Додаток 1

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 масиву”
Варіант 26

Виконав студент ІП-13 Паламарчук Олександр Олександрович
(шифр, прізвище, ім'я, по батькові)

Перевірила Вечерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 9

Дослідження алгоритмів обходу масиву

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

Варіант 26

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.

26	Задано матрицю дійсних чисел A[m,n]. При обході матриці по стовпчиках знайти в ній останній додатний елемент X і його місцезнаходження. Підрахувати кількість елементів над побічною діагоналлю, більших за X.

◆ Постановка задачі

Ініціювати зміну індексованого типу (двовимірний масив) розміру ***m***x***n***, що містить дійсні числа, випадковими числами, для цього розробимо дві функції *initializeMatrix*() та *getRandomFloatNumber*(). Обійти матрицю по стовпчиках і знайти в ній останній додатній елемент ***X*** і його місцезнаходження [***x***, ***y***], для цього розробимо функцію *getTheLastPositiveNumber*(). Знайти кількість елементів, що більші за елемент ***X*** і знаходяться над побічною діагоналлю, для цього розробимо функцію *getTheNumberOfElementsMoreThanX*(). Вихідними даними є елемент ***X*** та його місцезнаходження, кількість елементів над побічною діагоналлю, більших за ***X***.

◆ Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Призначення
ROWS	Натуральний	Початкове дане
COLS	Натуральний	Початкове дане
x	Ціле	Кінцеве дане
y	Ціле	Кінцеве дане
counterMoreThanX	Ціле	Кінцеве дане

X	Дійсний	Кінцеве дане
matrix[][]	Індексований	Проміжне значення
random	Дійсний	Проміжне значення
sign	Дійсний	Проміжне значення
temp	Дійсний	Проміжне значення
counter	Натуральний	Проміжне значення

Складемо таблицю функцій

Назва функції	Синтаксис	Призначення
Random	*rand*()	Генерація випадкового цілого числа з діапазону [0; 32767)

matrix[][] - змінна індексованого типу ***ROWS***x***COLS***, що містить дійсні числа. Для вирішення задачі використаємо допоміжні алгоритми(підпрограми), виклики яких мають вигляд: *initializeMatrix*(***matrix***[][], ***ROWS***, ***COLS***), *getRandomFloatNumber*(), *getTheLastPositiveNumber*(***matrix***[][], ***ROWS***, ***COLS***, ***ROW***, ***COL***), *getTheNumberOfElementsMoreThanX*(***matrix***[][], ***ROWS***, ***COLS***, ***X***), де ***matrix***[][], ***ROWS***, ***COLS***, ***ROW***, ***COL***, ***X*** - це формальні параметри функцій.

Функція *rand*() генерує випадкове ціле число з діапазону [0; 32767), за допомогою певних математичних операцій ми можемо задати потрібний нам діапазон.

◆ Розв’язання

Програмні специфікації запишемо у псевдокоді, графічні схемі у вигляді блок-схеми, та у вигляді коду.

1. Основна програма

Крок 1. Визначимо основні дії

Крок 2. Перевірка умови правильності заданих значень

Крок 3. Ініціалізація матриці.

Крок 4. Знаходження останього додатного.

Крок 5. Знаходження елементів над побічною діагоналлю більших за X

2. Підпрограма *getRandomFloatNumber*()

Крок 1. Визначимо основні дії

Крок 2. Генерація випадкового числа

Крок 3. Генерація випадкового числа для визначення знаку

Крок 4. Обробка числа.

3. Підпрограма *initializeMatrix*(***matrix***[][], ***ROWS***, ***COLS***)

Крок 1. Визначимо основні дії.

Крок 2. Генерація випадкового числа.

Крок 3. Запоанення матриці.

4. Підпрограма *getTheLastPositiveNumber*(***matrix***[][], ***ROWS***, ***COLS***, ***ROW***, ***COL***)

Крок 1. Визначимо основні дії.

Крок 2. Знайдемо останнє додатнє число.

Крок 3. Визначимо його місцезнаходження.

5. Підпрограма *getTheNumberOfElementsMoreThanX*(***matrix***[][], ***ROWS***, ***COLS***, ***X***)

Крок 1. Визначимо основні дії.

