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 масивів»

Варіант 15

Виконав студент ІП-12, Кириченко Владислав Сергійович
(шифр, прізвище, ім'я, по батькові)

Перевірив _____
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота № 9

Назва роботи: Дослідження алгоритмів обходу масивів

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

Варіант 15

Умова задачі*:*

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

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом (*Розмірність* - ***m*** x ***n*** . *Тип данних елементів* - ***Дійсний***).
2. Ініціювання змінної, що описана в п.1 даного завдання.
3. Обчислення змінної, що описана в п.1, згідно з варіантом
4. **Постановка задачі**:

Початкові дані **-** із початкових даних маємо лише розмір двовимірного масиву(***m*** *x* ***n***) *та дійсне число*.

Згенерувати двовимірний масив випадкових дійсних значень(представлення матриці у алгортитмі),визначити чи знаходиться задане число серед елементів масиву, я кщо так, то визначити його місцезнаходження, потім визначити кількість елементів, більших за введене число, під головною діагоналлю

Результат - одновимірний масив (розмір - 2, тип даних - цілочисельний) , що представдляє дані про розміщення елемента (якщо елемент наявний у масиві) і цілочислельна змінна значення якої - кількість елементів, більших за введене число, під головною діагоналлю.

Побудова математичної моделі:

Для реалізації алгоритму вирішення поставленої задачі нам портібен засіб генерації випадкового дійсного числа, нехай це буде функція ***randRealN()***, від
random real number(з англійської випадкове дійсне число). У коді програми буде використано стандартний метод генерації випадкового числа, саме: функція **rand().** Але т.я. ця функція повертає випадкове ціле значення, то дещо модивікуємо вираз і отримаємо **rand()%201 - 100 + double(rand()%100)/100**

(генерує дійсне число с проміжку [-100, 100])

Складемо таблицю змінних:

Змінна	Тип	Ім’я	Призначення
Представлення матрці	індексований	***arr***	Проміжні дані
Кількість рядків матриці	цілочисельний	***row***	Початкові дані
Кількість стовпців матриці	цілочисельний	***col***	Початкові дані
Задане дійсне число	дійсний	***num***	Початкові дані
Кількість елементів, більших за введене число, під головною діагоналлю	цілочисельний	***countGreater***	Проміжні дані
Визначення статусу (знайдено число, чи ще не знайдено)	булевий (логічний)	***found***	Проміжні дані (тільки підпрограма)
Кількість елементів, більших за введене число, під головною діагоналлю	цілочисельний	***counter***	Проміжні дані (тільки підпрограма)
Лічильник	цілочисельний	***i***	Проміжкове значення
Лічильник	цілочисельний	***j***	Проміжкове значення

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

Крок 1. Визначимо основні дії.
Крок 2.Деталізація ініціалізації ***arr, numCoord***
Крок 3.Деталізація заповнення масиву ***arr***
Крок 4.Деталізація знаходження місцезнаходження числа ***num*** *у матриці(якщо міститься)*
Крок 5.Деталізація обчислення кількості елементів, більших за введене число, під головною діагоналлю матриці.
Крок 6.Деталізація виведення

Псевдокод(основна програма):

Крок 1.

початок
введення **row, colm, num**
ініціалізація ***arr, numCoord***
заповнення масиву ***arr***
знаходження місцезнаходження числа ***num***
обчислення кількості елементів матриці, більших за введене число, під головною діагоналлю
виведення даних
виведення **numCoord** (в залежності чи знайдено число)**, countGreater**
кінець

Крок 2.

початок
введення **row, colm, num**
arr[row,col]
numCoord[2]
заповнення масиву ***arr***
знаходження місцезнаходження числа ***num***
обчислення кількості елементів матриці, більших за введене число, під головною діагоналлю
виведення даних
виведення **numCoord** (в залежності чи знайдено число)**, countGreater**
кінець

Крок 3.

початок
введення **row, colm, num**
arr[row,col]
numCoord[2]
fillArr(arr, row, col)
знаходження місцезнаходження числа ***num***
обчислення кількості елементів матриці, більших за введене число, під головною діагоналлю
виведення даних
виведення **numCoord** (в залежності чи знайдено число)**, countGreater**
кінець
Крок 4.

початок
введення **row, colm, num**
arr[row,col]
numCoord[2]
fillArr(arr, row, col)
numCoord = findNum(arr, row, col, num)
обчислення кількості елементів матриці, більших за введене число, під головною діагоналлю
виведення даних
виведення **numCoord** (в залежності чи знайдено число)**, countGreater**
кінець

Крок 5.

початок
введення **row, colm, num**
arr[row,col]
numCoord[2]
fillArr(arr, row, col)
numCoord = findNum(arr, row, col, num)
countGreater = countGreaterBelowDiagonal(arr,row,col,num)
виведення даних
виведення **numCoord** (в залежності чи знайдено число)**, countGreater**
кінець

Крок 6.

