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-11 Печковський Олександр Костянтинович

(шифр, прізвище, ім'я, по батькові)

Мартинова Оксана Петрівна

Перевірила

(прізвище, ім'я, по батькові)

Київ 2021

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

**Умова задачі:**

Задано матрицю дійсних чисел A[m,n]. При обході матриці по стовпчиках знайти в ній перший від’ємний елемент Х і його місцезнаходження. Підрахувати кількість елементів під побічною діагоналлю, менших за Х.

Постановка задачі:

Оскільки за умовою матриця має побічну діагональ, то ця матриця є квадратною, а отже m=n=size.

1. Створимо двовимірний масив A[size][size]. Ініціюємо його дійсними значеннями, згенерованими випадковим чином.

2. Обійдемо масив A[size][size] змійкою по стовпчиках і знайдемо в ньому перший від’ємний елемент Х і його місцезнаходження.

3. Підрахуємо кількість елементів під побічною діагоналлю, менших за Х.

Математична модель:

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| A | double[][] | Двовимірний масив | Початкове значення |
| size | int | Розмір масиву | Проміжне значення |
| X | double | Перший від’ємний елемент | Результат |
| MinusElementCounter | int | Лічильник кількості знайдених від’ємних елементів | Проміжне значення |
| ElementsLessThanXcounter | int | Кількість елементів під побічною діагоналлю, менших за Х | Результат |
| i | int | Лічильник циклів | Проміжне значення |
| j | int | Лічильник циклів | Проміжне значення |

**Псевдокод:**

Крок 1: Визначимо основні дії

Крок 2: Деталізуємо заповнення масиву A випадковими дійсними значеннями і виведення його на екран

Крок 3: Деталізуємо обхід масиву A змійкою по стовпчиках і знаходження в ньому першого від’ємного елемента Х і його місцезнаходження

Крок 4: Деталізуємо підрахунок кількості елементів під побічною діагоналлю, менших за Х

**Крок 1:**

**Початок**

Заповнення масиву A випадковими дійсними значеннями і виведення його на екран

Обхід масиву A змійкою по стовпчиках і знаходження в ньому першого від’ємного елемента Х і його місцезнаходження

Підрахунок кількості елементів під побічною діагоналлю, менших за Х

**Кінець**

**Крок 2:**

**Початок**

Для i в діапазоні (0, size):

Для j в діапазоні (0, size):

A[i][j] = (double)(-90 + rand() % 190) / 10

Виводимо A[i][j]

Обхід масиву A змійкою по стовпчиках і знаходження в ньому першого від’ємного елемента Х і його місцезнаходження

Підрахунок кількості елементів під побічною діагоналлю, менших за Х

**Кінець**

**Крок 3:**

**Початок**

Для i в діапазоні (0, size):

Для j в діапазоні (0, size):

A[i][j] = (double)(-90 + rand() % 190) / 10

Виводимо A[i][j]

Для i в діапазоні (0, size):

Для j в діапазоні (0, size):

Якщо (i%2=0):

Якщо (A[j][i] < 0 та MinusElementCounter = 0):

X = A[j][i]

Виводимо X

Виводимо j та i

MinusElementCounter++

Інакше:

Якщо (A[size-j-1][i] < 0 та MinusElementCounter = 0):

X = A[size-j-1][i]

Виводимо X

Виводимо (size-j-1) та i

MinusElementCounter++

Підрахунок кількості елементів під побічною діагоналлю, менших за Х

**Кінець**

**Крок 4:**

**Початок**

Для i в діапазоні (0, size):

Для j в діапазоні (0, size):

A[i][j] = (double)(-90 + rand() % 190) / 10

Виводимо A[i][j]

Для i в діапазоні (0, size):

Для j в діапазоні (0, size):

Якщо (i%2=0):

Якщо (A[j][i] < 0 та MinusElementCounter = 0):

X = A[j][i]

Виводимо X

Виводимо j та i

MinusElementCounter++

Інакше:

Якщо (A[size-j-1][i] < 0 та MinusElementCounter = 0):

X = A[size-j-1][i]

Виводимо X

Виводимо (size-j-1) та i

MinusElementCounter++

Для i в діапазоні (1, size):

Для j в діапазоні (size-i, size):

Якщо (A[i][j] < X):

ElementsLessThanXcounter++

Виводимо ElementsLessThanXcounter

**Кінець**

**Блок-схеми:**









Код програми:

#include <iostream>

#include "windows.h"

using namespace std;

typedef double Matrix[10][10];

