



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

Название: Алгоритмы умножения матриц

Дисциплина: Анализ алгоритмов

Студент ИУ7-55Б
(Группа)

(Подпись, дата)

Д.В. Сусликов
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

Л.Л. Волкова
(И.О. Фамилия)

Москва, 2020 г.

Оглавление

Введение	4
1 Аналитическая часть	5
1.1 Стандартный алгоритм умножения матриц	5
1.2 Алгоритм Винограда	6
Вывод	6
2 Конструкторская часть	7
2.1 Схемы алгоритмов	7
2.2 Отличие модифицированного алгоритма Винограда от обычного . .	11
2.3 Трудоемкость алгоритмов	11
2.3.1 Трудоемкость общей первичной проверки	12
2.3.2 Трудоемкость стандартного алгоритма умножения	12
2.3.3 Трудоемкость алгоритма Винограда	12
2.3.4 Трудоемкость модифицированного алгоритма Винограда . .	12
Вывод	13
3 Технологическая часть	14
3.1 Общие требования	14
3.2 Средства реализации	14
3.3 Реализация алгоритмов	15
Вывод	19
4 Экспериментальная часть	20
4.1 Примеры работы программы	20
4.2 Анализ времени работы алгоритмов	22
Вывод	24

Заключение	25
Литература	26

Введение

Цель данной лабораторной работы - изучение алгоритмов умножения матрицы, получение навыков улучшения алгоритмов и подсчёта их трудоёмкости. В ходе работы будут рассмотрены 3 алгоритма:

- 1) стандартный алгоритм умножения матриц;
- 2) алгоритм Винограда;
- 3) улучшенный алгоритм Винограда.

В лабораторной работе требуется:

- 1) изучить алгоритмы умножения матриц;
- 2) оптимизировать алгоритм Винограда;
- 3) дать теоритическую оценку стандартного алгоритма умножения матриц, алгоритма Винограда и улучшенного алгоритма Винограда;
- 4) реализовать три алгоритма умножения матриц;
- 5) сравнить алгоритмы умножения матриц.

1 | Аналитическая часть

В данном разделе представлены математические описания алгоритмов умножения матриц.

1.1 Стандартный алгоритм умножения матриц

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.

Умножение матриц — одна из основных операций над матрицами. Матрица, получаемая в результате операции умножения, называется произведением матриц.

Пусть даны две прямоугольные матрицы A и B размеров $[m * n]$ и $[n * k]$ соответственно. В результате произведения матриц A и B получим матрицу C размера $[m * k]$.

$$c_{i,j} = \sum_{r=1}^m a_{ir} b_{rj} \quad (i = 1, 2, \dots, l; j = 1, 2, \dots, n) \quad (1.1)$$

Операция умножения двух матриц выполняема только в том случае, если число столбцов в первой матрице совпадает с числом строк во второй, то эти две матрицы можно перемножить.

1.2 Алгоритм Винограда

Если посмотреть на результат умножения двух матриц, то видно, что каждый элемент в нем представляет собой скалярное произведение соответствующих строки и столбца исходных матриц. Можно заметить также, что такое умножение допускает предварительную обработку, позволяющую часть работы выполнить заранее. [1]

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:

$$V * W = v_1 w_1 + v_2 w_2 + v_3 w_3 + v_4 w_4 \quad (1.2)$$

Это равенство можно переписать в виде:

$$V * W = (v_1 + w_2)(v_2 + w_1) + (v_3 + w_4)(v_4 + w_3) - v_1 v_2 - v_3 v_4 - w_1 w_2 - w_3 w_4 \quad (1.3)$$

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. На практике это означает, что над предварительно обработанными элементами придется выполнять лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения.

Вывод

Таким образом, получилось узнать, что из себя представляет умножение матриц. Были описаны стандартный алгоритм умножения и Винограда, объяснено преимущество второго над первым.

2 | Конструкторская часть

В данном разделе представлены схемы алгоритмов и дана оценка их трудоемкости.

2.1 Схемы алгоритмов

Ниже на Рисунке 1 представлена схема стандартного алгоритма умножения матриц.

