МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

“МИРЭА - Российский технологический университет”

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4
по дисциплине «Структуры и алгоритмы обработки данных»

Тема «Многомерные массивы»

Выполнил студент группы ИКБО-50-23 Русаков М.Ю.

Принял старший преподаватель Скворцова Л.А.

Москва 2024

Оглавление

1. Условие задачи

Задания моего персонального варианта (№22):

Задание 1. Разработать АТД задачи варианта по управлению многомерными данными и реализовать на статическом многомерном массиве.

Задание 2. Разработать АТД задачи варианта по управлению многомерными данными и реализовать на динамическом многомерном массиве.

Задание 3. Разработать программу решения задачи варианта по управлению многомерными данными и реализовать с применением шаблона `<vector>` библиотеки STL.

2. Постановка задачи

Задания 1 и 2. Дана квадратная матрица. Найти определитель данной матрицы методом Гаусса.

Задание 3. На плоскости задано множество точек с целочисленными координатами. Необходимо найти количество отрезков, обладающих следующими свойствами:

1. Оба конца отрезка принадлежат заданному множеству;
2. Ни один конец отрезка не лежит на осях координат;
3. Отрезок пересекается ровно с одной осью координат.

Напишите эффективную по времени и по используемой памяти программу для решения этой задачи.

3. Математическая модель

3.1. Задания 1 и 2.

Допустим, у нас есть матрица следующего вида:

$$A = \begin{pmatrix} 1 & 3 & 3 & 7 \\ 3 & 4 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 6 & 9 & 6 & 9 \end{pmatrix}$$

Согласно методу Гаусса, для вычисления определителя нужно сначала привести матрицу к треугольному виду, а затем найти произведение элементов на главной диагонали. Значение полученного произведения (назовем его Δ) и будет определителем матрицы.

Приведем матрицу A к диагональному виду:

$$\Delta = \begin{vmatrix} 1 & 3 & 3 & 7 \\ 3 & 4 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 6 & 9 & 6 & 9 \end{vmatrix} = \begin{vmatrix} 1 & 3 & 3 & 7 \\ 0 & -5 & -6 & -17 \\ 0 & -9 & -8 & -27 \\ 0 & -9 & -12 & -33 \end{vmatrix} = \begin{vmatrix} 1 & 3 & 3 & 7 \\ 0 & -5 & -6 & -17 \\ 0 & 0 & \frac{14}{5} & \frac{18}{5} \\ 0 & 0 & -\frac{6}{5} & -\frac{12}{5} \end{vmatrix} = \begin{vmatrix} 1 & 3 & 3 & 7 \\ 0 & -5 & -6 & -17 \\ 0 & 0 & \frac{14}{5} & \frac{18}{5} \\ 0 & 0 & 0 & -\frac{6}{7} \end{vmatrix}$$

В данном случае $\Delta = 1 \times (-5) \times \frac{14}{5} \times \left(-\frac{6}{7}\right) = 12$.

Как видно из приведенного примера, приведение к треугольному виду представляет из себя «обнуление» элементов, стоящих под главной диагональю. Для того, чтобы добиться этого в общем случае, нужно из текущей строки вычесть строку, располагающуюся выше текущей и умноженную на некоторый коэффициент (назовем его λ).

В нашем примере $\lambda_1 = \frac{a_{21}}{a_{11}} = 3$, $\lambda_2 = \frac{a_{32}}{a_{21}} = \frac{9}{5}$, $\lambda_3 = \frac{a_{43}}{a_{32}} = -\frac{6}{14}$.

Видно, что общем случае для n-й строки ($n > 1, n \in \mathbb{N}$) получается:

$$\lambda_n = \frac{a_{(n+1)n}}{a_{nn}}$$

Тогда элемент n-й строки m-го столбца будет равен:

$$a_{nm} = a_{nm} - \lambda a_{(n-1)m}$$

Определитель же в общем виде будет находиться по следующей формуле:

$$\Delta = \Pi_1^n a_{nn}$$

3.2. Задание 3

Построим математическую модель задачи на основе рассмотрения примера (см. рис. 1).