Крок 2. Ініціалізуємо x та y.

Крок 3. Збільшимо x

Крок 4. Знайдемо елементи більші за X.

◆ Псевдокод алгоритму основної програми

Крок 5

Початок

Ввід ***ROWS***, ***COLS***

Якщо ***ROWS*** > 1 && ***COLS*** > 1

То

initializeMatrix(***matrix***[][], ***ROWS***, ***COLS***)

X = *getTheLastPositiveNumber*(***matrix***[][], ***ROWS***, ***COLS***, &***x***, &***y***)

counterMoreThanX = *getNumberOfElementsMoreThanX*(***matrix***[][], ***ROWS***, ***COLS***, ***X***)

Інакше

Вивід “Invalid input data”

Все якщо

Вивід ***X***, ***x***, ***y***, ***counterMoreThanX***

Кінець

◆ Псевдокод алгоритму *getRandomFloatNumber*()

getRandomFloatNumber()

random = *rand()* % 10000

sign = 1 + *rand() % 2*

Якщо ***sing*** > 1

То

random = (***random*** * -1) / 100

Інакше

random = ***random*** / 100

Все повторити

Повернути ***random***

Кінець *getRandomFloatNumber*

◆ Псевдокод алгоритму *initializeMatrix*(***matrix***[][], ***ROWS***, ***COLS***)

initializeMatrix(***matrix***[][], ***ROWS***, ***COLS***)

Повторити

Для *i* **від** 0 **до** ***ROWS***, **крок** 1

Повторити

Для *j* **від** 0 **до** ***COLS***, **крок** 1

random = *getRandomFloatNumber*()

matrix[*i*][*j*] = ***random***

Все повторити

Все повторити

Кінець

◆ Псевдокод алгоритму *getTheLastPositiveNumber*(***matrix***[][], ***ROWS***, ***COLS***, ***ROW***, ***COL***)

getTheLastPositiveNumber(***matrix***[][], ***ROWS***, ***COLS***, ***ROW***, ***COL***)

Повторити

Для *i* **від** 0 **до** ***COLS***, **крок** 1

Повторити

Для *j* **від** 0 **до** ***ROWS***, **крок** 1

Якщо ***matrix***[*j*][*i*] > 0

То

temp = ***matrix***[*j*][*i*]

(****ROW)*** =(*j* + 1)

(****COL)*** = (*i* + 1)

Інакше

Все якщо

Все повторити

Все повторити

Повернути ***temp***

Кінець *getTheLastPositiveNumber*

- **Псевдокод алгоритму** *getTheNumberOfElementsMoreThanX*(***matrix***[][], ***ROWS***, ***COLS***, ***X***)

getTheNumberOfElementsMoreThanX(***matrix***[][], ***ROWS***, ***COLS***, ***X***)

