Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное 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»

Лабораторная работа № 1
по дисциплине «Анализ алгоритмов»

Тема Умножение матриц

Студент Чириков Н. В.

Группа 56Б

Преподаватели Волкова Л. Л., Строганов Д. В.

Москва, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: сравнение алгоритмов умножения матриц стандартного и Винограда с оценкой трудоёмкости и времени выполнения.

Задачи лабораторной работы.

1) Описать теоретические основы стандартного алгоритма умножения матриц и алгоритма Винограда.

2) Разработать модель вычислений.

3) На основе модели вычислений оценить трудоёмкость трёх алгоритмов умножения матриц, учитывая чётный (лучший) и нечётный (худший) случаи.

4) Реализовать программное обеспечение с двумя режимами работы:

— одиночное умножение матриц, введённых пользователем;

— массированные замеры времени выполнения алгоритмов при варьируемом линейном размере квадратных матриц.

5) Выполнить замеры процессорного времени выполнения реализованных алгоритмов умножения матриц и построить графики зависимости времени выполнения от линейного размера матриц.

6) Провести сравнительный анализ стандартного алгоритма, алгоритма Винограда, оптимизированную версию алгоритма Винограда, на основе полученных результатов измерений.

1 Аналитическая часть

1.1 Алгоритмы умножения матриц

1.1.1 Стандартный алгоритм умножения матриц

Стандартный алгоритм реализует прямое определение операции матричного произведения, соответствующее основам линейной алгебры. [1] Пусть заданы матрицы A размером $n \times m$ и B размером $m \times p$. Результирующая матрица C имеет размер $n \times p$, а её элементы определяются как скалярные произведения соответствующих строк матрицы A и столбцов матрицы B. Вычисления выполняются по формуле (1.1).

$$C[i][j] = \sum_{k=1}^{m} A[i][k] \cdot B[k][j], \quad i = 1..n,\ j = 1..p \tag{1.1}$$

Реализация алгоритма предполагает использование трёх вложенных циклов: внешний цикл перебирает строки первой матрицы, средний — столбцы второй, а внутренний выполняет суммирование произведений соответствующих элементов.

1.1.2 Алгоритм Винограда

Метод Винограда представляет собой усовершенствование стандартного подхода, позволяющее уменьшить число умножений за счёт частичного предварительного вычисления. [2] Для каждой строки матрицы A и каждого столбца матрицы B заранее формируются вспомогательные значения, представленные в формулах (1.2) и (1.3).

$$row[i] = \sum_{k=1}^{\lfloor m/2 \rfloor} A[i][2k-1] \cdot A[i][2k] \tag{1.2}$$

$$col[j] = \sum_{k=1}^{\lfloor m/2 \rfloor} B[2k-1][j] \cdot B[2k][j] \tag{1.3}$$

После этого каждый элемент результирующей матрицы вычисляется по формуле (1.4).

$$C[i][j] = -row[i] - col[j] + \sum_{k=1}^{\lfloor m/2 \rfloor} (A[i][2k-1] + B[2k][j]) \cdot (A[i][2k] + B[2k-1][j]) \tag{1.4}$$

Если m нечётно, то добавляется дополнительное слагаемое (1.5):

$$C[i][j] = C[i][j] + A[i][m] \cdot B[m][j] \tag{1.5}$$

За счёт введения вспомогательных сумм количество умножений в формуле (1.4)

уменьшается по сравнению со стандартным алгоритмом.

Вывод

В данном разделе были рассмотрены два основных алгоритма умножения матриц — стандартный и алгоритм Винограда. Стандартный метод реализует прямое определение операции матричного умножения и требует тройного цикла вычислений, что приводит к высокой трудоёмкости при увеличении размерности матриц.

Алгоритм Винограда представляет собой оптимизацию стандартного подхода за счёт предварительного вычисления частичных произведений строк и столбцов. Это позволяет сократить количество умножений в основной части алгоритма, что особенно заметно при больших размерах матриц. Таким образом, метод Винограда достигает меньшей константы в выражении трудоёмкости при сохранении той же асимптотической сложности $O(N^3)$.

2 Конструкторская часть

2.1 Схемы алгоритмов умножения матриц

На рисунках 2.1 — 2.7 приведены схемы алгоритмов умножения матриц.

