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-15, Гуменюк Олександр Володимирович
(шифр, прізвище, ім'я, по батькові)
Перевірила Вєчерковська Анастасія Сергіївна
(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 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

Використовуємо підпрограму для ініціалізації матриці (двомірного масиву). Також використовуємо підпрограму для знаходження останнього від'ємного елемента і його місцезнаходження і в ній викликаємо допоміжну підпрограму для обміну знайденого елемента з діагональним елементом. Матрицю заповнюємо випадковими дійсними числами.

Результатом розв'язку є обчислення матриці і її обробкою за заданою умовою.

Побудова математичної моделі

Таблиця імен змінних

Змінна	***Тип***	***Ім'я***	***Призначення***
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	lastNegative	Проміжні дані
Індекс (місцезнаходження) останнього від’ємного елемента	int	lastNegativeIndex	Проміжні дані

Напрям обходу матриці «змійкою»	int	direction	Початкові дані
Змінна, яка зазначає чи знайдений від’ємний елемент	bool	isFound	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 вводимо значення змінної length (кількість рядків і кількість стовпців матриці). Зауваження: за умовою нам потрібно знаходити елементи головної діагоналі, яка існує тільки у квадратних матрицях, тому m = n = length. Після оголошення матриці matrix ініціалізуємо її, використовуючи процедуру initializeMatrix, яка приймає матрицю і її довжину як параметри (matrix, length). Процедура заповнює матрицю випадковими дійсними числами від -10 до 10, використовуючи функцію randomFloat(-10,10).

Після цього для обробки матриці виклакаємо процедуру swapDiagonalAndNegative, яка приймає матриці і її довжину як параметри (matrix , length). Ця процедура використовує обхід «змійкою», тому в ній є змінна direction, яка регулює напрям обходу, а також зовнішній арифметичний цикл (лічильник i), а також два внутрішніх альтернативних операторів умови і два внутрішніх арифметичних цикла (лічильники j). Якщо direction більше нуля, то обхід рядка йде зліва направо, а якщо менше нуля, то навпаки; && - це логічне і, ! – це логічне запаречення. За допомогою цих обходів, процедура знайде останній від'ємний елемент кожного рядочку (lastNegative) і його індекс (lastNegativeIndex).

Наприкінці кожного повторення зовнішнього циклу ми викликаємо допоміжну процедуру swapArrayElements, яка приймає масив (рядок) і два індекси як параметри (array, index1, index2). В нашому випадку array це буде matrix[i], index1 це i, а index2 це lastNegativeIndex. Використовуючи змінну temp, процедура замінить елементи з 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

lastNegative := matrix [i][j]

