**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 «Фундаментальные Науки»

КАФЕДРА ФН-12 «Математическое моделирование»

# ОТЧЕТ

# ПО ЛАБОРАТОРНОЙ РАБОТЕ НА ТЕМУ:

## Сжатое хранение разреженных матриц

Студент: _____ Мациевский И. М.

дата, подпись Ф.И.О.

Преподаватель: _____ Строганов Ю. В.

дата, подпись Ф.И.О.

*Москва, 2023*

# Содержание

# 1 Введение

**Цель:** Получить навык работы со схемами сжатого хранения разреженных матриц, для достижения поставленной цели требуется решить следующие **задачи**.

1. Описать две схемы хранения разреженных матриц — Дженнингса и Рейнбольдта-Местеньи.
2. Разработать алгоритм сжатия, распаковки, сложения матриц для обеих схем и алгоритм умножения матриц для схемы Рейнбольдта-Местеньи.
3. Реализовать разработанные алгоритмы.
4. Оценить эффективность сжатого хранения матриц в обеих схемах для матрицы действительных чисел с 5 % ненулевых элементов.

# 2 Матрица

**Матрица** — упорядоченный математический объект, обычно записываемый в виде прямоугольной таблицы элементов кольца/поля или другой структуры. Количество строк и столбцов задает размер этой матрицы. С матрицами можно производить различные действия, такие как транспонирование, сложение, умножения и др.

# 3 Разреженная матрица

**Разреженная матрица** — матрица, большая часть элементов которой являются нулевыми. Разреженные матрицы могут возникать при решении таких задач, как дифференциальное уравнение в частных производных или, например, анализ данных.

# 4 Схема Дженнингса

## 4.1 Описание

Схема Дженнингса — способ хранения симметричных разреженных матриц, который требует меньше памяти, чем хранение матрицы целиком. Идея заключается в том, чтобы в отдельном массиве $An$ хранить ненулевые элементы (а также все диагональные и некоторые нулевые в особых случаях, см. Алгоритм сжатия) и хранении в массиве $D$ индексов диагональных элементов по массиву $An$.

## 4.2 Алгоритм сжатия матрицы по схеме Дженнингса

Алгоритм заключается в следующем:
Для начала матрица делится своей главной диагональю на две части, далее рассматривается нижняя треугольная матрица.
Для хранения сжатой матрицы используются два массива: $An$ — массив ненулевых (в особых случаях там есть и нули) элементов, и массив $D$ индексов диагональных элементов по массиву $An$.
Требуется пройти по строкам нижней треугольной матрицы по следующему алгоритму: если элемент диагональный, записать в конец массива $An$, его индекс по $An$ записать в конец массива $D$, иначе если элемент ненулевой, записать его в конец массива $An$.
Иначе если элемент равен нулю, но при этом в этой строке уже встречался ненулевой элемент, помимо диагонального, записать его в массив $An$. Алгоритм по этапам:

1. $q = 1$, $count\_an = 0$ (с помощью переменной $q$ считывается нижняя часть матрицы, переменная $count\_an$ используется для заполнения массива $D$;

2. Перебор по строкам нижней треугольной матрицы, инициализация переменной $check$ типа bool, которая будет отслеживать встретился в строке ненулевой элемент после диагонального.

3. Перебор по элементам строки с индексами от 0 до $q$ не включительно;

   (a) Если элемент диагональный, добавить его в $An$, добавить его индекс по $An$ в $D$, увеличить $count\_an$ на 1, переход к шагу;

   (b) Иначе если $check == true$, добавить элемент в $An$ увеличить $count\_an$, переход к шагу 3;

   (c) Иначе если элемент ненулевой, добавить элемент в $An$ увеличить $count\_an$ на 1, $check = true$, переход к шагу 3;

4. Увеличить $q$ на 1, перейти к шагу 2.

## 4.3 Алгоритм суммы без распаковки двух матриц, упакованных по схеме Дженнингса

1. $c.an \leftarrow a.an[0] + b.an[0]$, $c.d[0] = 0$;

2. Для строк от 1 до N(нумерация с 0)

   (a) Если количество элементов i-й строки, записанных в $an$, разное для матриц A и B, то большее из двух значений для A и B количество элементов данной строки определяет количество элементов одноименной строки C. При сложении диагональных элементов $c.d$ обновляется индексом суммы по $c.an$;

(b) Если количество элементов в i-й строке для A и B совпадают, то они суммируются, нельзя забывать удалять лишние нули, если перед ними не идет ненулевых элементов, кроме диагональных. При сложении диагональных элементов $c.d$ обновляется индексом суммы по $c.an$.

### 4.4 Алгоритм распаковки матрицы, упакованной по схеме Дженнингса

1. $a[0][0] = an[0]$;
2. Для строки $i$ от 1 до $N$(нумерация с 0):
   (a) $a[i][i] = an[d[i]]$ — диагональный элемент сразу записывается в матрицу;
   (b) Если в массиве $an$ между двумя диагональными элементами есть еще элементы, они записываются в той же строке под диагональю(левее диагонального элемента), каждый такой элемент записывается в две ячейки матрицы с симметричными координатами, чтобы на выходе была симметричная матрица, а не ее половина.

## 5 Кольцевая схема Рейнбольдта-Местеньи

### 5.1 Описание

Кольцевая схема Рейнбольдта-Местеньи — способ хранения произвольных разреженных матриц. Для хранения используются 5 массивов: $An$ — массив ненулевых значений, $Nr$— индекс по $An$ следующего ненулевого элемента той же строки, $Nc$— индекс по $An$ следующего ненулевого элемента того же столбца, $Jr$— индекс по $An$ начальных элементов строк, $Jc$— индекс по $An$ начальных элементов столбцов. Изначально массивы $An, Nr, Nc$ являются пустыми векторами, массивы $Jr, Jc$ инициализируются $-1$, количество $-1$ в них равно количеству строк и столблцов матрицы.

### 5.2 Алгоритм добавления элемента в массив $an$

1. Если данный элемент равен 0, прервать функцию;
2. Если это первый элемент $i$-й строки, то есть $jr[i] == -1$, обновить $jr[i]$ индексом $count\_an$, в конец массива $nr$ добавить элемент $count\_an$;
3. Иначе пройти по массиву $nr$, пока не будет найдено элемента, который ссылается на первый элемент строки, обновить это значение индексом $count\_an$, в конец массива $nr$ добавить индекс первого элемента строки;

4. Если это первый элемент $j$-й строки, то есть $jc[j] == -1$, обновить $jc[j]$ значением $count\_an$, в конец массива $nc$ добавить элемент $count\_an$;

5. Иначе пройти по массиву $nc$, пока не будет найдено элемента, который ссылается на первый элемент столбца, обновить это значение индексом $count\_an$, в конец массива $nc$ добавить индекс первого элемента столбца;

6. Добавить элемент в $an$, увеличить счетчик количества элементов в $an$ на единицу.

## 5.3 Алгоритм сжатия матриц по кольцевой схеме Рейнбольдта-Местеньи

1. $count\_an = 0$ — счетчик количества добавленных в $an$ элементов;

2. Извлечь поэлементно элементы матрицы и для каждого из них запустить функцию добавления элемента в $an$.

## 5.4 Алгоритм определения столбцовой координаты элемента

1. Если индекс $coord$ по массиву $an$ есть в $jc$, то элемент является начальным элементом какого-то столбца, вывести индекс элемента $coord$ по массиву $jc$;

2. Иначе $coord = nc[coord]$, то есть взять следующий элемент этого столбца и вернуться к шагу 1.

## 5.5 Алгоритм определения строчной координаты элемента

1. Если индекс $coord$ по массиву $an$ есть в $jr$, то элемент является начальным элементом какой-то строки, вывести индекс элемента $coord$ по массиву $jr$;

2. Иначе $coord = nr[coord]$, то есть взять следующий элемент этой строки и вернуться к шагу 1.

## 5.6 Алгоритм распаковки матрицы, упакованной по кольцевой схеме Рейнбольдта-Местеньи

1. Для каждого элемента $an$ определить строчную и столбцовую координату, добавить элемент в ячейку массива с этими координатами.