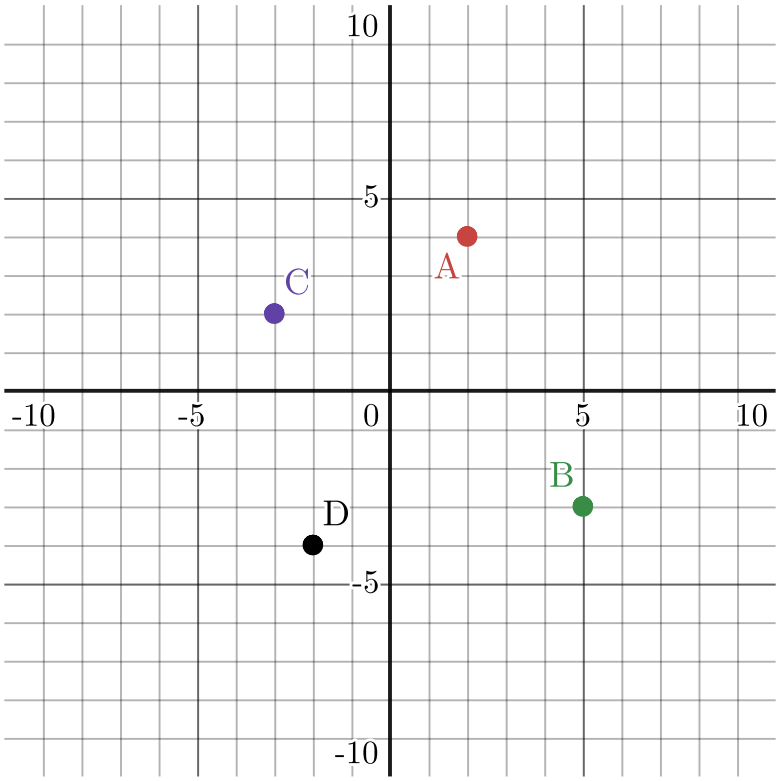

Рисунок 1 – Пример взаимного расположения точек на координатой плоскости

Как видно из рис. 1, точки, лежащие в соседних квадрантах, образуют отрезки, подходящие под критерии задачи (AB, AC, BD, CD), в то время как точки, лежащие в противолежащих квадрантах, таковых отрезков не образуют (AD, BC). Действительно, среди соответственных координат точек подходящих отрезков произведение ровно одной пары будет меньше нуля (например, для точек A и B : $y_A \times y_B = 2 \times (-3) = -6 < 0$, но $x_A \times x_B = 2 \times 5 = 10 > 0$), а для прочих отрезков произведение обеих пар будет одновременно либо больше, либо меньше нуля.

Отсюда получим следующую функцию:

$$F(x_1, x_2, y_1, y_2) = (x_1 \times x_2 < 0) \oplus (y_1 \times y_2 < 0),$$

где x_1, x_2, y_1, y_2 – координаты точек отрезка.

Значение функции F для отрезков, пересекающих ровно одну ось координат, всегда будет равно 1, а для прочих – 0.

4. АТД задачи

4.1. АТД для заданий 1 и 2

АТД matrix
{

Данные (описание свойств структуры данных задачи)

MAX_SIZE - максимальная размерность матрицы

size - размерность текущего массива

array - двумерный массив, содержащий действительные числа

Операции (объявления операций)

1. Метод, осуществляющий заполнение с клавиатуры
Предусловие: array, size
Постусловие: array, заполненный действительными числами

Заголовок: `fillManually()`

2. Метод, заполняющий массив случайными значениями
Предусловие: array, size
Постусловие: массив array, заполненный случайными значениями

Заголовок: `fillRandomly()`

3. Метод, осуществляющий вывод текущих значений массива
Предусловие: array, size
Постусловие: выведенные через пробел элементы массива

Заголовок: `print()`

4. Метод, возвращающий значение определителя матрицы
Предусловие: array, size
Постусловие: действительное число - значение определителя матрицы

