

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 ПІ-12, Васи́лишин Михайло Михайлович  
(шифр, прізвище, ім'я, по батькові)  
Перевірів Василишин Михайло Михайлович  
( прізвище, ім'я, по батькові)

Київ 2021

## Лабораторна робота №8 «Дослідження алгоритмів обходу масивів»

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

**Варіант – 5**

**Задача №5.**

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

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

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

**Розв'язок**

### 1. Постановка задачі.

Результатом розв'язку є змінна чисельного типу, що відповідає сумі елементів, розташованих нижче головної діагоналі. Саму матрицю ініціалізуємо випадковими значеннями, скориставшись відповідною стандартною функцією `rand()` та алгоритмом обходу по рядках. Визначимо функції генерації, виводу та обчислення суми елементів, розташованих нижче головної діагоналі. Для отримання розмірів матриці очікуватимемо вхідні дані, що репрезентовані змінною `n`. Інших функцій та змінних для розв'язку не потрібно.

## 2. Побудова математичної моделі. Складемо таблицю імен змінних.

Змінна	Тип	Ім'я	Призначення
Розмір масиву	Цілий	n	Вхідне дане
Лічильник циклу i	Цілий	i	Проміжне дане
Лічильник циклу j	Цілий	j	Проміжне дане
Початковий масив	Дійсний двовимірний масив	Array2D	Проміжне дане
Шукане значення суми	Дійсний	sum	Результату

Після вводу змінною n, отримуємо остаточні розміри масиву. Оскільки значення елементів генеруються випадково, то після операцію вводу та виклику функції, що відповідає за генерацію – масив буде ініціалізованим.

Функція generate2D(), яка генеруватиме масив, буде ініціалізувати його значення по рядках(відповідно до умови задачі). Очевидно, що її алгоритмічна інтерпретація полягає у виконанні складеного арифметичного циклу від 1 до n.

Функцію output2D() будемо використовувати для виводу масиву, її алгоритмічне представлення теж відображено у складених арифметичних циклах.

Опишемо функцію getAnswer(), яка повертає суму елементів, розташованих головної діагоналі. Нижче зображено схематичне зображення матриці:

-1	2	7	0
0	5	0	-1
0	0	7	0
0	0	0	0

Matrix A

На цьому рисунку: жовті елементи – елементи головної діагоналі, сині елементи – потрібні для розв’язку елементи(розташовані нижче головної діагоналі).

Сама функція `getAnswer()` буде представлена у вигляді двох арифметичних циклів від 1 до  $n$ . А необхідною умовою для перевірки приналежності елементи до таких, що розташовані нижче головної діагоналі є:  $(i > j)$ .

Примітка: лічильник  $i$  – по рядках,  $j$  – по стовпцях. Якщо логічний вираз приймає значення `true`, то додаватимемо до допоміжної змінної `sum` відповідний елемент з індексами  $[i,j]$ .

Зазначу, що у всіх циклах – крок 1, а умова невиходу – поки перший чисельне значення першого елемента є меншим, ніж другого.

Щодо використання стандартної функції `rand()`, будемо генерувати дійсні числа, ціла частина яких – на проміжку  $[0;999]$ , а кількість цифр у десятковому записі  $< 4$  ( тобто буде на проміжку  $[0.000; 0,999]$ . Тому випадкове число можна записати у вигляді - `rand()%1000 + (rand()%1000/1000.0)`.

Отож, всі необхідні функції описані.

**Крок 1.** Визначимо основні дії.

**Крок 2.** Деталізуємо функцію `generate2D`

**Крок 3.** Деталізуємо функцію `output2D`

**Крок 4.** Деталізуємо функцію `getAnswer`

### 3. Псевдокод алгоритму

Крок 1

**початок**

**ввід**  $n$

`generate2D(array2D,n)`

`output2D(array2D,n)`

sum:= getAnswer(array2D, n)

**вивід** sum

**Кінець**

**функція** generate2D(array2D[[]],n)

**функція** output2D(array2D[[]],n)

**функція** getAnswer(array2D[[]], n)

Крок 2

**початок**

**ввід** n

generate2D(array2D,n)

output2D(array2D,n)

sum:= getAnswer(array2D, n)

**вивід** sum

**Кінець**

**Початок функції** generate2D(array2D,n)

**повторити**

для i від 0 до n

**повторити**

для j від 0 до n

array2D[i+1,j+1]:= rand()%1000 + (rand()%1000/1000.0)