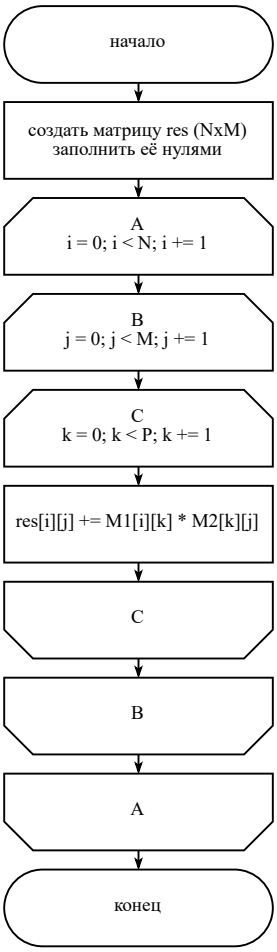

Рисунок 2.1 — Схема стандартного алгоритма умножения матриц

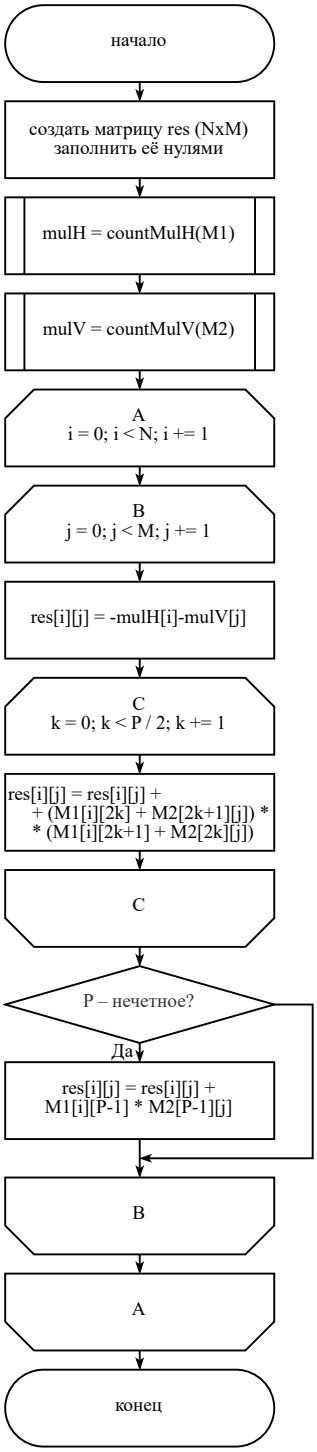

Рисунок 2.2 — Схема алгоритма Винограда умножения матриц

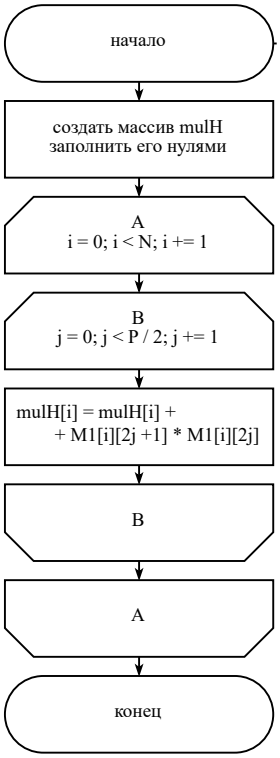

Рисунок 2.3 — Схема алгоритма вычисления сумм произведений пар соседних элементов строк матрицы

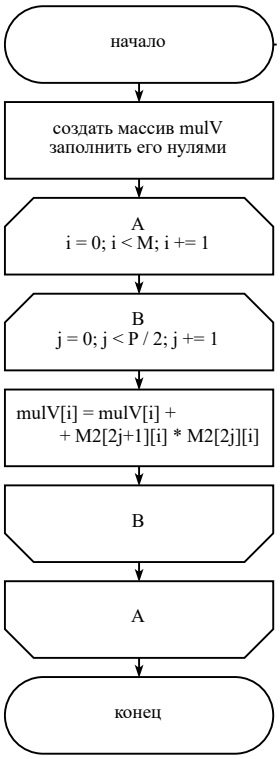

Рисунок 2.4 — Схема алгоритма вычисления сумм произведений пар соседних элементов столбцов матрицы

