

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 7

Виконав студент ІП-13, Гогіберідзе Торніке Лашаєвич

(шифр, прізвище, ім'я, по батькові)

Перевірила Вєчерковська 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.

**Варіант 7**

**Завдання**

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).

**7**

Задано матрицю дійсних чисел  $A[m,n]$ . В кожному рядку матриці знайти останній від'ємний елемент і його місцезнаходження. Обміняти знайдене значення  $X$  з елементом головної діагоналі.

*Постановка задачі*

Ми використовуємо підпрограму для ініціалізації матриці (двовимірного масиву). Ми також використовуємо підпрограму, щоб знайти останній негативний елемент і його позицію, і викликаємо допоміжну підпрограму для обміну знайденого елемента на діагональний елемент. Матриця заповнена випадковими дійсними числами.

Результатом рішення є розрахунок матриці та її обробка за заданих умов.

*Побудова математичної моделі*

Таблиця імен змінних

<i>Змінна</i>	<i>Typ</i>	<i>Im'я</i>	<i>Призначення</i>
Матриця	float	matrix	Проміжні дані/Результат

Довжина матриці (рядки і стопці)	int	length	Проміжні дані
Процедура для ініціалізації матриці	Процедура	initializeMatrix	Початкові дані
Лічильник i	int	i	Початкові дані
Лічильник j	int	j	Початкові дані
Головна процедура для обробки матриці	Процедура	swapDiagonalandNegative	Початкові дані
Останній від'ємний елемент	float	lastMinus	Проміжні дані
Індекс (місцезнаходження) останнього від'ємного елемента	int	lastMinusIndex	Проміжні дані
Напрям обходу матриці «змійкою»	int	direction	Початкові дані

Змінна, яка зазначає чи знайдений від'ємний елемент	bool	Found	Проміжні дані
Процедура, яка обмінює значення зазначених елементів	Процедура	swapArrayElements	Початкові дані
Змінна, яка тимчасово зберігає значення	float	temp	Проміжні дані
Масив (рядок), який приймає процедура	float	array	Проміжні дані

Спочатку введіть значення змінної довжини (кількість рядків і стовпців матриці). Примітка: за умовами нам потрібно знайти елемент головної діагоналі, який існує лише в квадратній матриці, тому  $m = n =$  довжина. Після оголошення матриці ми ініціалізуємо її за допомогою процедури initializeMatrix, яка приймає матрицю та її довжину як параметри (матриця, довжина). Цей процес використовує функцію randomFloat (-10..10) для заповнення матриці випадковими дійсними числами від -10 до 10.

Після цього, щоб обробити матрицю, ми викликаємо процедуру swapDiagonalandNegative, яка приймає матрицю та її довжину як параметри (matrix, length). Цей процес використовує змієвидний обхід, тому він має змінну напряму, яка регулює напрямок обходу, і зовнішній арифметичний цикл (лічильник i), два внутрішні оператори умови підстановки та два

внутрішні арифметичні цикли (лічильник  $j$ ). Якщо напрямок більше нуля, то обійти лінію зліва направо, якщо менше нуля, то навпаки;  $&&$  логічно,  $i != -1$  — Це логічне пограбування. Використовуючи ці обходи, процес знайде останній негативний елемент кожного рядка (`lastMinus`) та його індекс (`lastMinusIndex`).

У кінці кожного повторення зовнішнього циклу ми викликаємо допоміжну процедуру `swapArrayElements`, яка приймає масив (рядок) і два індекси як параметри (масив, індекс 1, індекс 2). У нашому прикладі масив — це матриця `[i]`, індекс1 — це  $i$ , а індекс2 — це `lastMinusIndex`. Використовуючи тимчасові змінні, процедура замінить елемент цими двома індексами.

### *Розв'язання*

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

*Крок 1.* Визначимо основні дії.