**все повторити**

**все повторити**

**кінець функції** generate2D(array2D[[]],size)

**функція** output2D(array2D[[]],n)

**функція** getAnswer(array2D[[]], n)

Крок 3

**початок**

**ввід** n

generate2D(array2D,n)

output2D(array2D,n)

sum:= getAnswer(array2D, n)

**вивід** sum

**Кінець**

**Початок функції** generate2D(array2D[[]],n)

**повторити**

для i від 0 до n

**повторити**

для j від 0 до n

array2D[i+1,j+1]:= rand()%1000 + (rand()%1000/1000.0)

**все повторити**

**все повторити**

**кінець функції** generate2D(array2D[[]],n)

**Початок функції** output2D(array2D[],n)

**повторити**

для i від 0 до n

**повторити**

для j від 0 до n

**вивід** array2D[i+1,j+1]

**все повторити**

**все повторити**

**кінець функції** output2D (array2D[],n)

**функція** getAnswer(array2D[], n)

Крок 4.

**початок**

**ввід** n

generate2D(array2D,n)

output2D(array2D,n)

sum:= getAnswer(array2D, n)

**вивід** sum

**Кінець**

**Початок функції** generate2D(array2D[],n)

**повторити**

для i від 0 до n

**повторити**

**для j від 0 до n**

**array2D[i+1,j+1]:= rand()%1000 + (rand()%1000/1000.0)**

**все повторити**

**все повторити**

**кінець функції generate2D(array2D[],n)**

**Початок функції output2D(array2D[],n)**

**повторити**

**для i від 0 до n**

**повторити**

**для j від 0 до n**

**вивід array2D[i+1,j+1]**

**все повторити**

**все повторити**

**кінець функції output2D (array2D[],n)**

**Початок функції getAnswer(array2D[],n)**

**Sum:= 0.0**

**повторити**

**для i від 0 до n**

**повторити**

**для j від 0 до n**



**якщо  $i > j$**

**то  $sum = sum + array2D[i+1][j+1]$**

**все якщо**

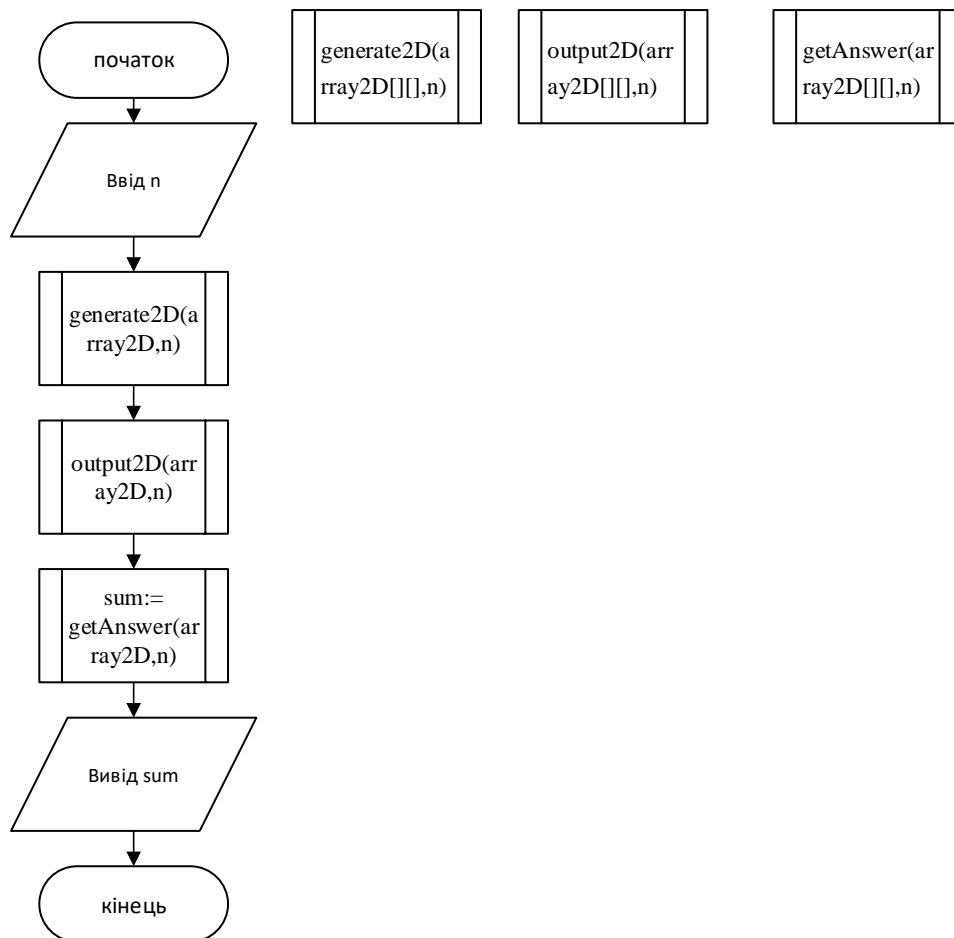
**все повторити**

**все повторити**

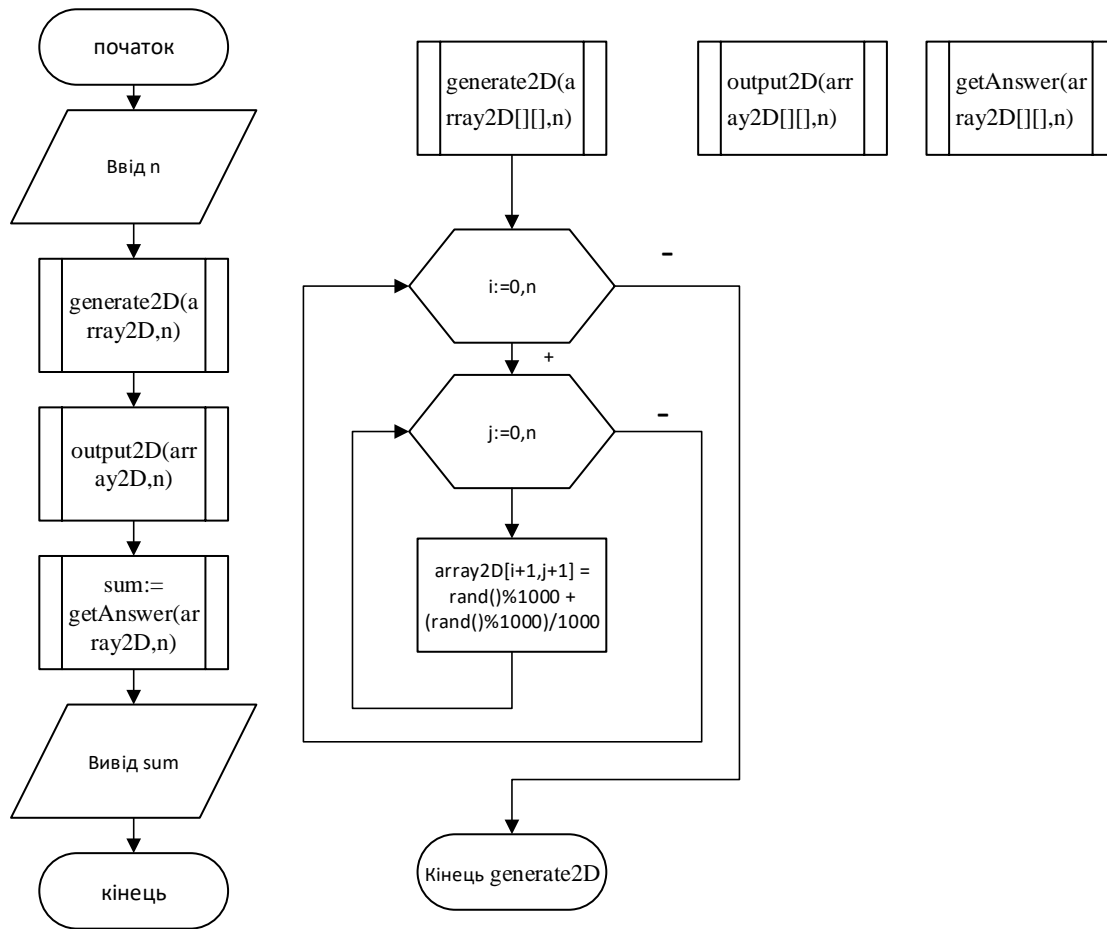
**кінець функції  $getAnswer(array2D[[],n])$**