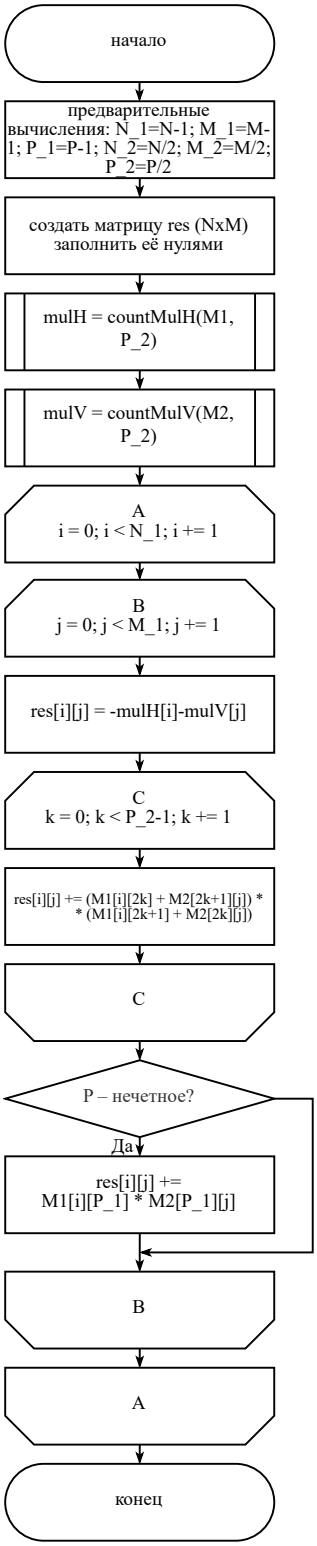

Рисунок 2.5 — Схема оптимизированного алгоритма Винограда умножения матриц

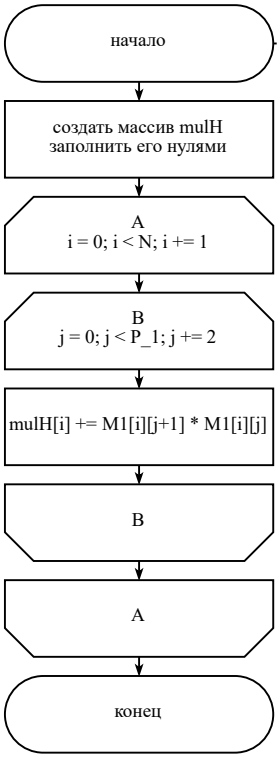

Рисунок 2.6 — Схема оптимизированного алгоритма вычисления сумм произведений пар соседних элементов строк матрицы

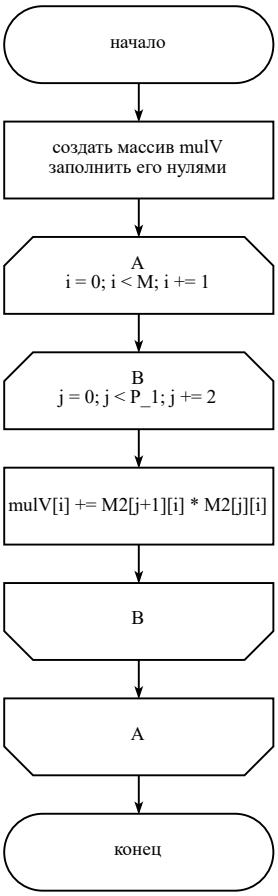

Рисунок 2.7 — Схема оптимизированного алгоритма вычисления сумм произведений пар соседних элементов столбцов матрицы

2.2 Модель вычислений

Модель вычислений для оценки трудоёмкости алгоритмов:

— операции с трудоёмкостью 1: =, +, -, +=, -=, <, >, ==, !=, >=, <=, [], «, », ++, –;

— операции с трудоёмкостью 2: *, /, //, %, *=, /=, //=;

— трудоёмкость оператора выбора `if` рассчитывается по формуле(2.1);

$$f_{if} = f_{\text{условия}} + \begin{cases} f_A, & \text{если условие выполняется;} \\ f_B, & \text{иначе} \end{cases} \tag{2.1}$$

— трудоёмкость цикла `for` рассчитывается по формуле(2.2);

$$f_{\text{for}} = f_{\text{инициализации}} + f_{\text{сравнения}} + N(f_{\text{тела}} + f_{\text{инкремента}} + f_{\text{сравнения}}) \tag{2.2}$$

— трудоёмкость вызова функции принимается равной 0.

Принятая модель отражает реальное распределение затрат по операциям и позволяет более точно сравнивать трудоёмкость различных алгоритмов.

2.3 Трудоёмкость стандартного алгоритма

Трудоёмкость вложенных циклов: внутреннего цикла по k, среднего цикла по j, и внешнего цикла по i. Итогом будет полная трудоёмкость алгоритма.

$$f_C = 2 + 11P \tag{2.3}$$

$$f_B = 2 + M(2 + f_C) = 11MP + 4M + 2 \tag{2.4}$$

$$f_A = 2 + N(2 + f_B) = 11NMP + 4NM + 4N + 2 \tag{2.5}$$

$$f_{\text{standart}} = f_A \tag{2.6}$$

Полная трудоёмкость стандартного алгоритма имеет вид:

$$f_{\text{standart}} = 11NMP + 4NM + 4N + 2, \tag{2.7}$$

где главный член $11NMP$ определяет асимптотическую сложность $O(NMP)$.

Принятая модель отражает реальное распределение затрат по операциям и позволяет более точно сравнивать трудоёмкость различных алгоритмов.

2.4 Алгоритм Винограда

Алгоритм Винограда предусматривает выполнение дополнительного подготовительного этапа, на котором для каждой строки первой матрицы и каждого столбца второй матрицы заранее вычисляются вспомогательные значения. Эти данные используются при последующем перемножении, что позволяет сократить количество умножений в основном цикле. После этапа предварительных вычислений выполняется основной цикл, формирующий элементы результирующей матрицы на основе уже готовых промежуточных сумм. Такой подход распределяет вычислительную нагрузку между фазами алгоритма и обеспечивает частичное снижение трудоёмкости операции умножения. В рамках модели вычислений трудоёмкость каждого из этапов — инициализации, подготовки строк и столбцов, а также основного цикла — может быть представлена следующими выражениями:

$$f_{init} = N + M, \tag{2.8}$$

$$f_{mulH} = 9.5PN + 6N + 2, \tag{2.9}$$

$$f_{mulV} = 9.5MP + 6M + 2, \tag{2.10}$$

$$f^{\wedge}_{main} = 16NMP + 16NM + 4N + 2, \tag{2.11}$$

$$f^{\vee}_{main} = 16NMP + 30NM + 4N + 2. \tag{2.12}$$

Полная трудоёмкость алгоритма Винограда для двух случаев вычислений имеет вид:

$$f^{\wedge}_{Winograd} = 16NMP + 16NM + 9.5MP + 9.5PN + 11N + 7M + 6, \tag{2.13}$$

$$f^{\vee}_{Winograd} = 16NMP + 30NM + 9.5MP + 9.5PN + 11N + 7M + 6, \tag{2.14}$$

где главный член $16NMP$ определяет асимптотическую сложность $O(NMP)$.

