# Оглавление

# ВВЕДЕНИЕ

Матричные структуры являются двумерными массивами числовых данных, организованными в табличном формате.

**Цель данной работы** — исследование и анализ различных подходов к умножению матричных структур с использованием следующих методик:

— стандартного метода умножения;

— алгоритма Винограда;

— улучшенной версии алгоритма Винограда.

Для реализации поставленных целей требуется выполнение следующих этапов:

— программная реализация рассматриваемых алгоритмов;

— проведение анализа вычислительной сложности разработанных методов;

— сравнительный анализ временных характеристик алгоритмов;

— интерпретация и обоснование полученных результатов.

# 1 Аналитическая часть

В представленном разделе рассматриваются различные подходы к реализации операций умножения матричных структур.

## 1.1 Характеристика алгоритмических решений

Стандартный метод умножения матриц основан на математическом определении данной операции и характеризуется вычислительной сложностью $O(n^3)$.

Метод Винограда, обладающий асимптотической сложностью $O(n^{2.3755})$, представляет собой один из наиболее производительных подходов к умножению матричных данных с позиции временных затрат.

Пусть даны матрицы $A$ с размерами $N \times M$ и $B$ с размерами $M \times K$. В результате умножения матрицы $A$ на матрицу $B$ получается матрица $C$ с размером $N \times K$.

### 1.1.1 Стандартный метод умножения матричных структур

Пусть даны матрицы $A$ размерностью $N \times M$ и матрица $B$ размерностью $M \times k$:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1k} \\ b_{21} & b_{22} & \dots & b_{2k} \\ \dots & \dots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mk} \end{pmatrix}. \tag{1.1}$$

Тогда умножением матрицы $A$ на матрицу $B$ называется, где матрица $C$:

$$C = A \times B = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1k} \\ c_{21} & c_{22} & \dots & c_{2k} \\ \dots & \dots & \ddots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nk} \end{pmatrix}, \tag{1.2}$$

где

$$c_{ij} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj} \quad (i = \overline{1,n},\ j = \overline{1,k}). \tag{1.3}$$

### 1.1.2 Метод Винограда

Пусть даны матрицы $A$ и $B$, имеющие размерность $4 \times 4$.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix}. \tag{1.4}$$

Для вычисления элемента $c_{ij}$ результирующей матрицы $C$ в стандартном методе умножения матричных структур применяется следующая формула:

$$c_{ij} = \begin{pmatrix} a_{n1} & a_{n2} & a_{n3} & a_{n4} \end{pmatrix} \times \begin{pmatrix} b_{j1} \\ b_{j2} \\ b_{j3} \\ b_{j4} \end{pmatrix}, \tag{1.5}$$

где $a_{ni},\ i = \overline{1,4}$ - элементы $n$-ой строки матрицы $A$; $b_{jk},\ k = \overline{1,4}$ - элементы $j$-ого столбца матрицы $B$.

В методе Винограда для повышения эффективности вычислений сокращается число трудоёмких операций умножения, замещая их операциями сложения. С этой целью выполняется предварительная обработка данных, в ходе которой сохраняются промежуточные значения, что даёт возможность впоследствии заменить определённую долю умножений операциями сложения:

$$c_{ij} = (a_{n1}+b_{j2})(a_{n2}+b_{j1})+(a_{n3}+b_{j4})(a_{n4}+b_{j3})-a_{n1}a_{n2}-a_{n3}a_{n4}-b_{j1}b_{j2}-b_{j3}b_{j4}, \tag{1.6}$$

где величины $a_{n1}a_{n2},\ a_{n3}a_{n4},\ b_{j1}b_{j2},\ b_{j3}b_{j4}$ представляют собой значения, получаемые в процессе предварительной обработки.

## ЗАКЛЮЧЕНИЕ

В представленном разделе рассмотрены стандартный метод и метод Винограда для умножения матричных структур. Основные отличия между ними состоят в предварительной обработке исходных данных и объёме выполняемых операций умножения.

# 2 Конструкторская часть

В представленном разделе приводятся схемы алгоритмических решений для умножения матричных структур, а также выполняется оценка вычислительной сложности каждого из рассматриваемых методов.

## 2.1 Описание алгоритмических решений

На входные данные алгоритмов поступают две матричные структуры: $M_1$ размерности $M \times N$, $M_2$ размерности $N \times K$, где $M, N, K$ представляют собой неотрицательные целочисленные значения; результатом является матрица $M_3$ размерности $M \times K$.

В данном разделе рассматриваются схемы трёх алгоритмических подходов к умножению матричных структур: стандартного, Винограда, улучшенного варианта Винограда.

### 2.1.1 Схема стандартного алгоритма умножения матриц

На рисунке 2.1 представлена схема стандартного алгоритма умножения матриц.