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

Matrix A;

int size;

double X;

int ElementsLessThanXcounter=0;

int MinusElementCounter = 0;

cout << "Введіть розмір квадратної матриці: ";

cin >> size;

for (int i = 0; i < size; ++i)

{

for (int j = 0; j < size; ++j)

{

A[i][j] = (double)(-90 + rand() % 190) / 10;

cout.width(6);

cout << A[i][j];

}

cout << endl;

}

for (int i = 0; i < size; ++i)

{

for (int j = 0; j < size; ++j)

{

if (i%2==0)

{

if (A[j][i] < 0 && MinusElementCounter == 0)

{

X = A[j][i];

cout << "\nПерший від'ємний елемент: " << X << endl;

cout << "Його місцезнаходження: " << j << ", " << i << endl;

MinusElementCounter++;

}

}

else

{

if (A[size-j-1][i] < 0 && MinusElementCounter == 0)

{

X = A[size - j - 1][i];

cout << "\nПерший від'ємний елемент: " << X << endl;

cout << "Його місцезнаходження: " << size - j - 1 << ", " << i << endl;

MinusElementCounter++;

}

}

}

}

for (int i = 1; i < size; i++)

{

for (int j = size - i; j < size; j++)

{

if (A[i][j] < X)

{

ElementsLessThanXcounter++;

}

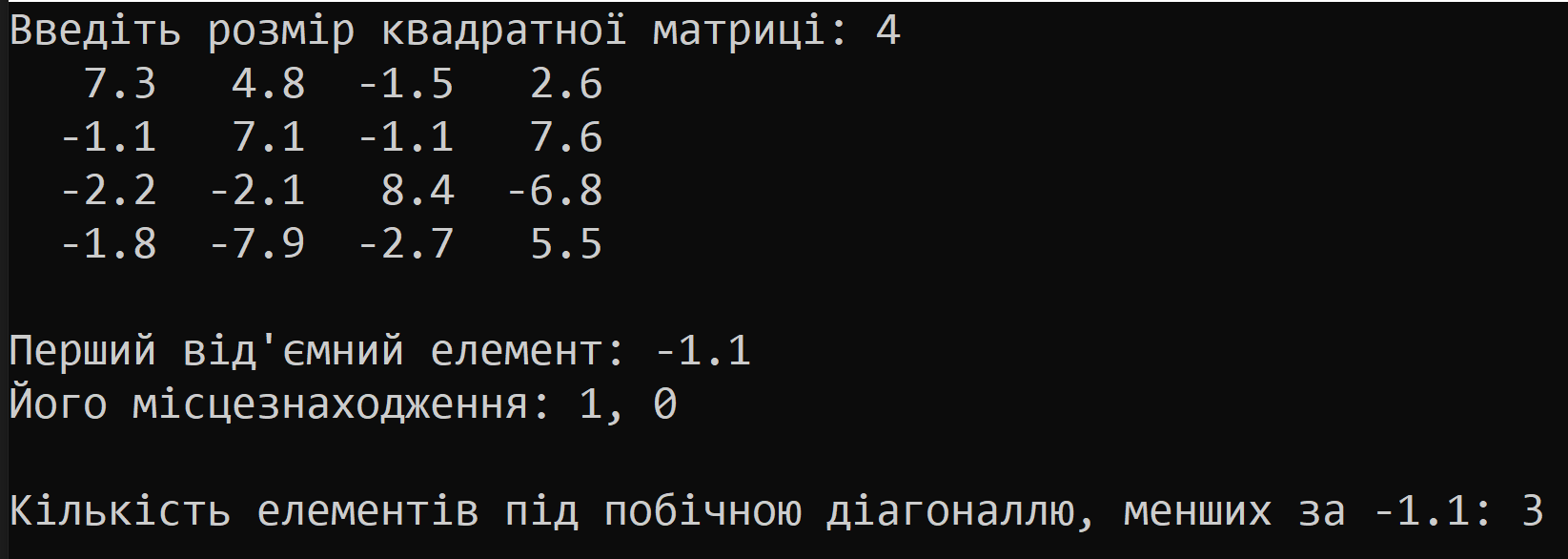
}

}

cout << "\nКількість елементів під побічною діагоналлю, менших за " << X << ": " << ElementsLessThanXcounter<<endl;

}

Робота програми:



Випробування алгоритму:

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Введення розміру матриці, генерація і виведення матриці |
| 2 | Перший від’ємний елемент: -1.1, рядок 1, стовпчик 0 |
| 3 | Елементи під побічною діагоналлю, менші за -1.1: -7.9, -2.7, -6.8 |
|  | Кінець |

Висновок: виконуючи лабораторну роботу, я дослідив алгоритми обходу масивів, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій.