



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»

## Отчёт по лабораторной работе №2 по курсу "Анализ алгоритмов"

Тема Алгоритмы умножения матриц

Студент Михаил Коротыч

Группа ИУ7-55Б

Оценка (баллы) \_\_\_\_\_

Преподаватели Волкова Л.Л., Строганов Ю.В.

Москва — 2021 г.

# Содержание

<b>Введение</b>	<b>3</b>
<b>1 Аналитическая часть</b>	<b>5</b>
1.1 Классический алгоритм умножения матриц . . . . .	5
1.2 Алгоритм Винограда . . . . .	6
1.3 Оптимизированный алгоритм Винограда . . . . .	7
1.4 Вывод . . . . .	7
<b>2 Конструкторская часть</b>	<b>8</b>
2.1 Схемы алгоритмов . . . . .	8
2.2 Вычисление трудоёмкости алгоритма . . . . .	18
2.3 Оценка трудоёмкости алгоритмов умножения матриц . . . .	18
2.4 Вывод . . . . .	20
<b>3 Технологическая часть</b>	<b>21</b>
3.1 Требования к вводу . . . . .	21
3.2 Требования к выводу . . . . .	21
3.3 Требования к программе . . . . .	21
3.4 Выбор языка программирования . . . . .	21
3.5 Сведения о модулях программы . . . . .	21
3.6 Функциональные тесты . . . . .	25
3.7 Вывод . . . . .	25
<b>4 Исследовательская часть</b>	<b>26</b>
4.1 Технические характеристики . . . . .	26
4.2 Демонстрация работы программы . . . . .	26
4.3 Время работы алгоритмов . . . . .	27

4.4 Вывод . . . . .	28
<b>5 Заключение</b>	<b>29</b>
<b>Список литературы</b>	<b>30</b>

## Введение

Термин «матрица» применяется во множестве разных областей: от программирования до кинематографии.

Матрица - это математический объект, представляющий из себя набор упорядоченных чисел (целых, дробных или даже комплексных). Эти числа записываются, как правило в виде квадратной или прямоугольной таблицы, над которой можно совершать различные операции. Матрицы — очень важный математический инструмент, позволяющий решать множество задач от систем уравнений до оптимизации поставок.

Мы встречаемся с матрицами каждый день, так как любая числовая информация, занесённая в таблицу, уже в какой-то степени считается матрицей.

Целью работы работы является изучение и реализация алгоритмов умножения матриц, вычисление трудоёмкости этих алгоритмов. В данной лабораторной работе рассматривается стандартный алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда.

Для достижения цели ставятся следующие задачи:

- изучить классический алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда;
- сравнить классический алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда;
- выявить достоинства и недостатки рассмотренных алгоритмов;
- дать оценку трудоёмкости алгоритмов;
- реализовать рассмотренные алгоритмы;
- замерить время работы алгоритмов;

- описать и обосновать полученные результаты в отчёте о выполненной лабораторной работе.

## 1 Аналитическая часть

Матрица – математический объект, эквивалентный двумерному массиву. Числа располагаются в матрице по строкам и столбцам. Две матрицы одинакового размера можно поэлементно сложить или вычесть друг из друга [?].

Если число столбцов в первой матрице совпадает с числом строк во второй, то эти две матрицы можно перемножить. У произведения будет столько же строк, сколько в первой матрице, и столько же столбцов, сколько во второй. При умножении матрицы размером  $3 \times 4$  на матрицу размером  $4 \times 7$  мы получаем матрицу размером  $3 \times 7$ . Умножение матриц некоммутативно: оба произведения  $AB$  и  $BA$  двух квадратных матриц одинакового размера можно вычислить, однако результаты, вообще говоря, будут отличаться друг от друга [?].

### 1.1 Классический алгоритм умножения матриц

Пусть даны две прямоугольные матрицы  $A$  (1) и  $B$  (2) размерности  $m$  на  $n$  и  $n$  на  $l$  соответственно:

$$\begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \dots & \dots & \dots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} b_{1,1} & \dots & b_{1,l} \\ \dots & \dots & \dots \\ b_{n,1} & \dots & b_{n,l} \end{bmatrix} \quad (2)$$

В результате получим матрицу  $C$  3 размерности  $m$  на  $l$ :

$$\begin{bmatrix} c_{1,1} & \dots & c_{1,l} \\ \dots & \dots & \dots \\ c_{m,1} & \dots & c_{m,l} \end{bmatrix} \quad (3)$$

Формула 4 - формула расчёта элемента, находящегося на  $i$ -ой строке  $j$ -ого столбца матрицы  $C$ :

$$c_{i,j} = \sum_{r=1}^n a_{i,r} \cdot b_{r,j} \quad (4)$$

## 1.2 Алгоритм Винограда

Если посмотреть на 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 [?].