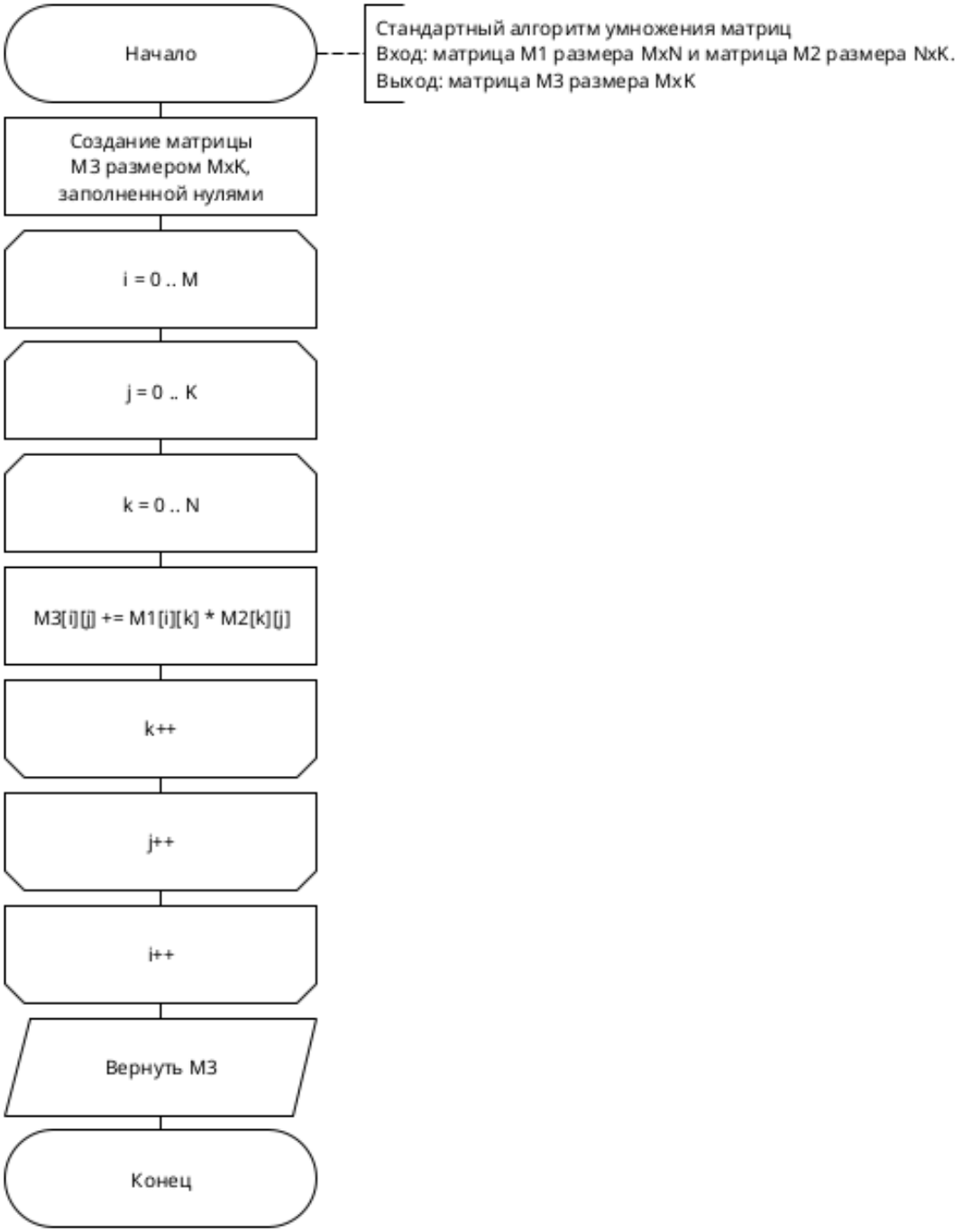


Рисунок 2.1 – Схема стандартного алгоритма умножения матриц

### 2.1.2 Схема алгоритма Винограда

На рисунке 2.2 представлена схема алгоритма Винограда для умножения матриц.

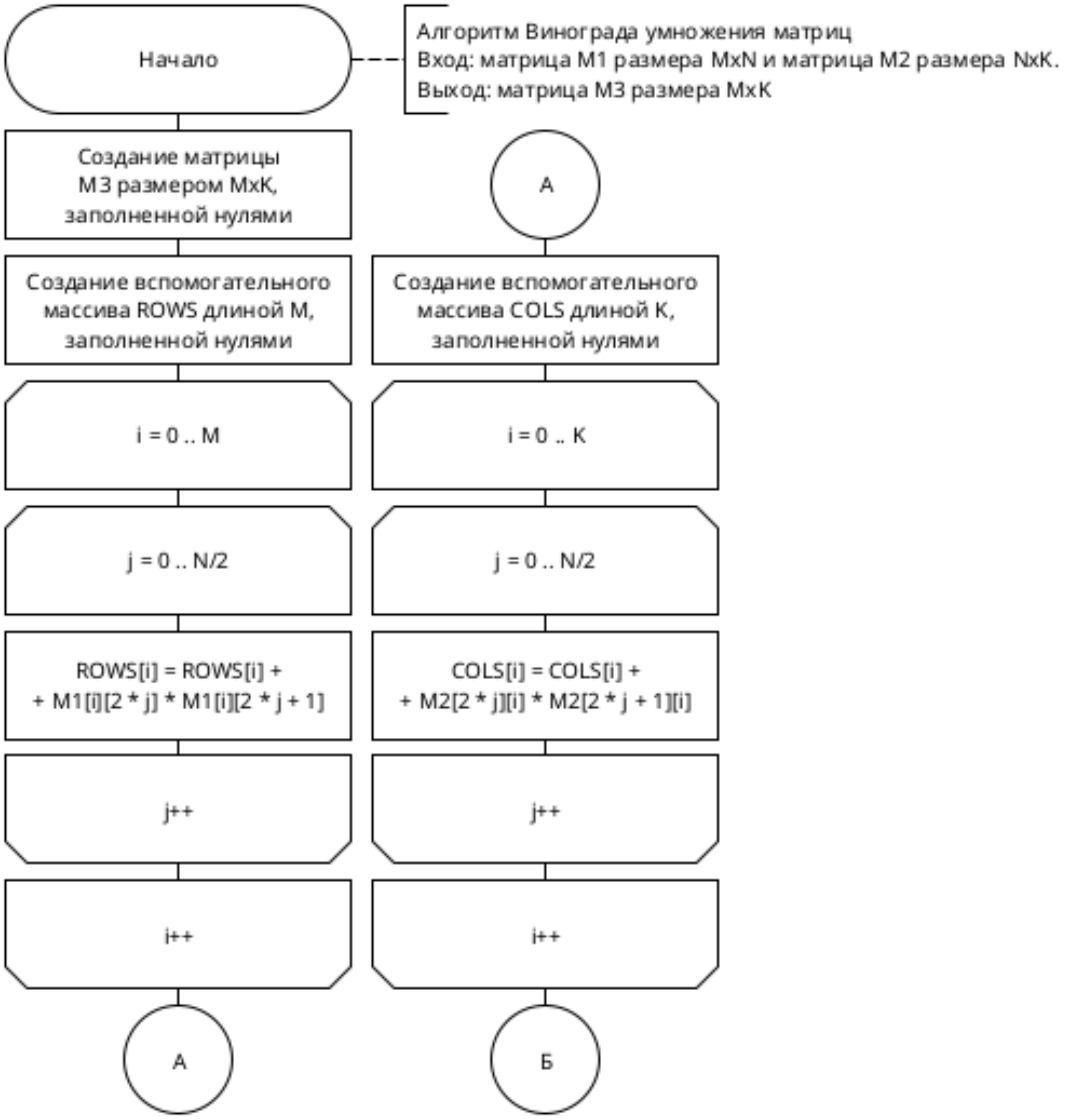


Рисунок 2.2 – Схема алгоритма Винограда умножения матриц

### 2.1.3 Схема оптимизированного алгоритма Винограда

На рисунке 2.3 представлена схема оптимизированного алгоритма Винограда для умножения матриц.