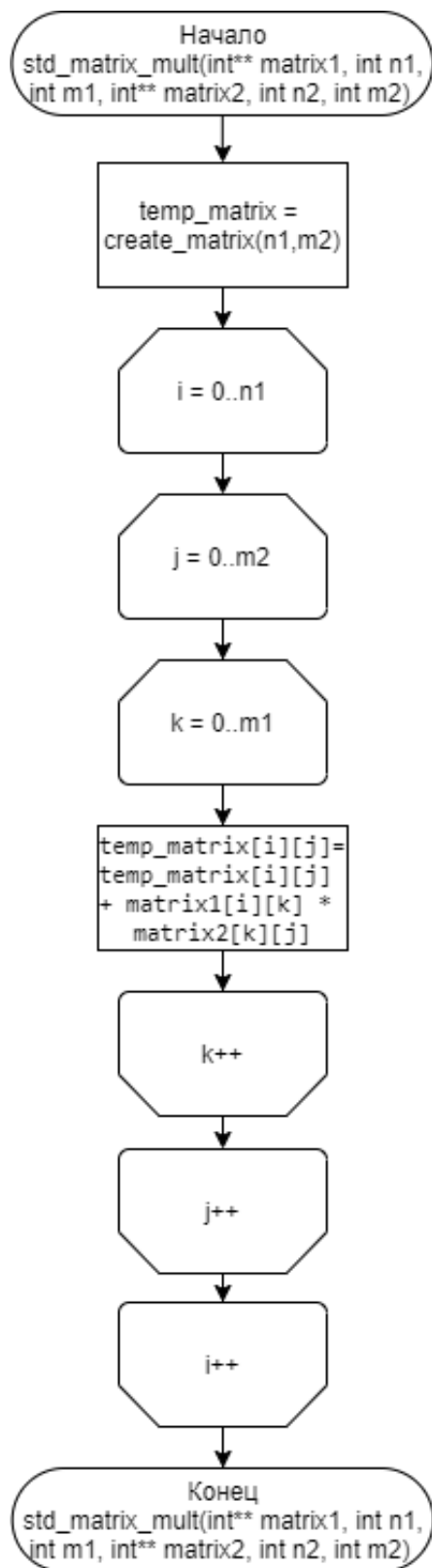


Рисунок 1 - Схема стандартного алгоритма умножения матриц

Далее на Рисунке 2 можно увидеть схему алгоритма Винограда.