Рассмотрим два вектора  $V = (v_1, v_2, v_3, v_4)$  и  $W = (w_1, w_2, w_3, w_4)$ . Их скалярное произведение равно:

$$V \cdot W = v_1 \cdot w_1 + v_2 \cdot w_2 + v_3 \cdot w_3 + v_4 \cdot w_4 \quad (5)$$

Это равенство можно переписать в виде 6:

$$V \cdot W = (v_1 + w_2) \cdot (v_2 + w_1) + (v_3 + w_4) \cdot (v_4 + w_3) - v_1 \cdot v_2 - v_3 \cdot v_4 - w_1 \cdot w_2 - w_3 \cdot w_4 \quad (6)$$

Менее очевидно, что выражение в правой части последнего равенства допускает предварительную обработку: его части можно вычислить заранее и запомнить для каждой строки первой матрицы и для каждого столбца второй. Это означает, что над предварительно обработанными элементами нам придётся выполнять лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения [?].

### 1.3 Оптимизированный алгоритм Винограда

Оптимизированный алгоритм Винограда представляет собой обычный алгоритм Винограда, за исключением следующих оптимизаций:

- вычисление происходит заранее;
- используется битовый сдвиг вместо деления на 2;
- последний цикл для нечётных элементов включён в основной цикл, используя дополнительные операции в случае нечётности  $N$ .

### 1.4 Вывод

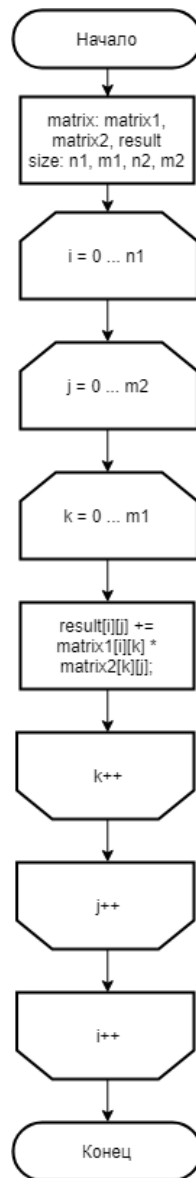
Были рассмотрены алгоритмы классического умножения матриц и алгоритм Винограда, основная разница которого - наличие предварительной обработки, а также уменьшение количества операций умножения.



## **2 Конструкторская часть**

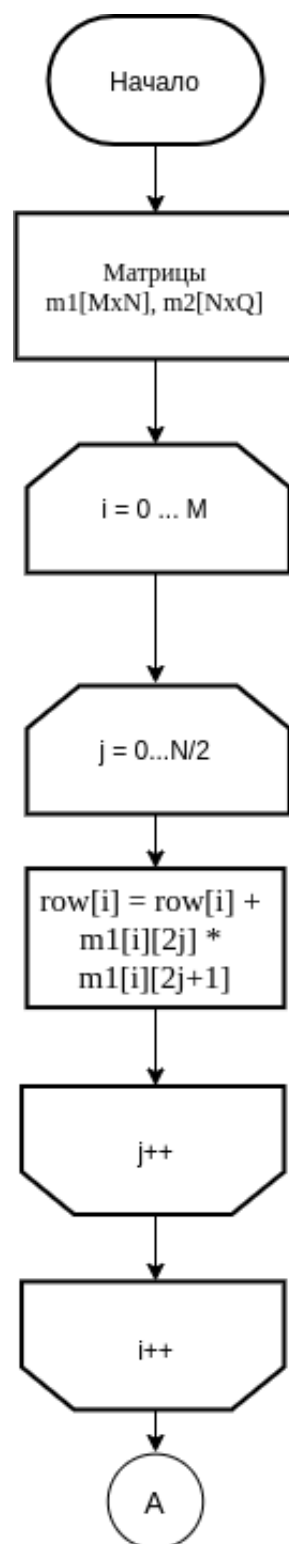
### **2.1 Схемы алгоритмов**

На рисунке 1 представлена схема классического умножения матриц.



**Рисунок 1** – Схема классического алгоритма умножения матриц

На рисунках 2, 3, 4, 5 представлена схема алгоритма умножения матриц Винограда.