## 5.7 Алгоритм суммы без распаковки двух матриц, упакованных по кольцевой схеме Рейнбольдта-Местеньи

1. Если матрицы разных размеров, завершить выполнение, их невозможно сложить;

2. Завести 3 счетчика $count\_a, count\_b, count\_c$ по массивам $a.an, b.an, c.an$.

3. Для элементов $a.an[count\_a]$ и $b.an[count\_b]$ найти строчные и столбцовые координаты;

4. Если элементы $a.an[count\_a]$ и $b.an[count\_b]$ в разных строках, то добавить верхний по строке элемент (с меньшей строчной координатой) в $c.an$ и увеличить соответствующий счетчик на единицу;

5. Если элементы $a.an[count\_a]$ и $b.an[count\_b]$ в одной строке:

   (a) Если столбцовые координаты разные, в $c.an$ добавить левый по столбцу элемент (с меньшей столбцовой координатой) и увеличить соответствующий счетчик на единицу;

   (b) Если столбцовые координаты равны и сумма элементов $a.an[count\_a]$ и $b.an[count\_b]$ ненулевая, добавить сумму в $c.an$, увеличить оба счетчика.

### 5.8 Алгоритм умножения без распаковки двух матриц, упакованных по кольцевой схеме Рейнбольдта-Местеньи

1. Если количество строк первой матрицы $\neq$ количеству столбцов второй матрицы или количество столбцов первой матрицы $\neq$ количеству строк второй матрицы, завершить выполнение, матрицы нельзя перемножить;

2. $\forall i$, $\forall i$, где $i$ — номер строки матрицы $A$, $j$ — номер столбца матрицы $B$. Если строка $i$ и столбец $j$ ненулевые:

   (a) Используя курсоры по элементам $i$-й строки и $j$-ого столбца, умножить строку на столбец.

   (b) Если произведение ненулевое, добавить его в $c.an$.

## 6 Реализация алгоритмов

Для реализации выбран язык C++. Функции, реализованные в проекте для схемы Дженнингса:

- $show\_one\_dim$ — вывод на экран одномерный вектор;
- $show\_two\_dim$ — вывод на экран двумерный вектор;
- $comp\_Djen$ — упаковка матрицы по схеме Дженнингса;
- $sum\_Djen$ — сумма двух матриц, упакованных по схеме Дженнингса;

- *unpack_Djen* — распаковка матрицы, упакованной по схеме Дженнингса.

Функции, реализованные в коде для кольцеовй схемы Рейнбольдта-Местеньи:

- *add_element* — добавления элемента в упакованную по кольцевой схеме матрицу;
- *comp_ring_RM* — упаковка матрицы по кольцевой схеме;
- *col_el* — поиск столбцовой координаты элемента;
- *row_el* — поиск строчной координаты элемента;
- *unpack_ring_RM* — распаковка матрицы, упакованной по кольцевой схеме;
- *sum_ring_RM* — сумма двух матриц, упакованных по кольцевой схеме;
- *mult_ring_RM* — произведение двух матриц, упакованных по кольцевой схеме.

Реализовано переключение между схемами упаковки матриц. (Реализация 1)

