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 ____________ Д.О. Склифасовский
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподователь ____________ Л.Л. Волкова
(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Оглавление

Введение

Цель работы: изучение алгоритмов умножения матриц. В данной лабораторной работе рассматриваются 3 алгоритма:

1) стандартный алгоритм умножения матриц;
2) алгоритм Винограда;
3) модифицированный алгоритм Винограда.

Также требуется изучить рассчет сложности алгоритмов. В ходе лабораторной работы необходимо:

1) изучить алгоритмы умножения матриц;
2) оптимизировать алгоритм Винограда;
3) дать теоритическую оценку стандартного алгоритма умножения матриц, алгоритма Винограда и модифицированного алгоритма Винограда;
4) реализовать три алгоритма умножения матриц на одном из языков программирования;
5) сравнить алгоритмы умножения матриц.

1 Аналитическая часть

В данном разделе представлено математическое описание алгоритмов умножения матриц.

1.1 Стандартный алгоритм умножения матриц

Матрица — математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых, действительных или комплексных чисел), которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы. Количество строк и столбцов задает размер матрицы. Хотя исторически рассматривались, например, треугольные матрицы, в настоящее время говорят исключительно о матрицах прямоугольной формы, так как они являются наиболее удобными и общими. 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 размером $[l * m]$ и $[m * n]$. В результате произведения матриц A и B получим матрицу C размером $[l * n]$, в которой:

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

Операция умножения двух матриц выполнима только в том случае, если число столбцов в первом сомножителе равно числу строк во втором; в этом случае говорят, что матрицы согласованы. В частности, умножение всегда выполнимо, если оба сомножителя — квадратные матрицы одного и того же порядка.

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.

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:

$$V * W = v_1w_1 + v_2w_2 + v_3w_3 + v_4w_4 \tag{1.2}$$

Это равенство можно переписать в виде:

$$V * W = (v_1 + w_2)(v_2 + w_1) + (v_3 + w_4)(v_4 + w_3) - v_1v_2 - v_3v_4 - w_1w_2 - w_3w_4 \tag{1.3}$$

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.

1.3 Вывод

Было представлено математическое описание стандартного алгоритма умножения матриц и алгоритма Винограда. Основное отличие - наличие предварительной обработки, а также количество операций умножения.

2 Конструкторская часть

В данном разделе представлены схемы разработанных алгоритмов. Также оценивается трудоемкость алгоритмов.

2.1 Разработка алгоритмов

На рисунке 1 изображена схема стандартного алгоритма умножения матриц.

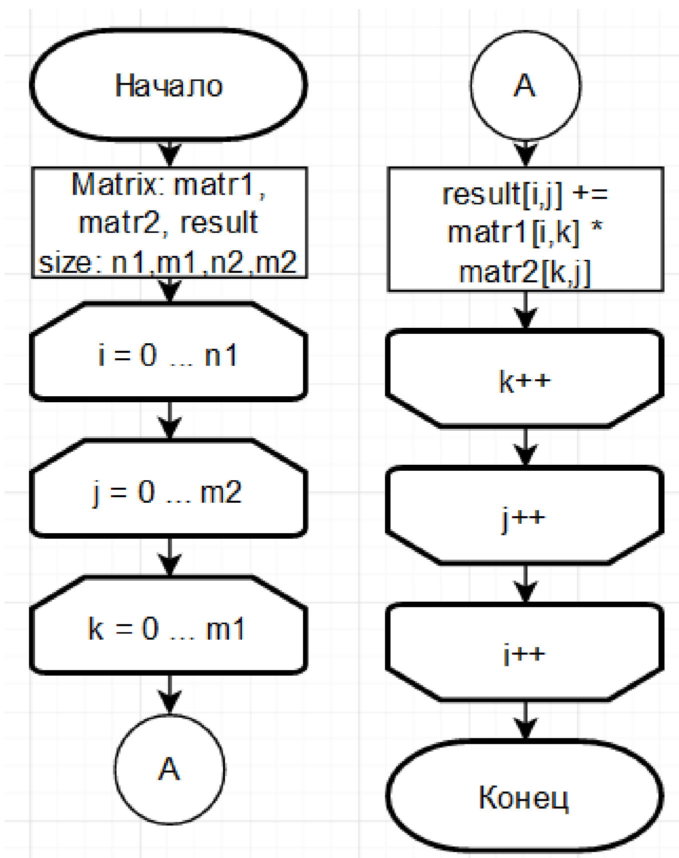


Рисунок 1. Схема стандартного алгоритма умножения матриц

На рисунке 2 изображена схема алгоритма Винограда.