**Рисунок 2** – Схема алгоритма Винограда (часть 1)

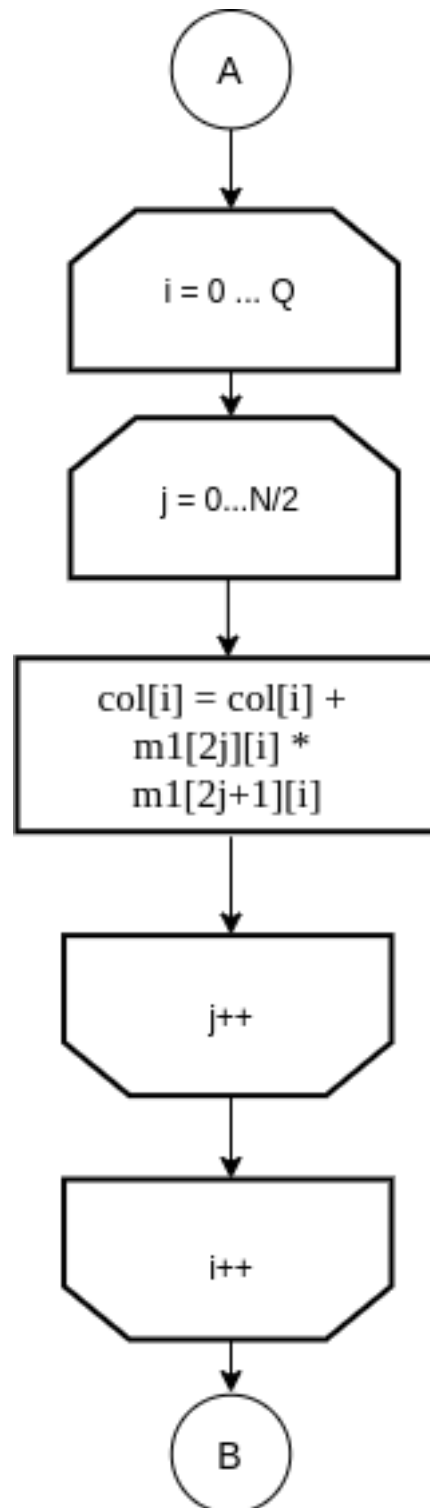
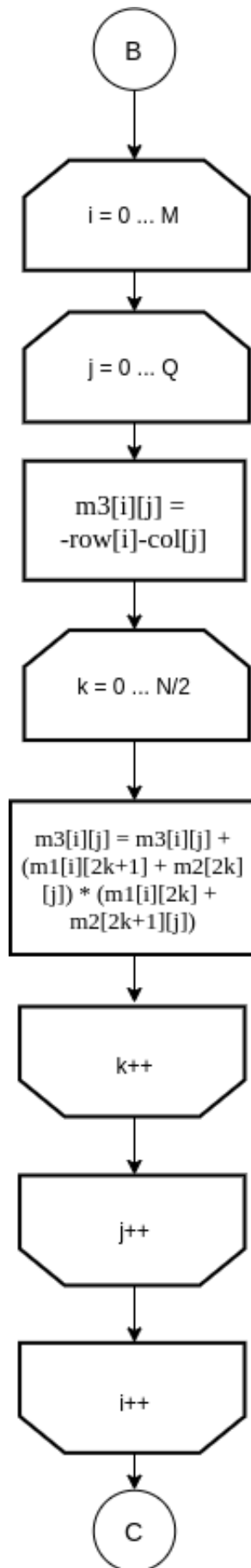
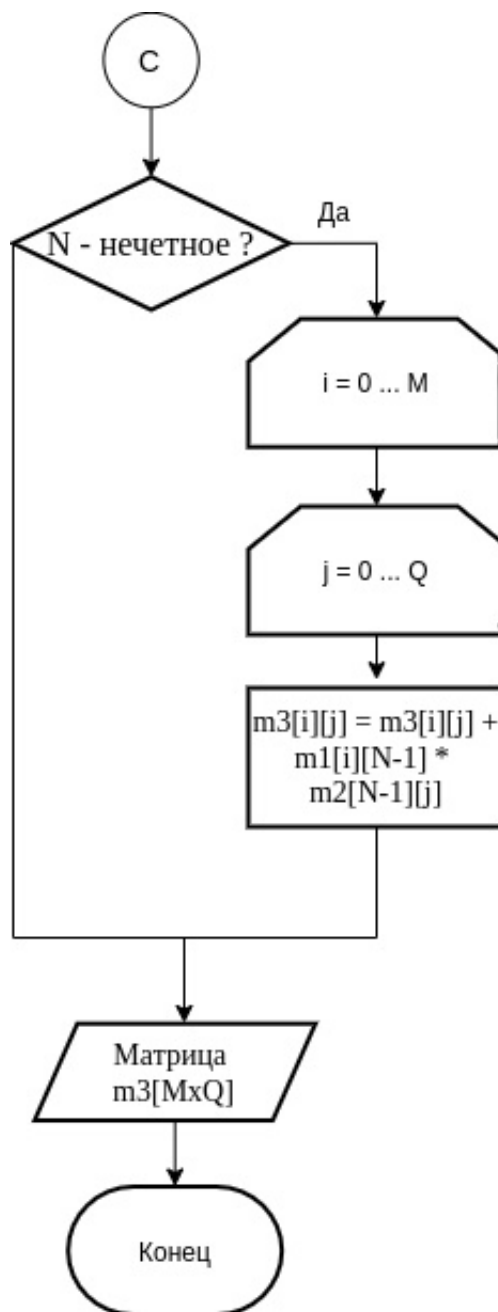


