)$$

Тогда трудоемкость описывается рекуррентной формулой (2.15).

$$T(n) = \begin{cases} 7, & n == 1 \\ 7T(\frac{n}{2}) + 18f_{sum} + 65, & n > 1 \end{cases} \quad (2.15)$$

Тогда по формуле (2.16) возможно рассчитать трудоемкость для матриц порядка n.

$$
\begin{aligned}
T(n) = 7^{\log_2 n} T(1) + \sum_{i=0}^{(\log_2 n)-1} (7^i (18 f_{sum} + 65)) = \\
= 7^{\log_2 n} T(1) + \frac{(7^{\log_2 n} - 1)(18 f_{sum} + 65)}{6}
\end{aligned}
\tag{2.16}
$$

Вывод

На основе теоретических данных, полученных из аналитического раздела были построены схемы требуемых алгоритмов. Была введена модель оценки трудоемкости алгоритма, были рассчитаны трудоемкости алгоритмов в соответствии с этой моделью.

3 Технологический раздел

4 Исследовательский раздел

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Конспект лекций, читаемых в курсе «Высшая математика» Южного федерального университета. — Режим доступа: `http://mmtb.uginfo.sfedu.ru/algebra/Print/print_I-1.pdf` (дата обращения: 12.10.2023).
2. *Головашкин Д. Л.* Векторные алгоритмы вычислительной линейной алгебры: учеб. пособие. //. — — Самара: Изд-во Самарского университета, 2019. — С. 28—35.
3. Gaussian Elimination is not Optimal. — Режим доступа: `https://www2.math.uconn.edu/~leykekhman/courses/MATH_5510/fa_2016/papers/fast_matrix.pdf` (дата обращения: 13.10.2023).