*Крок 2.* Ініціалізація матриці

*Крок 3.* Обробка матриці

### *Псевдокод*

#### Основна програма:

#### **початок введення**

length

    initializeMatrix (matrix, length)

    swapDiagonalandNegative (matrix, length)

#### **виведення matrix кінець**

#### Підпрограми:

initializeMatrix (matrix, length)

**повторити** для *i* від 1 до *length*

**повторити** для *j* від 1 до *length*

*matrix[i][j] = randomFloat(-10, 10)*

**все повторити**

**все повторити**

**кінець**

*swapDiagonalandNegative (matrix, length)*

*direction := 1*      **повторити** для *i* від 1

до *length*      **якщо** (*direction > 0*)

**то**

**повторити** для *j* від 1 до *length*

**якщо** (*matrix[i][j] < 0*)

**то**

*lastMinus := matrix [i][j]*

*lastMinusIndex := j*

**все якщо**

**все повторити**

**все якщо**

**інакше**

**то**      *Found := false*

**повторити** для *j* від *length* до 1, крок -1

**якщо** (*matrix[i][j] < 0 && !Found*)

**то**

*lastMinus := matrix [i][j]*

*lastMinusIndex := j*

*Found := true*

**все**

**якщо**

**все повторити**

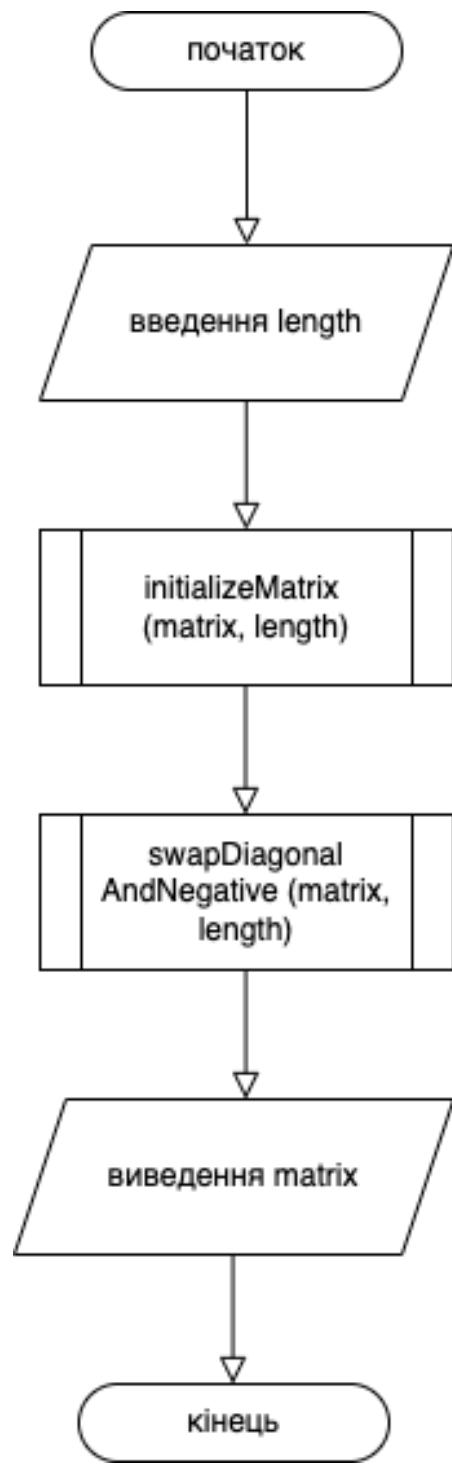
**все якщо**

```
swapArrayElements(matrix[i], i, lastMinusIndex)
direction := - direction    все повторити
кінець
```