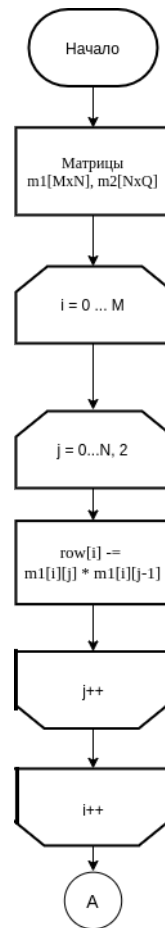
Рисунок 3 – Схема алгоритма Винограда (часть 2)



**Рисунок 4** – Схема алгоритма Винограда (часть 3)

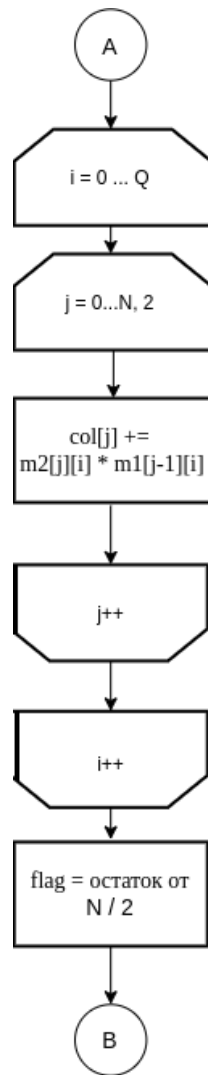


**Рисунок 5** – Схема алгоритма Винограда (часть 4)



**Рисунок 6** – Схема оптимизированного алгоритма Винограда(часть 1)

На рисунках 6, 7, 8, 9 представлена схема оптимизированного алгоритма умножения матриц Винограда.



**Рисунок 7** – Схема оптимизированного алгоритма Винограда(часть 2)



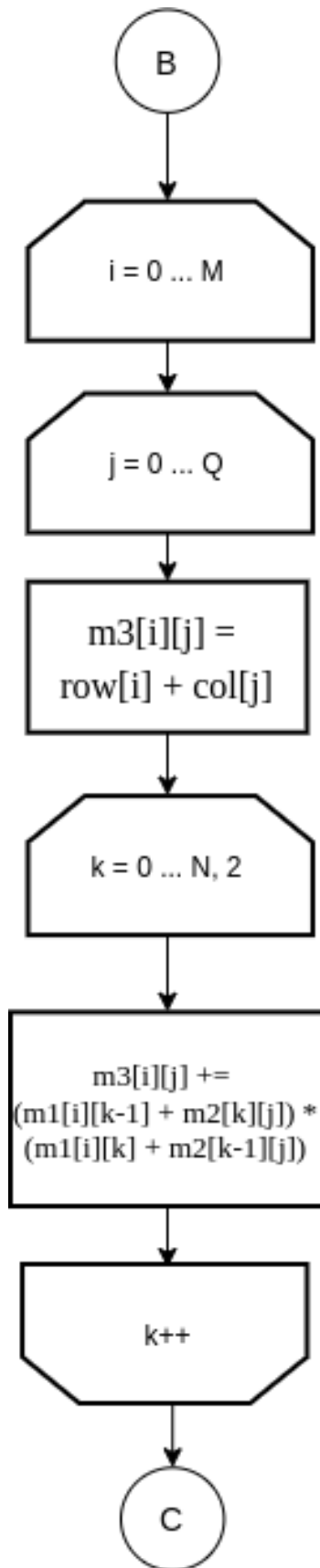
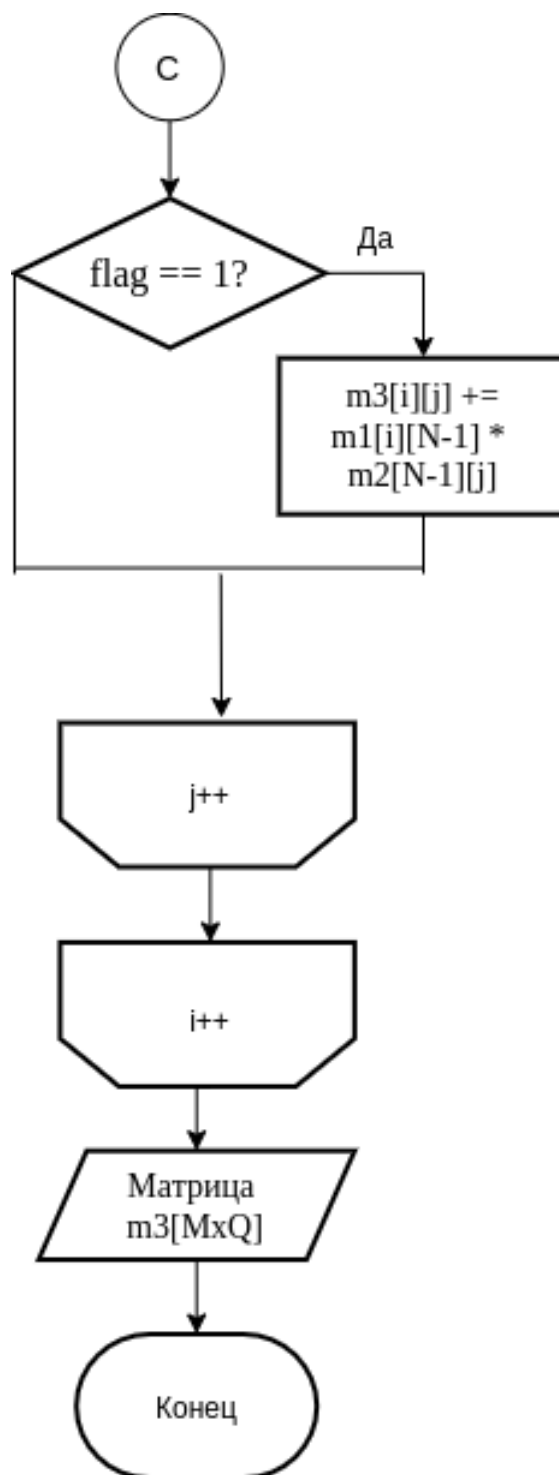