lastNegativeIndex := j

все якщо

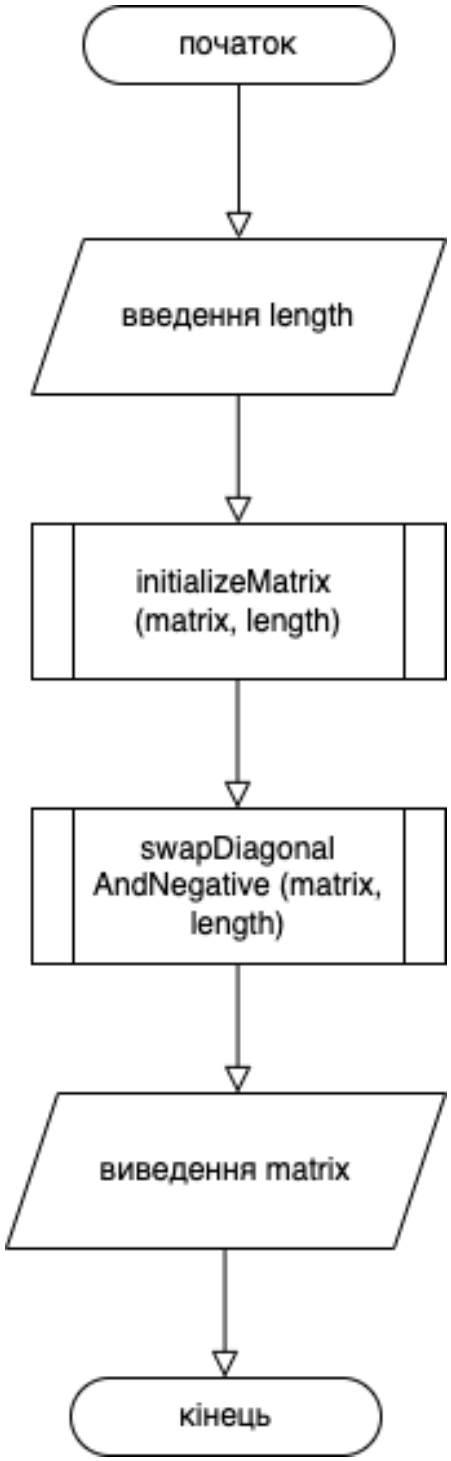
все повторити

все якщо

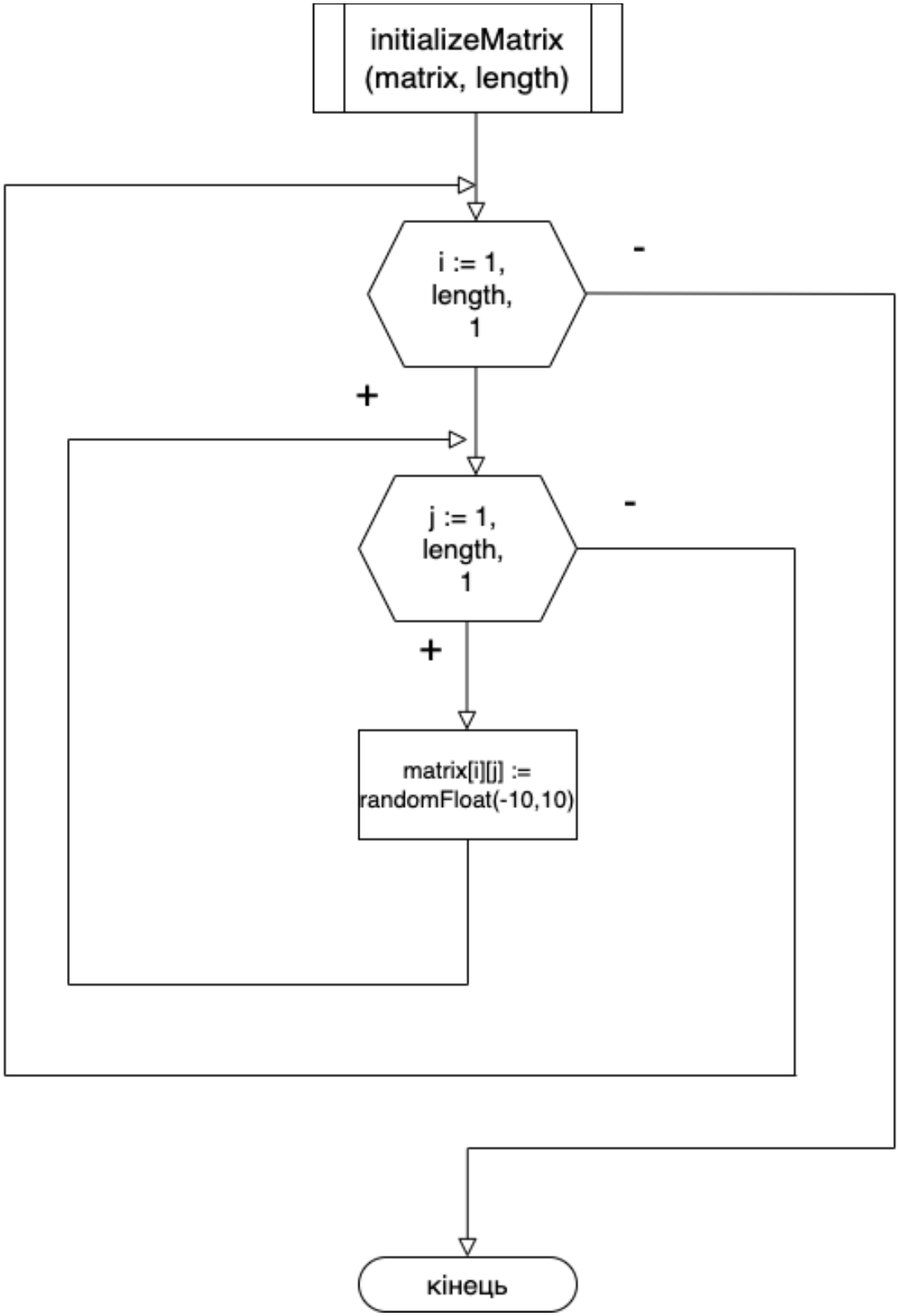
```
            інакше
              то
                isFound := false
                повторити  для j від length до 1, крок -1
                    якщо (matrix[i][j] < 0 && !isFound)
                      то
                        lastNegative := matrix [i][j]
                        lastNegativeIndex := j
                        isFound := true
                    все якщо
                все повторити
            все якщо