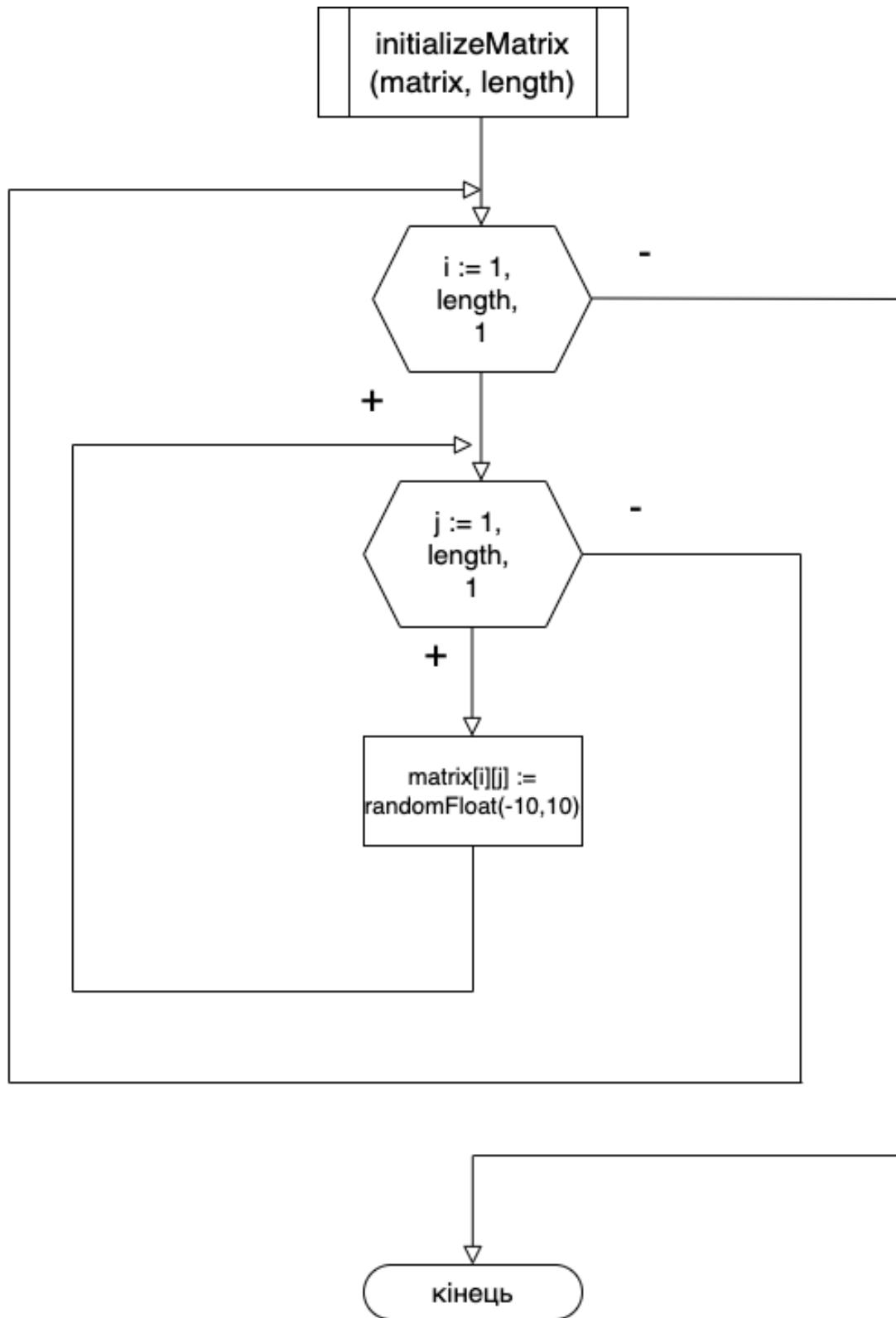
```
swapArrayElements(array, index1, index2)
temp = array[index1]    array[index1] =
array[index2]    array[index2] = temp
кінець
```