Листинг 1 – Исходный код

```
#include <iostream>
#include <fstream>
#include <vector>
#include <math.h>

using namespace std;

void show_one_dim(const vector<int>& vec) {
  for (int i = 0; i < vec.size(); i++) {
    cout << vec[i] << " ";
  }
}

void show_two_dim(const vector<vector<int>>& vec) {
    for(const auto& row : vec) {
        for(const auto& element : row) {
            cout << element << " ";
        }
        cout << endl;
    }
}

//Схема Дженнингса
```

```
void comp_Djen(vector<int>&a_an, vector<int>&a_d, int &rows, int &cols,
    int num) {
    string adress;
    if (num == 1) {
        adress = "/Users/ilya/DownloadsТипы/ и структуры данных 2 курс, 1
          семестр/laba2_structs/test6.1.txt";
    }
    else {
        adress = "/Users/ilya/DownloadsТипы/ и структуры данных 2 курс, 1
          семестр/laba2_structs/test6.2.txt";
    }
    ifstream file(adress);
    if (!file.is_open()) {
        cout << "Не удалось открыть файл." << endl;
        return;
    }
    file >> rows >> cols;
    int count_a_an = 0;
    if (rows != cols) {
        cout << "Матрица не симметрична" << endl;
        return;
    }
    int q = 1; //чтобы брать половину матрицы
    for (int i = 0; i < rows; i++) {
        bool check = false;
        for (int j = 0; j < q; j++) {
            int element;
            file >> element;
            if (i == j) {
                a_an.push_back(element);
                a_d.push_back(count_a_an);
                count_a_an++;
            }
            else if (check == true) {
                a_an.push_back(element);
                count_a_an++;
            }
            else if(element != 0) {
                a_an.push_back(element);
                check = true;
                count_a_an++;
```

```
            }
        }
        string line;
        getline(file, line);
        q++;
    }
    file.close();
}

void sum_Djen(vector<int>&a_an, vector<int>&a_d, vector<int>&b_an,
   vector<int>&b_d, vector<int>&c_an, vector<int>&c_d, int &rows_a, int
    &cols_a,
   int &rows_b, int &cols_b) {
    if (rows_a != rows_b or cols_a != cols_b) {
        cout << "Невозможно сложить матрицы" << endl;
        return;
    }
    c_an.push_back(a_an[0] + b_an[0]);
    c_d.push_back(0);
    int count_c = 1;
    int k;
    for (int i = 1; i < rows_a; i++) {
        k = a_d[i] - a_d[i - 1] - b_d[i] + b_d[i - 1]; //понять какая
   строка длиннее
        if (k > 0) {
            int j = 1;
            while (j <= k) {
                c_an.push_back(a_an[a_d[i - 1] + j]);
                count_c++;
                j++;
            }
            while (a_d[i - 1] + j <= a_d[i]) {
                c_an.push_back(a_an[a_d[i - 1] + j] + b_an[b_d[i - 1] +
    j - k]);
                count_c++;
                j++;
            }
            c_d.push_back(count_c - 1);
        }
        else if (k < 0) {
            k = -1 * k;
```

```
            int j = 1;
            while (j <= k) {
                c_an.push_back(b_an[b_d[i - 1] + j]);
                count_c++;
                j++;
            }
            while (b_d[i - 1] + j <= b_d[i]) {
                c_an.push_back(b_an[b_d[i - 1] + j] + a_an[a_d[i - 1] +
    j - k]);
                count_c++;
                j++;
            }
            c_d.push_back(count_c - 1);
        }
        else {
            int j = a_d[i - 1] + 1; //идем по a_n
            int z = b_d[i - 1] + 1; //идем по b_n
            bool check = false;
            while (j <= a_d[i]) {
                int sum = a_an[j] + b_an[z];
                if (sum != 0) {
                    c_an.push_back(sum);
                    count_c++;
                    check = true;
                }
                else if (check == true) {
                    c_an.push_back(sum);
                    count_c++;
                }
                j++;
                z++;
            }
            if (check == true) {
                c_d.push_back(count_c - 1);
            }
        }
    }
}

void unpack_Djen(vector<int>&a_an, vector<int>&a_d, vector<vector<int
   >>&a) {
```

```
    a[0][0] = a_an[0];
    for (int i = 1; i < a_d.size(); i++) {
        a[i][i] = a_an[a_d[i]];
        if (a_d[i] - a_d[i - 1] > 1) {
            int ind = a_d[i] - 1;
            int k = 1; //помогает понять куда ставить не диагональный элемент
            while (ind != a_d[i - 1]) {
                a[i - k][i] = a_an[ind];
                a[i][i - k] = a_an[ind];
                k++;
                ind--;
            }
        }
    }
}

//Кольцевая схема
void add_element(vector<int>&an, vector<int>&nr, vector<int>&nc, vector
   <int>&jr,
   vector<int>&jc, int element, int i, int j, int &count_an) {
    if (element != 0) {
        if (jr[i] == -1) {
            jr[i] = count_an;
            nr.push_back(count_an);
        }
        else {
            int k = jr[i];
            while (nr[k] != jr[i]) {
                k = nr[k];
            }
            nr[k] = count_an;
            nr.push_back(jr[i]);
        }
        if (jc[j] == -1) {
            jc[j] = count_an;
            nc.push_back(count_an);
        }
        else {
            int k = jc[j];
            while (nc[k] != jc[j]) {
                k = nc[k];
```

```
            }
            nc[k] = count_an;
            nc.push_back(jc[j]);
        }
        an.push_back(element);
        count_an++;
    }
}

void comp_ring_RM(vector<int>&an, vector<int>&nr, vector<int>&nc,
   vector<int>&jr, vector<int>&jc, int &rows, int &cols, ifstream &file
   ) {
    int count_an = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            int element;
            file >> element;
            add_element(an, nr, nc, jr, jc, element, i, j, count_an);
        }
    }
    file.close();
}

int col_el(vector<int>&an, vector<int>&nc, vector<int>&jc, int coord) {
    for (int i = 0; i < jc.size(); i++) {
        if (jc[i] == coord) {
            return i;
        }
    }
    while (1) {
        coord = nc[coord];
        for (int i = 0; i < jc.size(); i++) {
            if (jc[i] == coord) {
                return i;
            }
        }
    }
}

int row_el(vector<int>&an, vector<int>&nr, vector<int>&jr, int coord) {
    for (int i = 0; i < jr.size(); i++) {
```

```
        if (jr[i] == coord) {
            return i;
        }
    }
    while (1) {
        coord = nr[coord];
        for (int i = 0; i < jr.size(); i++) {
            if (jr[i] == coord) {
                return i;
            }
        }
    }
}

void unpack_ring_RM(vector<vector<int>>&a, vector<int>&an, vector<int>&
   nr,
   vector<int>&nc, vector<int>&jr, vector<int>&jc) {
    for (int i = 0; i < an.size(); i++) {
        int row = row_el(an, nr, jr, i);
        int col = col_el(an, nc, jc, i);
        a[row][col] = an[i];
    }
}

void sum_ring_RM(vector<int>&a_an, vector<int>&a_nr, vector<int>&a_nc,
   vector<int>&a_jr, vector<int>&a_jc, int &rows_a, int &cols_a,
   vector<int>&b_an, vector<int>&b_nr, vector<int>&b_nc, vector<int>&
   b_jr,
   vector<int>&b_jc, int &rows_b, int &cols_b, vector<int>&c_an,
   vector<int>&c_nr, vector<int>&c_nc, vector<int>&c_jr, vector<int>&
   c_jc) {
    if (rows_a != rows_b or cols_a != cols_b) {
        cout << "Невозможно сложить матрицы" << endl;
        return;
        }
    else if (a_an.size() == 0) {
        c_an = b_an;
        c_nr = b_nr;
        c_nc = b_nc;
        c_jr = b_jr;
        c_jc = b_jc;
```

```
        return;
    }
    else if (b_an.size() == 0) {
        c_an = a_an;
        c_nr = a_nr;
        c_nc = a_nc;
        c_jr = a_jr;
        c_jc = a_jc;
        return;
    }
    int count_a = 0;
    int count_b = 0;
    int count_c = 0;
    while (count_a != a_an.size() or count_b != b_an.size()) {
        int col_a = col_el(a_an, a_nc, a_jc, count_a);
        int row_a = row_el(a_an, a_nr, a_jr, count_a);
        int col_b = col_el(b_an, b_nc, b_jc, count_b);
        int row_b = row_el(b_an, b_nr, b_jr, count_b);
        if (row_a > row_b) {
            add_element(c_an, c_nr, c_nc, c_jr, c_jc, b_an[count_b],
row_b,
                col_b, count_c);
            count_b++;
        }
        else if (row_b > row_a) {
            add_element(c_an, c_nr, c_nc, c_jr, c_jc, a_an[count_a],
row_a,
                col_a, count_c);
            count_a++;
        }
        else {
            if (col_a > col_b) {
                add_element(c_an, c_nr, c_nc, c_jr, c_jc, b_an[count_b
], row_b,
                    col_b, count_c);
                count_b++;
            }
            else if (col_b > col_a) {
                add_element(c_an, c_nr, c_nc, c_jr, c_jc, a_an[count_a
], row_a,
                    col_a, count_c);
```

```
                count_a++;
            }
            else {
                int sum = b_an[count_b] + a_an[count_a];
                add_element(c_an, c_nr, c_nc, c_jr, c_jc, sum, row_a,
    col_a,
                    count_c);
                count_a++;
                count_b++;
            }
        }
    }
    while (count_a != a_an.size()) {
        int col_a = col_el(a_an, a_nc, a_jc, count_a);