x = -1

y = 0

Повторити

Поки ***y*** != ***ROWS*** && ***y*** ! = ***COLS*** - 1

x += 1

Якщо ***x*** == ***COLS*** - 1 - ***counter***

То

y++

x = -1

counter += 1

Інакше якщо ***matrix***[***y***][***x***] > ***X***

То

counterMoreThanX += 1

Все якщо

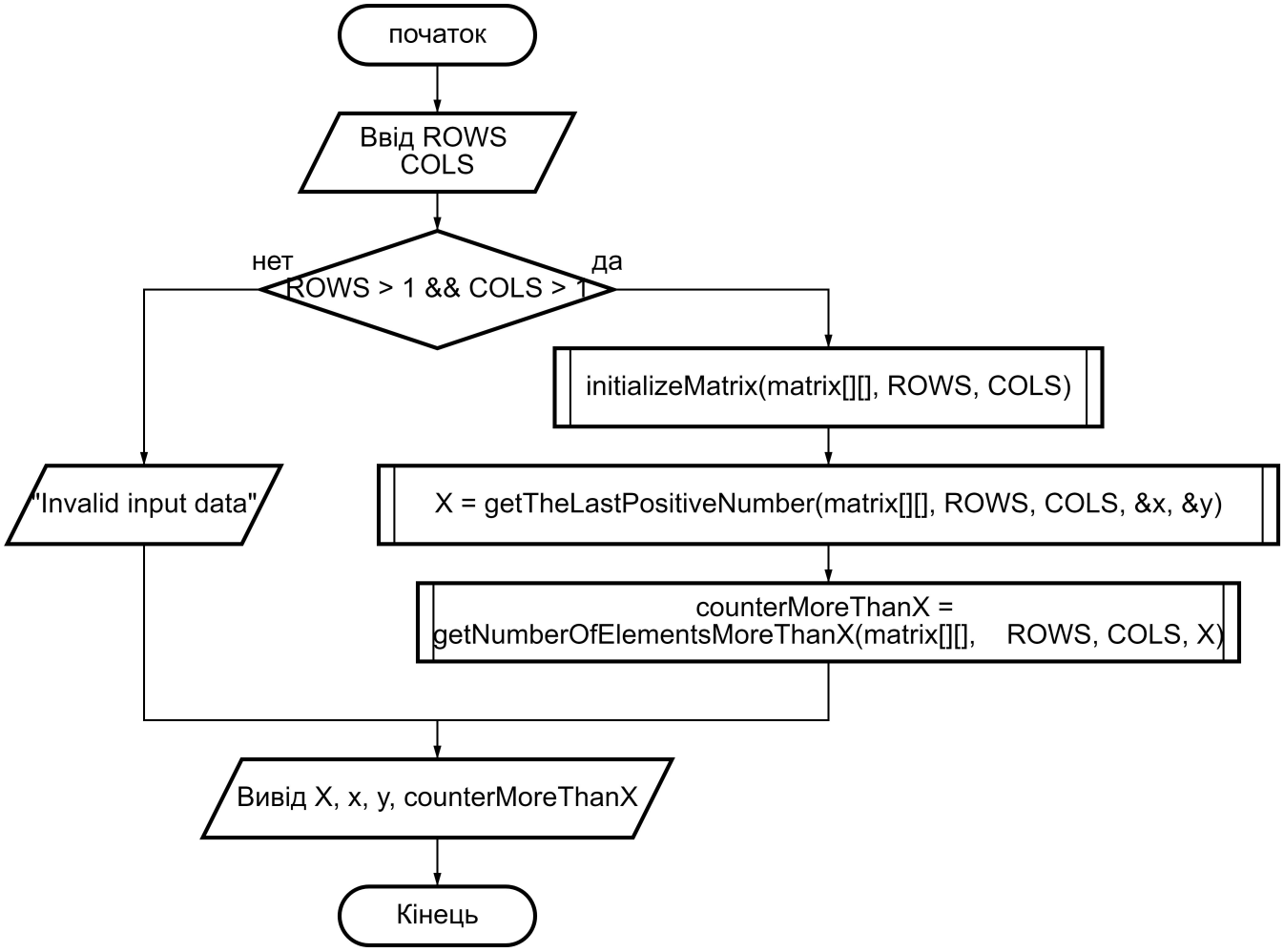
Інакше

Все повторити

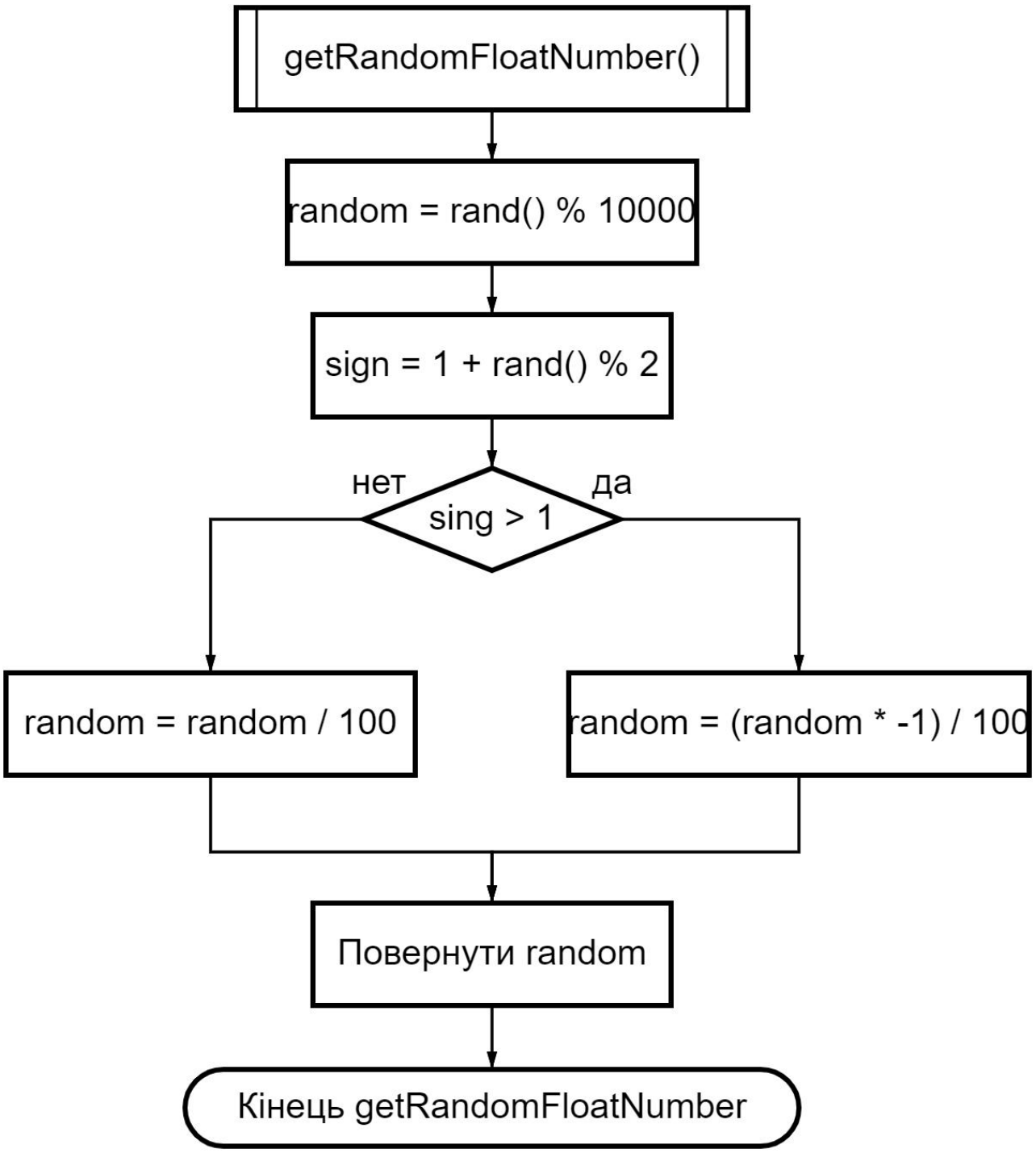