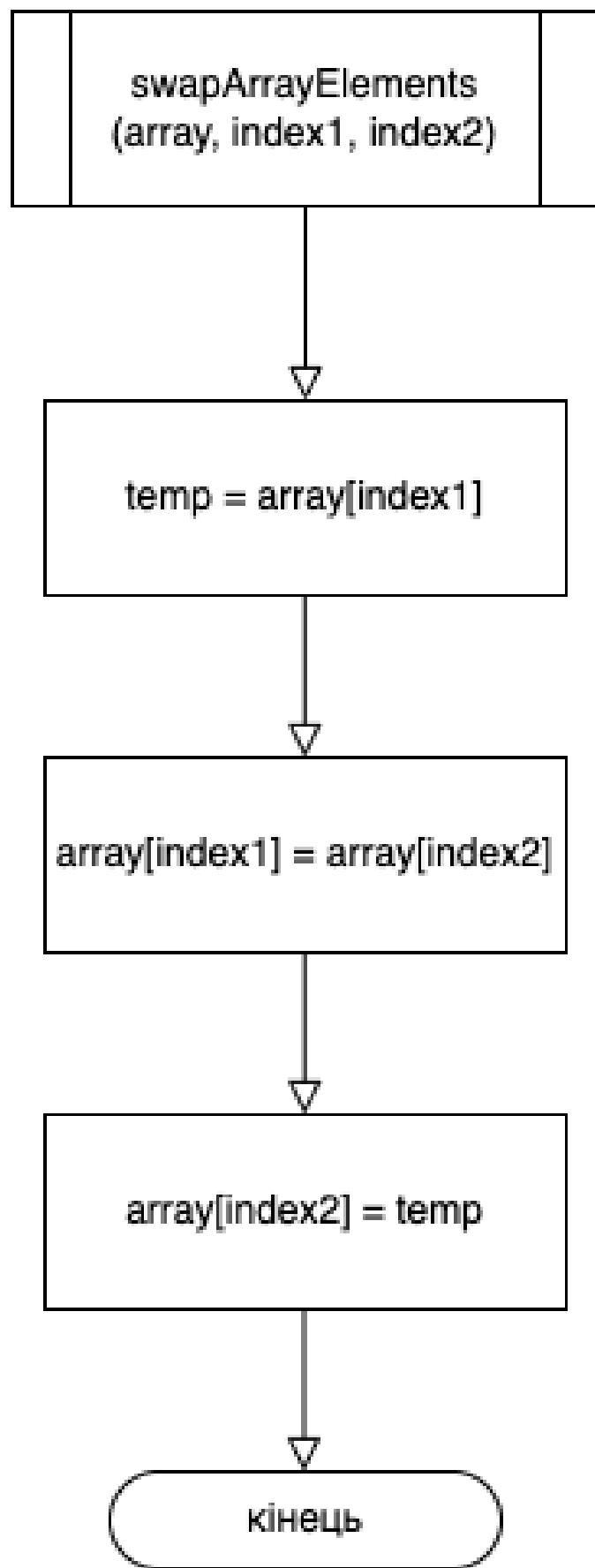
*Блок-схема:*

*Основна програма:*



Пiдпрограми:





*Kod:*

Основна програма:

```
int main() {  
  
    srand((float)time(NULL));  
  
    int length;  
  
    cout << "Enter number of rows and columns: ";  
    cin >> length;  
  
  
    float ** matrix = new float*[length];  
    initializeMatrix(matrix, length);  
  
    cout << "Matrix A" << endl;  
    for (int i = 0; i < length; i++){  
        for (int j = 0; j < length; j++){  
            printf("%8.3f", matrix[i][j]);  
        }  
        cout << endl;  
    }  
    cout << "\n\n";  
  
    swapDiagonalAndNegative(matrix,length);  
  
    cout << "\n\nUpdated Matrix A" << endl;  
    for (int i = 0; i < length; i++){  
        for (int j = 0; j < length; j++){  
            printf("%8.3f", matrix[i][j]);  
        }  
        cout << endl;  
    }  
  
  
    return 0;  
}
```

