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 24

Виконав студент ІП-14 Прокопенко Олексій Анатолійович

(шифр, прізвище, ім'я, по батькові)

Перевірив доц. Мартинова О. П.

(прізвище, ім'я, по батькові)

Київ 2021

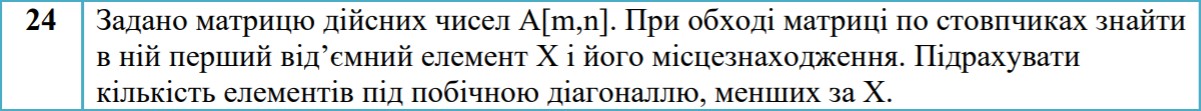
**Лабораторна робота 9**

**Дослідження алгоритмів обходу масивів**

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

**Варіант 24**

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.

**Побудова математичної моделі:**

Потрібно зазначити, що для вирішення задачі, а саме для підрахунку кількості елементів під побічною діагоналлю, необхідно, щоб матриця була квадратною (n = m), тому зручніше буде позначити це число змінною size.

Спочатку згенеруємо квадратну матрицю розмірністю size (яку вводитиме користувач) та заповнимо її випадковими елементами. Щоб знайти перший від’ємний елемент будемо проходити по матриці «змійкою» по стовбцям, ще можна реалізувати за допомогою умовного оператора, який перевірятиме парність номера стовбця та двох вкладених циклів. Підрахунок елементів під побічною діагоналлю також будемо здійснювати за допомогою двох вкладених циклів. При чому, ітератор в одному з яких буде змінюватись від 0 до size, а в іншому – від size – i до size.

Також, в процесі робити алгоритму будемо виводити матрицю за допомогою циклів задля візуалізації його роботи.

**Таблиця змінних**

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | Тип | Ім’я | Призначення |
| Розмірність матриці | int | size | Вхідні дані |
| Матриця | float | matr | Проміжні дані |
| Лічильник 1 | int | i | Проміжні дані |
| Лічильник 2 | int | j | Проміжні дані |
| Координата першого  від’ємного елемента | int | x | Вихідні дані |
| Координата першого  від’ємного елемента | int | y | Вихідні дані |
| Кількість елементів під  побічною діагоналлю | int | count | Вихідні дані |
| Перший від’ємний елемент | float | first\_negative | Вихідні дані |

**Таблиця функцій**

|  |  |  |
| --- | --- | --- |
| Назва | Позначення | Опис |
| Генерація псевдовипадкового числа | rand(a, b) | Генерація числа від a до b |
| Остача від ділення | % | 9 % 6 = 3 |

**Розв’язання:**

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

**Псевдокод:**

**підпрограма**

**FillMatrix**(matr, size):

**повторити** для і від 0 до size

**повторити** для j від 0 до size matr[i][j] = rand(-10, 10)

**все повторити все повторити**

**все підпрограма**

**підпрограма**

**OutputMatrix**(matr, size):

**повторити** для і від 0 до size

**повторити** для j від 0 до size

**виведення** matr[i][j]

**все повторити все повторити**

**все підпрограма**

**підпрограма**

**FindFirstNegative**(matr, x, y, size):

**повторити** для j від 0 до size

**якщо** j % 2 == 0

**то**

**повторити** для i від 0 до size

**якщо** (matr[i][j] < 0)

**то**

x = i y = j

**повернути** matr[i][j]

**все якщо все повторити**

**інакше**

**повторити** для i від size до 0

**якщо** (matr[i][j] < 0)

**то**

x = i y = j

**повернути** matr[i][j]

**все якщо все повторити**

**все якщо все повторити**

**все підпрограма**

**підпрограма**

**CountElements**(matr, first\_negative, size): count = 0

**повторити** для і від 0 до size

**повторити** для j від size - i до size **якщо** matr[i][j] < first\_negative **то**

count = count + 1

**все якщо все повторити**

**все повторити повернути** count

**все підпрограма**

**початок**

**введення** size matr = [size][size]

**FillMatrix**(matr, size)

**OutputMatrix**(matr, size)

first\_negative = **FindFirstNegative**(matr, x, y, size)