Несмотря на усложнение структуры программы, уменьшение числа умножений компенсирует рост количества сложений и обращений к памяти.

2.5 Оптимизированный алгоритм Винограда

Оптимизированная версия алгоритма Винограда реализует предварительные вычисления с применением буферизации, что позволяет эффективно хранить и повторно ис-

пользовать промежуточные результаты. Благодаря этому снижается количество выполняемых операций и уменьшается число обращений к памяти в основных циклах. Такое улучшение направлено на сокращение вычислительных затрат и повышение общей производительности метода. В соответствии с принятой моделью вычислений, трудоёмкость отдельных компонентов алгоритма — этапа инициализации, расчёта вспомогательных сумм и основного цикла умножения — выражается следующими зависимостями:

$$f_{init} = N + M, \tag{2.15}$$

$$f_{optMulH} = 6PN + 5N + 2, \tag{2.16}$$

$$f_{optMulV} = 6MP + 5M + 2, \tag{2.17}$$

$$f^{\wedge}_{optMain} = 10NMP + 14NM + 4N + 2, \tag{2.18}$$

$$f^{\vee}_{optMain} = 10NMP + 25NM + 4N + 2. \tag{2.19}$$

Полная трудоёмкость оптимизированного алгоритма Винограда имеет вид:

$$f^{\wedge}_{optWinograd} = 10NMP + 14NM + 6MP + 6PN + 10N + 6M + 6, \tag{2.20}$$

$$f^{\vee}_{optWinograd} = 10NMP + 25NM + 6MP + 6PN + 10N + 6M + 6, \tag{2.21}$$

где главный член $10NMP$ определяет асимптотическую сложность $O(NMP)$.

Снижение трудоёмкости достигается за счёт перераспределения вычислений и повторного использования ранее рассчитанных выражений.

Вывод

Разработаны схемы стандартного алгоритма умножения матриц, алгоритма Винограда и его оптимизированного варианта с буферизацией. Для каждого алгоритма введена модель вычислений и проведена оценка трудоёмкости.

Анализ показал, что все три алгоритма имеют асимптотическую сложность $O(NMP)$, но отличаются константными множителями. Стандартный алгоритм обладает наибольшей трудоёмкостью, алгоритм Винограда уменьшает количество операций за счёт предварительных вычислений, оптимизированный алгоритм Винограда с буферизацией дополнительно снижает трудоёмкость за счёт повторного использования ранее вычисленных значений.

3 Технологическая часть

3.1 Выбор языка и среды программирования

Для реализации программного обеспечения, выполняющего умножение матриц тремя алгоритмами, выбраны следующие инструменты и технические средства:

— операционная система: Windows 10 [4];

— среда разработки: Visual Studio Code [5];

— компилятор: `g++` версии 15.1.0 [6];

— центральный процессор (CPU): Intel Core i5-11400H [7];

— оперативная память (RAM): 16 ГБ [8];

— язык программирования: C++17 [9];

— библиотеки:

  – стандартная библиотека STL [10] — контейнеры `vector`, ввод-вывод (`iomanip`, `iostream`);

  – библиотека `chrono` [11] — измерение времени выполнения алгоритмов;

  – библиотека `fstream` [12] — запись результатов замеров производительности в CSV-файлы;

  – генератор случайных чисел `<random>` [13] — заполнение матриц случайными элементами.

Для реализации выбран язык программирования C++. Это обусловлено его высокой скоростью работы, возможностью прямого управления памятью и наличием средств для точного измерения времени выполнения программ. Эти свойства делают C++ удобным инструментом для проведения исследований и сравнения эффективности алгоритмов.

3.2 Установка и настройка среды

Компиляция проекта выполняется с помощью команды g++, что показано в листинге 3.1.

Листинг 3.1 — Компиляция проекта

```
1  g++ main.cpp mult_algos.cpp matrix_utils.cpp -std=c++17 -o
    matrix_mult.exe
```

Для запуска программы используется команда, представленная в листинге 3.2.

Листинг 3.2 — Запуск программы

```
1  ./matrix_mult.exe
```

При запуске пользователь выбирает один из двух режимов:

— ручной ввод матриц и вычисление результата;

— автоматическое сравнение времени работы трёх алгоритмов при разных размерах матриц.

3.3 Реализация алгоритмов

В данном разделе представлены реализации трёх алгоритмов умножения матриц: стандартного, алгоритма Винограда и его оптимизированной версии. Каждый алгоритм оформлен в виде отдельной функции на языке C++.

3.3.1 Стандартный алгоритм

Стандартный алгоритм умножения матриц основан на трёх вложенных циклах. Каждый элемент результирующей матрицы вычисляется как скалярное произведение строки первой матрицы и столбца второй. Код представлен в листинге 3.3.