Повернути ***counterMoreThanX***

Кінець *getTheNumberOfElementsMoreThanX*

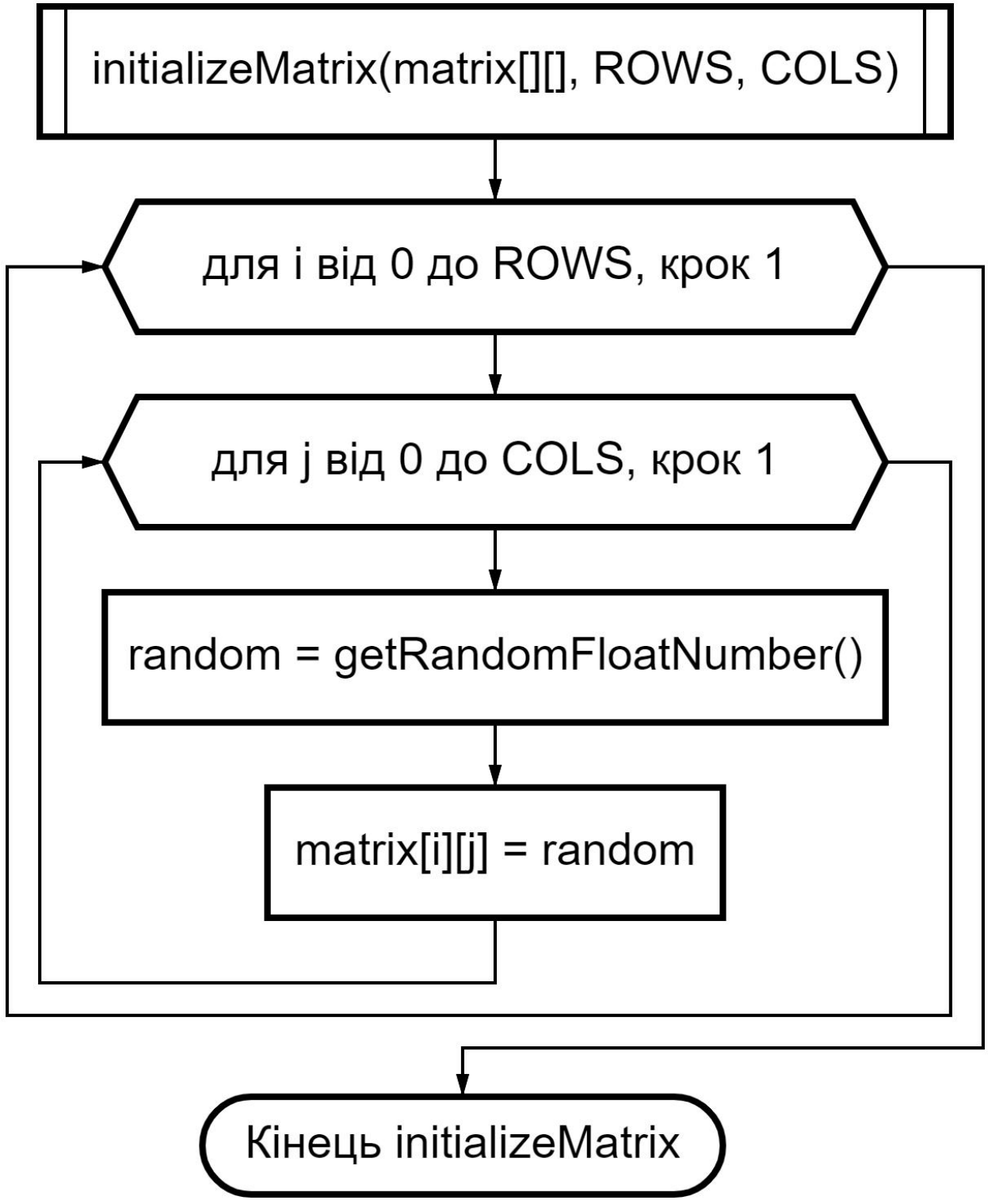
◆ Блок-схема алгоритму основної програми