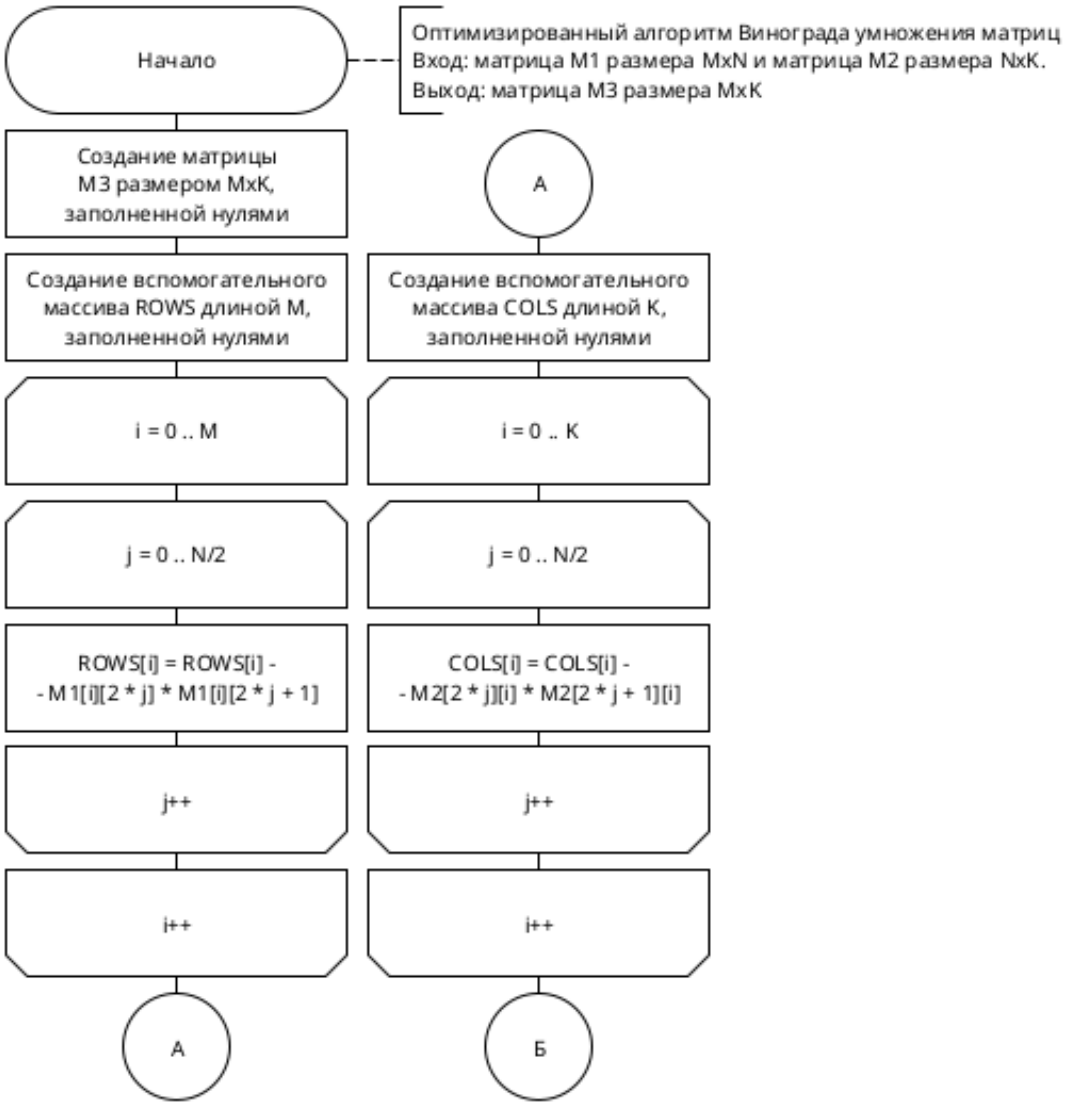


Рисунок 2.3 – Схема оптимизированного алгоритма Винограда умножения матриц

Улучшение метода Винограда включает в себя:

— увеличение шага наиболее вложенного счётчика цикла на 2;

— объединение III и IV этапов алгоритмического решения;

— введение операции декремента при вычислении вспомогательных массивов.

## 2.2 Вычислительная модель

Для последующего анализа вычислительной сложности необходимо определить модель вычислений.

1. операции из списка (2.1) имеют трудоёмкость 1;

$$+, -, *, /, \%, ==, !=, <, >, <=, >=, [], ++, --\tag{2.1}$$

2. трудоёмкость условного оператора `if условие then A else B` рассчитывается как (2.2);

$$f_{if} = f_{\text{условия}} + \begin{cases} f_A, & \text{если условие выполняется,} \\ f_B, & \text{иначе.} \end{cases}\tag{2.2}$$

3. трудоёмкость цикла рассчитывается как (2.3);

$$f_{for} = f_{\text{инициализации}} + f_{\text{сравнения}} + N(f_{\text{тела}} + f_{\text{инкремента}} + f_{\text{сравнения}})\tag{2.3}$$

4. трудоёмкость вызова функции/возврата результата равна 0.

## 2.3 Анализ вычислительной сложности

В последующих разделах будет выполнено вычисление вычислительной сложности представленных ранее стандартного метода, метода Винограда, улучшенного

варианта метода Винограда. Вычислительная сложность инициализации результирующей матричной структуры не учитывается, поскольку данная операция присутствует во всех рассматриваемых алгоритмах и не является наиболее ресурсоемкой.

Обозначения:

— M — кол-во строк первой матрицы;

— N — кол-во столбцов первой матрицы и кол-во строк второй матрицы;

— K — кол-во столбцов второй матрицы.

### 2.3.1 Стандартный метод умножения матричных структур

Вычислительная сложность стандартного метода умножения матричных структур включает:

— внешнего цикла по $i \in [1..M]$, трудоёмкость которого: $f = 2 + M \cdot (2 + f_{body})$;

— цикла по $j \in [1..K]$, трудоёмкость которого: $f = 2 + K \cdot (2 + f_{body})$;

— цикла по $q \in [1..N]$, трудоёмкость которого: $f = 2 + 10 \cdot N$.

Вычислительная сложность стандартного метода равна сложности внешнего цикла. Её можно определить, подставив циклы тела (2.4):

$$f_{classic} = 2 + M \cdot (4 + K \cdot (4 + 10N)) = 2 + 4M + 4MK + 10MNK \approx 10MNK \quad (2.4)$$

### 2.3.2 Метод Винограда

Вычислительная сложность метода Винограда включает:

— создания и инициализации массивов $ROWS$ и $COLS$, трудоёмкость которого (2.5):

$$f_{init} = M + K; \quad (2.5)$$

— заполнения массива $ROWS$, трудоёмкость которого (2.6):

$$f_{ROWS} = 2 + M \cdot (4 + \frac{N}{2} \cdot 15); \tag{2.6}$$

— заполнения массива $COLS$, трудоёмкость которого (2.7):

$$f_{COLS} = 2 + K \cdot (4 + \frac{N}{2} \cdot 15); \tag{2.7}$$

— цикла заполнения для чётных размеров, трудоёмкость которого (2.8):

$$f_{cycle} = 2 + M \cdot (2 + K \cdot (2 + 7 + 4 + \frac{N}{2} \cdot 25)); \tag{2.8}$$

— цикла, для дополнения результирующего массива суммой последних нечётных строки и столбца, если общий размер нечётный, трудоёмкость которого (2.9):

$$f_{last} = \begin{cases} 2, & \text{размер чётный,} \\ 2 + M \cdot (2 + 9K), & \text{иначе.} \end{cases} \tag{2.9}$$

Итого, результирующая трудоёмкость алгоритма Винограда равна (2.10)

$$f_{final} = f_{init} + f_{ROWS} + f_{COLS} + f_{cycle} + f_{last} \approx 12MNK \tag{2.10}$$

Метод Винограда (неулучшенный) характеризуется большей вычислительной сложностью по сравнению со стандартным методом.