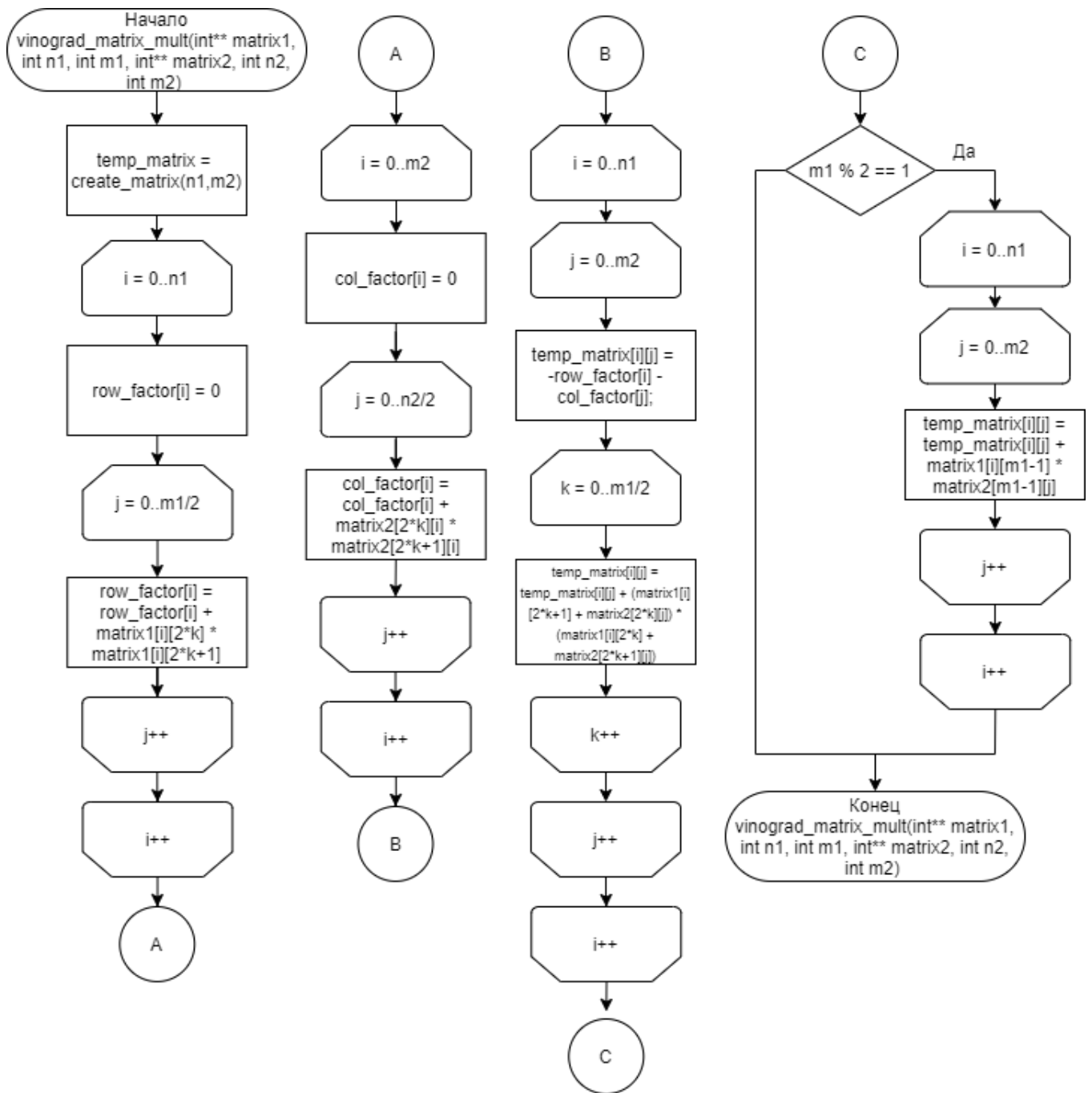


Рисунок 2 - Схема алгоритма Винограда

Ниже на Рисунке 3 показана схема модифицированного алгоритма Винограда.