Рисунок 8 – Схема оптимизированного алгоритма Винограда(часть 3)



**Рисунок 9** – Схема оптимизированного алгоритма Винограда(часть 4)

## 2.2 Вычисление трудоёмкости алгоритма

Введём модель трудоёмкости для оценки алгоритмов.

1. Базовые операции стоимостью 1: +, -, \*, /, =, ==, <=, >=, !=, +=, [], получение полей класса.

2. Оценка трудоёмкости цикла:

$$f_{\text{цикла}} = f_{\text{инициализации}} + f_{\text{сравнения}} + N * (f_{\text{инкремента}} + f_{\text{сравнения}} + f_{\text{тела}}).$$

3. Стоимость условного перехода возьмём за 0, стоимость вычисления условия остаётся. В условном операторе может возникнуть лучший и худший случаи по трудоёмкости в зависимости от выполнения условия и в зависимости от входных данных алгоритма.

## 2.3 Оценка трудоёмкости алгоритмов умножения матриц

Оценка трудоёмкости дана согласно введённой выше модели вычислений.

1. Стандартный алгоритм

$$f = 2 + M(2 + 2 + Q(2 + 2 + N(2 + 8 + 1 + 1 + 1))) = 13 \cdot$$

$$\cdot MNQ + 4MQ + 4M + 2 \approx 13 \cdot MNQ$$

2. Алгоритм Винограда

(а) Трудоёмкость алгоритма Винограда:

(b) Первый цикл:  $\frac{15}{2} \cdot NQ + 5 \cdot M + 2$

Второй цикл:  $\frac{15}{2} \cdot MN + 5 \cdot M + 2$

Третий цикл:  $13 \cdot MNQ + 12 \cdot MQ + 4 \cdot M + 2$

(с) Условный переход:

$$\left[ \begin{array}{ll} 2 & , \text{лучший случай при чётном } N \\ 15 \cdot QM + 4 \cdot M + 4 & , \text{худший случай} \end{array} \right]$$

(d) Итого:

$$f = \frac{15}{2} \cdot MN + \frac{15}{2} \cdot QN + 9 \cdot M + 8 + 5 \cdot Q + 13 \cdot MNQ + 12 \cdot MQ +$$

$$\left[ \begin{array}{ll} 2 & , \text{в лучшем случае} \\ 15 \cdot QM + 4 \cdot M + 4 & , \text{в худшем случае} \end{array} \right]$$

$$f \approx 13 \cdot MNQ$$

### 3. Оптимизированный алгоритм Винограда

Введём оптимизации:

- (a) замена операции  $=$  на  $+=$  или  $-=$ ;
- (b) избавление от деления в условиях цикла ( $j < N$ ,  $j += 2$ );
- (c) заносение проверки на нечётность количества строк внутрь основных циклов;
- (d) расчёт условия для последнего цикла один раз, а далее использование флага;

Первый цикл:  $4 \cdot NQ + 4 \cdot M + 2$

Второй цикл:  $4 \cdot MN + 4 \cdot M + 2$

Третий цикл:  $9 \cdot MNQ + 10 \cdot MQ + 4 \cdot M + 2$

Условный переход:

$$\left[ \begin{array}{ll} 2 & , \text{лучший случай (при четном } N) \\ 10 \cdot QM & , \text{худший случай} \end{array} \right]$$

Трудоёмкость оптимизированного алгоритма Винограда:

$$f = 4 \cdot NQ + 4 \cdot M + 2 + 4 \cdot MN + 4 \cdot M + 2 + 9 \cdot MNQ + 10 \cdot MQ + 4 \cdot M + 2 + \\ + \left[ \begin{array}{cc} 2 & , \text{л.с} \\ 10 \cdot QM & , \text{х.с} \end{array} \right] \approx 9 \cdot MNQ$$

## 2.4 Вывод

На основе теоретических данных, полученных из аналитического раздела были построены схемы требуемых алгоритмов и вычислены их трудоёмкости.