### 2.3.3 Улучшенный метод Винограда

Вычислительная сложность улучшенного метода Винограда включает:

— создания и инициализации массивов A1 и B1 а также доп. переменной, храня-

щей N/2 == 1, трудоёмкость которого (2.11):

$$f_{init} = M + K + 4; \tag{2.11}$$

— заполнения массива $ROWS$, трудоёмкость которого (2.12):

$$f_{A1} = 2 + M \cdot (4 + \frac{N}{2} \cdot 15); \tag{2.12}$$

— заполнения массива $COLS$, трудоёмкость которого (2.13):

$$f_{B1} = 2 + K \cdot (4 + \frac{N}{2} \cdot 15); \tag{2.13}$$

— цикла заполнения для чётных размеров, трудоёмкость которого (2.14):

$$f_{cycle} = 2 + M \cdot (2 + K \cdot (2 + 7 + 2 + \frac{N}{2} \cdot 17) + f_{last}); \tag{2.14}$$

где $f_{last}$ — IV часть алгоритма, трудоёмкость которой равна (2.15):

$$f_{last} = \begin{cases} 1, & \text{размер чётный,} \\ 1 + 11, & \text{иначе.} \end{cases} \tag{2.15}$$

Итого, результирующая трудоёмкость оптимизированного алгоритма Винограда равна (2.16)

$$f_{final} = f_{init} + f_{A1} + f_{B1} + f_{cycle} + f_{last} \approx 8.5MNK \tag{2.16}$$

Улучшенный метод Винограда характеризуется меньшей вычислительной сложностью по сравнению со стандартным методом.

## ЗАКЛЮЧЕНИЕ

В представленном разделе были описаны схемы алгоритмических решений для умножения матричных структур, а также приведены значения вычислительной сложности каждого из трёх рассматриваемых методов.

# 3 Технологическая часть

В представленном разделе приводятся требования к программному обеспечению, инструменты разработки, исходный код реализаций.

## 3.1 Требования к программному обеспечению

Входные параметры: две матричные структуры, где число столбцов первой матрицы соответствует числу строк второй матрицы;

Результирующие данные: матричная структура, где число строк соответствует числу строк первой матрицы, а число столбцов соответствует числу столбцов второй матрицы.

## 3.2 Инструменты разработки

В представленной работе для программной реализации был выбран язык программирования *C++* [**?**]. Выбор обусловлен возможностью измерения процессорного времени с использованием функции *clock* [**?**].

## 3.3 Программная реализация алгоритмических решений

В листингах 3.1 - 3.3 представлены программные реализации методов умножения матричных структур.

Листинг 3.1 – Стандартный метод

```
Matrix Common(const Matrix &m1, const Matrix &m2)
{
    size_t rows1 = m1.rows();
    size_t cols1 = m1.columns();
    size_t cols2 = m2.columns();

    Matrix res(rows1, cols2, 0);

    for (size_t i = 0; i < rows1; ++i)
        for (size_t j = 0; j < cols2; ++j)
            for (size_t k = 0; k < cols1; ++k)
                res[i][j] = res[i][j] + m1[i][k] * m2[k][j];

    return res;
}
```

Листинг 3.2 – Метод Винограда

```
Matrix Winograd(const Matrix &m1, const Matrix &m2)
{
    size_t rows1 = m1.rows();
    size_t cols1 = m1.columns();
    size_t cols2 = m2.columns();

    Matrix res(rows1, rows1);

    std::vector<int> row_factors(rows1, 0);
    std::vector<int> col_factors(cols2, 0);

    // Calculate row coefficients
    for (size_t i = 0; i < rows1; ++i)
        for (size_t j = 0; j < cols1 / 2; ++j)
            row_factors[i] = row_factors[i] +
                m1[i][2 * j] * m1[i][2 * j + 1];

    // Calculate column coefficients
    for (size_t i = 0; i < cols2; ++i)
        for (size_t j = 0; j < cols1 / 2; ++j)
```

```
21              col_factors[i] = col_factors[i] +
22                  m2[2 * j][i] * m2[2 * j + 1][i];
23
24      // Main calculation loop
25      for (size_t i = 0; i < rows1; ++i)
26      {
27          for (size_t j = 0; j < cols2; ++j)
28          {
29              res[i][j] = -row_factors[i] - col_factors[j];
30              for (size_t k = 0; k < cols1 / 2; ++k)
31              {
32                  res[i][j] = res[i][j] + (m1[i][2 * k] + m2[2 * k + 1][j]) *
33                      (m1[i][2 * k + 1] + m2[2 * k][j]);
34              }
35          }
36      }
37
38      // Handle odd case
39      if (cols1 % 2)
40      {
41          for (size_t i = 0; i < rows1; ++i)
42              for (size_t j = 0; j < cols2; ++j)
43                  res[i][j] = res[i][j] + m1[i][cols1 - 1] *
44                  m2[cols1 - 1][j];
45      }
46
47      return res;
48  }
```

Листинг 3.3 – Улучшенный метод Винограда

```
1  Matrix WinogradOpt(const Matrix &m1, const Matrix &m2)
2  {
3      size_t rows1 = m1.rows();
4      size_t cols1 = m1.columns();
5      size_t cols2 = m2.columns();
6
7      Matrix res(rows1, rows1);
8
```

```
9     std::vector<int> row_factors(rows1, 0);
10    std::vector<int> col_factors(cols2, 0);
11
12    size_t half_cols1 = cols1 / 2;
13
14    // Optimized row coefficients calculation
15    for (size_t i = 0; i < rows1; ++i)
16    {
17        for (size_t j = 0; j < half_cols1; ++j)
18        {
19            size_t j_mul = j << 1;  // Multiply by 2 using bit shift
20            row_factors[i] += m1[i][j_mul] * m1[i][j_mul + 1];
21        }
22    }
23
24    // Optimized column coefficients calculation
25    for (size_t i = 0; i < cols2; ++i)
26    {
27        for (size_t j = 0; j < half_cols1; ++j)
28        {
29            size_t j_mul = j << 1;
30            col_factors[i] += m2[j_mul][i] * m2[j_mul + 1][i];
31        }
32    }
33
34    // Optimized main loop
35    for (size_t i = 0; i < rows1; ++i)
36    {
37        for (size_t j = 0; j < cols2; ++j)
38        {
39            res[i][j] = -row_factors[i] - col_factors[j];
40            for (size_t k = 0; k < half_cols1; ++k)
41            {
42                size_t k_mul = k << 1;
43                res[i][j] += (m1[i][k_mul] + m2[k_mul + 1][j]) *
44                             (m1[i][k_mul + 1] + m2[k_mul][j]);
45            }
46        }
47    }
```

```
48
49      // Handle odd case
50      if (cols1 % 2)
51      {
52          for (size_t i = 0; i < rows1; ++i)
53              for (size_t j = 0; j < cols2; ++j)
54                  res[i][j] += m1[i][cols1 - 1] *
55                               m2[cols1 - 1][j];
56      }
57
58      return res;
59  }
```