Листинг 3.3 — Стандартный алгоритм умножения матриц

```
1  Matrix mult_standard(const Matrix &A, const Matrix &B)
2  {
3      int n = A.size(), m = B[0].size(), k = B.size();
4      Matrix C(n, std::vector<int>(m, 0));
5      for (int i = 0; i < n; ++i)
6          for (int j = 0; j < m; ++j)
7              for (int t = 0; t < k; ++t)
8                  C[i][j] += A[i][t] * B[t][j];
9      return C;
10 }
```

3.3.2 Алгоритм Винограда

Алгоритм Винограда представляет собой усовершенствованный вариант стандартного способа умножения матриц, направленный на уменьшение количества арифметических операций. Основная идея данного метода заключается в использовании предварительных вычислений, позволяющих частично сократить число умножений, выполняемых в основном цикле.

Перед началом основного этапа производится вычисление вспомогательных массивов для строк первой матрицы и для столбцов второй матрицы. Эти массивы содержат результаты промежуточных произведений, которые используются при формировании итоговых элементов результирующей матрицы. Такой подход позволяет снизить количество умножений за счёт замены части из них на операции сложения, что повышает общую эффективность алгоритма.

Реализация алгоритма на языке C++ представлена в листинге 3.4. Программная

реализация сохраняет структуру базового метода, но дополнена этапами вычисления и использования вспомогательных массивов, что позволяет более рационально использовать ресурсы процессора и уменьшить время выполнения операции умножения.

Листинг 3.4 — Алгоритм Винограда умножения матриц

```
1  Matrix mult_vinograd(const Matrix &A, const Matrix &B)
2  {
3      int n = A.size(), m = B[0].size(), k = B.size();
4      Matrix C(n, std::vector<int>(m, 0));
5      std::vector<int> row_factor(n, 0), col_factor(m, 0);
6      for (int i = 0; i < n; ++i)
7          for (int t = 0; t < k/2; ++t)
8              row_factor[i] += A[i][2*t] * A[i][2*t+1];
9      for (int j = 0; j < m; ++j)
10         for (int t = 0; t < k/2; ++t)
11             col_factor[j] += B[2*t][j] * B[2*t+1][j];
12     for (int i = 0; i < n; ++i)
13         for (int j = 0; j < m; ++j)
14         {
15             C[i][j] = -row_factor[i] - col_factor[j];
16             for (int t = 0; t < k/2; ++t)
17                 C[i][j] += (A[i][2*t] + B[2*t+1][j]) *
18                            (A[i][2*t+1] + B[2*t][j]);
19         }
20     if (k % 2 == 1)
21         for (int i = 0; i < n; ++i)
22             for (int j = 0; j < m; ++j)
23                 C[i][j] += A[i][k-1] * B[k-1][j];
24
25     return C;
26 }
```

3.3.3 Оптимизированный алгоритм Винограда

Оптимизированный вариант алгоритма Винограда содержит ряд дополнительных усовершенствований, направленных на повышение эффективности вычислений. Основные изменения включают вынесение повторяющихся выражений в отдельные переменные, использование локальных аккумуляторов и уменьшение числа обращений к памяти, что снижает вычислительные затраты. Благодаря этим доработкам достигается более высокая скорость выполнения программы при сохранении точности результатов. Фрагмент реализации алгоритма представлен в листинге 3.5.

Листинг 3.5 — Оптимизированный алгоритм Винограда

```
Matrix mult_vinograd_opt(const Matrix &A, const Matrix &B)
{
    int n = A.size(), m = B[0].size(), k = B.size();
    Matrix C(n, std::vector<int>(m, 0));
    std::vector<int> row_factor(n, 0), col_factor(m, 0);
    int k2 = k / 2;
    for (int i = 0; i < n; ++i)
    {
        int sum = 0;
        for (int t = 0; t < k2; ++t)
            sum += A[i][2*t] * A[i][2*t+1];
        row_factor[i] = sum;
    }
    for (int j = 0; j < m; ++j)
    {
        int sum = 0;
        for (int t = 0; t < k2; ++t)
            sum += B[2*t][j] * B[2*t+1][j];
        col_factor[j] = sum;
    }
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
        {
            int temp = -row_factor[i] - col_factor[j];
            for (int t = 0; t < k2; ++t)
                temp += (A[i][2*t] + B[2*t+1][j]) *
                        (A[i][2*t+1] + B[2*t][j]);
            C[i][j] = temp;
        }
    if (k % 2 == 1)
    {
        int last = k-1;
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < m; ++j)
                C[i][j] += A[i][last] * B[last][j];
    }
    return C;
}
```

3.4 Результаты работы

Разработанная программа поддерживает два режима работы: режим ручного ввода данных и режим автоматического измерения времени выполнения алгоритмов.