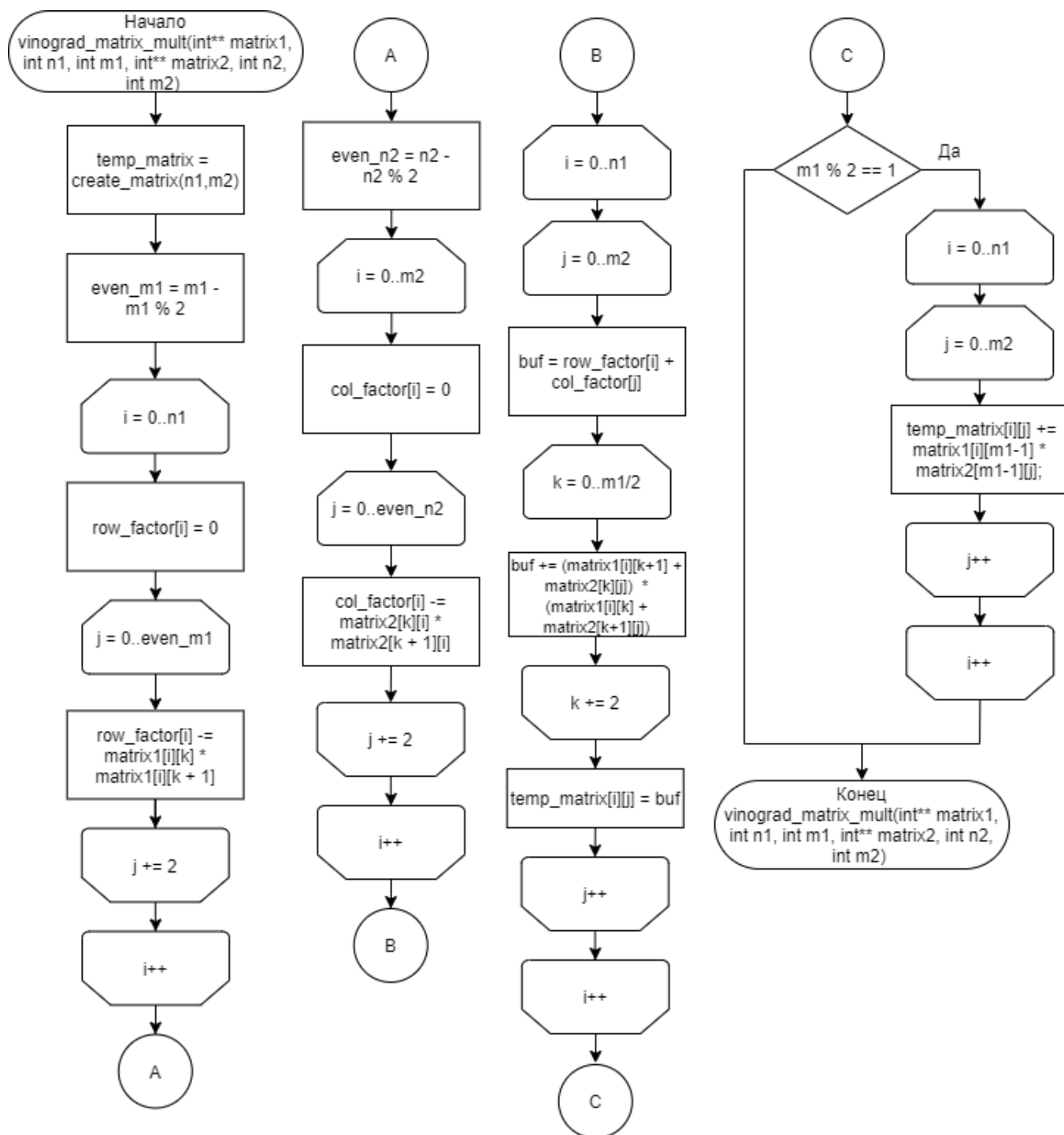


Рисунок 3 - Схема модифицированного алгоритма Винограда

2.2 Отличие модифицированного алгоритма Винограда от обычного

Улучшения:

- 1) нет нужды внутри цикла каждый раз пересчитывать $m1/2$ и $n2/2$, так как заранее посчитано $even_m1 = m1 - m\%2$ и $even_n2 = n2 - n\%2$;
- 2) так как есть $even_m1$ и $even_n2$, то индекс будет изменяться на $2 - j + = 2$, следовательно нет нужды умножать на 2, как в обычном алгоритме, где индекс матрицы считался $matrix1[i][2 * j]$;
- 3) в обычном алгоритме значение $-row_factor[i] - col_factor[j]$ присваивается элементу матрицы умножения, и в следующем цикле каждый раз происходит обращение к нему. В модифицированном алгоритме Винограда это значение присваивается буферу, и лишь потом после выполнения цикла записывается в матрицу-результат;
- 4) подставлены $+ =$ и $- =$, где возможно.

2.3 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

- 1) Операции, чья стоимость 1: $=, +, *, \simeq, <, >, \geq, \leq, ==, !=, [], + =, - =, * =, / =, ++, --$;

- 2) стоимость цикла:

$$f_{for} = f_{init} + f_{comp} + M(f_{body} + f_{increment} + f_{comp})$$

Пример: $for(i = 0, i < M; i++) * body * /$

Результат: $2 + M(2 + f_{body})$;

- 3) стоимость условного оператора

Пусть goto (переход к одной из ветвей) стоит 0, тогда

$$f_f = \begin{cases} \min(f_A, f_B), & \text{лучший случай} \\ \max(f_A, f_B), & \text{худший случай} \end{cases}$$

Оценим трудоемкость алгоритмов.

2.3.1 Трудоемкость общей первичной проверки

Трудоемкость проверки $if((m1! = n2)||n1 == 0||n2 == 0) return; - 5$.

2.3.2 Трудоемкость стандартного алгоритма умножения

Расчёт:

$$2 + n_1(2 + 2 + m_2(2 + 3 + 2 + m_1(2 + 11))) = 13n_1m_1m_2 + 7n_1m_2 + 4n_1 + 2$$

2.3.3 Трудоемкость алгоритма Винограда

Расчёт:

$$\text{Первый цикл: } 2 + n_1(2 + 2 + 3 + \frac{m_1}{2}(3 + 12)) = 2 + n_1(7 + \frac{15m_1}{2}) = \frac{15}{2}n_1m_1 + 7n_1 + 2$$

$$\text{Второй цикл: аналогично, } \frac{15}{2}m_2n_2 + 7m_2 + 2$$

$$\text{Третий цикл: } 2 + n_1(2 + 2 + m_2(2 + 7 + 3 + \frac{m_1}{2}(3 + 23))) = 13n_1m_1m_2 + 12n_1m_2 + 4n_1 + 2$$

$$\text{Условие: } \begin{bmatrix} 2 & , \text{ невыполнение} \\ 15n_1m_2 + 4n_1 + 4 & , \text{ выполнение} \end{bmatrix}$$

$$\text{Результат: } 13n_1m_1m_2 + \frac{15}{2}n_1m_1 + \frac{15}{2}m_2n_2 + 12n_1m_2 + 7n_1 + 7m_2 + 4n_1 + 6 + \begin{bmatrix} 2 & , \text{ невыполнение} \\ 15n_1m_2 + 4n_1 + 4 & , \text{ выполнение} \end{bmatrix}$$