## 3 Технологическая часть

### 3.1 Требования к вводу

На вход подаются две матрицы. Операция умножения двух матриц выполняма только в том случае, если число столбцов в первом сомножителе равно числу строк во втором; в этом случае говорят, что матрицы согласованы.

### 3.2 Требования к выводу

Матрица (результат умножения матриц, поданных на вход).

### 3.3 Требования к программе

Две пустые матрицы - корректный ввод. Программа не должна аварийно завершаться.

### 3.4 Выбор языка программирования

Языком программирования был выбран C++.

Время работы алгоритмов было замерено с помощью функции из `<Windows.h>`.

### 3.5 Сведения о модулях программы

Программа состоит из модуля `main.cpp`. В модуле `main` одна за другой вызываются подпрограммы каждого рассматриваемого алгоритма, а также подпрограммы подготовки к использованию этих алгоритмов, например создание и инициализация матриц.

На листингах 1, 2, 3 представлен код алгоритмов.

#### Листинг 1 – Классический алгоритм умножения матриц

```
0 my_matrix classical_multiplication(my_matrix matrix1, my_matrix matrix2)
1 {
```

```
2     my_matrix result
3     {
4         std::vector<std::vector<float>>(matrix1.n, std::vector<float>(
matrix2.m, 0)),
5         matrix1.n,
6         matrix2.m
7     };
8
9     for (int i = 0; i < result.n; i++)
10         for (int j = 0; j < result.m; j++)
11             for (int k = 0; k < matrix1.m; k++)
12                 result.values[i][j] += matrix1.values[i][k] * matrix2.
values[k][j];
13
14     return result;
15 }
16
```

## Листинг 2 – Алгоритм Винограда

```
0   my_matrix result
1   {
2       std::vector<std::vector<float>>(matrix1.n, std::vector<float>(
matrix2.m, 0)),
3       matrix1.n,
4       matrix2.m
5   };
6
7   std::vector<float> row(matrix1.n, 0);
8   for (int i = 0; i < matrix1.n; i++)
9       for (int j = 0; j < matrix1.m / 2; j++)
10          row[i] += matrix1.values[i][2 * j] * matrix1.values[i][2 * j +
11          1];
12
13   std::vector<float> column(matrix2.m, 0);
14   for (int i = 0; i < matrix2.m; i++)
15       for (int j = 0; j < matrix1.m / 2; j++)
16          column[i] += matrix2.values[2 * j][i] * matrix2.values[2 * j +
17          1][i];
18
19   for (int i = 0; i < matrix1.n; i++)
20       for (int j = 0; j < matrix2.m; j++)
21       {
22          result.values[i][j] = -row[i] - column[j];
23          for (int k = 0; k < matrix1.m / 2; k++)
24              result.values[i][j] += (matrix1.values[i][2 * k + 1] +
matrix2.values[2 * k][j]) * (matrix1.values[i][2 * k] + matrix2.values
25              [2 * k + 1][j]);
26      }
27
28   if (matrix1.m % 2)
29       for (int i = 0; i < matrix1.n; i++)
30           for (int j = 0; j < matrix2.m; j++)
31              result.values[i][j] += matrix1.values[i][matrix1.m - 1] *
matrix2.values[matrix1.m - 1][j];
32
33   return result;
34 }
```



### Листинг 3 – Оптимизированный алгоритм Винограда

```
0 my_matrix optimized_winograd_algorithm(my_matrix matrix1, my_matrix
  matrix2)
1 {
2     my_matrix result
3     {
4         std::vector<std::vector<float>>(matrix1.n, std::vector<float>(<
matrix2.m, 0)),
5         matrix1.n,
6         matrix2.m
7     };
8
9     std::vector<float> row(matrix1.n, 0);
10    for (int i = 0; i < matrix1.n; i++)
11        for (int j = 1; j < matrix1.m; j += 2)
12            row[i] -= matrix1.values[i][j] * matrix1.values[i][j - 1];
13
14    std::vector<float> column(matrix2.m, 0);
15    for (int i = 0; i < matrix2.m; i++)
16        for (int j = 1; j < matrix1.m; j += 2)
17            column[i] -= matrix2.values[j][i] * matrix2.values[j - 1][i];
18
19    for (int i = 0; i < matrix1.n; i++)
20        for (int j = 0; j < matrix2.m; j++)
21        {
22            result.values[i][j] += row[i] + column[j];
23            for (int k = 1; k < matrix1.m; k += 2)
24            {
25                result.values[i][j] += (matrix1.values[i][k - 1] + matrix2
.values[k][j]) * (matrix1.values[i][k] + matrix2.values[k - 1][j]);
26                if (matrix1.m % 2)
27                    result.values[i][j] += matrix1.values[i][matrix1.m -
1] * matrix2.values[matrix1.m - 1][j];
28            }
29        }
30
31    return result;
32 }
33
```