**якщо** first\_negative == 0

**то**

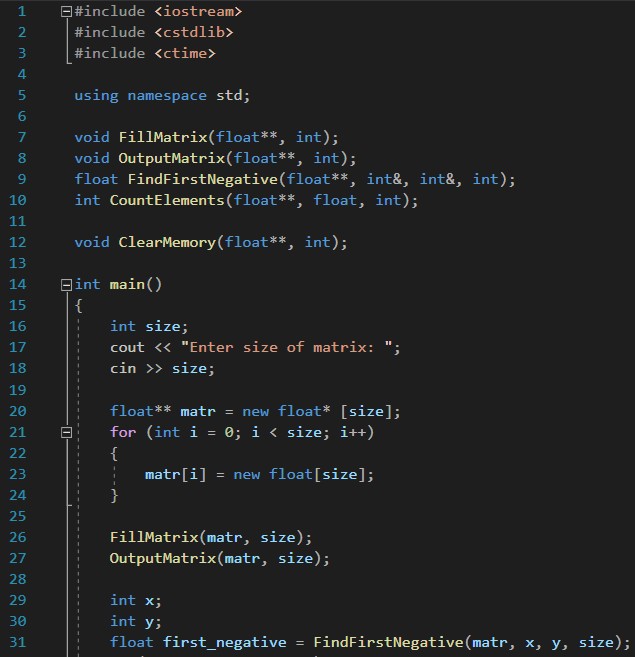
**виведення** “Всі числа додатні”