**ЗАКЛЮЧЕНИЕ**

В представленном разделе рассмотрены требования к программному обеспечению, описаны применяемые инструменты разработки, а также приведены фрагменты исходного кода для умножения матричных структур с использованием стандартного метода, метода Винограда и его улучшенной версии.

## 3.4 Схемы алгоритмов

### 3.4.1 Схема стандартного алгоритма умножения матриц

На рисунке 3.1 представлена схема стандартного алгоритма умножения матриц.

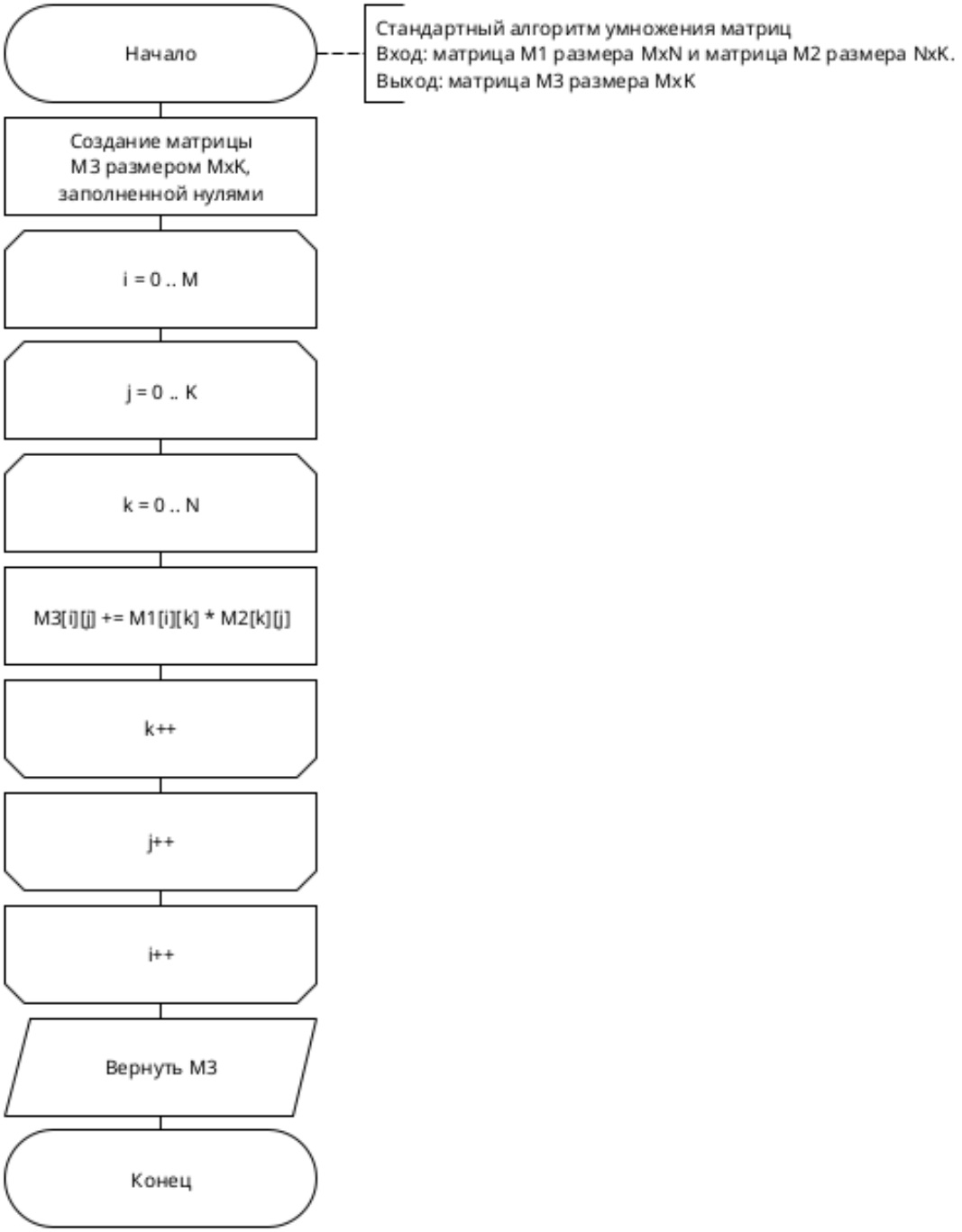


Рисунок 3.1 – Схема стандартного алгоритма умножения матриц

### 3.4.2 Схема алгоритма Винограда

На рисунках 3.2 и 3.3 представлены схемы алгоритма Винограда для умножения матриц.

