

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 9 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження алгоритмів обходу масивів»

Варіант 5

Виконав студент

ІП-15, Буяло Дмитро Олександрович
(шифр, прізвище, ім'я, по батькові)

Перевірів

Вечерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 9

Дослідження алгоритмів обходу масивів

Мета – дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Індивідуальне завдання

Варіант 5

Завдання

Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом
2. Ініціювання змінної, що описана в п.1 даного завдання.
3. Обчислення змінної, що описана в п.1, згідно з варіантом

№	Опис варіанту
5	Задано матрицю дійсних чисел $A[n,n]$, ініціалізувати матрицю обходом по рядках. Знайти суму елементів, розташованих нижче головної діагоналі матриці.

1. Постановка задачі

1) Термінологія в формуванні задачі повністю зрозуміла та не потребує пояснень. 2) Маємо розмірність матриці, тобто двовимірний масив, та дійсний тип даних значень елементів. 3) Необхідно знайти суму елементів, розташованих нижче головної діагоналі матриці, використовуючи вхідні дані, користуючись методом обходу змійкою по рядках, маємо ініціалізувати матрицю. 4) Як загальну властивість можна виділити те, що в нас задана розмірність, утворюється квадратна матриця. 5) Існує багато розв'язків даної задачі, будемо використовувати, на мою думку, найпростіший, найефективніший та найшвидший. 6) Даних цілком достатньо, всі потрібні та припущень робити не потрібно.

2. Побудова математичної моделі

Складемо таблицю імен змінних

<i>Змінна</i>	<i>Тип</i>	<i>Ім'я</i>	<i>Призначення</i>
Кількість рядків та стовпчиків	Цілочисельний	n	Вхідні дані
Двовимірний масив	Дійсний	matrix	Проміжні дані
Елемент матриці	Дійсний	count	Проміжні дані
Змінна напрямку	Цілочисельний	dir	Проміжні дані
Лічильник 1	Цілочисельний	i	Проміжні дані
Лічильник 2	Цілочисельний	j	Проміжні дані
Генерація масиву	Процедура	createM	Проміжні дані
Вивід матриці	Процедура	output	Проміжні дані
Сума елементів	Процедура	sumEl	Проміжні дані, результат
Сума	Дійсний	sum	Результат

Щоб ініціалізувати матрицю обходом по рядках та знайти суму елементів, розташованих нижче головної діагоналі матриці, нам потрібно застосувати підпрограми, в яких ми будемо генерувати двовимірний масив з заданою розмірністю, виводити в консоль отриману матрицю та знаходити суму елементів, розташованих нижче головної діагоналі матриці.

Вхідними параметрами функції createM є пустий двовимірний масив та його розмірність, підпрограма буде створювати масив-матрицю.

Вхідними параметрами функції output є двовимірний масив та його розмірність, підпрограма буде виводити у консоль матрицю, з якою ми працюємо.

Вхідними параметрами функції sumEl є двовимірний масив та його розмірність, підпрограма рахує суму елементів, розташованих нижче головної діагоналі матриці.

Всі виклики функцій відбуваються в основній програмі.

Також зазначимо, що оператор += - це оператор складання з привласненням, додає значення правого операнда до змінної і привласнює змінній результат.

Розв'язання

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначимо основні дії

Крок 2. Деталізуємо дію знаходження суми елементів, розташованих нижче головної діагоналі матриці

3. Псевдокод алгоритму

Основна програма

<i>Крок 1</i>	<i>Крок 2</i>
Початок	Початок
Задання розмірності	Введення n
Створення та вивід двовимірного масиву	matrix[n][n]
	createM(n, matrix)
	output(n, matrix)
Вивід суми	Вивід sumEl(n, matrix)
Кінець	Кінець