        int row_a = row_el(a_an, a_nr, a_jr, count_a);
        add_element(c_an, c_nr, c_nc, c_jr, c_jc, a_an[count_a], row_a,
    col_a, count_c);
        count_a++;
    }
    while (count_b != b_an.size()) {
        int col_b = col_el(b_an, b_nc, b_jc, count_b);
        int row_b = row_el(b_an, b_nr, b_jr, count_b);
        add_element(c_an, c_nr, c_nc, c_jr, c_jc, b_an[count_b], row_b,
    col_b, count_c);
        count_b++;
    }
}

void mult_ring_RM(vector<int>&a_an, vector<int>&a_nr, vector<int>&a_nc,
    vector<int>&a_jr, vector<int>&a_jc, int &rows_a, int &cols_a,
    vector<int>&b_an, vector<int>&b_nr, vector<int>&b_nc, vector<int>&
    b_jr,
    vector<int>&b_jc, int &rows_b, int &cols_b, vector<int>&c_an,
    vector<int>&c_nr, vector<int>&c_nc, vector<int>&c_jr, vector<int>&
    c_jc) {
    if (rows_a != cols_b or cols_a != rows_b) {
        cout << "Невозможно перемножить матрицы" << endl;
        return;
    }
    if (a_an.size() == 0 or b_an.size() == 0) {
        return;
```

```
 }
 int count_c = 0;
 for (int i = 0; i < rows_a; i++) {
     int now_row = a_jr[i];
     int start_row = now_row;
     if (now_row == -1) {
         continue;
     }
     for (int j = 0; j < cols_b; j++) {
         int now_col = b_jc[j];
         int start_col = now_col;
         now_row = a_jr[i];
         if (now_col == -1) {
             continue;
         }
         int sum = 0;
         while (a_nr[now_row] != start_row and b_nc[now_col] !=
start_col) {
             int col_a = col_el(a_an, a_nc, a_jc, now_row);
             int row_b = row_el(b_an, b_nr, b_jr, now_col);
             if (col_a > row_b) {
                 now_col = b_nc[now_col];
             }
             else if (row_b > col_a) {
                 now_row = a_nr[now_row];
             }
             else{
                 sum += a_an[now_row] * b_an[now_col];
                 now_col = b_nc[now_col];
                 now_row = a_nr[now_row];
             }
         }
         int col_a = col_el(a_an, a_nc, a_jc, now_row);
         int row_b = row_el(b_an, b_nr, b_jr, now_col);
         if (a_nr[now_row] == start_row) {
             if (col_a == row_b) {
                 sum += a_an[now_row] * b_an[now_col];
             }
             else if (col_a > row_b) {
                 do {
                     now_col = b_nc[now_col];
```

```
                        row_b = row_el(b_an, b_nr, b_jr, now_col);
                        if (row_b == col_a) {
                            sum += a_an[now_row] * b_an[now_col];
                            break;
                        }
                    } while (now_col != start_col);
                }
            }
            else if (b_nc[now_col] == start_col) {
                if (col_a == row_b) {
                    sum += a_an[now_row] * b_an[now_col];
                }
                else if (col_a < row_b) {
                    do {
                        now_row = a_nr[now_row];
                        col_a = col_el(a_an, a_nc, a_jc, now_row);
                        if (row_b == col_a) {
                            sum += a_an[now_row] * b_an[now_col];
                            break;
                        }
                    } while (now_row != start_row);
                }
            }
            add_element(c_an, c_nr, c_nc, c_jr, c_jc, sum, i, j,
   count_c);
        }
    }
}

int main() {
    cout << "Выберите схему, с которой хотите работать:" << endl;
    cout << "1 - Схема Дженнингса, 2 - Кольцевая схема" << endl;
    int k;
    cin >> k;
    if (k == 1) {
        //Работа с первым массивомсжатие(, распаковка, вывод на экран)
        vector<int> a_an;
        vector<int> a_d;
        int rows_a;
        int cols_a;
        comp_Djen(a_an, a_d, rows_a, cols_a, 1);
```

```
    vector<vector<int>> a(rows_a, vector<int>(rows_a, 0));
    cout << "a_an: ";
    show_one_dim(a_an);
    cout << endl;
    cout << "a_d: ";
    show_one_dim(a_d);
    cout << endl;
    unpack_Djen(a_an, a_d, a);
    cout << "a: " << endl;
    show_two_dim(a);
    cout << endl;

    //Работа со вторым массивомсжатие(, распаковка, вывод на экран)
    vector<int> b_an;
    vector<int> b_d;
    int rows_b;
    int cols_b;
    comp_Djen(b_an, b_d, rows_b, cols_b, 2);
    vector<vector<int>> b(rows_b, vector<int>(rows_b, 0));
    cout << "b_an: ";
    show_one_dim(b_an);
    cout << endl;
    cout << "b_d: ";
    show_one_dim(b_d);
    cout << endl;
    unpack_Djen(b_an, b_d, b);
    cout << "b: " << endl;
    show_two_dim(b);
    cout << endl;

    //Поиск суммы
    vector<int> c_an;
    vector<int> c_d;
    sum_Djen(a_an, a_d, b_an, b_d, c_an, c_d, rows_a, cols_a,
rows_b, cols_b);
    if (c_an.size() == 0) {
        return 0;
    }
    vector<vector<int>> c(rows_b, vector<int>(rows_b, 0));
    cout << "c_an: ";
    show_one_dim(c_an);
```

```
        cout << endl;
        cout << "c_d: ";
        show_one_dim(c_d);
        cout << endl;
        unpack_Djen(c_an, c_d, c);
        cout << "c: " << endl;
        show_two_dim(c);
        cout << endl;
    }
    else {
        //упаковка и вывод первой матрицы
        ifstream file1("/Users/ilya/DownloadsТипы/ и структуры данных 2
курс, 1
           семестр/laba2_structs/test18.1.txt");
        vector<int> a_an;
        vector<int> a_nr;
        vector<int> a_nc;
        int rows_a, cols_a;
        file1 >> rows_a >> cols_a;
        vector<int> a_jr(rows_a, -1);
        vector<int> a_jc(cols_a, -1);
        comp_ring_RM(a_an, a_nr, a_nc, a_jr, a_jc, rows_a, cols_a,
file1);
        cout << "a.an: ";
        show_one_dim(a_an);
        cout << endl;
        cout << "a.nr: ";
        show_one_dim(a_nr);
        cout << endl;
        cout << "a.nc: ";
        show_one_dim(a_nc);
        cout << endl;
        cout << "a.jr: ";
        show_one_dim(a_jr);
        cout << endl;
        cout << "a.jc: ";
        show_one_dim(a_jc);
        cout << endl;
        cout << "a:" << endl;
        vector<vector<int>> a(rows_a, vector<int>(cols_a, 0));
        unpack_ring_RM(a, a_an, a_nr, a_nc, a_jr, a_jc);
```

```
    show_two_dim(a);
    cout << endl;

    // упаковка и вывод второй матрицы
    ifstream file2("/Users/ilya/DownloadsТипы/ и структуры данных 2
курс, 1
       семестр/laba2_structs/test18.2.txt");
    vector<int> b_an;
    vector<int> b_nr;
    vector<int> b_nc;
    int rows_b, cols_b;
    file2 >> rows_b >> cols_b;
    vector<int> b_jr(rows_b, -1);
    vector<int> b_jc(cols_b, -1);
    comp_ring_RM(b_an, b_nr, b_nc, b_jr, b_jc, rows_b, cols_b,
file2);
    cout << "b.an: ";
    show_one_dim(b_an);
    cout << endl;
    cout << "b.nr: ";
    show_one_dim(b_nr);
    cout << endl;
    cout << "b.nc: ";
    show_one_dim(b_nc);
    cout << endl;
    cout << "b.jr: ";
    show_one_dim(b_jr);
    cout << endl;
    cout << "b.jc: ";
    show_one_dim(b_jc);
    cout << endl;
    cout << "b:" << endl;
    vector<vector<int>> b(rows_b, vector<int>(cols_b, 0));
    unpack_ring_RM(b, b_an, b_nr, b_nc, b_jr, b_jc);
    show_two_dim(b);
    cout << endl;

    //сумма двух матриц
    vector<int> c_an;
    vector<int> c_nr;
    vector<int> c_nc;
```

```
    vector<int> c_jr(rows_a, -1);
    vector<int> c_jc(cols_b, -1);
    sum_ring_RM(a_an, a_nr, a_nc, a_jr, a_jc, rows_a, cols_a, b_an,
 b_nr,
        b_nc, b_jr, b_jc, rows_b, cols_b, c_an, c_nr, c_nc, c_jr,
c_jc);
    if (rows_a == rows_b and cols_a == cols_b) {
        cout << "c" << endl;
        cout << "c.an: ";
        show_one_dim(c_an);
        cout << endl;
        cout << "c.nr: ";
        show_one_dim(c_nr);
        cout << endl;
        cout << "c.nc: ";
        show_one_dim(c_nc);
        cout << endl;
        cout << "c.jr: ";
        show_one_dim(c_jr);
        cout << endl;
        cout << "c.jc: ";
        show_one_dim(c_jc);
        cout << endl;
        cout << "c:" << endl;
        vector<vector<int>> c(rows_a, vector<int>(cols_a, 0));
        unpack_ring_RM(c, c_an, c_nr, c_nc, c_jr, c_jc);
        show_two_dim(c);
        cout << endl;
    }

    //произведение двух матриц
    vector<int> d_an;
    vector<int> d_nr;
    vector<int> d_nc;
    vector<int> d_jr(rows_a, -1);
    vector<int> d_jc(cols_b, -1);
    mult_ring_RM(a_an, a_nr, a_nc, a_jr, a_jc, rows_a, cols_a, b_an
, b_nr,
        b_nc, b_jr, b_jc, rows_b, cols_b, d_an, d_nr, d_nc, d_jr,
d_jc);
    if (rows_a == cols_b and rows_b == cols_a) {
```

```
            cout << "d" << endl;
            cout << "d.an: ";
            show_one_dim(d_an);
            cout << endl;
            cout << "d.nr: ";
            show_one_dim(d_nr);
            cout << endl;
            cout << "d.nc: ";
            show_one_dim(d_nc);
            cout << endl;
            cout << "d.jr: ";
            show_one_dim(d_jr);
            cout << endl;
            cout << "d.jc: ";
            show_one_dim(d_jc);
            cout << endl;
            cout << "d:" << endl;
            vector<vector<int>> d(rows_a, vector<int>(cols_b, 0));
            unpack_ring_RM(d, d_an, d_nr, d_nc, d_jr, d_jc);
            show_two_dim(d);
            cout << endl;
        }
    }
}
```