В режиме ручного ввода пользователь задаёт размеры матриц и их элементы с клавиатуры. Программа выполняет умножение матриц с использованием трёх методов: стандартного алгоритма, алгоритма Винограда и его оптимизированной версии. Результаты вычислений выводятся в консоль в виде матриц. Пример работы программы в данном режиме представлен на рисунке 3.1.

```
PS C:\Users\k01ya\Desktop\AA\1\code> .\matrix_mult.exe
Выберите режим:
1. Ручной ввод
2. Автоматический замер времени
> 2

===Лучший случай для алгоритма Винограда (чётные размеры)===

Размер 100x100, Standard: 29.1052 ms
Размер 100x100, Vinograd: 29.8503 ms
Размер 100x100, Vinograd_Opt: 25.0559 ms

Размер 200x200, Standard: 199.78 ms
Размер 200x200, Vinograd: 165.208 ms
Размер 200x200, Vinograd_Opt: 114.429 ms

Размер 300x300, Standard: 532.564 ms
Размер 300x300, Vinograd: 449.934 ms
Размер 300x300, Vinograd_Opt: 360.952 ms

Размер 400x400, Standard: 1203.94 ms
Размер 400x400, Vinograd: 1169.89 ms
Размер 400x400, Vinograd_Opt: 835.045 ms

Размер 500x500, Standard: 2407.89 ms
Размер 500x500, Vinograd: 2304.95 ms
Размер 500x500, Vinograd_Opt: 1813.21 ms

===Худший случай для алгоритма Винограда (нечетные размеры)===

Размер 101x101, Standard: 20.5505 ms
Размер 101x101, Vinograd: 20.0238 ms
Размер 101x101, Vinograd_Opt: 16.3164 ms

Размер 201x201, Standard: 150.896 ms
Размер 201x201, Vinograd: 128.038 ms
Размер 201x201, Vinograd_Opt: 99.3982 ms

Размер 301x301, Standard: 497.039 ms
Размер 301x301, Vinograd: 525.12 ms
Размер 301x301, Vinograd_Opt: 425.57 ms

Размер 401x401, Standard: 1240.19 ms
Размер 401x401, Vinograd: 1211.07 ms
Размер 401x401, Vinograd_Opt: 959.859 ms

Размер 501x501, Standard: 2458.19 ms
Размер 501x501, Vinograd: 2317.12 ms
Размер 501x501, Vinograd_Opt: 1801.87 ms
```

Рисунок 3.1 — Пример работы программы в режиме ручного ввода матриц

Во втором режиме осуществляется автоматический замер времени выполнения алгоритмов.Пример вывода программы в этом режиме показан на рисунке 3.2.

```
PS C:\Users\k01ya\Desktop\AA\1\code> .\matrix_mult.exe
Выберите режим:
1. Ручной ввод
2. Автоматический замер времени
> 1

Введите размеры матриц: n1 m1 m2 (для умножения n1×m1 * m1×m2): 4 2 5

Введите матрицу 4x2 построчно:
3 4
7 6
4 3
2 6

Введите матрицу 2x5 построчно:
5 4 3 8 3
5 4 3 2 5

Результат (Standard):
35 28 21 32 29
65 52 39 68 51
35 28 21 38 27
40 32 24 28 36

Результат (Vinograd):
35 28 21 32 29
65 52 39 68 51
35 28 21 38 27
40 32 24 28 36

Результат (Vinograd_Opt):
35 28 21 32 29
65 52 39 68 51
35 28 21 38 27
40 32 24 28 36
```

Рисунок 3.2 — Пример работы программы в режиме автоматического замера времени

Полученные данные показывают устойчивую тенденцию к ускорению при использовании оптимизированной версии, что подтверждает эффективность предложенных улучшений.

Вывод

В ходе работы были выбраны инструменты разработки и обоснован выбор языка программирования C++. Программа реализована по модульному принципу, что позволяет разделить логику обработки матриц и реализацию алгоритмов умножения.

Программное обеспечение обеспечивает два режима работы: ручной ввод матриц с выводом результата и автоматическое измерение времени выполнения алгоритмов. Полученные реализации позволяют провести сравнение производительности стандартного алгоритма, алгоритма Винограда и его оптимизированной версии, а также подтвердить эффективность внесённых оптимизаций.

4 Исследовательская часть

4.1 Цель исследования

Целью исследования является сравнение эффективности трёх алгоритмов умножения матриц: стандартного, алгоритма Винограда и оптимизированного алгоритма Винограда. Для анализа производительности выполнены замеры времени работы алгоритмов при различных размерах матриц в двух режимах:

— худший случай — матрицы нечётного размера;

— лучший случай — матрицы чётного размера.

4.2 Худший случай

График зависимости времени работы алгоритмов от размера матрицы в худшем случае приведён на рисунке 4.1.