            swapArrayElements(matrix[i], i, lastNegativeIndex)
            direction := - direction
        все повторити
кінець

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

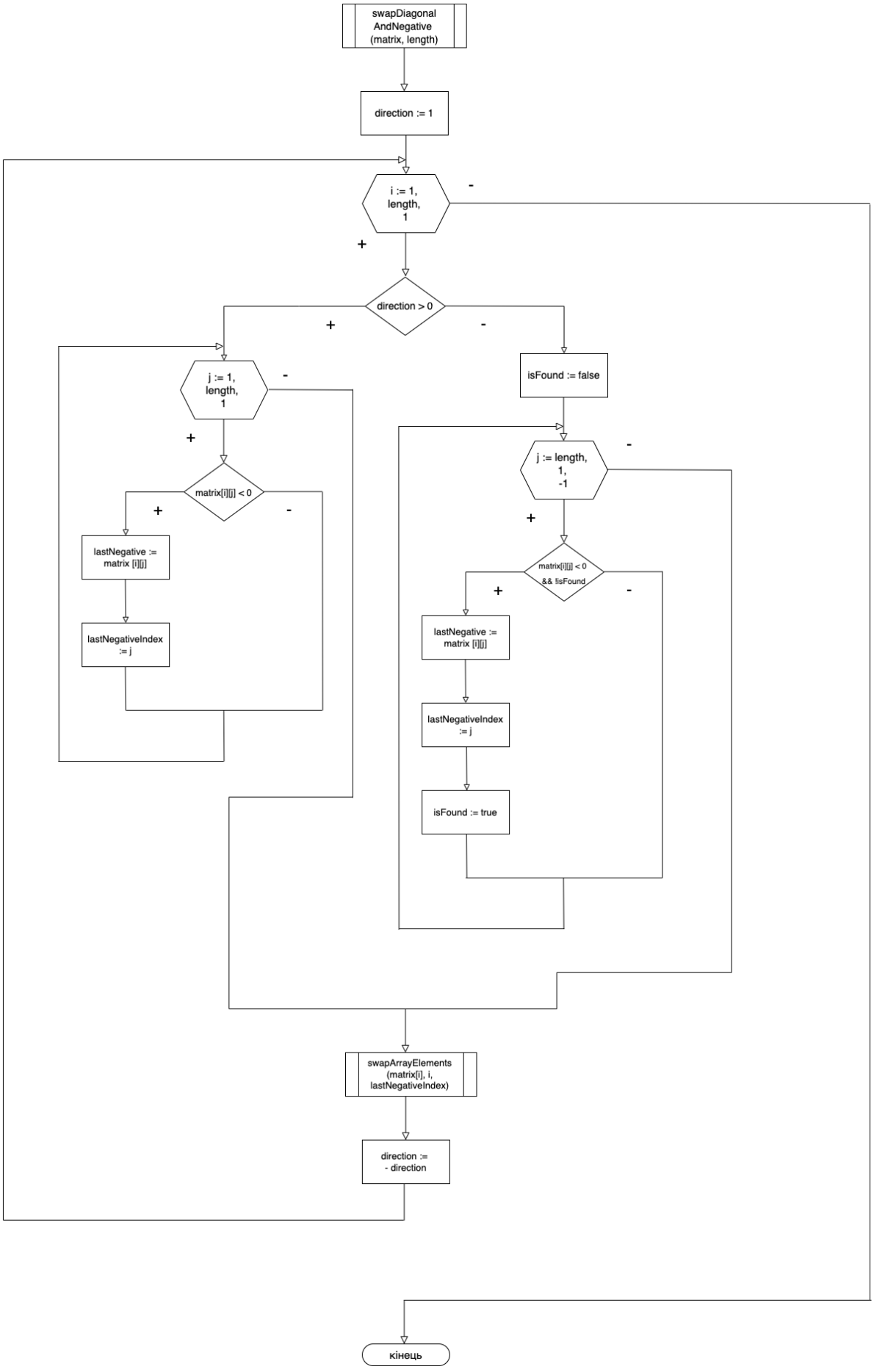
Блок-схема:

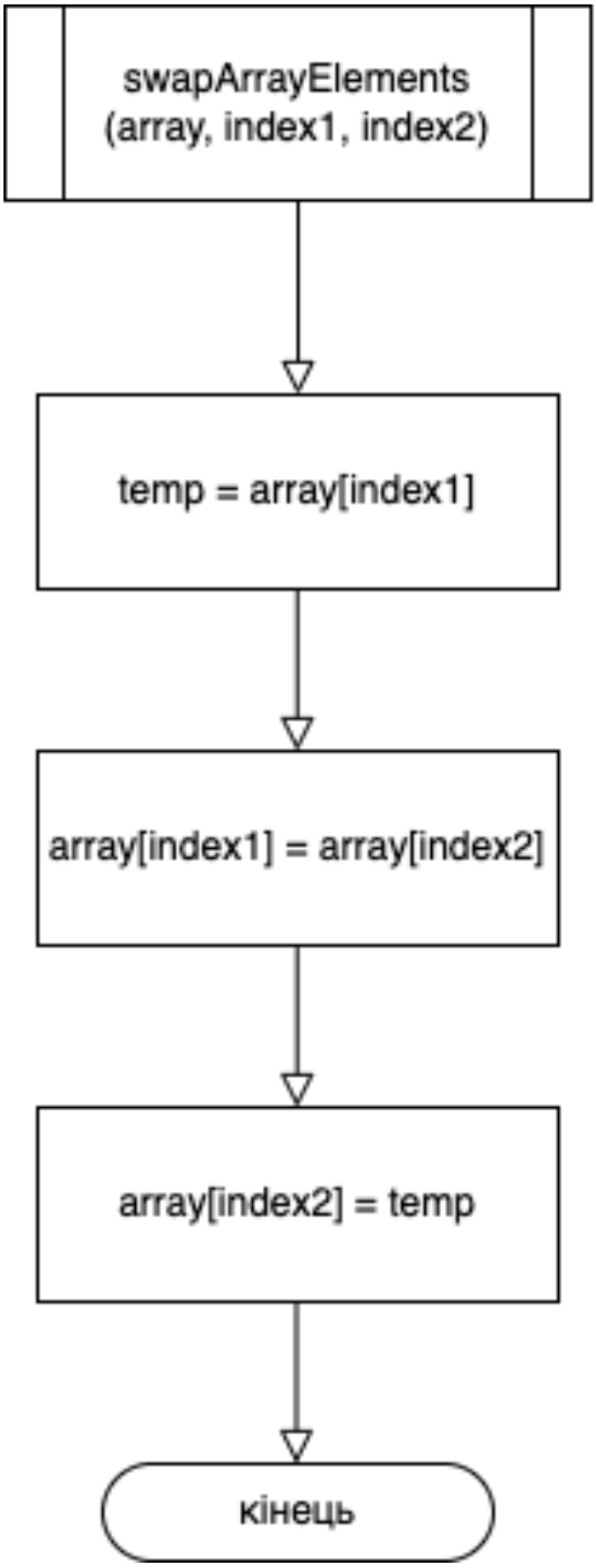
Основна програма:



Підпрограми:




swapDiagonal AndNegative (matrix, length)
direction := 1
i := 1, length, 1
-
+
direction > 0
+
-
j := 1, length, 1
-
+
matrix[i][j] < 0
+
-
lastNegative := matrix [i][j]
lastNegativeIndex := j
isFound := false
j := length, 1, -1
-
+
matrix[i][j] < 0 && !isFound
+
-
lastNegative := matrix [i][j]
lastNegativeIndex := j
isFound := true
swapArrayElements (matrix[i], i, lastNegativeIndex)
direction := - direction
кінець


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

Код:

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;
}
```

Підпрограми:

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


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

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 ініціалізую матрицю випадковими дійсними числами, а також оброблює цю матрицю за даною умовою, використовуючи алгоритму обходу.