# 7 Анализ эффективности

Для анализа эффективности схем хранения матриц, посчитаем затраты памяти на хранение разреженной матрицы без сжатия. Пусть матрицы имеет размеры $M \times N$, одна переменная типа *int* занимает 4 байта, значит все элементы матрицы займут $4MN$ байтов, 95% процентов из которых занимают нули, а значит если хранить только ненулевые элементы (значащими нулями в схеме Дженнингса в общем случае можно пренебречь), то затраты памяти будут $0.38MN$, то есть в 10.5 раз меньше.

# 8 Тесты

## 8.1 Схема Дженнингса

Ниже приведены тесты работы программы, которая суммирует симметричные матрицы, упакованные по схеме Дженнингса

1. Тест 1. Входные файлы: matrix1.txt, matrix2.txt — две симметричные ненулевые матрицы

   Вывод программы — их ненулевая сумму:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
1
a_an: 1 2 3 7 0 4 -1 8 0 5
a_d: 0 1 2 5 8 9
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b_an: 10 20 30 40 0 -7 0 1 -10 1 2 3 7
b_d: 0 2 4 7 11 12
b:
10 20 0 0 0 0
20 30 40 -7 -10 0
0 40 0 0 1 0
0 -7 0 1 2 0
0 -10 1 2 3 0
0 0 0 0 0 7
```

```
c_an: 11 20 32 40 3 5 -10 0 10 3 12
c_d: 0 2 4 5 9 10
c:
11 20 0 0 0 0
20 32 40 0 -10 0
0 40 3 0 0 0
0 0 0 5 10 0
0 -10 0 10 3 0
0 0 0 0 0 12

Program ended with exit code: 0
```

Рис. 1 – Вывод теста 1

2. Тест 2: Входные файлы: test2.1.txt, test2.2.txt — две нулевые матрицы
   Вывод программы — их нулевая сумма:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
1
a_an: 0 0 0 0 0 0
a_d: 0 1 2 3 4 5
a:
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

b_an: 0 0 0 0 0 0
b_d: 0 1 2 3 4 5
b:
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

```
c_an: 0
c_d: 0
c:
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

Program ended with exit code: 0
```

Рис. 2 – Вывод теста 2

3. Тест 3: Входные файлы: test2.1.txt, matrix2.txt — нулевая матрица A и ненулевая матрица B

   Вывод программы — их сумма, то есть ненулевая матрица B:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
1
a_an: 0 0 0 0 0 0
a_d: 0 1 2 3 4 5
a:
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

b_an: 10 20 30 40 0 -7 0 1 -10 1 2 3 7
b_d: 0 2 4 7 11 12
b:
10 20 0 0 0 0
20 30 40 -7 -10 0
0 40 0 0 1 0
0 -7 0 1 2 0
0 -10 1 2 3 0
0 0 0 0 0 7
```

```
c_an: 10 20 30 40 0 -7 0 1 -10 1 2 3 7
c_d: 0 2 4 7 11 12
c:
10 20 0 0 0 0
20 30 40 -7 -10 0
0 40 0 0 1 0
0 -7 0 1 2 0
0 -10 1 2 3 0
0 0 0 0 0 7

Program ended with exit code: 0
```

Рис. 3 – Вывод теста 3

4. Тест 4: Входные файлы: matrix1.txt, test2.2.txt — ненулевая матрица A и нулевая матрица B

   Вывод программы — их сумма, то есть ненулевая матрица B:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
1
a_an: 1 2 3 7 0 4 -1 8 0 5
a_d: 0 1 2 5 8 9
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b_an: 0 0 0 0 0 0
b_d: 0 1 2 3 4 5
b:
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

```
c_an: 1 2 3 7 0 4 -1 8 0 5
c_d: 0 1 2 5 8 9
c:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

Program ended with exit code: 0
```

Рис. 4 – Вывод теста 4

5. Тест 5: Входные файлы: matrix1.txt, test5.txt — матрицы, которые невозможно сложить

   Вывод программы — ошибка:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
1
a_an: 1 2 8 3 9 14 4 10 15 19 5 11 16 20 23 6 12 17 21 24 26 7 13 18 22 25 27 28
a_d: 0 2 5 9 14 20 27
a:
1 2 3 4 5 6 7
2 8 9 10 11 12 13
3 9 14 15 16 17 18
4 10 15 19 20 21 22
5 11 16 20 23 24 25
6 12 17 21 24 26 27
7 13 18 22 25 27 28

b_an: 10 20 30 40 0 -7 0 1 -10 1 2 3 7
b_d: 0 2 4 7 11 12
b:
10 20 0 0 0 0
20 30 40 -7 -10 0
0 40 0 0 1 0
0 -7 0 1 2 0
0 -10 1 2 3 0
0 0 0 0 0 7

Невозможно сложить матрицы
Program ended with exit code: 0
```

Рис. 5 – Вывод теста 5

6. Тест 6: Входные файлы: test6.1.txt, test6.2.txt — матрицы 3 на 3 Вывод программы — сумма двух матриц 3 на 3:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
1
a_an: 1 5 2 4 0 3
a_d: 0 2 5
a:
1 5 4
5 2 0
4 0 3

b_an: 4 8 5 -4 9 6
b_d: 0 2 5
b:
4 8 -4
8 5 9
-4 9 6

c_an: 5 13 7 9 9
c_d: 0 2 4
c:
5 13 0
13 7 9
0 9 9

Program ended with exit code: 0
```

Рис. 6 – Вывод теста 6

## 8.2 Кольцевая схема Рейнбольдта-Местеньи, сложение матриц

Ниже приведены тесты работы программы, которая суммирует произвольные матрицы, упакованные по кольцевой схеме Рейнбольдта-Местеньи

1. Тест 7: Входные файлы: matrix1.txt, matrix2.txt — две ненулевые матрицы
   Вывод программы — их ненулевая сумму:

```
a.an: 1 2 7 3 -1 7 4 8 -1 8 5
a.nr: 0 2 1 4 3 6 7 5 9 8 10
a.nc: 0 5 6 8 7 1 9 4 3 2 10
a.jr: 0 1 3 5 8 10
a.jc: 0 1 3 2 4 10
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b.an: 10 20 30 40 -7 1 -10 1 2 3 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
b.nr: 1 2 3 0 5 6 4 8 9 7 11 12 13 10 15 16 17 18 19 14 21 22 23 24 25 20
b.nc: 7 4 9 12 8 11 13 14 15 10 16 17 18 19 20 21 22 23 24 25 0 1 2 5 3 6
b.jr: 0 4 7 10 14 20
b.jc: 0 1 2 5 3 6
b:
10 20 30 0 40 0
0 -7 0 1 0 -10
1 2 3 0 0 0
0 0 7 7 7 7
7 7 7 7 7 7
7 7 7 7 7 7

c
c.an: 11 20 30 40 -5 8 -10 1 2 6 -1 7 7 11 15 7 7 7 6 15 7 7 7 7 7 7 7 12
c.nr: 1 2 3 0 5 6 4 8 9 10 7 12 13 14 15 11 17 18 19 20 21 16 23 24 25 26 27 22
c.nc: 7 4 9 10 8 13 15 16 11 12 14 17 18 19 20 21 22 23 24 25 26 27 0 1 2 5 3 6
c.jr: 0 4 7 11 16 22
c.jc: 0 1 2 5 3 6
c:
11 20 30 0 40 0
0 -5 0 8 0 -10
1 2 6 0 -1 0
0 7 7 11 15 7
7 7 6 15 7 7
7 7 7 7 7 12
```

Рис. 7 – Вывод теста 7

2. Тест 8: Входные файлы: test2.1.txt, test2.2.txt — две нулевые матрицы
   Вывод программы — их нулевая сумма:

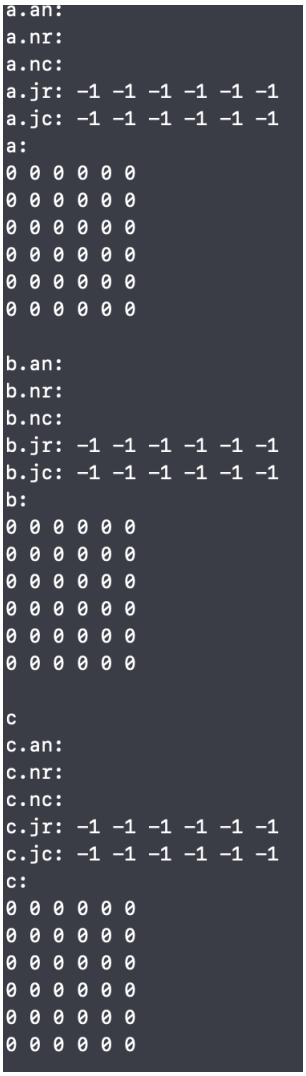


Рис. 8 – Вывод теста 8

3. Тест 9: Входные файлы: test2.1.txt, matrix2.txt — нулевая матрица A и ненулевая матрица B