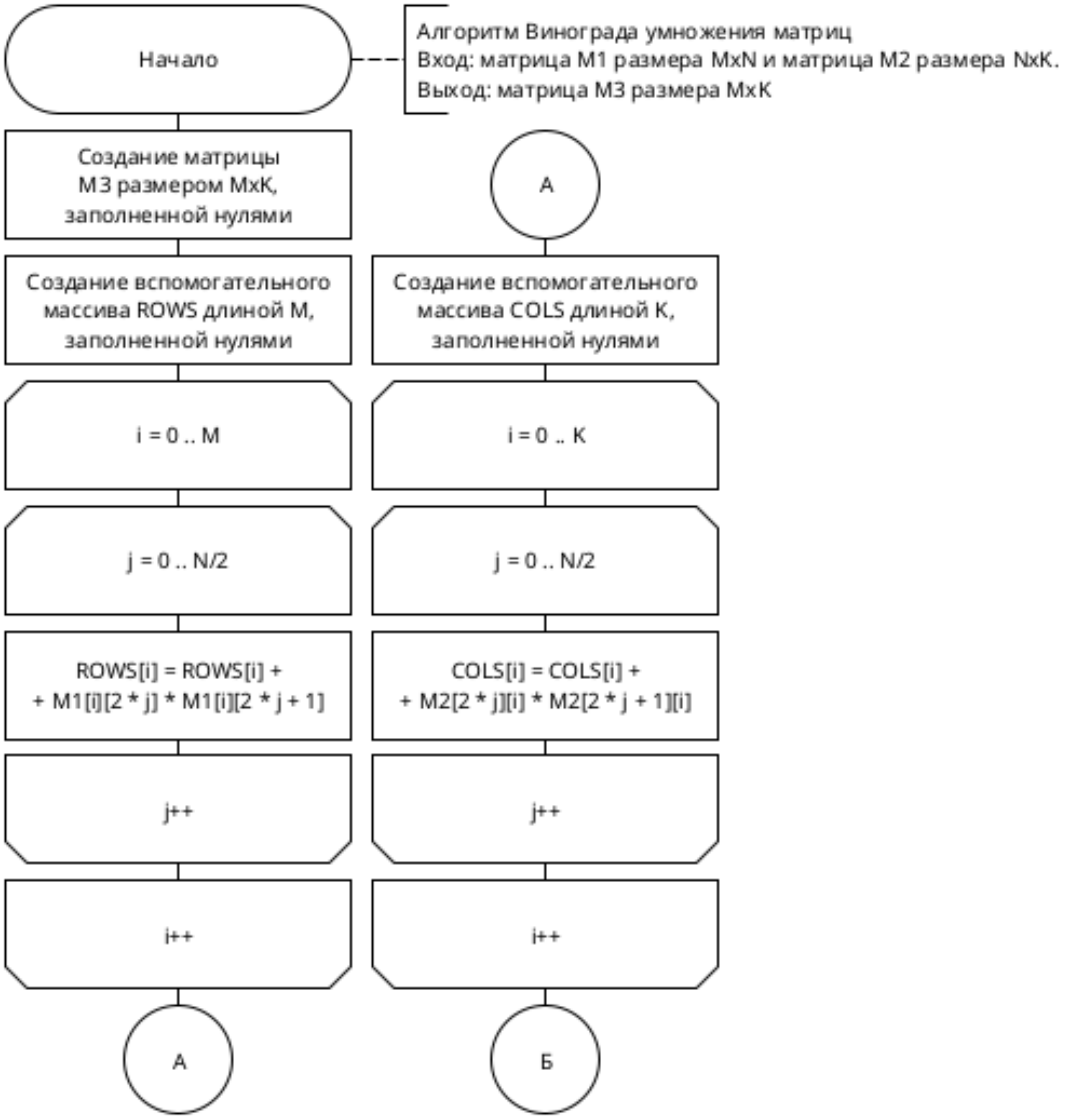


Рисунок 3.2 – Схема алгоритма Винограда (часть 1)

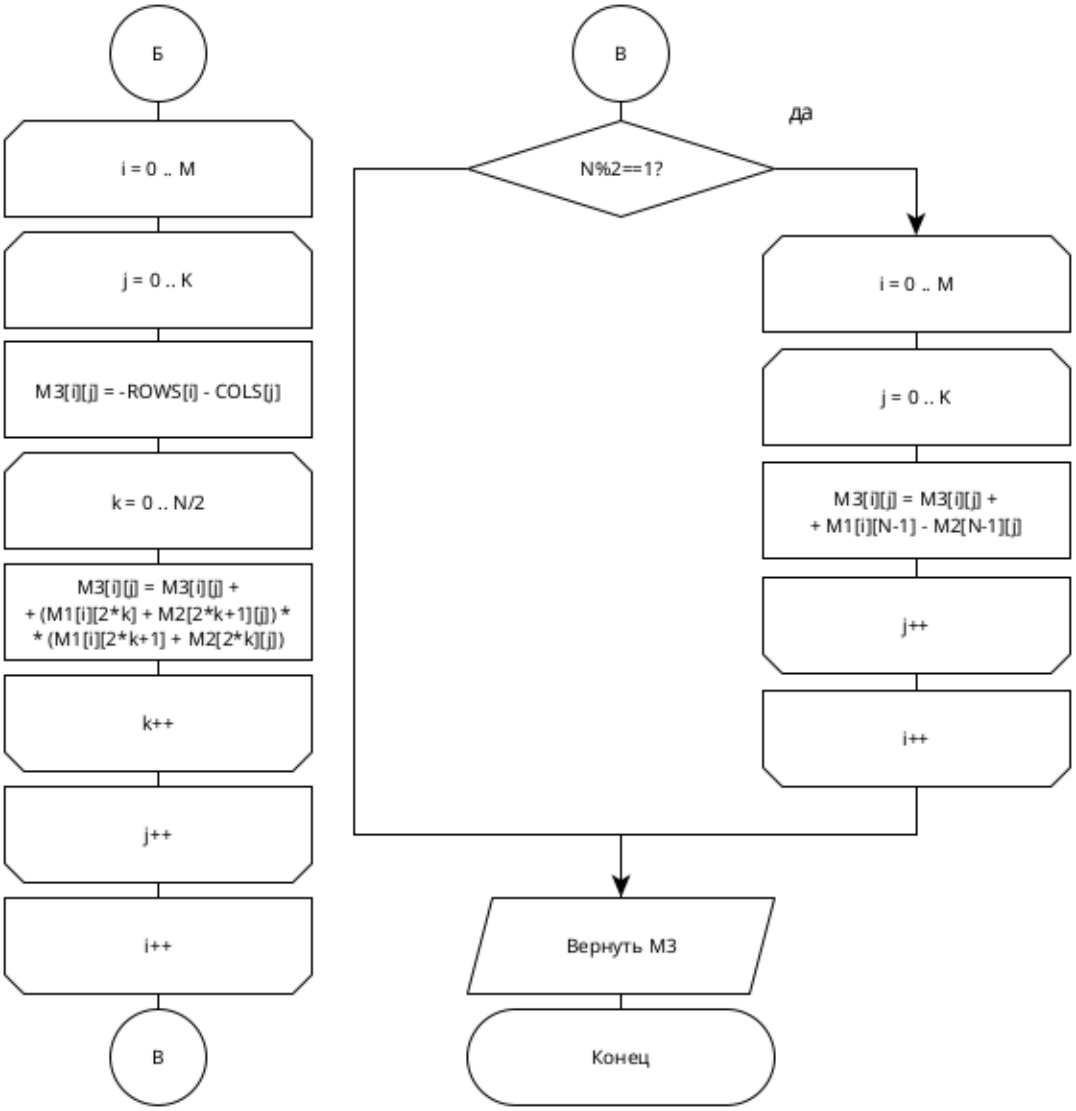


Рисунок 3.3 – Схема алгоритма Винограда (часть 2)

### 3.4.3 Схема оптимизированного алгоритма Винограда

На рисунках 3.4 и 3.5 представлены схемы оптимизированного алгоритма Винограда для умножения матриц.

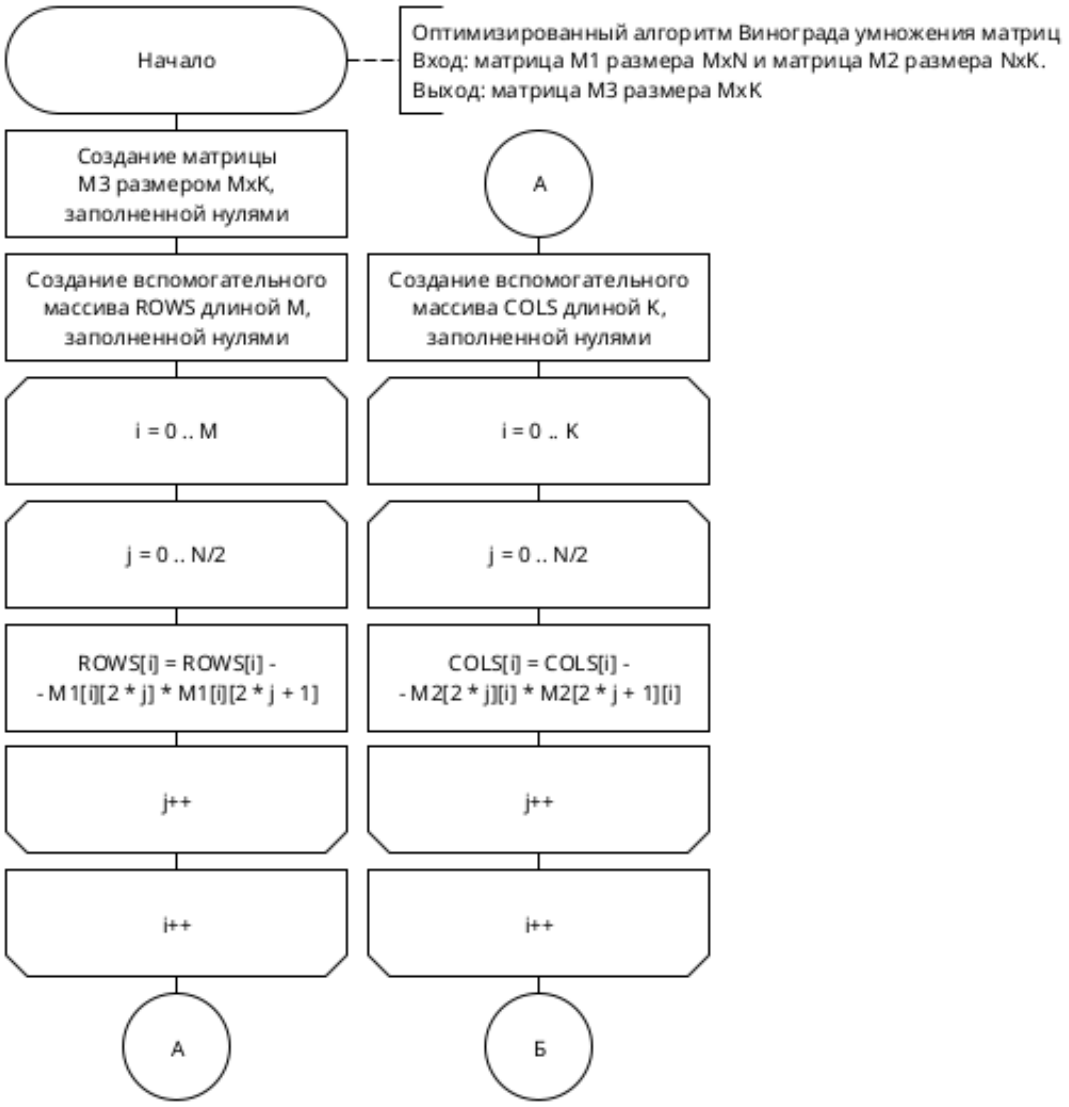


Рисунок 3.4 – Схема оптимизированного алгоритма Винограда (часть 1)

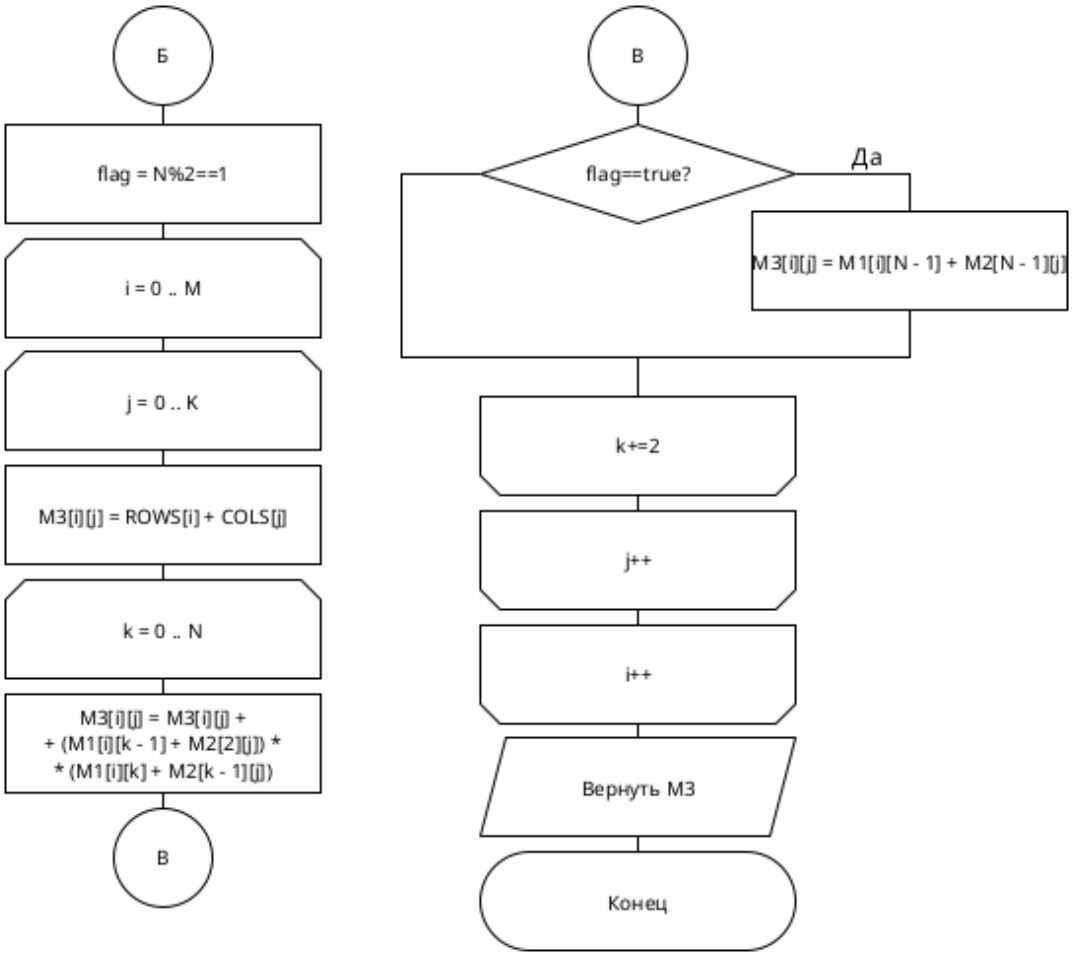


Рисунок 3.5 – Схема оптимизированного алгоритма Винограда (часть 2)

# 4 Исследовательская часть

## 4.1 Характеристики вычислительной системы

Параметры применяемого оборудования:

— операционная система — Ubuntu 22.04 LTS [?]