#### 4. Блок-схема

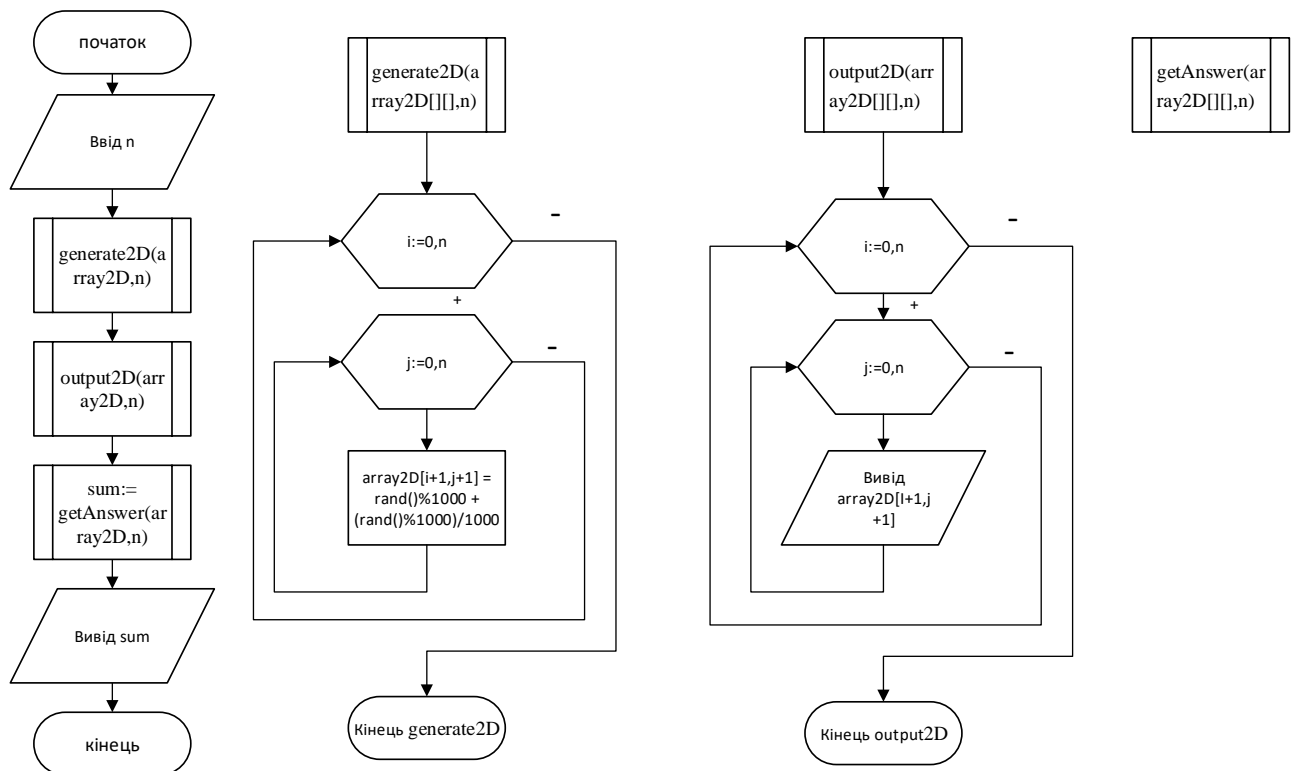
Крок 1



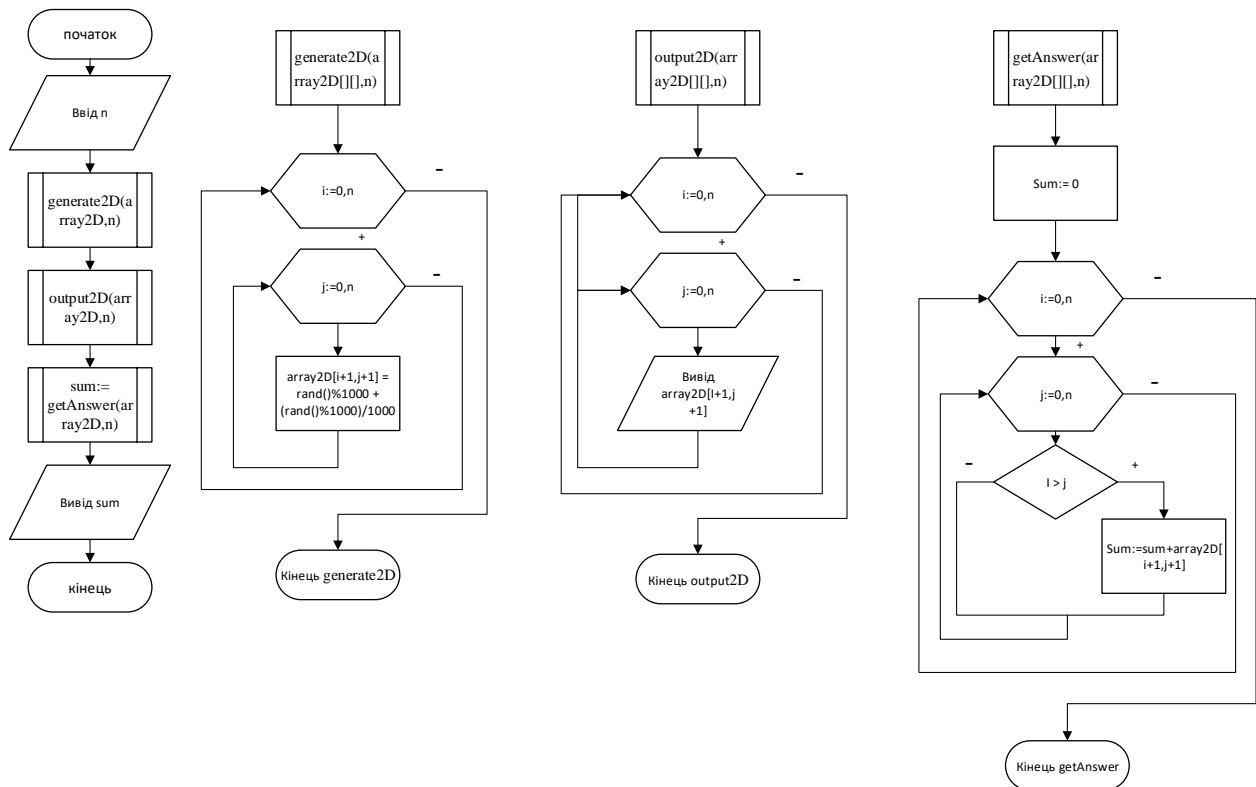
## Крок 2.



## Крок 3.



## Крок 4.



## 5. Код програми(c++)

```
#include<iostream>
```

```
#include<time.h>
```

```
#include<iomanip>
```

```
using namespace std;
```

```
void generate2D(double** , int);
```

```
void output2D(double** , int);
```

```
double getAnswer(double** ,int);
```

```
int main()
```

```
{
```

```
    srand(time(NULL));
```

```
    double **array2D;
```

```

    cout << "Please, enter the size of matrix: " << endl;
    int n;
    cin >> n;

    array2D = new double *[n];
    for(int i = 0; i < n; i++)array2D[i] = new double[n];