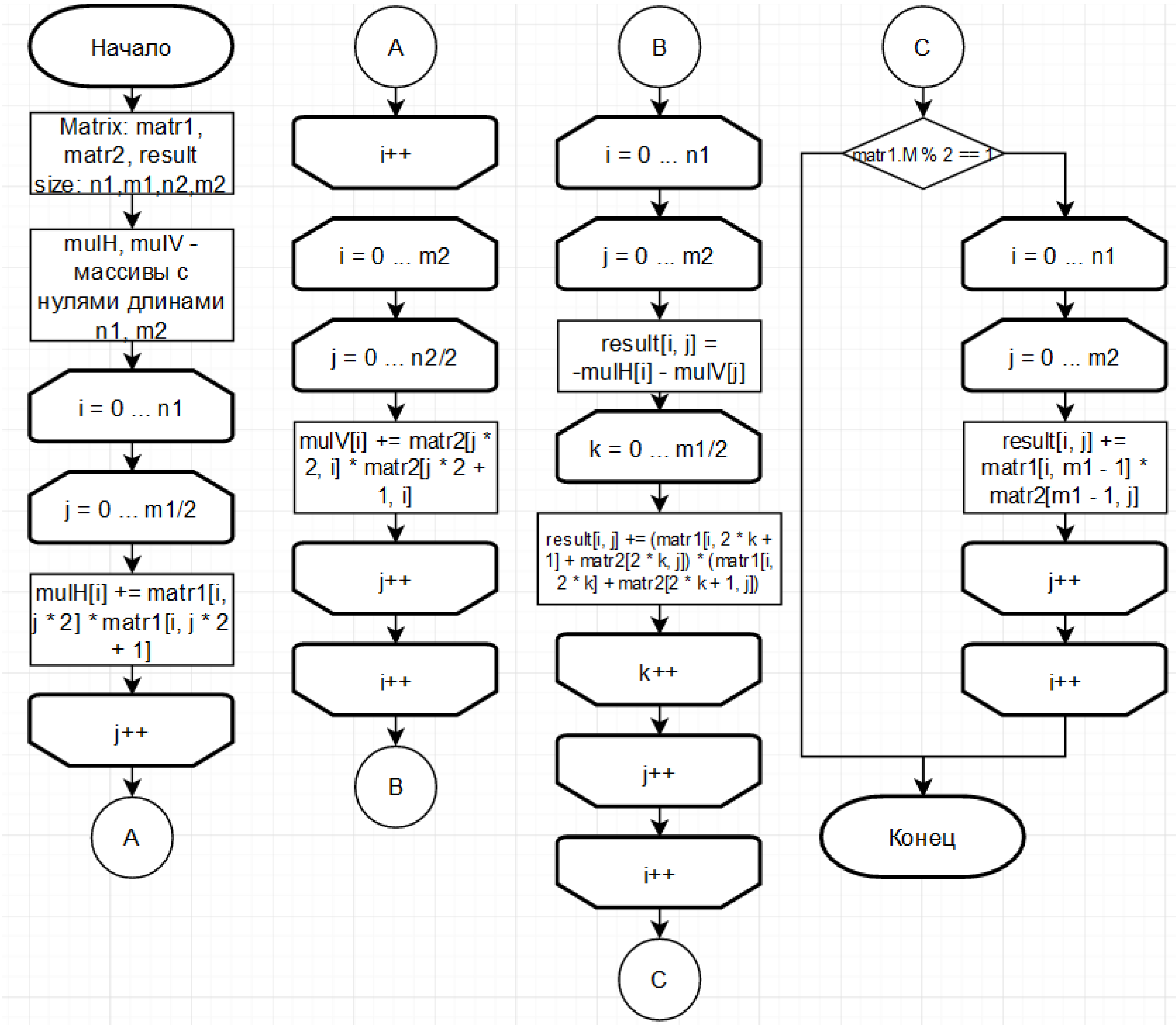


Рисунок 2. Схема алгоритма Винограда

На рисунке 3 изображена схема модифицированного алгоритма Винограда.

