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»

«Організація циклічних процесів. Ітераційні цикли»

Варіант 32

Виконав студент ІП-11 Фукс Вікторія Ігорівна

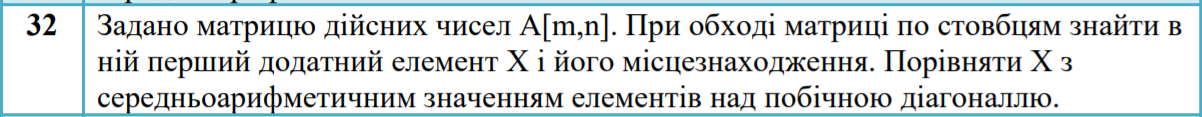
Мартинова Оксана Петрівна

Перевірив \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.

**Варіант 32:** **Постановка задачі**: Описуємо двовимірний масив mхn випадковими цілими числами. Обходимо по стовпцям та знаходимо перший додатній елемент і місцезнаходження. Порівнюємо Х з середньоарифметичним значенням елементів над побічною діагоналлю.

**Математична модель**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| сума | дійсний | suma | проміж.результ |
| лічильник | цілочисельний | count | проміж.результ |
| розмір | цілочисельний | size | проміж.результ |
| строка | цілочисельний | row | проміж.результ |
| стовбець | цілочисельний | col | проміж.результ |
| матриця | дійсний | matrix | результат |
| середнє арифм. | дійсний | aver | результат |
| перший позит. ел. | дійсний | positive\_elem | результат |
| індекс строки | цілочисельний | row\_index | результат |
| індекс стовбця | цілочисельний | col\_index | результат |

**Псевдокод:**

**Крок 1:** Визначимо основні дії

**Крок 2:** Деталізуємо умову з використанням підпрограми та умовних операторів.

**Крок 1:**

**Початок**

ввести row, col

Заповнення матриці

Знаходження першого позитивного елементу

Знаходження середнього арифметичного під побічною діагоналлю

**Кінець**

**Крок 2:**

**Підпрограма** generate\_matrix(row, col)**:**

matrix= [[random(-20, 11 for j in range(col)] for i in range(row)]

**повернути** matrix

**Кінець підпрограми**

**Підпрограма** output\_matrix(mtx)**:**

**виконати** row in matrix**:**

**вивести** row

**все виконати**

**Кінець підпрограми**

**Підпрограма** find\_positive\_elem(matrix, row, col)**:**

**виконати** j in range(col)**:**

**виконати** i in range(row) if (j % 2 == 0) **інакше** range(row - 1, -1, -1):

**якщо** matrix[i][j] > 0:

**повернути** matrix[i][j], i, j

**все якщо**

**все виконати**

**все виконати**

**Кінець підпрограми**

**Підпрограма** average\_under\_side\_diagonal(matrix, size):

suma = 0

count = 0

**виконати** i in range(size):

**виконати** j in range(size):

**якщо** (i + j) <= size:

suma += matrix[i][j]

count += 1

**все якщо**

**все виконати**

**повернути** ((suma/ count),3)

**все виконати**

**Кінець підпрограми**

**Початок**

**ввести** row, col

matrix = generate\_matrix(row, col)

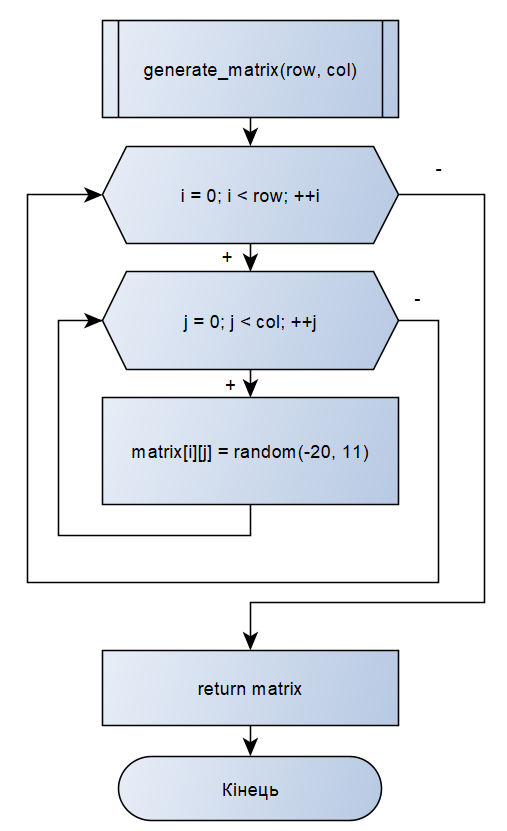
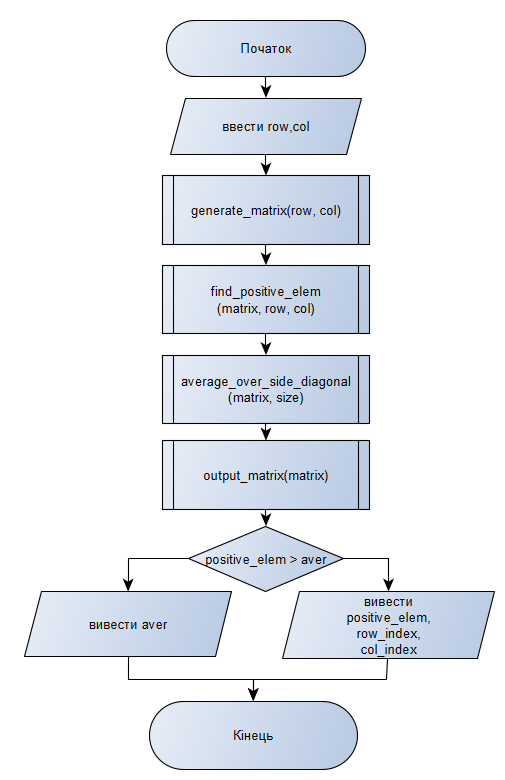
positive\_elem, row\_index, col\_index = find\_positive\_elem(matrix, row, col)