    generate2D(array2D,n);
    output2D(array2D,n);

    double sum = getAnswer(array2D,n);
    cout << sum << endl;
    return 0;
}

void generate2D(double** array2D, int n)
{
    for(int i = 0; i < n;i++)
        for(int j = 0; j < n;j++)array2D[i][j] = rand()%1000 +
(rand()%1000)/1000.0;
}

void output2D(double** array2D, int n)
{
    for(int i = 0; i < n;i++)
    {
        for(int j = 0; j < n;j++)cout << setw(8) << array2D[i][j];
        cout << endl;
    }
}

double getAnswer(double** array2D,int n)

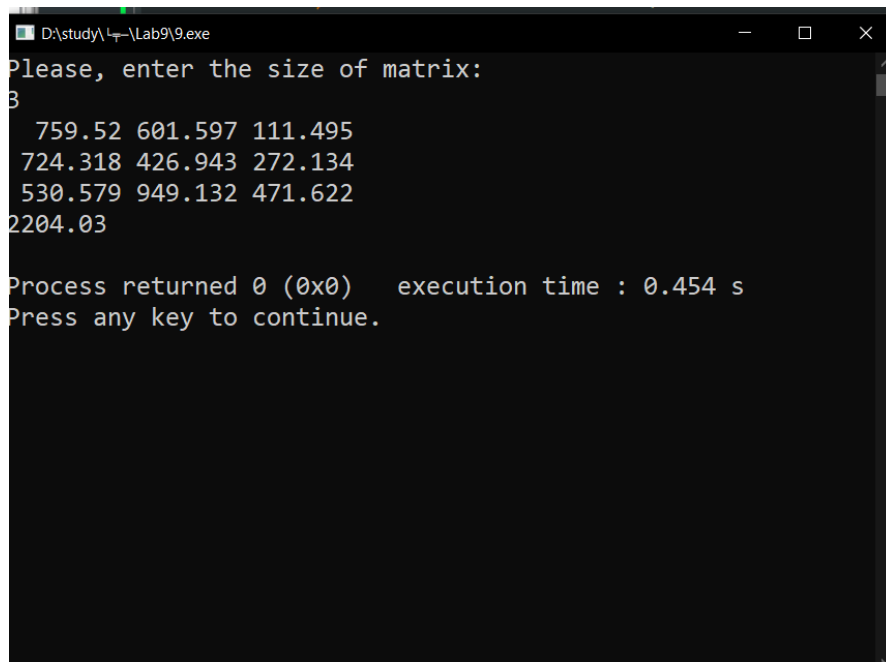
```

```

{
    double sum = 0;
    for(int i = 0; i < n;i++)
        for(int j = 0; j < n;j++)
            if(i > j)sum+= array2D[i][j];
    return sum;
}