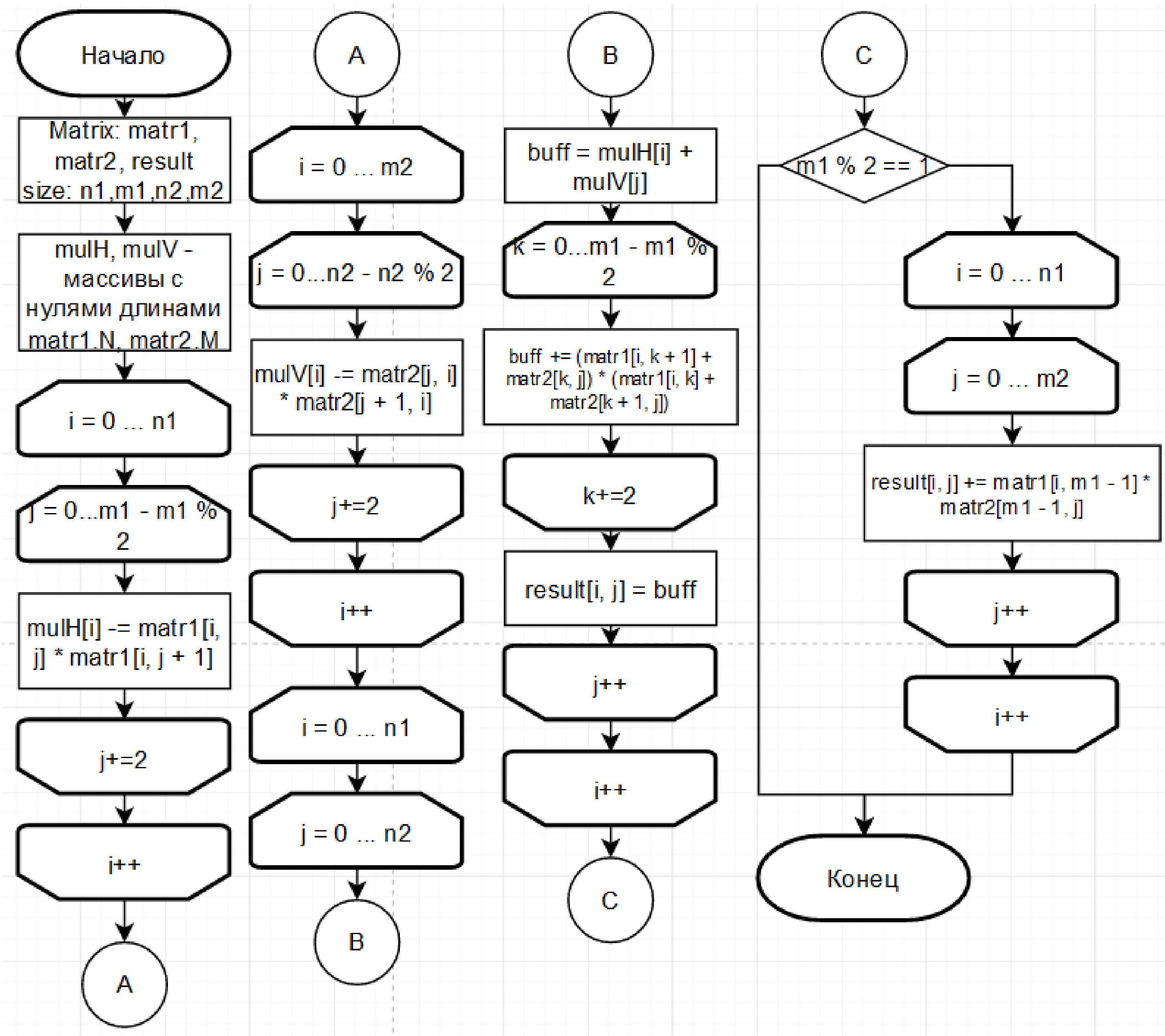


Рисунок 3. Схема модифицированного алгоритма Винограда

2.2 Модель трудоемкости

Модель трудоемкости для оценки алгоритмов:

1) стоимость базовых операций единица:

   $=, +, *, \simeq, <, >, \geq, \leq, ==, != , [], +=, -=, *=, /=, ++, --;$

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

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

4) операция обращения к ячейки матрицы [i, j] имеет трудоёмкость равную двум.