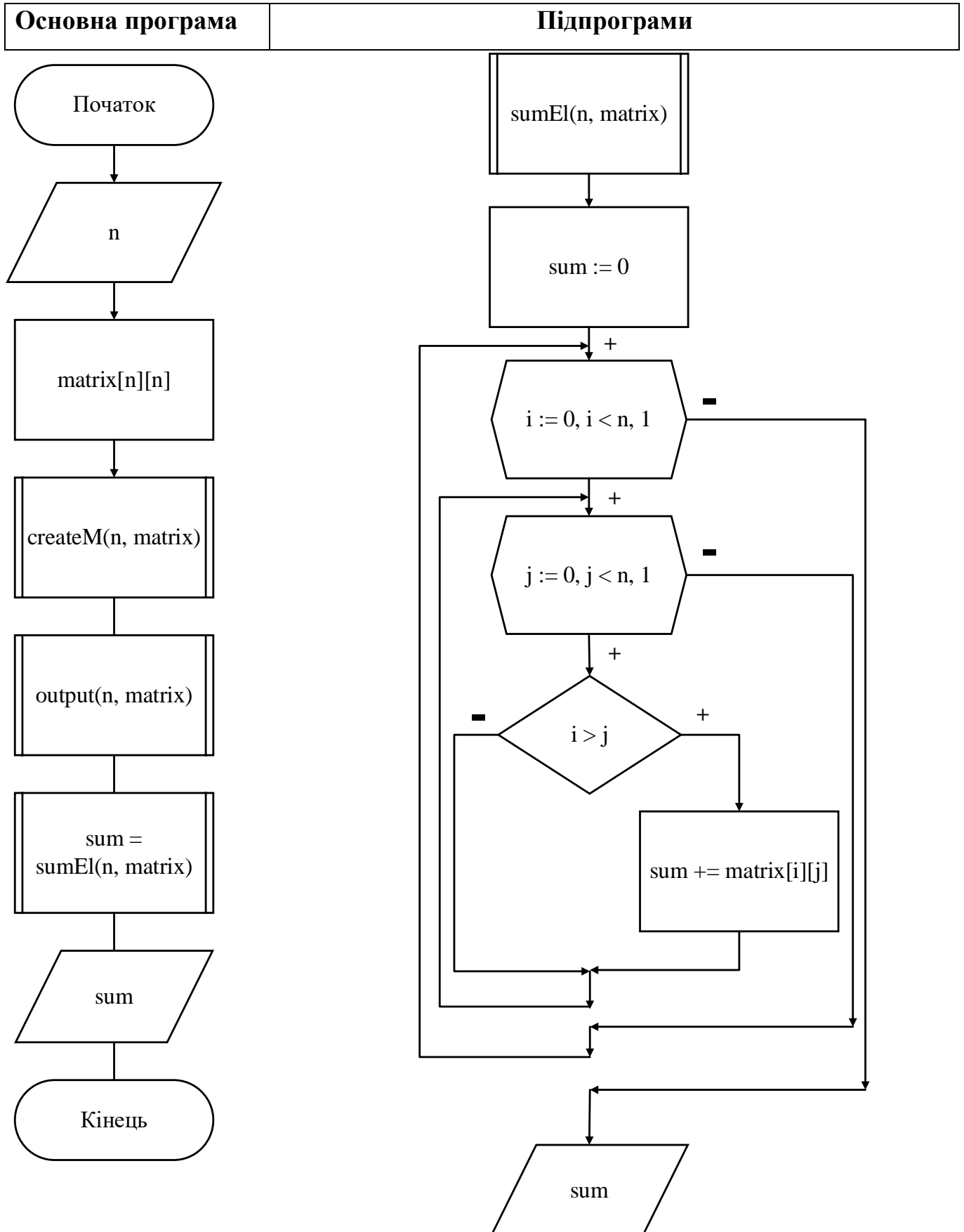
Підпрограми:

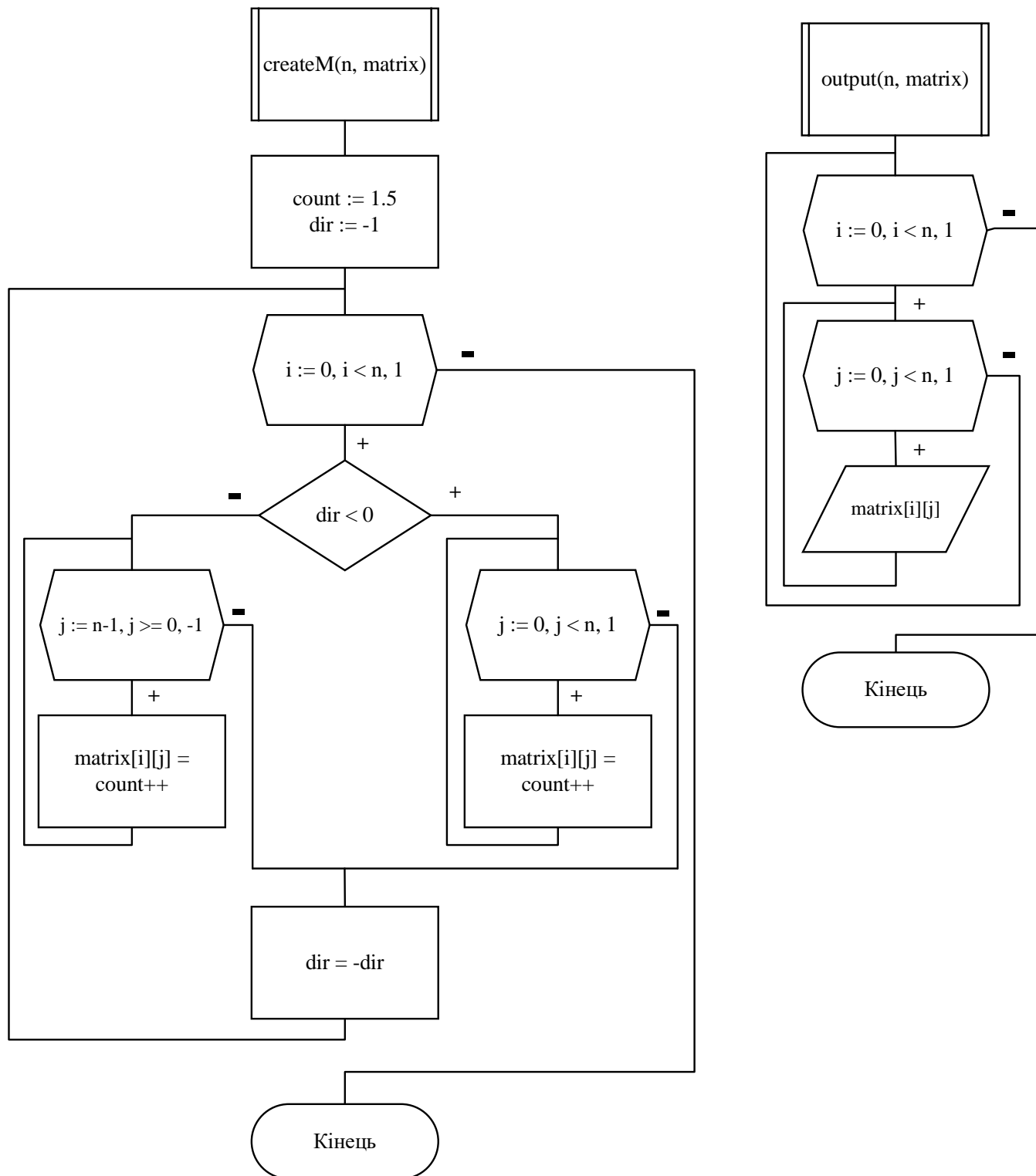
outputM(n, matrix)
Початок
Повторити для i від 0 до m
Повторити для j від 0 до n
Виведення matrix[i][j]
Все повторити
Все повторити
Кінець