Заголовок: `det()`

}

4.2. АТД для задания 3

АТД Set
{

Данные (описание свойств структуры данных задачи)

points - вектор, содержащий пары целочисленных значений (координаты точки)

1. Метод, осуществляющий заполнение с клавиатуры
Предусловие: array, size
Постусловие: array, заполненный действительными числами

Заголовок: `fillManually()`

2. Метод, заполняющий массив случайными значениями
Предусловие: array, size
Постусловие: массив array, заполненный случайными значениями

Заголовок: `fillRandomly()`

3. Метод, осуществляющий вывод текущих значений массива
Предусловие: array, size
Постусловие: выведенные через пробел элементы массива

Заголовок: `print()`

4. Метод, возвращающий `true`, если хотя бы одна точка отрезка лежит на осях координат, иначе `false`
Предусловие: координаты точек концов отрезка
Постусловие: `true`/`false`

Заголовок: `onAxis(std::pair<int, int>& point1, std::pair<int, int>& point2)`

5. Метод, возвращающий `true`, если отрезок пересекает ровно одну ось координат, иначе `false`
Предусловие: координаты конца точек отрезка
Постусловие: `true`/`false`

Заголовок: `singleAxisIntersection(std::pair<int, int>& point1, std::pair<int, int>& point2`)

6. Метод, возращающий случайно сгенерированное целое число в диапазоне $[-100, 100]$
Предусловие: *сильно зависит от конкретного языка*

программирования
Постусловие: целое число из диапазона [−100, 100]

Заголовок: `generate(std::uniform_int_distribution<>& dis, std::mt19937& generator)`

7. Метод, возвращающий количество отрезков, удовлетворяющих условию задачи 3
Предусловие: points
Постусловие: целое число - количество отрезков

Заголовок: `validSegments()`

}

5. Разработка и реализация задачи

5.1. Код программы

5.1.1. Задание 1

1. Код файла `matrixStatic.h`

```
struct matrix {
private:
    static const int MAX_SIZE = 10;
    int size;
    double array[MAX_SIZE][MAX_SIZE];

public:
    matrix(int size);

    void fillRandomly();

    void fillManually();

    void print();

    double det();
};
```

2. Код файла `matrixStatic.cpp`

```
#include "matrixStatic.h"
#include <iostream>
#include <random>
#include <ctime>
```

```
matrix::matrix(int size)
{
    this->size = size;
}

void matrix::fillManually()
{
    for (int i{}; i < size; i++)
    {
        for (int j{}; j < size; j++)
        {
            std::cin >> array[i][j];
        }
    }
}

void matrix::fillRandomly()
{
    srand(std::time(0));

    for (int i{}; i < size; i++)
    {
        for (int j{}; j < size; j++)
        {
            int num = rand() % 101;
            array[i][j] = num;
        }
    }
}

void matrix::print()
{
    for (int i{}; i < size; i++)
    {
        for (int j{}; j < size; j++)
        {
            std::cout << array[i][j] << ' ';
        }

        std::cout << '\n';
    }
}

double matrix::det()
{
    double det = 1;
```

```
auto copy = array;

for (int i = 0; i < size; ++i)
{
    bool found = false;
    if (copy[i][i] == 0)
    {
        found = true;
        for (int j = i; j < size; ++j)
        {
            if (copy[j][i] != 0)
            {
                det *= -1.0;
                for (int k = 0; k < size; ++k)
                {
                    double t = copy[i][k];
                    copy[i][k] = copy[j][k];
                    copy[j][k] = t;
                    found = false;
                }
            }
        }
    }

    if (found == true)
    {
        return 0.0;
    }

    else
    {

        for (int j = i+1; j < size; ++j)
        {
            double store = copy[j][i];
            for (int k = i; k < size; ++k)
            {
                copy[j][k] -= (copy[i][k]*store)/copy[i][i];
            }
        }
        det *= copy[i][i];
    }
}
```

```
    return det;
}
```