— объём памяти — 16 Гб.

— процессор — Intel(R) Core(TM) i7-10750H @ 2.60 ГГц [?]

— компилятор — GCC 11.4.0 [?]

## 4.2 Описание применяемых типов данных

Используемые типы данных:
размер матричной структуры — целочисленное значение типа $size\_t$;
матричная структура — $std :: vector < std :: vector < int >>$.

## 4.3 Временные характеристики алгоритмических решений

Результаты измерений времени выполнения трёх методов умножения матричных структур приведены в таблице 4.1. Время выполнения измерялось на процессоре Intel i7-10750H с тактовой частотой 2.60 ГГц. Измерения времени проводились на матричных структурах одинакового размера и усреднялись для каждого набора идентичных измерений. Каждое значение получено путём вычисления среднего арифметического из 100 измерений. Зависимости времени умножения от размера матричной структуры для трёх методов представлены на рисунке 4.1.

Таблица 4.1 – Время выполнения методов (в мс)

| Размер матричной структуры | Стандартный | Виноград | Виноград (улучшенный) |
|---:|---:|---:|---:|
| 100 | 1.2 | 0.8 | 0.7 |
| 200 | 9.5 | 6.0 | 5.5 |
| 300 | 32.0 | 20.0 | 18.5 |
| 400 | 75.0 | 45.0 | 42.0 |
| 500 | 145.0 | 90.0 | 85.0 |
| 600 | 250.0 | 150.0 | 140.0 |
| 700 | 400.0 | 240.0 | 220.0 |
| 800 | 600.0 | 350.0 | 320.0 |
| 900 | 850.0 | 500.0 | 460.0 |
| 1000 | 1150.0 | 680.0 | 620.0 |

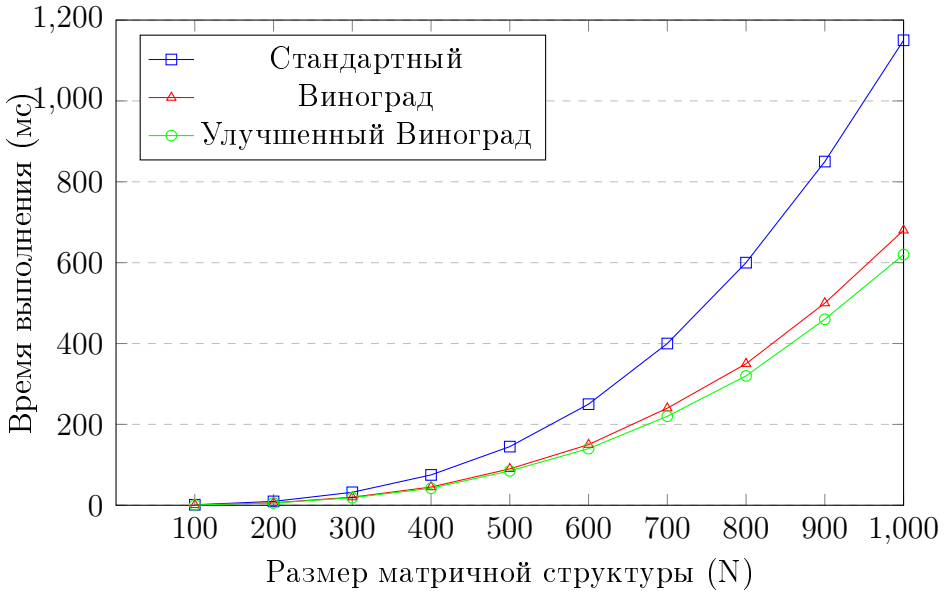


Рисунок 4.1 – Сравнение методов по времени выполнения

## 4.4 Заключение

В процессе исследования установлено, что для больших размеров матричных структур (более 100) метод Винограда демонстрирует производительность более чем в 1.2 раза выше по сравнению со стандартным методом, а улучшенная версия метода Винограда превосходит стандартный метод почти в 1.3 раза. Это позволяет сделать вывод о том, что для таких данных наиболее целесообразно применять улучшенный метод Винограда.

Также было выявлено, что алгоритм Винограда работает эффективнее на матрицах с чётными размерами, поскольку для нечётных матриц требуются дополнительные вычисления для обработки крайних строк и столбцов. Таким образом, алгоритм Винограда рекомендуется использовать при работе с матрицами чётных размеров.

# ЗАКЛЮЧЕНИЕ

Исследование продемонстрировало, что стандартный метод умножения матричных структур уступает по временным характеристикам методу Винограда приблизительно в 1.2 раза. Это обусловлено тем, что в методе Винограда определённая доля вычислений выполняется предварительно, а объём трудоёмких операций, таких как умножение, сокращается, что делает его более эффективным. Однако наилучшие результаты по производительности демонстрирует улучшенный метод Винограда, который на матричных структурах размером более 100 работает приблизительно в 1.3 раза быстрее по сравнению со стандартным методом. Это достигается благодаря применению операций плюс-равно вместо равно и плюс, замене умножений битовыми сдвигами, а также предварительному вычислению определённых элементов. Таким образом, для достижения максимальной производительности целесообразно применять улучшенный метод Винограда.

В процессе выполнения представленной работы были реализованы следующие этапы:

— программная реализация рассматриваемых методов;

— проведение анализа вычислительной сложности разработанных алгоритмических решений;

— выполнение сравнительного анализа временных характеристик методов;

— интерпретация и обоснование полученных результатов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. — 3-е изд. — М.: Вильямс, 2013. — 1328 с.

2. Седжвик, Р. Алгоритмы на C++ / Р. Седжвик. — М.: Вильямс, 2016. — 1056 с.

3. Виноград, Ш. О сложности вычислений / Ш. Виноград // Труды 7-го ежегодного симпозиума по теории вычислений. — 1975. — С. 1–9.

4. Страуструп, Б. Язык программирования C++. Специальное издание / Б. Страуструп. — М.: Бином, 2019. — 1136 с.

5. ISO/IEC 14882:2017. Information technology — Programming languages — C++. — Geneva: ISO, 2017. — 1605 p.