createM(n, matrix)
Початок
count := 1.5
dir := -1
Повторити для i від 0 до n
Якщо dir < 0
то
Повторити для j від 0 до n
matrix[i][j] = count++
Все повторити
інакше
Повторити для j від n-1 до 0
matrix[i][j] = count++
Все повторити
dir = -dir
Все повторити
Кінець

sumEl(n, matrix)
Початок
sum=0
Повторити для j від 0 до n
Повторити для i від 0 до n
sum += matrix[i][j]
Все повторити
Все повторити
Кінець

4. Блок-схема алгоритму





5. Код програми

Код написаний на мові програмування Java

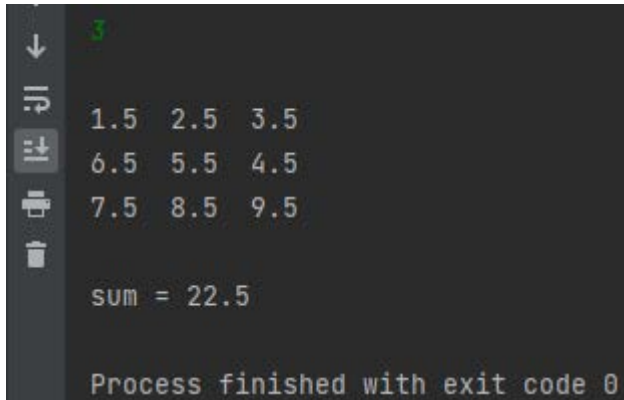
```
package com.company;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        double[][] matrix = new double[n][n];
        createMatrix(n,matrix);
        output(n,matrix);
        System.out.println("\nsum = "+sumEl(n,matrix));
    }
    public static void createMatrix(int n, double[][] matrix) {
        double count=1.5;
        int dir =-1;
        for(int i=0;i<n;i++) {
            if(dir<0) {
                for(int j = 0;j<n;j++) {
                    matrix[i][j] = count++;
                }
            }
            else {
                for(int j=n-1;j>=0;j--) {
                    matrix[i][j] = count++;
                }
            }
            dir=-dir;
        }
    }
    public static void output(int n, double[][] matrix) {
        for(int i=0; i<n;i++) {
            System.out.println();
            for(int j=0;j<n;j++) {
                System.out.print(matrix[i][j]+" ");
            }
            System.out.println();
        }
    }
    public static double sumEl(int n, double[][] matrix) {
        double sum = 0;
        for(int i=0;i<n;i++) {
            for(int j=0;j<n;j++) {
                if(i>j) {
                    sum+=matrix[i][j];
                }
            }
        }
        return sum;
    }
}
```


Результат виконання:



```
3
1.5  2.5  3.5
6.5  5.5  4.5
7.5  8.5  9.5

sum = 22.5

Process finished with exit code 0
```

Випробування алгоритм пройшов відмінно, видавши правильний кінцевий результат.

6. Висновки

Ми дослідили алгоритми обходу масивів та практичних навичок використання цих алгоритмів під час складання програмних специфікацій. В результаті виконання лабораторної роботи ми отримали алгоритм для ініціалізування матриці обходом по рядках, знаходження суми елементів, розташованих нижче головної діагоналі матриці. Дискретували задачу на 2 кроки: визначили основні дії, деталізували дію знаходження суми елементів, розташованих нижче головної діагоналі матриці. Алгоритм є ефективним та результативним, бо забезпечує розв'язок за мінімальний час із мінімальними витратами ресурсів та отримує чіткий кінцевий результат.