2.3.4 Трудоемкость модифицированного алгоритма Винограда

Расчёт:

$$\text{Первый цикл: } 3 + 2 + n_1(2 + 2 + 2 + \frac{m_1}{2}(2 + 8)) = 5n_1m_1 + 6n_1 + 5$$

(3 - на переменную вместо $m_1/2$)

$$\text{Второй цикл: аналогично, } 5m_2n_2 + 6m_2 + 5$$

Третий цикл: $2 + n_1(2 + 2 + m_2(2 + 4 + 2 + \frac{m_1}{2}(2 + 14) + 3)) = 8n_1m_1m_2 + 11n_1m_2 + 4n_1 + 2$

Условие: $\left[\begin{array}{ll} 2 & , \text{ невыполнение} \\ 12n_1m_2 + 4n_1 + 4 & , \text{ выполнение} \end{array} \right]$

Результат: $8n_1m_1m_2 + 5n_1m_1 + 5m_2n_2 + 11n_1m_2 + 6n_1 + 6m_2 + 12 +$
 $\left[\begin{array}{ll} 2 & , \text{ невыполнение} \\ 12n_1m_2 + 4n_1 + 4 & , \text{ выполнение} \end{array} \right]$

Вывод

В итоге, код алгоритма умножения матриц Винограда больше, чем у стандартного, но трудоемкость меньше. Так же есть возможность модифицировать алгоритм Винограда, чтоб трудоемкость была еще меньше.

3 | Технологическая часть

В данном разделе даны общие требования к программе, средства реализации и реализация алгоритмов.

3.1 Общие требования

Требования к вводу:

- 1) вводятся размеры матриц;
- 2) вводятся (или автоматически генерируются) матрицы.

Требования к программе:

- 1) при вводе неправильных размеров матриц программа не должна завершаться аварийно;
- 2) должно выполняться корректное умножение матриц.

3.2 Средства реализации

В лабораторной работе был использован язык *C++*[1], так как он известен, и на нём было написано множество предыдущих работ.

Среда разработки - *Qt*[2].

Для замеров процессорного времени была использована функция *clock()*[3].

3.3 Реализация алгоритмов

Листинг 1 - Алгоритм стандартного умножения матриц

```
1  void std_matrix_mult(int** matrix1, int n1, int m1,\n2  int** matrix2, int n2, int m2)\n3  {\n4      if ((m1 != n2) || n1 == 0 || n2 == 0)\n5      {\n6          std::cout << "Incorrect matrixes" << std::endl;\n7          return;\n8      }\n9\n10     int** temp_matrix = create_matrix(n1, m2);\n11\n12     for (int i = 0; i < n1; i++)\n13     {\n14         for (int j = 0; j < m2; j++)\n15         {\n16             temp_matrix[i][j] = 0;\n17             for (int k = 0; k < m1; k++)\n18                 temp_matrix[i][j] = temp_matrix[i][j] + matrix1[i][k] *\n19                     matrix2[k][j];\n20         }\n21     }\n22\n23     print_matrix(temp_matrix, n1, m2);\n24     delete_matrix(temp_matrix, m1);\n25 }
```

Листинг 2 - Алгоритм Винограда

```
1 void vinograd_matrix_mult(int** matrix1, int n1, int m1,\n2 int** matrix2, int n2, int m2)\n3 {\n4     if ((m1 != n2) || n1 == 0 || n2 == 0)\n5     {\n6         std::cout << "Incorrect matrixes" << std::endl;\n7         return;\n8     }\n9\n10    int** temp_matrix = create_matrix(n1, m2);\n11\n12    int row_factor[n1];\n13    for (int i = 0; i < n1; i++)\n14    {\n15        row_factor[i] = 0;\n16        for (int k = 0; k < m1 / 2; k++)\n17            row_factor[i] = row_factor[i] + matrix1[i][2 * k] * matrix1[i\n18                ][2 * k + 1];\n19    }\n20\n21    int col_factor[m2];\n22    for (int i = 0; i < m2; i++)\n23    {\n24        col_factor[i] = 0;\n25        for (int k = 0; k < n2 / 2; k++)\n26            col_factor[i] = col_factor[i] + matrix2[2 * k][i] * matrix2[2 *\n27                k + 1][i];\n28    }\n29\n30    for (int i = 0; i < n1; i++)\n31    {\n32        for (int j = 0; j < m2; j++)\n33        {\n34            temp_matrix[i][j] = -row_factor[i] - col_factor[j];\n35            for (int k = 0; k < m1 / 2; k++)\n36                temp_matrix[i][j] = temp_matrix[i][j] + (matrix1[i][2 * k +\n37                    1] + matrix2[2 * k][j])
```



```

35         * (matrix1[i][2 * k] + matrix2[2 * k + 1][j]);
36     }
37 }
38
39 if (m1 % 2 == 1)
40 {
41     for (int i = 0; i < n1; i++)
42     {
43         for (int j = 0; j < m2; j++)
44             temp_matrix[i][j] = temp_matrix[i][j] + matrix1[i][m1 - 1] *
45                 matrix2[m1 - 1][j];
46     }
47
48     print_matrix(temp_matrix, n1, m2);
49     delete_matrix(temp_matrix, m1);
50 }