Подпрограми:

```
void initializeMatrix(float ** matrix, int length){

    for (int i = 0; i < length; i++){
        matrix[i] = new float[length];
    }

    for (int i = 0; i < length; i++){
        for (int j = 0; j < length; j++){
            matrix[i][j] = (float)(rand())/(float)RAND_MAX*20-10;
        }
    }
}
```

```
void swapArrayElements(float *array, int index1, int index2{

    float temp;

    temp = array[index1];
    array[index1] = array[index2];
    array[index2] = temp;
}
```

```
void swapDiagonalAndNegative(float ** matrix, int length){

    int direction = 1, lastNegativeIndex = 0;
    float lastNegative = 0;

    for (int i = 0; i < length; i ++){

        if (direction > 0){

            for (int j = 0; j < length; j++){

                if (matrix[i][j] < 0){

                    lastNegative = matrix[i][j];
                    lastNegativeIndex = j;
                }
            }
        }

        else {

            bool isFound = false;

            for (int j = length - 1; j >= 0; j--){

                if (matrix[i][j] < 0 && !isFound){

                    lastNegative = matrix[i][j];
                    lastNegativeIndex = j;
                    isFound = true;
                }
            }
        }
    }
}
```

```

        cout << "The last negative element of row #" << i+1 << " is " << setprecision(4) << lastNegative << " with
        index " << lastNegativeIndex << endl;

        swapArrayElements(matrix[i],i,lastNegativeIndex);

        direction = -direction;
    }
}

```

**Enter number of rows and columns: 6**

**Matrix A**

<b>6.448</b>	<b>4.721</b>	<b>4.307</b>	<b>0.277</b>	<b>-9.264</b>	<b>4.465</b>
<b>5.294</b>	<b>7.926</b>	<b>5.550</b>	<b>5.802</b>	<b>-9.721</b>	<b>-9.065</b>
<b>7.490</b>	<b>1.546</b>	<b>-0.089</b>	<b>6.540</b>	<b>3.908</b>	<b>-2.895</b>
<b>5.120</b>	<b>-2.606</b>	<b>6.901</b>	<b>4.862</b>	<b>1.748</b>	<b>-4.898</b>
<b>-2.604</b>	<b>2.720</b>	<b>-6.143</b>	<b>0.626</b>	<b>-0.839</b>	<b>-6.788</b>
<b>-6.772</b>	<b>2.952</b>	<b>0.776</b>	<b>-4.979</b>	<b>2.189</b>	<b>7.186</b>

The last negative element of row #1 is -9.264 with index 4  
The last negative element of row #2 is -9.065 with index 5  
The last negative element of row #3 is -2.895 with index 5  
The last negative element of row #4 is -4.898 with index 5  
The last negative element of row #5 is -6.788 with index 5  
The last negative element of row #6 is -4.979 with index 3

**Updated Matrix A**

<b>-9.264</b>	<b>4.721</b>	<b>4.307</b>	<b>0.277</b>	<b>6.448</b>	<b>4.465</b>
<b>5.294</b>	<b>-9.065</b>	<b>5.550</b>	<b>5.802</b>	<b>-9.721</b>	<b>7.926</b>
<b>7.490</b>	<b>1.546</b>	<b>-2.895</b>	<b>6.540</b>	<b>3.908</b>	<b>-0.089</b>
<b>5.120</b>	<b>-2.606</b>	<b>6.901</b>	<b>-4.898</b>	<b>1.748</b>	<b>4.862</b>
<b>-2.604</b>	<b>2.720</b>	<b>-6.143</b>	<b>0.626</b>	<b>-6.788</b>	<b>-0.839</b>
<b>-6.772</b>	<b>2.952</b>	<b>0.776</b>	<b>7.186</b>	<b>2.189</b>	<b>-4.979</b>

## Висновки

Протягом дев'ятої лабораторної роботи я дослідив алгоритми обходу масивів і набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій. В результаті я отримав алгоритм, що ініціалізує матрицю випадковими дійсними числами, а також оброблює цю матрицю за даною умовою, використовуючи алгоритму обходу.