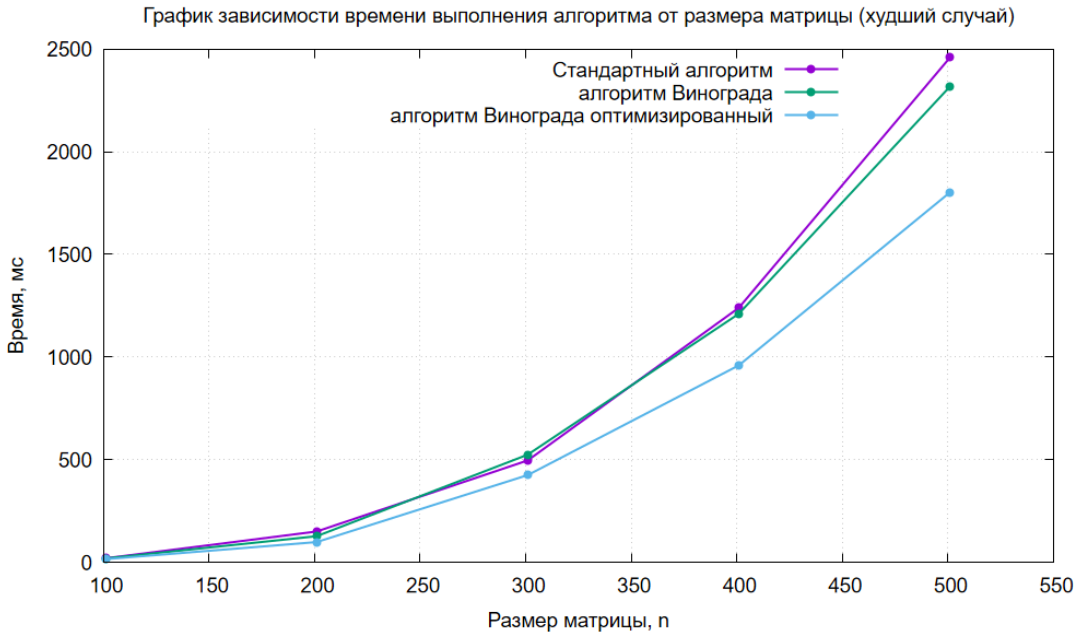

Рисунок 4.1 — Сравнение алгоритмов умножения матриц (худший случай)

В таблице 4.1 приведены результаты замеров времени работы трёх алгоритмов при умножении квадратных матриц нечётного размера.

Таблица 4.1 — Результаты замеров времени (в миллисекундах) для худшего случая

Размер n	Стандартный алгоритм	Алгоритм Винограда	Оптимизированный Виноград
101	15.06	13.78	10.93
201	116.48	107.50	85.38
301	387.57	355.84	282.34
401	930.25	851.76	670.05
501	1800.02	1636.03	1299.95

4.3 Лучший случай

В таблице 4.2 приведены результаты замеров времени работы трёх алгоритмов при умножении квадратных матриц чётного размера.

График зависимости времени работы алгоритмов от размера матрицы в лучшем случае приведён на рисунке 4.2.

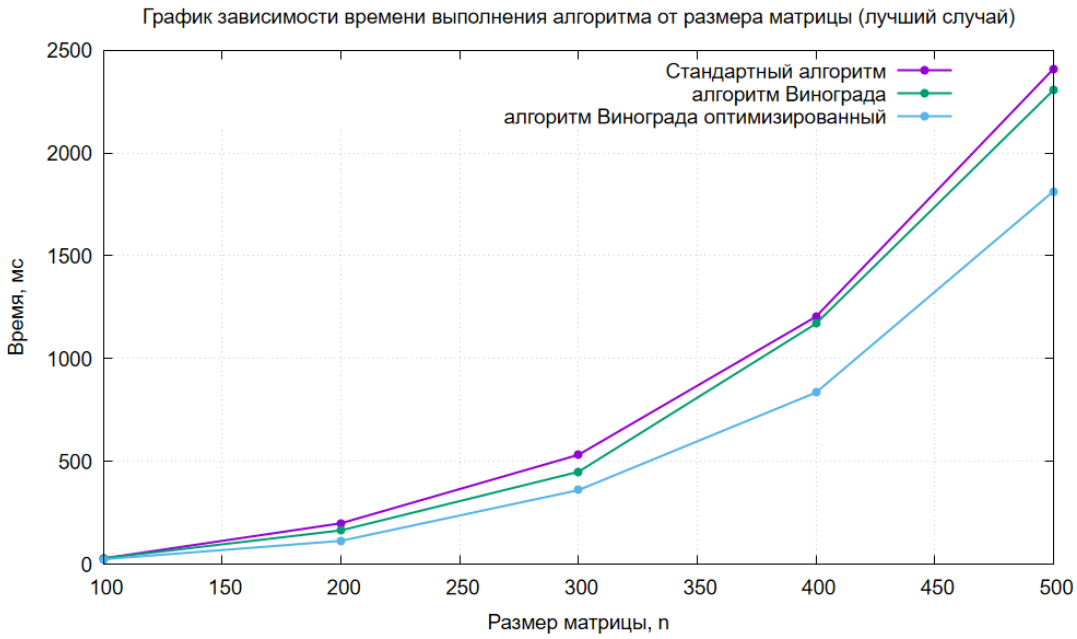

Рисунок 4.2 — Сравнение алгоритмов умножения матриц (лучший случай)

Таблица 4.2 — Результаты замеров времени (в миллисекундах) для лучшего случая

Размер n	Стандартный алгоритм	Алгоритм Винограда	Оптимизированный Виноград
100	14.25	13.41	10.34
200	114.41	104.30	82.89
300	386.43	354.84	281.87
400	920.57	850.07	669.74
500	1818.12	1637.69	1293.03

Для чётных размеров матриц эффективность алгоритма Винограда достигает максимума благодаря полному использованию парных индексов.