```

Листинг 3 - Модифицированный алгоритм Винограда

```

1 void vinograd_modified_matrix_mult(int** matrix1, int n1, int m1,\
2 int** matrix2, int n2, int m2)
3 {
4     if ((m1 != n2) || n1 == 0 || n2 == 0)
5     {
6         std::cout << "Incorrect matrixes" << std::endl;
7         return;
8     }
9
10    int** temp_matrix = create_matrix(n1, m2);
11
12    int row_factor[n1];
13    int even_m1 = m1 - m1 % 2;
14    for (int i = 0; i < n1; i++)
15    {
16        row_factor[i] = 0;
17        for (int k = 0; k < even_m1; k += 2)
18            row_factor[i] += matrix1[i][k] * matrix1[i][k + 1];
19    }
20

```

```

21  int col_factor[m2];
22  int even_n2 = n2 - n2 % 2;
23  for (int i = 0; i < m2; i++)
24  {
25      col_factor[i] = 0;
26      for (int k = 0; k < even_n2; k += 2)
27          col_factor[i] -= matrix2[k][i] * matrix2[k + 1][i];
28  }
29
30  for (int i = 0; i < n1; i++)
31  {
32      for (int j = 0; j < m2; j++)
33      {
34          int buf = row_factor[i] + col_factor[j];
35          for (int k = 0; k < even_m1; k += 2)
36              buf += (matrix1[i][k + 1] + matrix2[k][j])
37                  * (matrix1[i][k] + matrix2[k + 1][j]);
38
39          temp_matrix[i][j] = buf;
40      }
41  }
42
43  if (m1 % 2 == 1)
44  {
45      for (int i = 0; i < n1; i++)
46      {
47          for (int j = 0; j < m2; j++)
48              temp_matrix[i][j] += matrix1[i][m1 - 1] * matrix2[m1 - 1][j];
49      }
50  }
51
52  print_matrix(temp_matrix, n1, m2);
53  delete_matrix(temp_matrix, m1);
54  }

```

Вывод

По итогу, написанная программа соответствует всем описанным выше требованиям, алгоритмы были реализованы на C++, так как данный язык известен, выполнено на нём много прошлых работ.

4 | Экспериментальная часть

В данном разделе представлены результаты работы программы и приведен анализ времени работы каждого из алгоритмов.

4.1 Примеры работы программы

На Рисунке 4 представлены меню и ввод матриц.

```
0 - Exit
1 - Input matrixes
2 - Standart
3 - Vinograd
4 - Vinograd modified
5 - Timing tests
Your choice: 1

Input rows amount: 3
Input columns amount: 2
Input matrix
1 2
3 4
5 6

Input matrix
1 2 3
4 5 6
```

Рисунок 4 - Меню выбора и ввод матриц

На Рисунке 5 изображен результат работы стандартного алгоритма умножения матриц.

```
0 - Exit
1 - Input matrixes
2 - Standart
3 - Vinograd
4 - Vinograd modified
5 - Timing tests
Your choice: 2

9 12 15
19 26 33
29 40 51
```

Рисунок 5 - Стандартный алгоритм умножения матриц

На Рисунке 6 показан результат работы алгоритма Винограда.

```
0 - Exit
1 - Input matrixes
2 - Standart
3 - Vinograd
4 - Vinograd modified
5 - Timing tests
Your choice: 3

9 12 15
19 26 33
29 40 51
```

Рисунок 6 - Алгоритм Винограда

На Рисунке 7 показан результат работы модифицированного алгоритма Винограда.

```
0 - Exit
1 - Input matrixes
2 - Standart
3 - Vinograd
4 - Vinograd modified
5 - Timing tests
Your choice: 4

9 12 15
19 26 33
29 40 51
```

Рисунок 7 - Модифицированный алгоритм Винограда

4.2 Анализ времени работы алгоритмов

Эксперементы проводятся на квадратных матрицах. Элементы матриц задаются случайным образом.