aver = average\_under\_side\_diagonal(matrix, row)

output\_matrix(matrix)

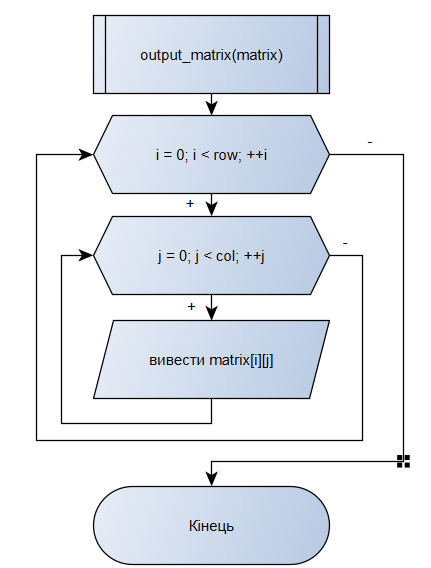
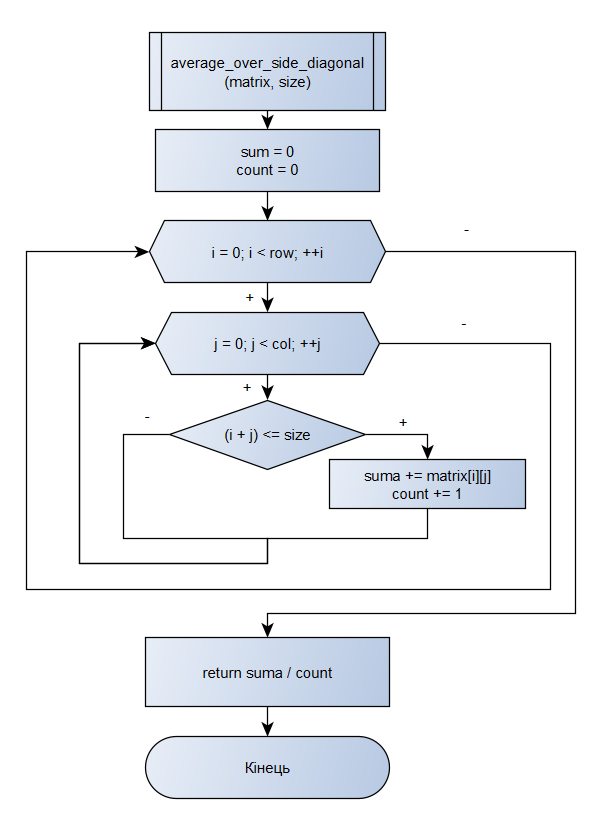
**вивести**("First positive element is bigger") **якщо** (positive\_elem > aver) **інакше вивести**("Average is bigger")