Расхождения между теоретическими и данными, полученными программным путём находятся в допустимых пределах и объясняются аппаратными особенностями тестовой системы.

Вывод

Результаты исследований подтверждают сделанные теоретические выводы:

— наименьшее время выполнения демонстрирует оптимизированный алгоритм Винограда;

— базовая версия метода Винограда в среднем быстрее классического способа, но уступает оптимизированной;

— наблюдается зависимость времени работы от чётности размеров матриц: при нечётных размерах производительность незначительно снижается.

ЗАКЛЮЧЕНИЕ

В ходе работы были рассмотрены три метода умножения матриц: стандартный алгоритм, алгоритм Винограда и его оптимизированная модификация. Для каждого метода выполнен анализ трудоёмкости на основе модели вычислений, а также реализованы программные версии для проверки их эффективности на практике.

Теоретический анализ показал, что все рассмотренные алгоритмы имеют одинаковую асимптотическую сложность $O(N^3)$, однако различаются по количеству выполняемых базовых операций. Стандартный алгоритм является исходным вариантом для сравнения, алгоритм Винограда уменьшает количество умножений за счёт предварительных вычислений, а оптимизированная версия дополнительно снижает трудоёмкость благодаря буферизации и повторному использованию промежуточных данных.

Проведённые вычислительные исследования подтвердили результаты теоретического анализа. На тестовых примерах с различными размерами матриц алгоритм Винограда показал преимущество по скорости перед стандартным методом, а оптимизированный вариант продемонстрировал наилучшие показатели времени выполнения. Также отмечено влияние чётности размеров матриц: при нечётных значениях время работы увеличивается, что соответствует теоретическим оценкам.

Цель работы — исследование и сравнение эффективности алгоритмов умножения матриц — достигнута. В ходе выполнения работы были решены следующие задачи:

— были рассмотрены теоретические основы стандартного алгоритма и алгоритма Винограда;

— была разработана модель вычислений для оценки трудоёмкости;

— на основе модели вычислений был выполнен анализ трудоёмкости алгоритмов в лучшем и худшем случаях;

— было реализовано программное обеспечение, включающее два режима работы:

  – режим ручного ввода матриц и вывода результата умножения;

  – режим автоматического измерения времени выполнения алгоритмов при изменении размера матриц;

— были произведены измерения процессорного времени выполнения алгоритмов и построены графики зависимости времени от размера матриц;

— было выполнено сравнение эффективности стандартного алгоритма, алгоритма Винограда и его оптимизированной версии на основе полученных данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Самарский А. А., Гулин А. В. Численные методы. – М.: Наука, 1989. – 432 с.

2. Анисимов Н. С., Строганов Ю. В. Реализация алгоритма умножения матриц по Винограду на языке Haskell. – Новые информационные технологии в автоматизированных системах, 2018. – № 21. – С. 390–395.

3. Погорелов Д. А. Оптимизация классического алгоритма Винограда для перемножения матриц. – Информационные технологии и вычислительные системы, 2019. – № 3. – С. 45–52.

4. Microsoft. Официальный сайт Windows 10 [Электронный ресурс]. — Режим доступа: `https://www.microsoft.com/ru-ru/software-download/windows10` (дата обращения: 04.10.2025).

5. Visual Studio Code. Официальный сайт и документация [Электронный ресурс]. — Режим доступа: `https://code.visualstudio.com/` (дата обращения: 04.10.2025).

6. GNU Project. Компилятор GNU C++ (g++) версии 15.1.0 [Электронный ресурс]. — Режим доступа: `https://gcc.gnu.org/` (дата обращения: 04.10.2025).

7. Intel. Процессор Intel Core i5-11400H: техническая спецификация [Электронный ресурс]. — Режим доступа: `https://ark.intel.com/content/www/ru/ru/ark/products/212285/intel-core-i511400h-processor-12m-cache-up-to-4-50-ghz.html` (дата обращения: 04.10.2025).

8. Kingston Technology. Обзор модулей оперативной памяти DDR4 16 ГБ [Электронный ресурс]. — Режим доступа: `https://www.kingston.com/ru/memory/ddr4` (дата обращения: 04.10.2025).

9. ISO/IEC. Язык программирования C++17. Международный стандарт ISO/IEC 14882:2017 [Электронный ресурс]. — Режим доступа: `https://isocpp.org/std/the-standard` (дата обращения: 04.10.2025).

10. Cplusplus.com. Стандартная библиотека STL [Электронный ресурс]. — Режим доступа: `https://cplusplus.com/reference/stl/` (дата обращения: 04.10.2025).

11. Cplusplus.com. Справка по <chrono> [Электронный ресурс]. — Режим доступа: `https://cplusplus.com/reference/chrono/` (дата обращения: 04.10.2025).

12. Cplusplus.com. Справка по std::fstream [Электронный ресурс]. — Режим доступа: https://cplusplus.com/reference/fstream/fstream/ (дата обращения: 04.10.2025).

13. Cplusplus.com. Справка по <random> [Электронный ресурс]. — Режим доступа: https://cplusplus.com/reference/random/ (дата обращения: 04.10.2025).