3. Код файла `main.cpp`

```
#include "matrixStatic.h"
#include <iostream>

int main()
{
    matrix m(3);

    m.fillManually();

    std::cout << "Определитель матрицы = " << m.det();
}
```

5.1.2. Задание 2

1. Код файла `matrixDynamic.h`

```
// Квадратная матрица
struct matrix {
private:
    int size;
    double** array;

public:
    matrix(int size);

    void fillRandomly();

    void fillManually();

    void print();

    double det();
};
```

2. Код файла `matrixDynamic.cpp`

```
#include "../headers/matrix_dynamic_array.h"
#include <iostream>

matrix::matrix(int size) {
    this->size = size;

    array = new double*[size]{};
    for (int i{}; i < size; i++)
```

```
    {
        array[i] = new double[size]{};
    }
}

void matrix::fillManually() {
    for (int i{}; i < size; i++) {
        for (int j{}; j < size; j++) {
            std::cin >> array[i][j];
        }
    }
}

// Числа генерируются в диапазоне [0;100]
void matrix::fillRandomly() {
    for (int i{}; i < size; i++) {
        for (int j{}; j < size; j++) {
            double num = static_cast<double>(rand() % 101);
            array[i][j] = num;
        }
    }
}

void matrix::print() {
    for (int i{}; i < size; i++) {
        for (int j{}; j < size; j++) {
            std::cout << array[i][j] << ' ';
        }

        std::cout << '\n';
    }
}

double matrix::det() {
    double det = 1;

    for (int i = 0; i < size; ++i) {
        bool found = false;
        if (array[i][i] == 0) {
            found = true;
            for (int j = i; j < size; ++j) {
                if (array[j][i] != 0) {
                    det *= -1;
                    for (int k = 0; k < size; ++k) {
                        double t = array[i][k];
                        array[i][k] = array[j][k];
```

```
                    array[j][k] = t;
                    found = false;
                }
            }
        }

        if (found == true) {
            return 0.0;
        }

        else {

            for (int j = i+1; j < size; ++j) {
                double store = array[j][i];
                for (int k = i; k < size; ++k) {
                    array[j][k] -= (array[i][k]*store)/array[i][i];

                }
            }
            det *= array[i][i];
        }
    }

    return det;
}
```

3. Код файла `main.cpp`

```
#include <iostream>
#include "matrixDynamic.h"

int main()
{
    matrix m(5);

    m.fillRandomly();

    m.print();

    std::cout << m.det();
}
```

5.1.3. Задание 3

1. Код файла `matrixVector.h`

```
#include <vector>
#include <random>
#include <ctime>


struct Set {
private:
    std::vector<std::pair<int, int>> points;

    bool onAxis(std::pair<int, int>& point1, std::pair<int, int>&
point2);
    bool singleAxisIntersection(std::pair<int, int>& point1,
std::pair<int, int>& point2);
    int generate(std::uniform_int_distribution<>& dis, std::mt19937&
generator);

public:
    Set(int size);

    void fillRandomly(std::uniform_int_distribution<>& dis,
std::mt19937& generator);

    void fillManually();

    void print();

    int validSegments();
};
```

2. Код файла matrixVector.cpp

```
#include <iostream>
#include <ctime>
#include <random>
#include "matrixVector.h"

Set::Set(int size)
{
    points.assign(size, std::make_pair(0, 0));
}

void Set::fillManually()
{
    for (int i{}; i < points.size(); i++)
    {
        std::cin >> points[i].first;
        std::cin >> points[i].second;
```

```
    }
}

int Set::generate(std::uniform_int_distribution<>& dis, std::mt19937&
generator)
{
    return dis(generator);
}

void Set::fillRandomly(std::uniform_int_distribution<>& dis,
std::mt19937& generator)
{
    for (int i{}; i < points.size(); i++)
    {
        points[i].first  = generate(dis, generator);
        points[i].second = generate(dis, generator);
    }
}

void Set::print()
{
    for (int i{}; i < points.size(); i++)
    {
        printf("(%d;%d)\n", points[i].first, points[i].second);
    }
}

bool Set::onAxis(std::pair<int, int>& point1, std::pair<int, int>&
point2)
{
    return (point1.first == 0 || point1.second == 0 || point2.first ==
0 || point2.second == 0);
}

bool Set::singleAxisIntersection(std::pair<int, int>& point1,
std::pair<int, int>& point2)
{
    return !(point1.first * point2.first < 0 == point1.second *
point2.second < 0);
}

int Set::validSegments()
{
    int number{};

    for (int i{}; i < points.size() - 1; i++)
```

```
    {
        for (int j = i + 1; j < points.size(); j++)
        {
            if (singleAxisIntersection(points[i], points[j]) && !
onAxis(points[i], points[j]))
            {
                number++;
            }
        }
    }

    return number;
}
```