### 3.6 Функциональные тесты

Таблица 3.1 – Функциональные тесты

Матрица1	Матрица2	Ожидаемый результат	Фактический результат
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{bmatrix}$	$\begin{bmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{bmatrix}$
$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	Матрицы не могут быть перемножены	Матрицы не могут быть перемножены
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 9 & 12 & 15 \\ 19 & 26 & 33 \\ 29 & 40 & 51 \end{bmatrix}$	$\begin{bmatrix} 9 & 12 & 15 \\ 19 & 26 & 33 \\ 29 & 40 & 51 \end{bmatrix}$

Все реализации алгоритмов на практике показали ожидаемый результат.

### 3.7 Вывод

В этом разделе была представлена реализация алгоритмов классического умножения матриц, алгоритма Винограда, оптимизированного алгоритма Винограда. Тестирование показало, что алгоритмы реализованы правильно и работают корректно.

## 4 Исследовательская часть

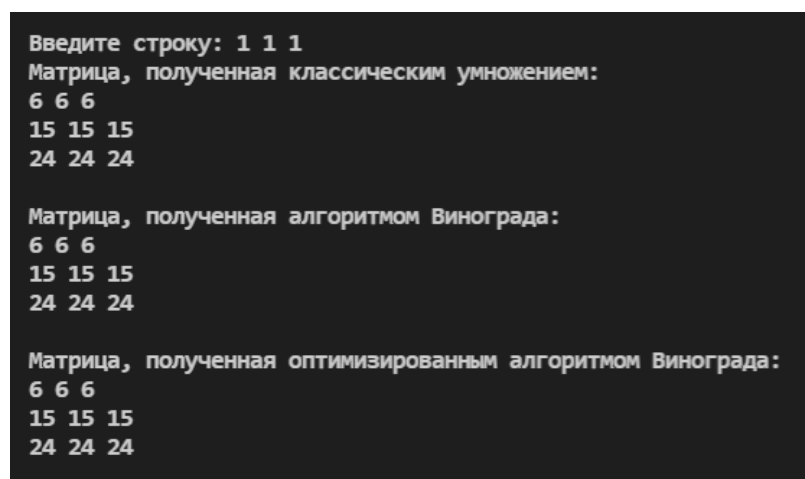
### 4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование, следующие:

- Операционная система: Windows 10 64-bit;
- Память 8 ГБ;
- Процессор: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz.

### 4.2 Демонстрация работы программы

На рисунке 10 представлен результат работы программы.



```
Введите строку: 1 1 1
Матрица, полученная классическим умножением:
6 6 6
15 15 15
24 24 24

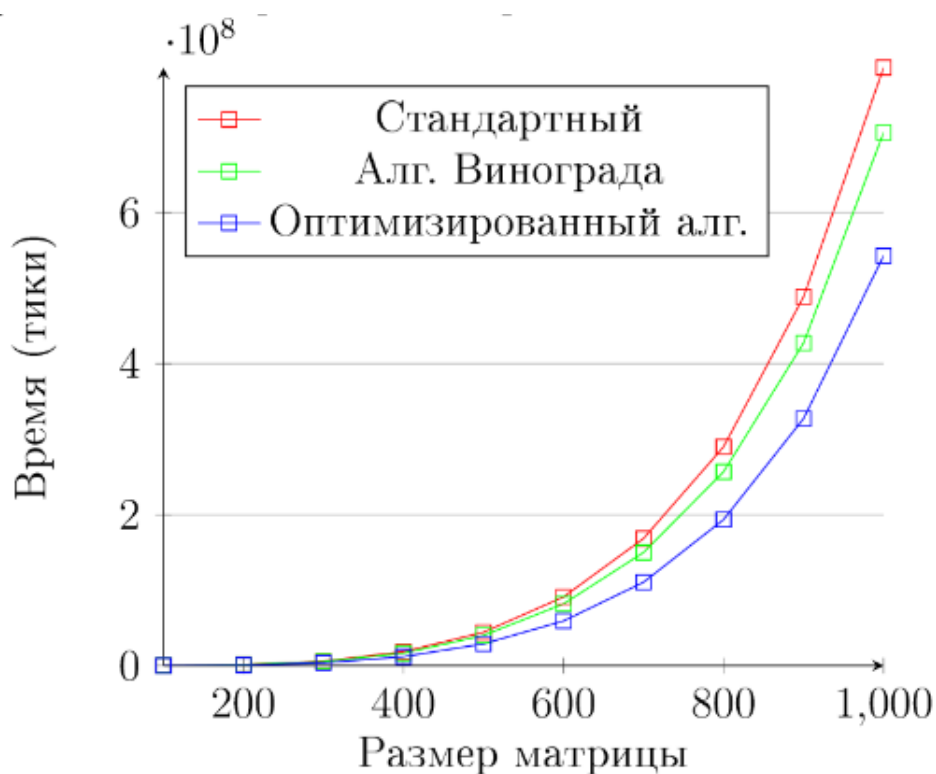
Матрица, полученная алгоритмом Винограда:
6 6 6
15 15 15
24 24 24

Матрица, полученная оптимизированным алгоритмом Винограда:
6 6 6
15 15 15
24 24 24
```

Рисунок 10 – Результат работы программы.