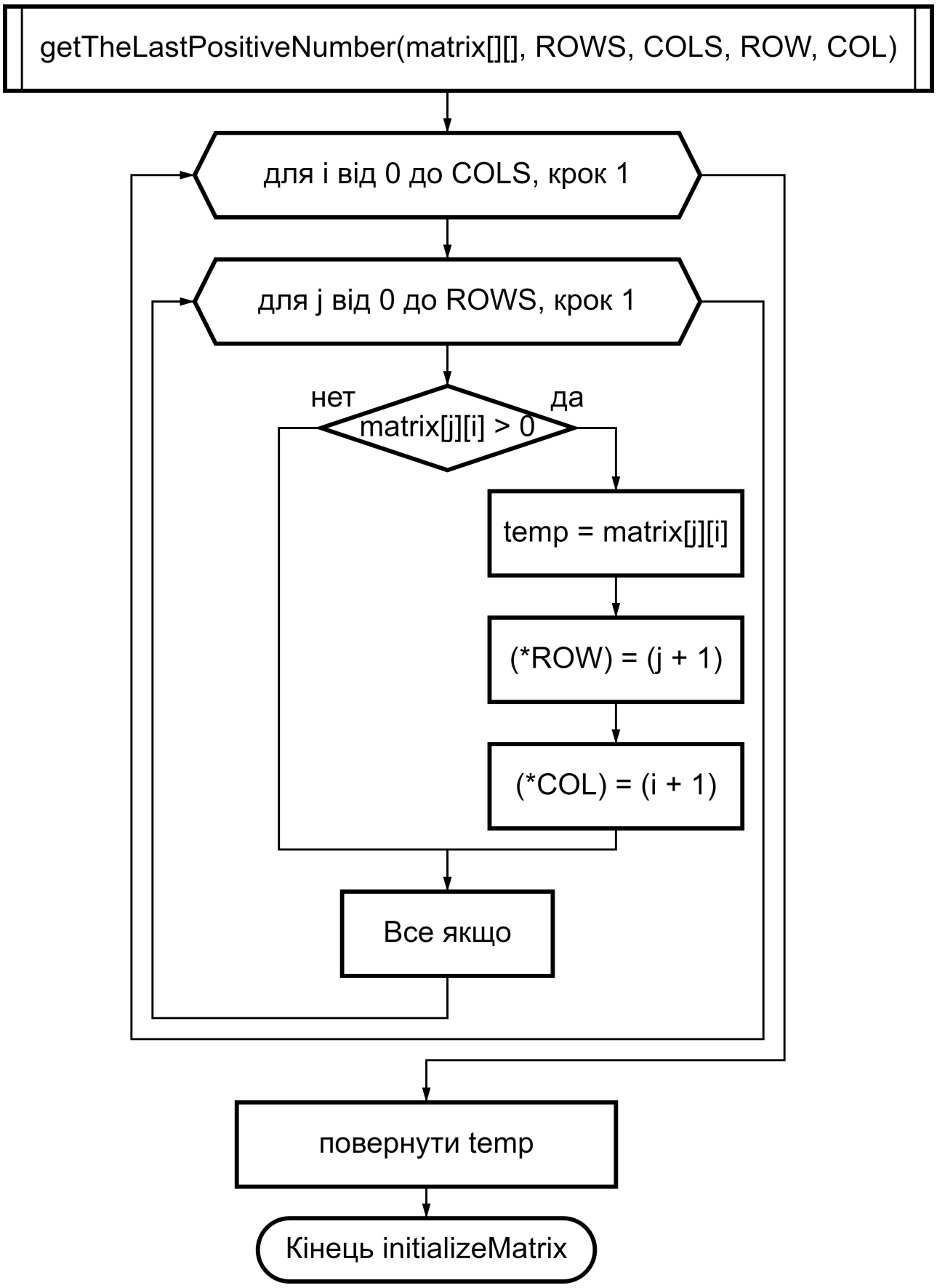
◆ Блок-схема алгоритму *getRandomFloatNumber*()