2.3 Трудоемкость алгоритмов

Оценим трудоемкости алгоритмов.

2.3.1 Трудоемкость предварительной проверки

Таблица 2.1: Оценка веса

Код	Вес
int n1 = matr1.N;	2
int n2 = matr2.N;	2
int m1 = matr1.M;	2
int m2 = matr2.M;	2
if ((m1 != n2) || n1 == 0 || n2 == 0) return null;	5

2.3.2 Трудоемкость стандартного алгоритма

Подсчет: $1+1+n_1(2+1+1+m_2(2+1+1+m_1(2+8))) = 2+n_1(4+m_2(4+m_1 10)) = 2 + n_1(4 + 4m_2 + 10m_1m_2) = 10n_1m_1m_2 + 4n_1m_2 + 4n_1 + 2$

2.3.3 Трудоемкость алгоритма Винограда

Подсчет:

Первый цикл: $\frac{15}{2}n_1m_1 + 5n_1 + 2$

Второй цикл: $\frac{15}{2}m_2n_2 + 5m_2 + 2$

Третий цикл: $13n_1m_2m_1 + 12n_1m_2 + 4n_1 + 2$

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

Результат: $13n_1m_2m_1 + \frac{15}{2}n_1m_1 + \frac{15}{2}m_2n_2 + 12n_1m_2 + 5n_1 + 5m_2 + 4n_1 + 6 + \begin{bmatrix} 2 & \text{, невыполнение условия} \\ 15n_1m_2 + 4n_1 + 2 & \text{, выполнение} \end{bmatrix}$

2.3.4 Трудоемкость модифицированного алгоритма Винограда

Подсчет:

Первый цикл: $\frac{11}{2}n_1m_1 + 4n_1 + 2$

Второй цикл: $\frac{11}{2}m_2n_2 + 4m_2 + 2$

Третий цикл: $\frac{17}{2}n_1m_2m_1 + 9n_1m_2 + 4n_1 + 2$

Условный оператор: $\begin{bmatrix} 2 & \text{, невыполнение} \\ 10n_1m_2 + 4n_1 + 2 & \text{, выполнение} \end{bmatrix}$

Результат: $\frac{17}{2}n_1m_2m_1 + \frac{11}{2}n_1m_1 + \frac{11}{2}m_2n_2 + 9n_1m_2 + 8n_1 + 4m_2 + 6 + \begin{bmatrix} 2 & \text{, невыполнение} \\ 10n_1m_2 + 4n_1 + 2 & \text{, выполнение} \end{bmatrix}$

2.3.5 Вывод

В данном разделе были рассмотрены схемы алгоритмов умножения матриц, введена модель оценки трудоемкости алгоритма и были рассчитаны трудоемкости алгоритмов.

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 качестве языка программирования был выбран C#, так как я знаком с данным языком программирования, имею представление о способах тестирования программы.

Средой разработки Visual Studio.

Для замеров процессорного времени используется функция $Stopwatch$.

3.3 Сведения о модулях программы

Программа состоит из:

1) Program.cs - главный файл программы, в котором располагается точка входа в программу.

2) Matrix.cs - файл класса Matrix. Класс реализует матрицу размером $n * m$, а также он содержит методы для работы с матрицами.

3) Array.cs - файл класса Array. Класс реализует массив размером n, а также он содержит методы для работы с массивами.

4) MultMatr.cs - файл класса MultMatr. В нем находятся алгоритмы умножения матриц.