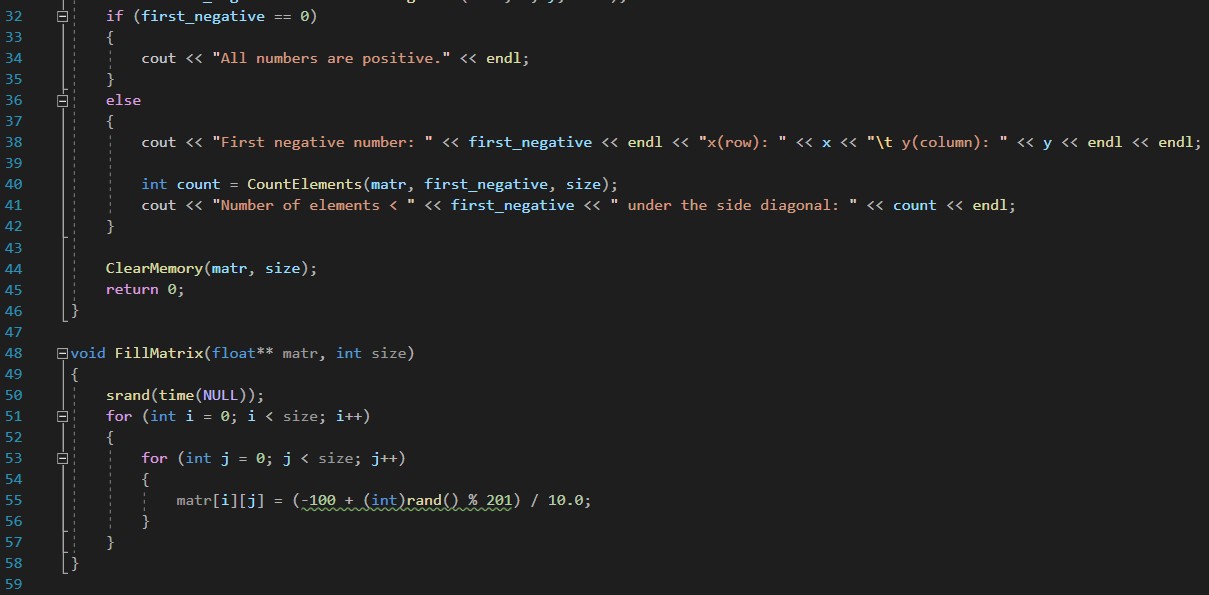
**інакше**

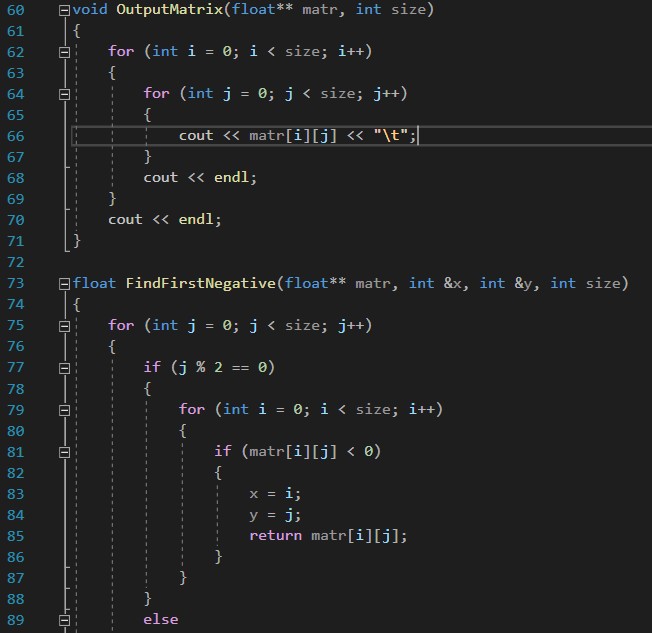
count = **CountElements**(matr, first\_negative, size)

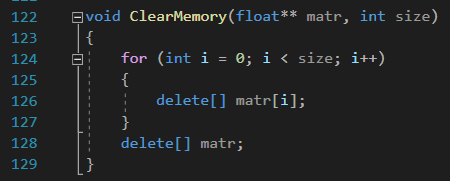
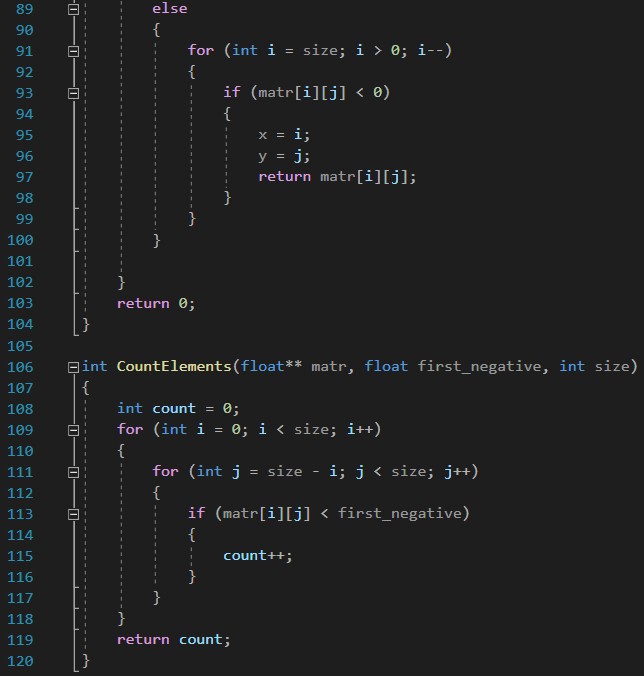
**виведення** first\_negative, count

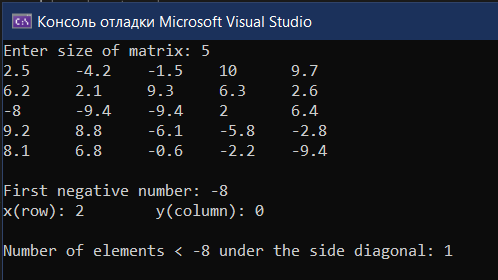
**все якщо кінець**

**Код програми**







**Тестування**

**Висновок:**

Я набув практичних навичок використання алгоритмів обходу масивів в багатовимірних масивах під час складання програмних специфікацій. Під час виконання роботи було створено підпрограми для заповнення і виведення матриці (це робилося за допомогою двох ітераційних циклів), також було створено підпрограму для обходу матриці по стовпцях (за допомогою умовного оператора парні рядки обходилися зверху вниз, а не парні знизу вгору) та пошуку першого від'ємного елемена за допомогою умовного оператора який перевіряв належність числа до від'ємних. Також підпрограму бул створено для поідрахунку кількості емементів, які менші за перший від'ємний елемент.