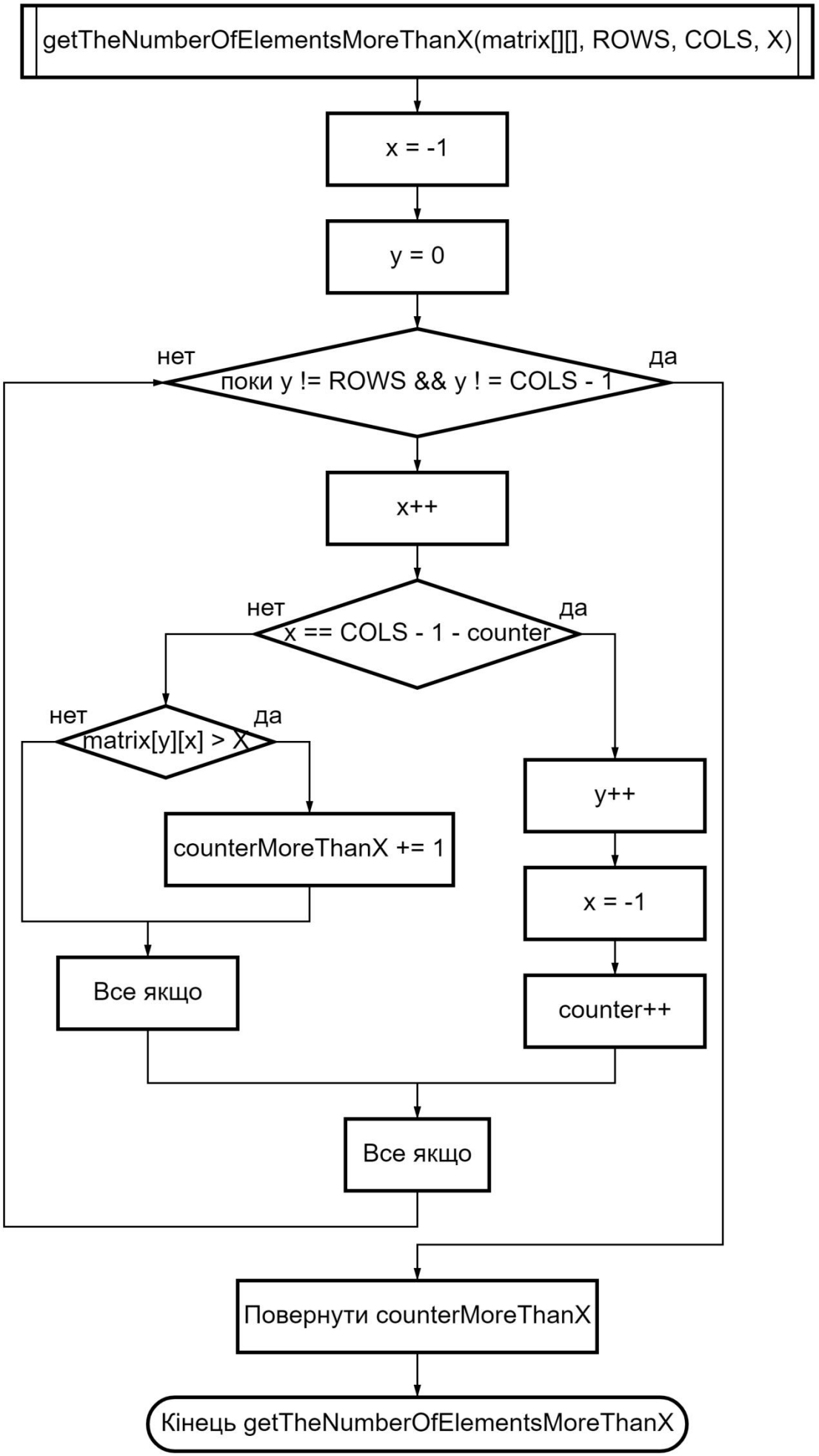
- **Блок-схема алгоритму** *initializeMatrix*(***matrix***[][], ***ROWS***, ***COLS***)



◆ **Блок-схема алгоритму** *getTheLastPositiveNumber*(***matrix***[][], ***ROWS***, ***COLS***, ***ROW***, ***COL***)



◆ **Блок-схема алгоритму** *getTheNumberOfElementsMoreThanX*(***matrix***[][], ***ROWS***, ***COLS***, ***X***)


getTheNumberOfElementsMoreThanX(matrix[][], ROWS, COLS, X)
x = -1
y = 0
нет
поки y != ROWS && y ! = COLS - 1
да
x++
нет
x == COLS - 1 - counter
да
нет
matrix[y][x] > X
да
y++
counterMoreThanX += 1
x = -1
Все якщо
counter++
Все якщо
Повернути counterMoreThanX
Кінець getTheNumberOfElementsMoreThanX