   Вывод программы — их сумма, то есть ненулевая матрица B:

```
a.an:
a.nr:
a.nc:
a.jr: -1 -1 -1 -1 -1 -1
a.jc: -1 -1 -1 -1 -1 -1
a:
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

b.an: 10 20 20 30 40 -7 -10 40 1 -7 1 2 -10 1 2 3 7
b.nr: 1 0 3 4 5 6 2 8 7 10 11 9 13 14 15 12 16
b.nc: 2 3 0 7 13 10 8 9 11 12 14 15 1 4 5 6 16
b.jr: 0 2 7 9 12 16
b.jc: 0 1 4 5 6 16
b:
10 20 0 0 0 0
20 30 40 -7 -10 0
0 40 0 0 1 0
0 -7 0 1 2 0
0 -10 1 2 3 0
0 0 0 0 0 7

c
c.an: 10 20 20 30 40 -7 -10 40 1 -7 1 2 -10 1 2 3 7
c.nr: 1 0 3 4 5 6 2 8 7 10 11 9 13 14 15 12 16
c.nc: 2 3 0 7 13 10 8 9 11 12 14 15 1 4 5 6 16
c.jr: 0 2 7 9 12 16
c.jc: 0 1 4 5 6 16
c:
10 20 0 0 0 0
20 30 40 -7 -10 0
0 40 0 0 1 0
0 -7 0 1 2 0
0 -10 1 2 3 0
0 0 0 0 0 7
```

Рис. 9 – Вывод теста 9

4. Тест 10: Входные файлы: matrix1.txt, test2.2.txt — ненулевая матрица A и нулевая матрица B

   Вывод программы — их сумма, то есть ненулевая матрица A:

```
a.an: 1 2 7 3 -1 7 4 8 -1 8 5
a.nr: 0 2 1 4 3 6 7 5 9 8 10
a.nc: 0 5 6 8 7 1 9 4 3 2 10
a.jr: 0 1 3 5 8 10
a.jc: 0 1 3 2 4 10
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b.an:
b.nr:
b.nc:
b.jr: -1 -1 -1 -1 -1 -1
b.jc: -1 -1 -1 -1 -1 -1
b:
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

c
c.an: 1 2 7 3 -1 7 4 8 -1 8 5
c.nr: 0 2 1 4 3 6 7 5 9 8 10
c.nc: 0 5 6 8 7 1 9 4 3 2 10
c.jr: 0 1 3 5 8 10
c.jc: 0 1 3 2 4 10
c:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5
```

Рис. 10 – Вывод теста 10

5. Тест 11: Входные файлы: matrix1.txt, test5.txt — матрицы, которые невозможно сложить

   Вывод программы — ошибка:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
2
a.an: 1 2 7 3 -1 7 4 8 -1 8 5
a.nr: 0 2 1 4 3 6 7 5 9 8 10
a.nc: 0 5 6 8 7 1 9 4 3 2 10
a.jr: 0 1 3 5 8 10
a.jc: 0 1 3 2 4 10
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b.an: 1 2 3 4 5 6 7 2 8 9 10 11 12 13 3 9 14 15 16 17 18 4 10 15 19 20 21 22 5 11 16 20 23 24 25 6
12 17 21 24 26 27 7 13 18 22 25 27 28
b.nr: 1 2 3 4 5 6 0 8 9 10 11 12 13 7 15 16 17 18 19 20 14 22 23 24 25 26 27 21 29 30 31 32 33 34
28 36 37 38 39 40 41 35 43 44 45 46 47 48 42
b.nc: 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
39 40 41 42 43 44 45 46 47 48 0 1 2 3 4 5 6
b.jr: 0 7 14 21 28 35 42
b.jc: 0 1 2 3 4 5 6
b:
1 2 3 4 5 6 7
2 8 9 10 11 12 13
3 9 14 15 16 17 18
4 10 15 19 20 21 22
5 11 16 20 23 24 25
6 12 17 21 24 26 27
7 13 18 22 25 27 28

Невозможно сложить матрицы
```

Рис. 11 – Вывод теста 11

6. Тест 12: Входные файлы: test6.1.txt, test6.2.txt — матрицы 3 на 3 Вывод программы — сумма матриц размера 3 на 3:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
2
a.an: 1 5 4 5 2 4 3
a.nr: 1 2 0 4 3 6 5
a.nc: 3 4 6 5 1 0 2
a.jr: 0 3 5
a.jc: 0 1 2
a:
1 5 4
5 2 0
4 0 3

b.an: 4 8 -4 8 5 9 -4 9 6
b.nr: 1 2 0 4 5 3 7 8 6
b.nc: 3 4 5 6 7 8 0 1 2
b.jr: 0 3 6
b.jc: 0 1 2
b:
4 8 -4
8 5 9
-4 9 6

c
c.an: 5 13 13 7 9 9 9
c.nr: 1 0 3 4 2 6 5
c.nc: 2 3 0 5 6 1 4
c.jr: 0 2 5
c.jc: 0 1 4
c:
5 13 0
13 7 9
0 9 9
```

Рис. 12 – Вывод теста 12

## 8.3 Кольцевая схема Рейнбольдта-Местеньи, умножение матриц

Ниже приведены тесты работы программы, которая умножает произвольные матрицы, упакованные по кольцевой схеме Рейнбольдта-Местеньи

1. Тест 13: Входные файлы: matrix4.txt, matrix5.txt — две ненулевые матрицы
   Вывод программы — произведение двух матриц:

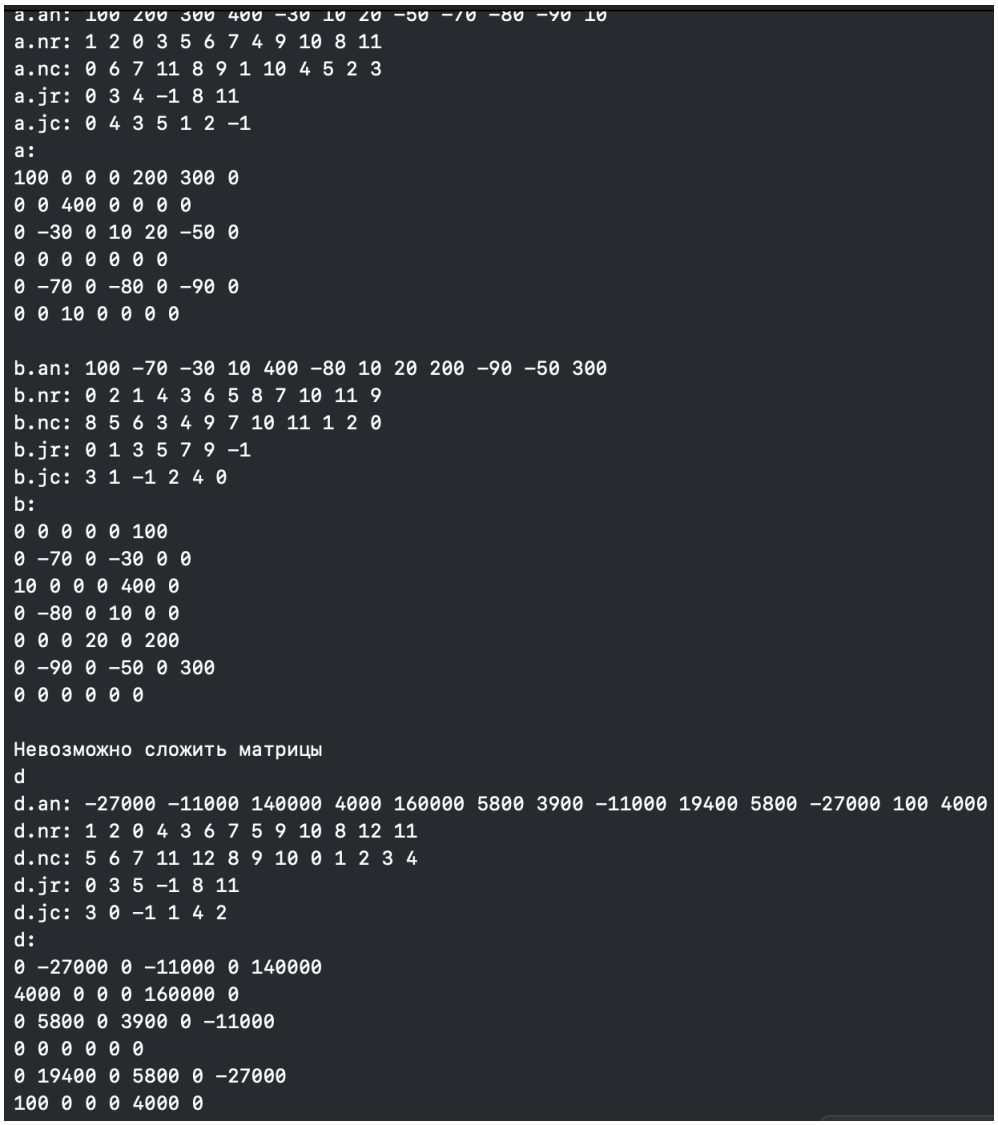


Рис. 13 – Вывод теста 13

2. Тест 14: Входные файлы: test14.1.txt, test14.2.txt — две нулевые матрицы
   Вывод программы — нулевое произведение двух нулевых матриц:

```
Выберите схему, с которой хотите работать:
1 – Схема Дженнингса, 2 – Кольцевая схема
2
a.an:
a.nr:
a.nc:
a.jr: -1 -1 -1
a.jc: -1 -1 -1 -1
a:
0 0 0 0
0 0 0 0
0 0 0 0

b.an:
b.nr:
b.nc:
b.jr: -1 -1 -1 -1
b.jc: -1 -1 -1
b:
0 0 0
0 0 0
0 0 0
0 0 0

Невозможно сложить матрицы
d
d.an:
d.nr:
d.nc:
d.jr: -1 -1 -1
d.jc: -1 -1 -1
d:
0 0 0
0 0 0
0 0 0
```

Рис. 14 – Вывод теста 14

3. Тест 15: Входные файлы: test14.1.txt, matrix5.txt — нулевая матрица A и ненулевая матрица B

   Вывод программы — нулевое произведение одной нулевой и одной ненулевой матрицы:

```
a.an:
a.nr:
a.nc:
a.jr: -1 -1 -1 -1 -1 -1
a.jc: -1 -1 -1 -1 -1 -1 -1
a:
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0

b.an: 100 -70 -30 10 400 -80 10 20 200 -90 -50 300
b.nr: 0 2 1 4 3 6 5 8 7 10 11 9
b.nc: 8 5 6 3 4 9 7 10 11 1 2 0
b.jr: 0 1 3 5 7 9 -1
b.jc: 3 1 -1 2 4 0
b:
0 0 0 0 0 100
0 -70 0 -30 0 0
10 0 0 0 400 0
0 -80 0 10 0 0
0 0 0 20 0 200
0 -90 0 -50 0 300
0 0 0 0 0 0

Невозможно сложить матрицы
d
d.an:
d.nr:
d.nc:
d.jr: -1 -1 -1 -1 -1 -1
d.jc: -1 -1 -1 -1 -1 -1 -1
d:
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
```

Рис. 15 – Вывод теста 15

4. Тест 16: Входные файлы: matrix4.txt, test14.2.txt — ненулевая матрица A и нулевая матрица B

   Вывод программы — нулевое произведение одной нулевой и одной ненулевой матрицы:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
2
a.an: 100 200 300 400 -30 10 20 -50 -70 -80 -90 10
a.nr: 1 2 0 3 5 6 7 4 9 10 8 11
a.nc: 0 6 7 11 8 9 1 10 4 5 2 3
a.jr: 0 3 4 -1 8 11
a.jc: 0 4 3 5 1 2 -1
a:
100 0 0 0 200 300 0
0 0 400 0 0 0 0
0 -30 0 10 20 -50 0
0 0 0 0 0 0 0
0 -70 0 -80 0 -90 0
0 0 10 0 0 0 0

b.an:
b.nr:
b.nc:
b.jr: -1 -1 -1 -1 -1 -1 -1
b.jc: -1 -1 -1 -1 -1 -1
b:
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

Невозможно сложить матрицы
d
d.an:
d.nr:
d.nc:
d.jr: -1 -1 -1 -1 -1 -1
d.jc: -1 -1 -1 -1 -1 -1
d:
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

Рис. 16 – Вывод теста 16

5. Тест 17: Входные файлы: matrix1.txt, matrix3.txt — матрицы, которые невозможно перемножить

   Вывод программы — ошибка:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
2
a.an: 1 2 7 3 -1 7 4 8 -1 8 5
a.nr: 0 2 1 4 3 6 7 5 9 8 10
a.nc: 0 5 6 8 7 1 9 4 3 2 10
a.jr: 0 1 3 5 8 10
a.jc: 0 1 3 2 4 10
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b.an: 10 20 30 40 50 60 70 80 90
b.nr: 1 0 3 4 2 5 7 8 6
b.nc: 3 1 5 7 8 6 2 0 4
b.jr: 0 -1 2 5 6 -1
b.jc: -1 2 -1 0 -1 4 1
b:
0 0 0 10 0 0 20
0 0 0 0 0 0 0
0 30 0 40 0 50 0
0 60 0 0 0 0 0
0 70 0 80 0 90 0
0 0 0 0 0 0 0

Невозможно сложить матрицы
Невозможно перемножить матрицы
Program ended with exit code: 0
```

Рис. 17 – Вывод теста 17

6. Тест 18: Входные файлы: test18.1.txt, test18.2.txt — матрица 3 на 2 и 2 на 3 Вывод программы — произведение двух матриц:

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
2
a.an: 4 8 1 -5 6
a.nr: 1 0 2 4 3
a.nc: 2 4 3 0 1
a.jr: 0 2 3
a.jc: 0 1
a:
4 8
1 0
-5 6

b.an: 5 4
b.nr: 0 1
b.nc: 0 1
b.jr: 0 1
b.jc: 1 0 -1
b:
0 5 0
4 0 0

Невозможно сложить матрицы
d
d.an: 32 20 5 24 -25
d.nr: 1 0 2 4 3
d.nc: 3 2 4 0 1
d.jr: 0 2 3
d.jc: 0 1 -1
d:
32 20 0
0 5 0
24 -25 0

Program ended with exit code: 0
```

Рис. 18 – Вывод теста 18

# 9 Примеры работы

## 9.1 Схема Дженнингса

Ниже представлены примеры работы функций для матриц, упакованных по схеме Дженнингса:

1. На вход подаются две произвольные матрицы, программа выводит их сумму

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
1
a_an: 1 2 3 7 0 4 -1 8 0 5
a_d: 0 1 2 5 8 9
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b_an: 10 20 30 40 0 -7 0 1 -10 1 2 3 7
b_d: 0 2 4 7 11 12
b:
10 20 0 0 0 0
20 30 40 -7 -10 0
0 40 0 0 1 0
0 -7 0 1 2 0
0 -10 1 2 3 0
0 0 0 0 0 7
```

```
c_an: 11 20 32 40 3 5 -10 0 10 3 12
c_d: 0 2 4 5 9 10
c:
11 20 0 0 0 0
20 32 40 0 -10 0
0 40 3 0 0 0
0 0 0 5 10 0
0 -10 0 10 3 0
0 0 0 0 0 12

Program ended with exit code: 0
```

Рис. 19 – Пример работы 1

2. На вход подаются матрицы разных размерностей, которые невозможно сложить, программа выдает ошибку

```
Выберите схему, с которой хотите работать:
1 – Схема Дженнингса, 2 – Кольцевая схема
1
a_an: 1 2 8 3 9 14 4 10 15 19 5 11 16 20 23 6 12 17 21 24 26 7 13 18 22 25 27 28
a_d: 0 2 5 9 14 20 27
a:
1 2 3 4 5 6 7
2 8 9 10 11 12 13
3 9 14 15 16 17 18
4 10 15 19 20 21 22
5 11 16 20 23 24 25
6 12 17 21 24 26 27
7 13 18 22 25 27 28

b_an: 10 20 30 40 0 -7 0 1 -10 1 2 3 7
b_d: 0 2 4 7 11 12
b:
10 20 0 0 0 0
20 30 40 -7 -10 0
0 40 0 0 1 0
0 -7 0 1 2 0
0 -10 1 2 3 0
0 0 0 0 0 7

Невозможно сложить матрицы
Program ended with exit code: 0
```

Рис. 20 – Пример работы 2

## 9.2 Кольцевая схема Рейнбольдта-Местеньи

Ниже представлены примеры работы функций для матриц, упакованных по схеме кольцевой схема Рейнбольдта-Местеньи

1. На вход подаются две произвольные матрицы, программа выводит их произведение

```
a.an: 1 2 7 3 -1 7 4 8 -1 8 5
a.nr: 0 2 1 4 3 6 7 5 9 8 10
a.nc: 0 5 6 8 7 1 9 4 3 2 10
a.jr: 0 1 3 5 8 10
a.jc: 0 1 3 2 4 10
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b.an: 10 20 30 40 -7 1 -10 1 2 3 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
b.nr: 1 2 3 0 5 6 4 8 9 7 11 12 13 10 15 16 17 18 19 14 21 22 23 24 25 20
b.nc: 7 4 9 12 8 11 13 14 15 10 16 17 18 19 20 21 22 23 24 25 0 1 2 5 3 6
b.jr: 0 4 7 10 14 20
b.jc: 0 1 2 5 3 6
b:
10 20 30 0 40 0
0 -7 0 1 0 -10
1 2 3 0 0 0
0 0 7 7 7 7
7 7 7 7 7 7
7 7 7 7 7 7

c
c.an: 11 20 30 40 -5 8 -10 1 2 6 -1 7 7 11 15 7 7 7 6 15 7 7 7 7 7 7 7 12
c.nr: 1 2 3 0 5 6 4 8 9 10 7 12 13 14 15 11 17 18 19 20 21 16 23 24 25 26 27 22
c.nc: 7 4 9 10 8 13 15 16 11 12 14 17 18 19 20 21 22 23 24 25 26 27 0 1 2 5 3 6
c.jr: 0 4 7 11 16 22
c.jc: 0 1 2 5 3 6
c:
11 20 30 0 40 0
0 -5 0 8 0 -10
1 2 6 0 -1 0
0 7 7 11 15 7
7 7 6 15 7 7
7 7 7 7 7 12
```

Рис. 21 – Пример работы 3

2. На вход подаются матрицы разных размерностей, которые невозможно сложить, программа выдает ошибку

```
Выберите схему, с которой хотите работать:
1 - Схема Дженнингса, 2 - Кольцевая схема
2
a.an: 1 2 7 3 -1 7 4 8 -1 8 5
a.nr: 0 2 1 4 3 6 7 5 9 8 10
a.nc: 0 5 6 8 7 1 9 4 3 2 10
a.jr: 0 1 3 5 8 10
a.jc: 0 1 3 2 4 10
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b.an: 1 2 3 4 5 6 7 2 8 9 10 11 12 13 3 9 14 15 16 17 18 4 10 15 19 20 21 22 5 11 16 20 23 24 25 6
12 17 21 24 26 27 7 13 18 22 25 27 28
b.nr: 1 2 3 4 5 6 0 8 9 10 11 12 13 7 15 16 17 18 19 20 14 22 23 24 25 26 27 21 29 30 31 32 33 34
28 36 37 38 39 40 41 35 43 44 45 46 47 48 42
b.nc: 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
39 40 41 42 43 44 45 46 47 48 0 1 2 3 4 5 6
b.jr: 0 7 14 21 28 35 42
b.jc: 0 1 2 3 4 5 6
b:
1 2 3 4 5 6 7
2 8 9 10 11 12 13
3 9 14 15 16 17 18
4 10 15 19 20 21 22
5 11 16 20 23 24 25
6 12 17 21 24 26 27
7 13 18 22 25 27 28

Невозможно сложить матрицы
```

Рис. 22 – Пример работы 3

3. На вход подаются две произвольные матрицы, программа выводит их сумму

```
a.an: 100 200 300 400 -30 10 20 -50 -70 -80 -90 10
a.nr: 1 2 0 3 5 6 7 4 9 10 8 11
a.nc: 0 6 7 11 8 9 1 10 4 5 2 3
a.jr: 0 3 4 -1 8 11
a.jc: 0 4 3 5 1 2 -1
a:
100 0 0 0 200 300 0
0 0 400 0 0 0 0
0 -30 0 10 20 -50 0
0 0 0 0 0 0 0
0 -70 0 -80 0 -90 0
0 0 10 0 0 0 0

b.an: 100 -70 -30 10 400 -80 10 20 200 -90 -50 300
b.nr: 0 2 1 4 3 6 5 8 7 10 11 9
b.nc: 8 5 6 3 4 9 7 10 11 1 2 0
b.jr: 0 1 3 5 7 9 -1
b.jc: 3 1 -1 2 4 0
b:
0 0 0 0 0 100
0 -70 0 -30 0 0
10 0 0 0 400 0
0 -80 0 10 0 0
0 0 0 20 0 200
0 -90 0 -50 0 300
0 0 0 0 0 0

Невозможно сложить матрицы
d
d.an: -27000 -11000 140000 4000 160000 5800 3900 -11000 19400 5800 -27000 100 4000
d.nr: 1 2 0 4 3 6 7 5 9 10 8 12 11
d.nc: 5 6 7 11 12 8 9 10 0 1 2 3 4
d.jr: 0 3 5 -1 8 11
d.jc: 3 0 -1 1 4 2
d:
0 -27000 0 -11000 0 140000
4000 0 0 0 160000 0
0 5800 0 3900 0 -11000
0 0 0 0 0 0
0 19400 0 5800 0 -27000
100 0 0 0 4000 0
```

Рис. 23 – Пример работы 4

4. На вход подаются матрицы, которые невозможно умножить, программа выдает ошибку

```
Выберите схему, с которой хотите работать:
1 – Схема Дженнингса, 2 – Кольцевая схема
2
a.an: 1 2 7 3 -1 7 4 8 -1 8 5
a.nr: 0 2 1 4 3 6 7 5 9 8 10
a.nc: 0 5 6 8 7 1 9 4 3 2 10
a.jr: 0 1 3 5 8 10
a.jc: 0 1 3 2 4 10
a:
1 0 0 0 0 0
0 2 0 7 0 0
0 0 3 0 -1 0
0 7 0 4 8 0
0 0 -1 8 0 0
0 0 0 0 0 5

b.an: 10 20 30 40 50 60 70 80 90
b.nr: 1 0 3 4 2 5 7 8 6
b.nc: 3 1 5 7 8 6 2 0 4
b.jr: 0 -1 2 5 6 -1
b.jc: -1 2 -1 0 -1 4 1
b:
0 0 0 10 0 0 20
0 0 0 0 0 0 0
0 30 0 40 0 50 0
0 60 0 0 0 0 0
0 70 0 80 0 90 0
0 0 0 0 0 0 0

Невозможно сложить матрицы
Невозможно перемножить матрицы
Program ended with exit code: 0
```

Рис. 24 – Пример работы 4

# 10 Заключение

Цель достигнута:

1. Рассмотрены две схемы хранения матриц и реализованы функции:
   (a) $show\_one\_dim$ — вывод на экран одномерный вектор;
   (b) $show\_two\_dim$ — вывод на экран двумерный вектор;
   (c) $comp\_Djen$ — упаковка матрицы по схеме Дженнингса;
   (d) $sum\_Djen$ — сумма двух матриц, упакованных по схеме Дженнингса;
   (e) $unpack\_Djen$ — распаковка матрицы, упакованной по схеме Дженнингса.
   (f) $add\_element$ — добавления элемента в упакованную по кольцевой схеме матрицу;
   (g) $comp\_ring\_RM$ — упаковка матрицы по кольцевой схеме;
   (h) $col\_el$ — поиск столбцовой координаты элемента;
   (i) $row\_el$ — поиск строчной координаты элемента;
   (j) $unpack\_ring\_RM$ — распаковка матрицы, упакованной по кольцевой схеме;
   (k) $sum\_ring\_RM$ — сумма двух матриц, упакованных по кольцевой схеме;
   (l) $mult\_ring\_RM$ — произведение двух матриц, упакованных по кольцевой схеме.
2. Выполнено тестирование реализации разработанных функций.
3. Проведен анализ эффективности сжатого хранения матриц в обоих схемах для матрицы действительных чисел с 5 % ненулевых элементов.

# Список литературы

1. Писсанецки, С. Технология разреженных матриц [Текст]: монография / С.Писсанецки.; пер. с анг. под ред. Х.Д.Икрамова. - М.: Мир, 1988. - 410 с.