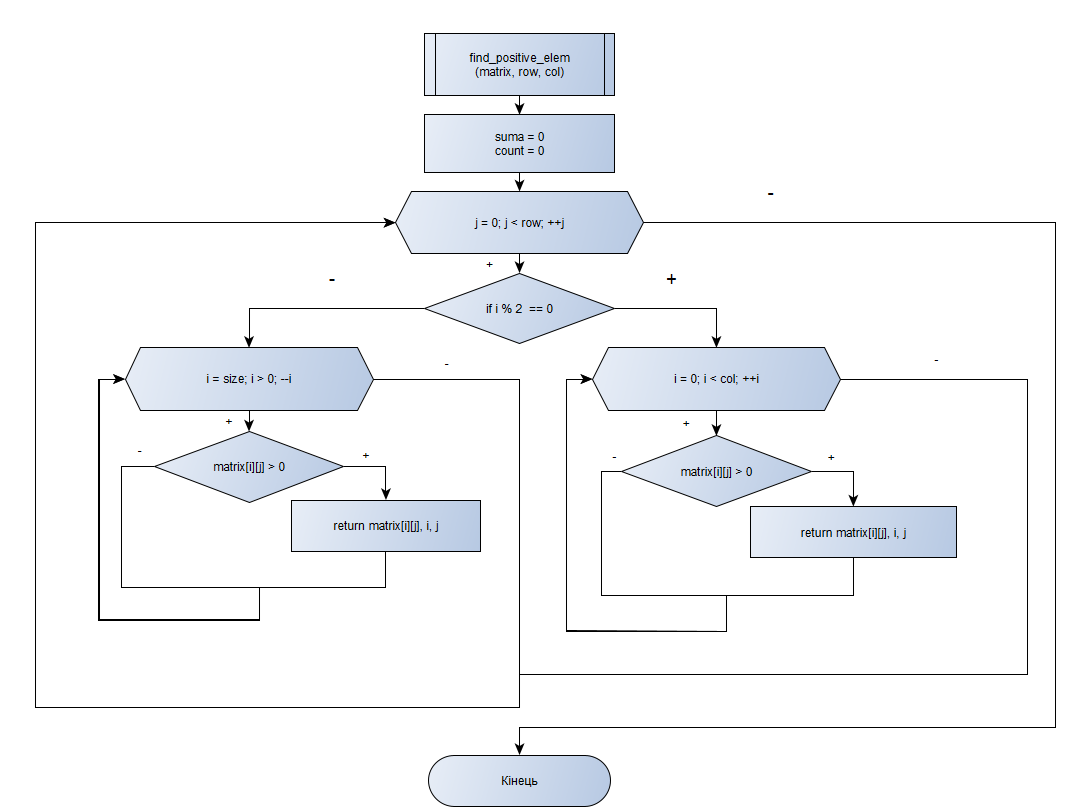
**Кінець**

**Блок-схеми:**

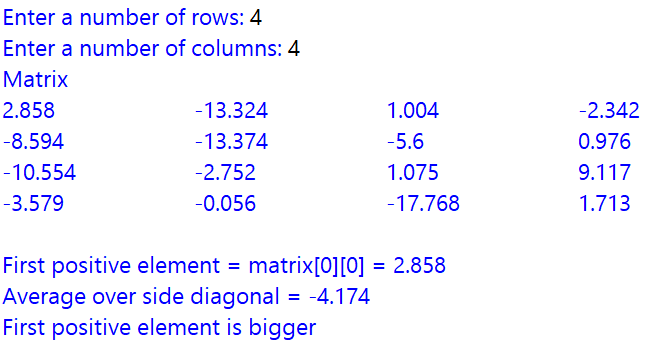
**Крок 1:**

**Крок 2:**

****

****

**Випробування алгоритму:**

****

**Код:**

**Python**

import random

def generate\_matrix(row, col):

matrix = [[round(random.uniform(-20, 11), 3) for j in range(col)] for i in range(row)]

return matrix

def output\_matrix(matrix):

print('Matrix')

for row in matrix:

print(\*row, sep='\t\t')

def find\_positive\_elem(matrix, row, col):

for j in range(col):

for i in range(row) if (j % 2 == 0) else range(row - 1, -1, -1):

if matrix[i][j] > 0:

return matrix[i][j], i, j

def average\_over\_side\_diagonal(matrix, size):

suma = 0

count = 0

for i in range(size):

for j in range(size):

if (i + j) <= size:

suma += matrix[i][j]

count += 1

return round((suma / count), 3)

def main():

row = int(input("Enter a number of rows: "))

col = int(input("Enter a number of columns: "))

matrix = generate\_matrix(row, col)

positive\_elem, row\_index, col\_index = find\_positive\_elem(matrix, row, col)

aver = average\_over\_side\_diagonal(matrix, row)

output\_matrix(matrix)

print(f'\nFirst positive element = matrix[{row\_index}][{col\_index}] = {positive\_elem}')

print(f"Average over side diagonal = {aver}")

print("First positive element is bigger") if (positive\_elem > aver) else print("Average is bigger")

main()

**Висновок:**

Ми опанували та дослідили алгоритми обходу масивів, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій. У результаті лабораторної роботи розробили код, математичну модель, що відповідає постановці задачі, псевдокод, розробили покрокові блок-схеми, які пояснюють логіку алгоритму. Написали код на Python. Зробили випробування алгоритму.