◆ Код алгоритму

```
#include <iostream>
#include <ctime>
#include <Windows.h>
#include <iomanip>

using namespace std;

void initializeMatrix(float **matrix, int, int);
float getTheLastPositiveNumber(float **matrix, int, int, int *ROW, int *COL);
int getTheNumberOfElementsMoreThanX(float **matrix, int, int, float);
void display(float **matrix, int, int, float, int, int, int);
float getRandomFloatNumber();
int main() {

   srand(time(NULL));
   SetConsoleCP(CP_UTF8);
   SetConsoleOutputCP(CP_UTF8);

   int ROWS, COLS, x = 0, y = 0, counterMoreThanX;
   float X;

   cout << "Write a number of rows:";
   cin >> ROWS;
   cout << "Write a number of columns:";
   cin >> COLS;
   cout << endl;

   if(ROWS > 1 && COLS > 1) {
      float** matrix = new float *[ROWS];
      for (int i = 0; i < ROWS; i++) {
         matrix[i] = new float[COLS];
      }
      initializeMatrix(matrix, ROWS, COLS);
      X = getTheLastPositiveNumber(matrix, ROWS, COLS, &x, &y);
      counterMoreThanX = getTheNumberOfElementsMoreThanX(matrix, ROWS, COLS, X);
      display(matrix, ROWS, COLS, X, x, y, counterMoreThanX);

      for (int i = 0; i < ROWS; i++) {
         delete[] matrix[i];
      }
      delete[] matrix;
   }else{
      cout << "Invalid input data";
   }
}

float getRandomFloatNumber(){
   float random = rand() % 10000;
   int sign = 1 + rand() % 2;
   if(sign > 1){
      random = (random * -1) / 100;
```

```
    }
    else{
        random = random / 100;
    }
    return random;
}

void initializeMatrix(float **matrix, int ROWS, int COLS){
    for(int i = 0; i < ROWS; i++){
        for(int j = 0; j < COLS; j++){
            float random = getRandomFloatNumber();
            matrix[i][j] = random;
        }
    }
}

float getTheLastPositiveNumber(float **matrix, int ROWS, int COLS, int *ROW, int *COL){
    float temp = 0;
    for(int i = 0; i < COLS; i++){
        for(int j = 0; j < ROWS; j++){
            if(matrix[j][i] > 0){
                temp = matrix[j][i];
                (*ROW) = (j + 1);
                (*COL) = (i + 1);
            }
        }
    }
    return temp;
}

int getTheNumberOfElementsMoreThanX(float **matrix, int ROWS, int COLS, float X){
    int x = -1, y = 0, counter = 0, counterMoreThanX = 0;
    while(y != ROWS && y != COLS - 1){
        x++;
        if(x == COLS - 1 - counter){
            y++;
            x = -1;
            counter++;
        }
        else if (matrix[y][x] > X){
            counterMoreThanX++;
        }
    }
    return counterMoreThanX;
}

void display(float **matrix, int ROWS, int COLS, float X, int x, int y, int moreThanX){

    for(int i = 0; i < ROWS; i++){
        for(int j = 0; j < COLS; j++){
            cout << matrix[i][j] << "\t" ;
        }
        cout << "\n";
```

```
    }
    cout << endl;
    cout << "The last positive number is " << X << endl;
    cout << X << " has position: " << "[" << x << ", " << y << "]" << endl;
    cout << "The number of elements more than " << X << ": "<< moreThanX;
}
```

◆ Тестування алгоритму

```
-13.23  -57.98  -14.43  -54.15  -43.04  14.02   -32.1   50.92
2.48    -9.1    95.07   4.35    89.73   -1.08   26.25   44.89
-83.48  99.08   -91.64  -37.77  -6.32   55.12   -36.17  32.38
75.92   88.32   -1.12   -88.89  -67.24  41.51   -35.64  26.57
18.04   48.91   57.74   32.22   -39.06  85.45   47.7    73.48
-7.63   60.95   -1.69   6.23    -87.53  96.01   13.33   86.65

The last positive number is 86.65
86.65 has position: [6, 8]
The number of elements more than 86.65: 4
Process finished with exit code 0
```

◆ Висновок

На лабораторній роботі було декомпозовано задачу на такі етапи основна програма: перевірка умови правильності заданих значень, ініціалізація матриці, знаходження останього додатного, знаходження елементів над побічною діагоналлю більших за X. Підпрограма *getRandomFloatNumber*(): генерація випадкового числа, генерація випадкового числа для визначення знаку, обробка числа. Підпрограма *initializeMatrix:* генерація випадкового числа, заповнення матриці. Підпрограма *getTheLastPositiveNumber:* знайдемо останнє додатнє число, визначимо його місцезнаходження. Підпрограма *getTheNumberOfElementsMoreThanX*: ініціалізуємо x та y, збільшимо x, знайдемо елементи більші за X. Було досліджено алгоритми обходу масивів та набуто практичних навичок використання цих алгоритмів під час складання програмних специфікацій.