```

## 6. Тестування програми

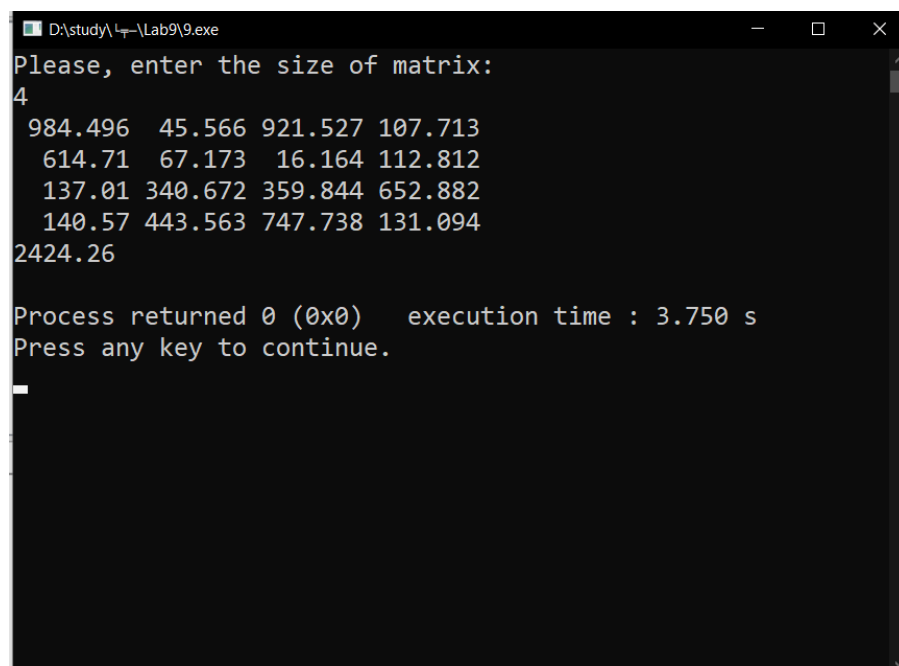


```

D:\study\ІТ\Lab9\9.exe
Please, enter the size of matrix:
3
759.52 601.597 111.495
724.318 426.943 272.134
530.579 949.132 471.622
2204.03

Process returned 0 (0x0)   execution time : 0.454 s
Press any key to continue.

```



```

D:\study\ІТ\Lab9\9.exe
Please, enter the size of matrix:
4
984.496 45.566 921.527 107.713
614.71 67.173 16.164 112.812
137.01 340.672 359.844 652.882
140.57 443.563 747.738 131.094
2424.26

Process returned 0 (0x0)   execution time : 3.750 s
Press any key to continue.

```

## 7. Висновок.

На даній лабораторній роботі було досліджено алгоритми обходу масивів, набуто практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Особливістю моєї роботи була ініціалізація масиву випадковими значеннями по рядках. Також особливістю було, те що кожна логічна окрема дія для масиву реалізована через функцію.