3. Код файла `main.cpp`

```
#include <iostream>
#include <random>
#include <chrono>
#include "matrixVector.h"

int main()
{
    unsigned seed =
std::chrono::system_clock::now().time_since_epoch().count();
    std::mt19937 generator(seed);
    std::uniform_int_distribution<> dis(-100, 100);

    Set points(4);

    points.fillManually();

    points.print();

    std::cout << "Количество отрезков, удовлетворяющих условию: " <<
points.validSegments() << '\n';
}
```

5.2. Набор тестов

5.2.1. Задания 1 и 2

Таблица 1 – Таблица тестов для заданий 1 и 2

Номер	Входные данные	Ожидаемые выходные данные
1	size = 3 array = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}	0
2	size = 4 array = {{1, 3, 3, 7}, {3, 4, 3, 4}, {5, 6, 7, 8}, {6, 9, 6, 9}}	18

5.2.2. Задание 3

Таблица 2 – Таблица тестов для задания 3

Номер	Входные данные	Ожидаемые выходные данные
1	points = {{1,2}, {3,4}, {5,-2}, {-3,-5}}	3
2	points = {{1,0}, {0,1}, {0,0}, {6,9}}	0

5.3. Результаты тестирования

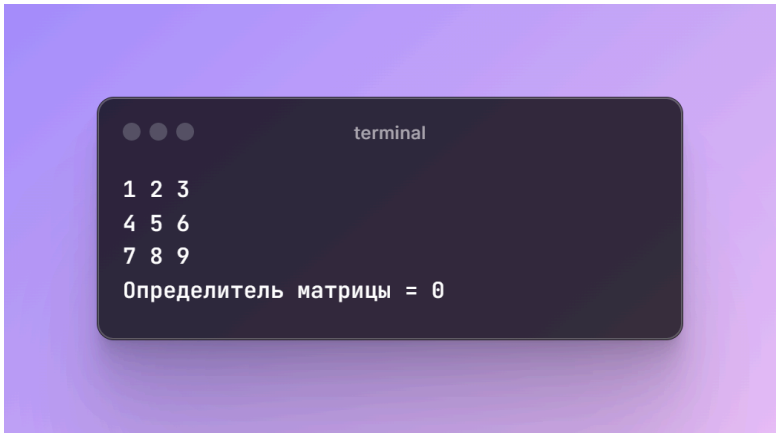

Рисунок 2 – Результаты тестирования теста №1 для заданий 1 и 2

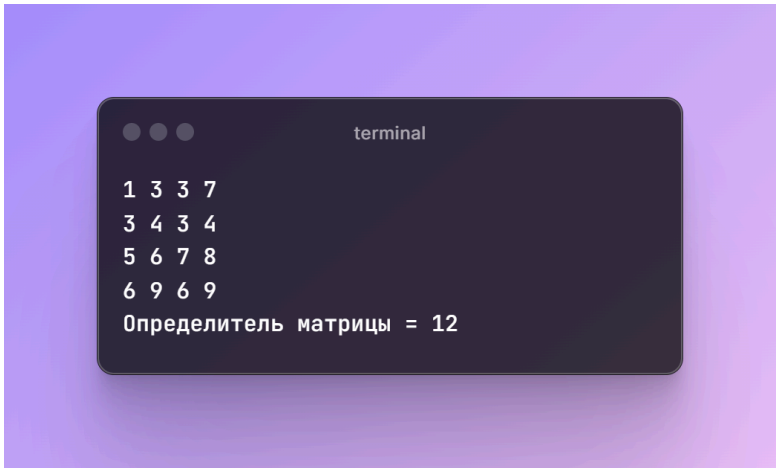

Рисунок 3 – Результаты тестирования теста №2 для заданий 1 и 2

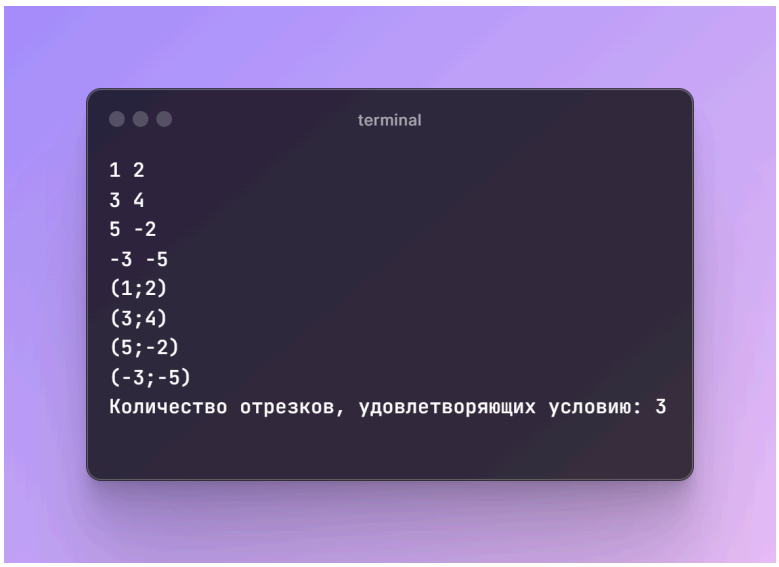

Рисунок 4 – Результаты тестирования теста №1 для задания 3

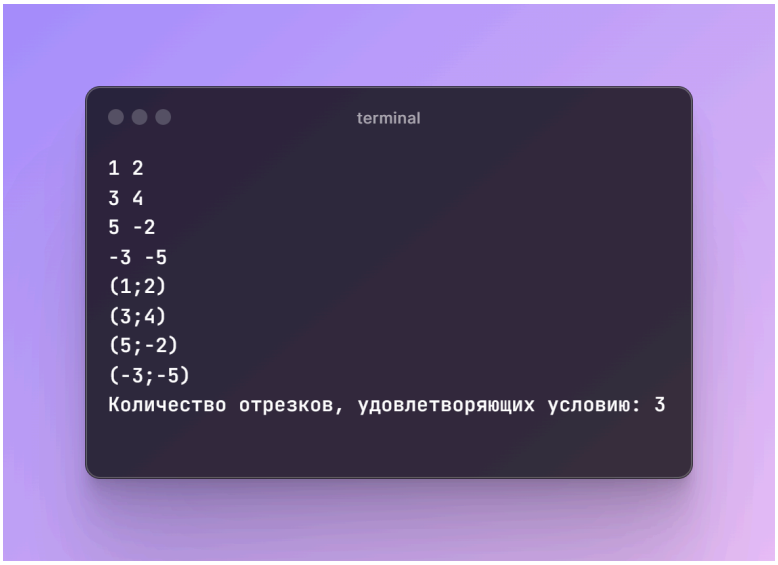

Рисунок 5 – Результаты тестирования теста №2 для задания 3

6. Вывод

В результате работе были получены навыки по определению многомерного статического и динамического массивов в программе, их представлению в оперативной памяти, определению структуры данных для хранения данных задачи и ее наиболее оптимальной реализации, а также разработке алгоритмов операций на многомерном (двумерном массиве) в соответствии с задачей.