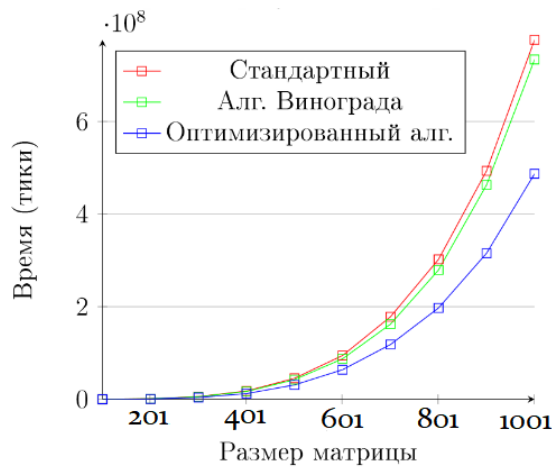
### 4.3 Время работы алгоритмов

Был проведён замер времени работы из алгоритмов. Каждый замер времени проводился 10 раз и результат усреднялся. Первый эксперимент производится для лучшего случая на матрицах размерами от 100 на 100 до 500 на 500 с шагом 100. Рисунок 11.



**Рисунок 11** – Замеры времени на чётном количестве строк и столбцов квадратных матриц.

Второй эксперимент производится для худшего случая, когда заданы матрицы с нечётными размерами от 101 на 101 до 501 на 501 с шагом 100. Рисунок 12.



**Рисунок 12** – Замеры времени на нечётном количестве строк и столбцов квадратных матриц.

По результатам тестирования, все рассматриваемые алгоритмы реализованы правильно. Самым медленным оказался алгоритм классического умножения матриц, самым быстрым - оптимизированный алгоритм Винограда.

#### 4.4 Вывод

В данном разделе протестированы алгоритмы умножения матриц. Классический алгоритм показал худшие результаты, что и следовало ожидать. Оптимизированный алгоритм Винограда проявил себя лучше остальных. Экспериментально было подтверждено различие по временной эффективности алгоритмов умножения матриц на материале замеров процессорного времени выполнения реализации на варьирующихся размерах матриц. Так, самым быстрым является оптимизированный алгоритм Винограда, на размере матрицы 400 на 400 он работает в 1,3 раза быстрее алгоритма Винограда без оптимизации и в 1,23 раз быстрее классического алгоритма умножения. Алгоритмы Винограда и классического умножения примерно сопоставимы.

## 5 Заключение

Цель лабораторной работы достигнута. В ходе выполнения работы решены следующие задачи:

- изучены классический алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда;
- реализованы классический алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда;
- дана оценка трудоёмкости алгоритмов;
- замерено время работы алгоритмов;
- описаны и обоснованы полученные результаты в отчёте о выполненной лабораторной работе.

Экспериментально было подтверждено различие по временной эффективности алгоритмов умножения матриц на материале замеров процессорного времени выполнения реализации на варьирующихся размерах матриц. Так, самым быстрым является оптимизированный алгоритм Винограда, на размере матрицы 400 на 400 он работает в 1,3 раза быстрее алгоритма Винограда без оптимизации и в 1,23 раз быстрее классического алгоритма умножения. Алгоритмы Винограда и классического умножения примерно сопоставимы.

На основании сравнения данных алгоритмов был сделан вывод, что классический алгоритм является более эффективным, чем алгоритм Винограда, однако после ряда оптимизаций алгоритм Винограда становится значительно быстрее классического.

## Список литературы

- [1] Умножение матриц.[Электронный ресурс]. Режим доступа: [https://life-prog.ru/2\\_90314\\_umnozhenie-matrits.html](https://life-prog.ru/2_90314_umnozhenie-matrits.html)
- [2] Умножение матриц по Винограду [Электронный ресурс]. Режим доступа: <http://algotlib.narod.ru/Math/Matrix.html>
- [3] CPP Reference [Электронный ресурс]. Режим доступа: <https://en.cppreference.com/>
- [4] Процессор Intel Core i7-8550U [Электронный ресурс] Режим доступа: <https://ark.intel.com/content/www/ru/ru/ark/products/122589/intel-core-i7-8550u-processor-8m-cache-up-to-4-00-ghz.html>
- [5] Windows 10 [Электронный ресурс] Режим доступа: <https://www.microsoft.com/ru/windows/get-windows-10>