В первом случае размеры матриц: 100x100, 200x200, 300x300, 400x400, 500x500. Графики зависимости времени работы алгоритмов от размеров матриц изображены на Рисунке 8.

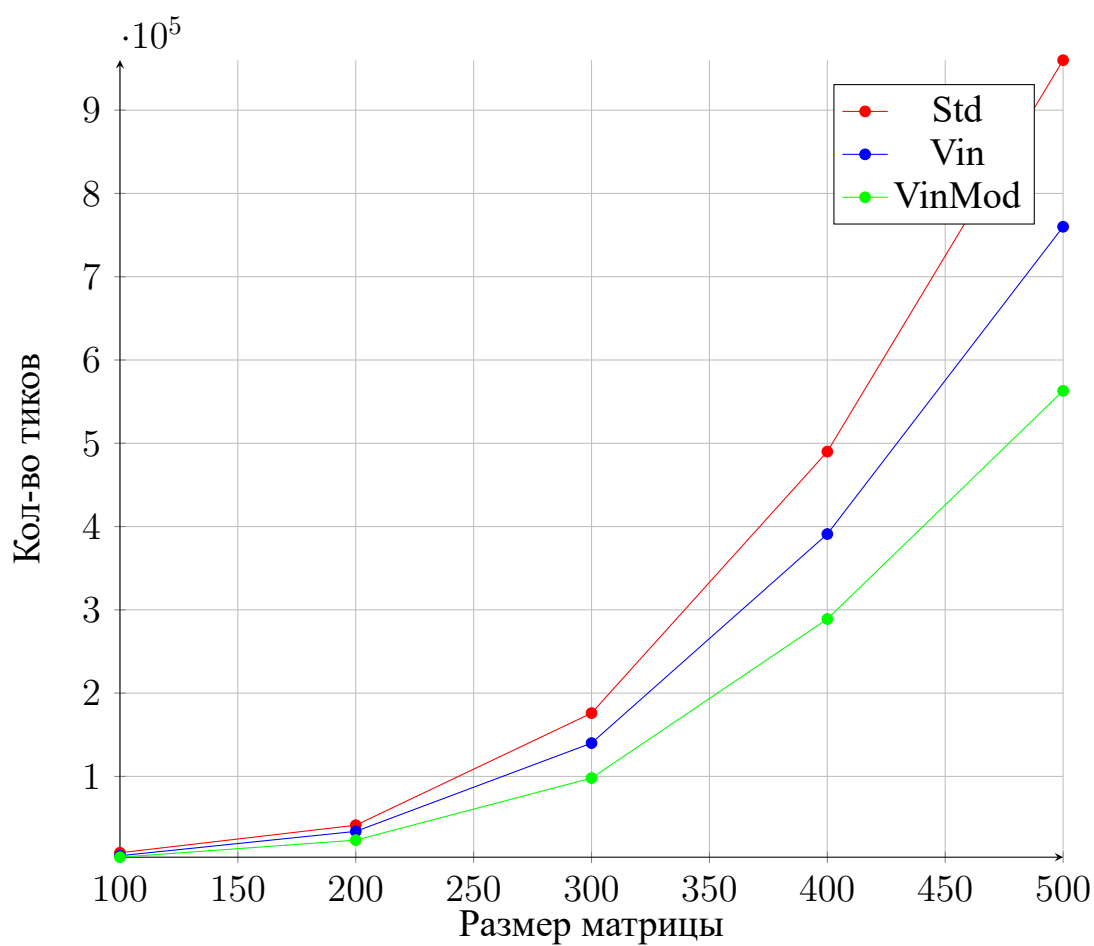


Рисунок 8 - Графики зависимости времени работы алгоритмов от размеров

Во втором случае размеры матриц: 101x101, 201x201, 301x301, 401x401, 501x501. Графики зависимости времени работы алгоритмов от размеров матриц показаны на Рисунке 9.