3.4 Листинг кода программы

Листинг 3.1: Класс Matrix для работы с матрицами

```
class Matrix
{
  private int n;
  private int m;
  private int[,] matrix;

  public Matrix() { }
  public Matrix(int n, int m)
  {
    this.n = n;
    this.m = m;
    matrix = new int[n, m];
  }

  public int N
  {
```

```
            get { return n; }
            set { if (value > 0) n = value; }
        }
        public int M
        {
            get { return m; }
            set { if (value > 0) m = value; }
        }

        public int this[int i, int j]
        {
            get { return matrix[i, j]; }
            set { matrix[i, j] = value; }
        }

        public void InputMatr()
        {
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    Console.WriteLine("                {0}:{1}", i + 1, j + 1);
                    matrix[i, j] = Convert.ToInt32(Console.ReadLine());
                }
            }
        }

        public void ReadMatr()
        {
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    Console.Write(matrix[i, j] + "\t");
                }
                Console.WriteLine();
            }
        }
```

```
    public void FillMatr()
    {
        Random rand = new Random();
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < m; j++)
            {
                matrix[i, j] = rand.Next(100);
            }
        }
    }
}
```

Листинг 3.2: Класс Array для работы с массивами

```
class Array
{
    private int[] array;
    private int n;
    public Array() { }

    public Array(int n)
    {
        this.n = n;
        array = new int[n];
    }

    public int N
    {
        get { return n; }
        set { if (value > 0) n = 0; }
    }

    public int this[int i]
    {
        get { return array[i]; }
        set { array[i] = value; }
    }
```

```
    }
```

Листинг 3.3: Стандартный алгоритм умножения матриц

```
    public static Matrix StandartMult(Matrix matr1, Matrix matr2)
    {
      int n1 = matr1.N;
      int n2 = matr2.N;
      int m1 = matr1.M;
      int m2 = matr2.M;
      if ((m1 != n2) || n1 == 0 || n2 == 0)
      {
        return null;
      }

      Matrix result = new Matrix(n1, m2);

      for (int i = 0; i < n1; i++)
      {
        for (int j = 0; j < m2; j++)
        {
          for (int k = 0; k < m1; k++)
          {
            result[i, j] += matr1[i, k] * matr2[k, j];
          }
        }
      }
      return result;
    }
```

Листинг 3.4: Алгоритм Винограда

```
    public static Matrix VinogradMult(Matrix matr1, Matrix matr2)
    {
      int n1 = matr1.N;
      int n2 = matr2.N;
      int m1 = matr1.M;
      int m2 = matr2.M;
      if ((m1 != n2) || n1 == 0 || n2 == 0)
```

```
    {
      return null;
    }

    Matrix result = new Matrix(n1, m2);
    Array mulH = new Array(n1);
    Array mulV = new Array(m2);

    for (int i = 0; i < n1; i++)
    {
      for (int j = 0; j < m1 / 2; j++)
      {
        mulH[i] += matr1[i, j * 2] * matr1[i, j * 2 + 1];
      }
    }
    for (int i = 0; i < n1; i++)
    {
      Console.WriteLine(mulH[i]);
    }
    for (int i = 0; i < m2; i++)
    {
      for (int j = 0; j < n2 / 2; j++)
      {
        mulV[i] += matr2[j * 2, i] * matr2[j * 2 + 1, i];
      }
    }
    for (int i = 0; i < m2; i++)
    {
      Console.WriteLine(mulV[i]);
    }
    for (int i = 0; i < n1; i++)
    {
      for (int j = 0; j < m2; j++)
      {
        result[i, j] = -mulH[i] - mulV[j];
        for (int k = 0; k < m1 / 2; k++)
        {
          result[i, j] += (matr1[i, 2 * k + 1] + matr2[2 * k, j]) *
```

```
                (matr1[i, 2 * k] + matr2[2 * k + 1, j]);
            }
        }
    }

    if (m1 % 2 == 1)
    {
        for (int i = 0; i < n1; i++)
        {
            for (int j = 0; j < m2; j++)
            {
                result[i, j] += matr1[i, m1 - 1] * matr2[m1 - 1, j];
            }
        }
    }

    return result;
}
```