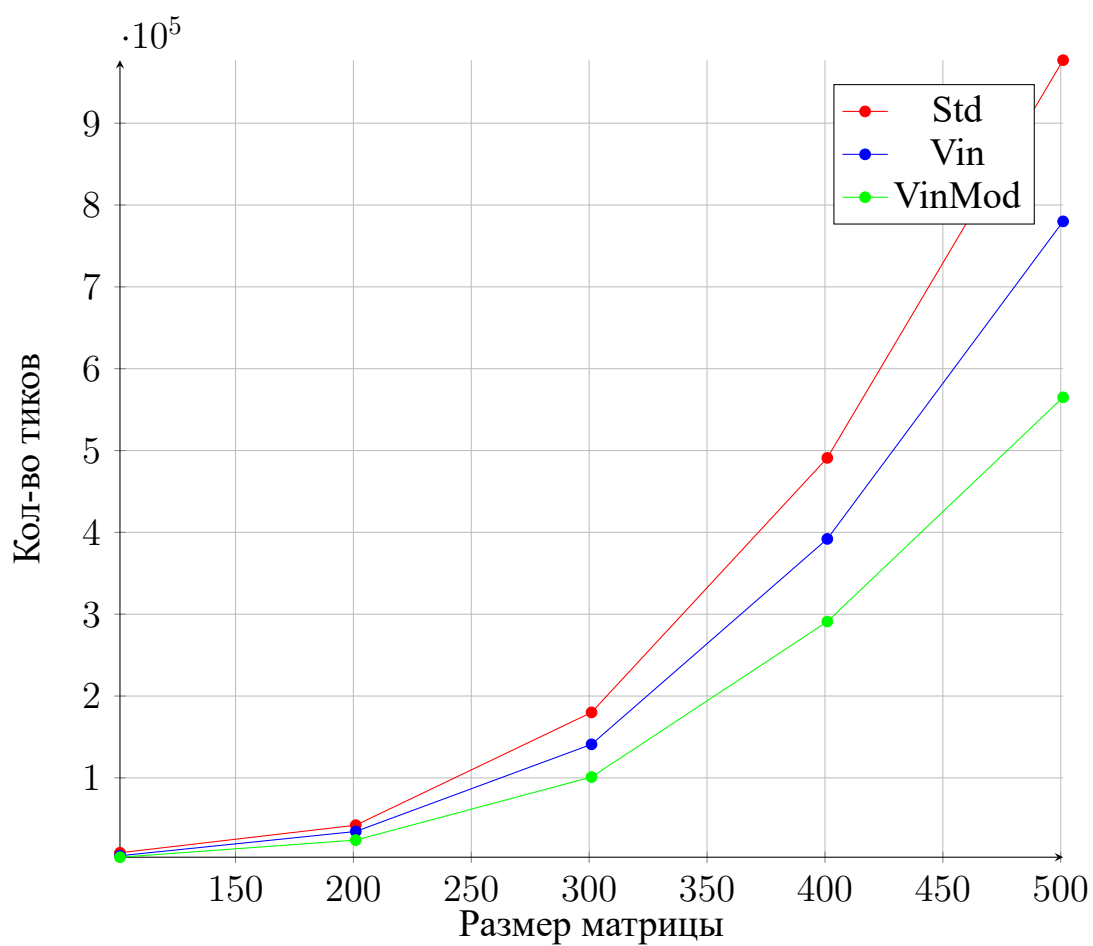


Рисунок 9 - Графики зависимости времени работы алгоритмов от размеров

Вывод

По итогу, все алгоритмы дают верный результат. По значениям замеров можно сделать вывод, что модифицированный алгоритм Винограда работает быстрее всех, самым медленным оказался стандартный

Заключение

В ходе лабораторной работы были изучены алгоритмы умножения матриц: стандартный и Винограда. Также был модифицирован алгоритм Винограда. Были даны теоритические оценки алгоритмов умножения матриц. Была оценена трудоемкость алгоритмов. Также было произведено сравнение времени работы алгоритмов, в результате которого стало понятно, что модифицированный алгоритм Винограда почти в 2 раза быстрее, чем стандартный алгоритм, и в 1.5 раза быстрее, чем алгоритм Винограда.

Литература

- 1) Бьерн Страуструп. Язык программирования C++. -URL:

https://codernet.ru/books/c_plus/bern_straustруп_yazyk_programmirovaniya_c_specia

(дата обращения: 01.10.2020). Текст: электронный.

- 2) Qt. -URL:

<https://www.qt.io/> (дата обращения: 01.10.2020). Текст: электронный.

- 3) Функция *clock*. -URL:

<https://docs.microsoft.com/ru-ru/cpp/c-runtime-library/reference/clock?view=vs-2019>

(дата обращения: 01.10.2020). Текст: электронный.

- 4) Дж. Макконнелл. Основы современных алгоритмов.

2-е дополненное издание

Москва: Техносфера, 2004. - 368с. ISBN 5-94836-005-9

с. 130 - 133