Листинг 3.5: Модифицированный алгоритм Винограда

```
public static Matrix VinogradModMult(Matrix matr1, Matrix matr2)
{
    int n1 = matr1.N;
    int n2 = matr2.N;
    int m1 = matr1.M;
    int m2 = matr2.M;
    if ((m1 != n2) || n1 == 0 || n2 == 0)
    {
        return null;
    }

    Matrix result = new Matrix(n1, m2);
    Array mulH = new Array(n1);
    Array mulV = new Array(m2);

    int smallerM = m1 % 2;
    for (int i = 0; i < n1; i++)
    {
```

```
        for (int j = 0; j < (m1 - smallerM); j += 2)
        {
            mulH[i] -= matr1[i, j] * matr1[i, j + 1];
        }
    }

    int smallerN = n2 % 2;
    for (int i = 0; i < m2; i++)
    {
        for (int j = 0; j < (n2 - smallerN); j += 2)
        {
            mulV[i] -= matr2[j, i] * matr2[j + 1, i];
        }
    }

    for (int i = 0; i < n1; i++)
    {
        for (int j = 0; j < m2; j++)
        {
            int buff = mulH[i] + mulV[j];
            for (int k = 0; k < (m1 - smallerM); k += 2)
            {
                buff += (matr1[i, k + 1] + matr2[k, j]) * (matr1[i, k] +
                    matr2[k + 1, j]);
            }
            result[i, j] = buff;
        }
    }

    if (smallerM == 1)
    {
        for (int i = 0; i < n1; i++)
        {
            for (int j = 0; j < m2; j++)
            {
                result[i, j] += matr1[i, m1 - 1] * matr2[m1 - 1, j];
            }
        }
```

```
        }

        return result;
    }
```

3.5 Вывод

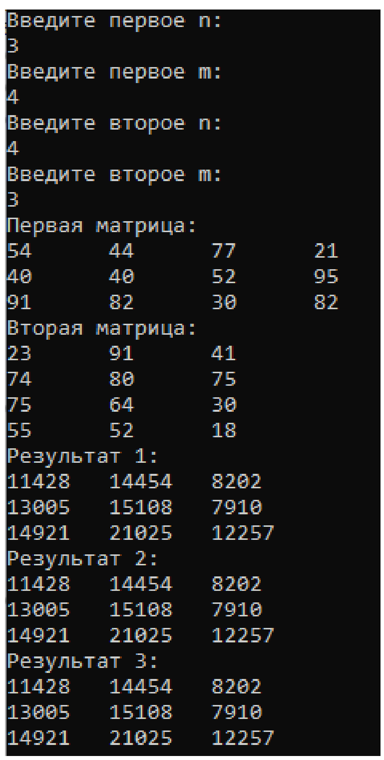
В данном разделе были даны общие требования к программе, описаны средства реализации, были представлены сведения о модулях программы, а также реализованы три алгоритма умножения матриц (стандартный, Винограда, модифицированный винограда).

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 представлен результат работы алгоритмов.



```
Введите первое n:
3
Введите первое m:
4
Введите второе n:
4
Введите второе m:
3
Первая матрица:
54      44      77      21
40      40      52      95
91      82      30      82
Вторая матрица:
23      91      41
74      80      75
75      64      30
55      52      18
Результат 1:
11428   14454   8202
13005   15108   7910
14921   21025   12257
Результат 2:
11428   14454   8202
13005   15108   7910
14921   21025   12257
Результат 3:
11428   14454   8202
13005   15108   7910
14921   21025   12257
```

Рисунок 4. Первый результат работы алгоритмов

На рисунке 5 представлен результат работы алгоритмов.

```
Введите второе n:
4
Введите второе m:
4
Первая матрица:
80      60      55      85
94      48      16      0
19      58      57      73
80      5       55      8
Вторая матрица:
39      28      7       67
10      96      50      27
47      79      75      12
38      27      91      27
Результат 1:
9535    14640   15420   9935
4898    8504    4258    7786
6774    12574   13951   5494
6059    7281    5663    6371
Результат 2:
9535    14640   15420   9935
4898    8504    4258    7786
6774    12574   13951   5494
6059    7281    5663    6371
Результат 3:
9535    14640   15420   9935
4898    8504    4258    7786
6774    12574   13951   5494
6059    7281    5663    6371
```

Рисунок 5. Второй результат работы алгоритмов

4.2 Анализ времени работы алгоритмов

литература Виноград: http://www.algolib.narod.ru/Math/Matrix.html