початок
введення **row, colm, num**
arr[row,col]
numCoord[2]
fillArr(arr, row, col)
numCoord = findNum(arr, row, col, num)
countGreater = countGreaterBelowDiagonal(arr,row,col,num)
якщо numCoord[0]==-1
то
*виведення "Число не було знайдено серед елементівматриці",***countGreater**
інакше
виведення **numCoord, countGreater**
все якщо
кінець

Підпрограма ***fillArr***

Крок 1

функція ***fillArr(arr[][], row, col)***
проходження по всіх елементах матриці за допомогою вкладеного циклу
присвоєння кожному елементу матриці випадкового дійсного значення
кінець

Крок 2.

функція ***fillArr(arr[][], row, col)***
повторити
для i від 0 до row із кроком 1
повторити
для j від 0 до col із кроком 1
присвоєння елементу матріці випадкового дійсного значення
все повторити
все повторити
кінець

Крок 3.

функція ***fillArr(arr[][], row, col)***
повторити
для i від 0 до row із кроком 1
повторити
для j від 0 до col із кроком 1
arr[i][j]= randRealN()
все повторити
все повторити
кінець

Підпрограма ***findNum***

Крок 1.

функція ***findNum(arr[][], row,col, num)***
  ініціалізація змінних **found, numCoord**
  проходження по всіх елементах матриці за допомогою вкладеного циклу (змійка)
  перевірка чи поточний елемент масиву дорівнює ***num***
  знаходження місцезнаходження у випадку справдження умови
кінець

Крок 2.

функція ***findNum(arr[][], row,col, num)***
  found = false
  numCoord[2] = {-1, -1}
  проходження по всіх елементах матриці за допомогою вкладеного циклу (змійка)
  перевірка чи поточний елемент масиву дорівнює ***num***
  знаходження місцезнаходження у випадку справдження умови
кінець

Крок 3.

функція ***findNum(arr[][], row,col, num)***

```
 found = false
 numCoord[2] = {-1, -1}
 i = 0
 j = 0
 повторити
          поки i < row &&  found != true
          якщо i%2==0
            то
                  повторити
                        поки  j < col &&  found != true
                        перевірка чи поточний елемент масиву дорівнює num
                        знаходження місцезнаходження у випадку справдження
                  умови
                        j++
            інакше
                  повторити
                        поки  j > -1 &&  found != true
                        перевірка чи поточний елемент масиву дорівнює num
                        знаходження місцезнаходження у випадку справдження
                  умови
                        j--
            все повторити
            i++
 все повторити

кінець
```

Крок 4.

функція ***findNum(arr[][], row,col, num)***

```
 found = false
 numCoord[2] = {-1, -1}
  j = 0
 повторити
            поки i < row &&  found != true
            якщо i%2==0
              то
                    повторити
                          поки  j < col &&  found != true
                          якщо arr[i][j] == num
                          то
                                знаходження місцезнаходження у випадку
                          справдження умови
                          все якщо

                          j++
              інакше
                    повторити
                          поки  j > -1 &&  found != true
                          якщо arr[i][j] == num
                          то
                                знаходження місцезнаходження у випадку
                          справдження умови
                          все якщо
                          j--
              все повторити
              i++
  все повторити
кінець
```

Крок 5.
функція ***findNum(arr[][], row,col, num)***

```
 found = false
 numCoord[2] = {-1, -1}
  j = 0
 повторити
            поки i < row &&  found != true
            якщо i%2==0
              то
                    повторити
                          поки  j < col &&  found != true
                          якщо arr[i][j] == num
                          то
                                numCoord[0] = i+1
                                numCoord[1] = j+1
                                 found = true

                          все якщо

                          j++
              інакше
                    повторити
                          поки  j > -1 &&  found != true
                          якщо arr[i][j] == num
                          то
                                numCoord[0] = i+1
                                numCoord[1] = j+1
                                 found = true
                          все якщо
                          j--
              все повторити
              i++
 все повторити
кінець
```

Підпрограма ***countGreaterBelowDiagonal***

Крок 1.

функція ***countGreaterBelowDiagonal(arr[][],row,col,num)***
 ініціалізація змінної **counter**
 проходження по всіх елементах матриці нижче головної діагоналі за допомогою вкладеого циклу
 перевірка чи поточний елемент масиву дорівнює ***num***
 збільшення значення ***counter*** *на 1*

кінець

Крок 2.

функція ***countGreaterBelowDiagonal(arr[][],row,col,num)***
 counter = 0
 проходження по всіх елементах матриці нижче головної діагоналі за допомогою вкладеого циклу
 перевірка чи поточний елемент масиву дорівнює ***num***
 збільшення значення ***counter*** *на 1*
кінець

Крок 2.

функція ***countGreaterBelowDiagonal(arr[][],row,col,num)***
```
 counter = 0
 повторити
       для і від 0 до row із кроком 1
            j =0
            повторити
               поки j < i && j < col
                     перевірка чи поточний елемент масиву дорівнює num
                     збільшення значення counter на 1
                     j++
            все повторити
 все повторити
кінець
```

Крок 3.

функція ***countGreaterBelowDiagonal(arr[][],row,col,num)***

```
 counter = 0
 повторити
       для і від 0 до row із кроком 1
             j =0
             повторити
                поки j < i && j < col
                      якщо arr[i][j] > num
                            то
                              збільшення значення counter на 1
                      все якщо
             все повторити
 все повторити
кінець
```

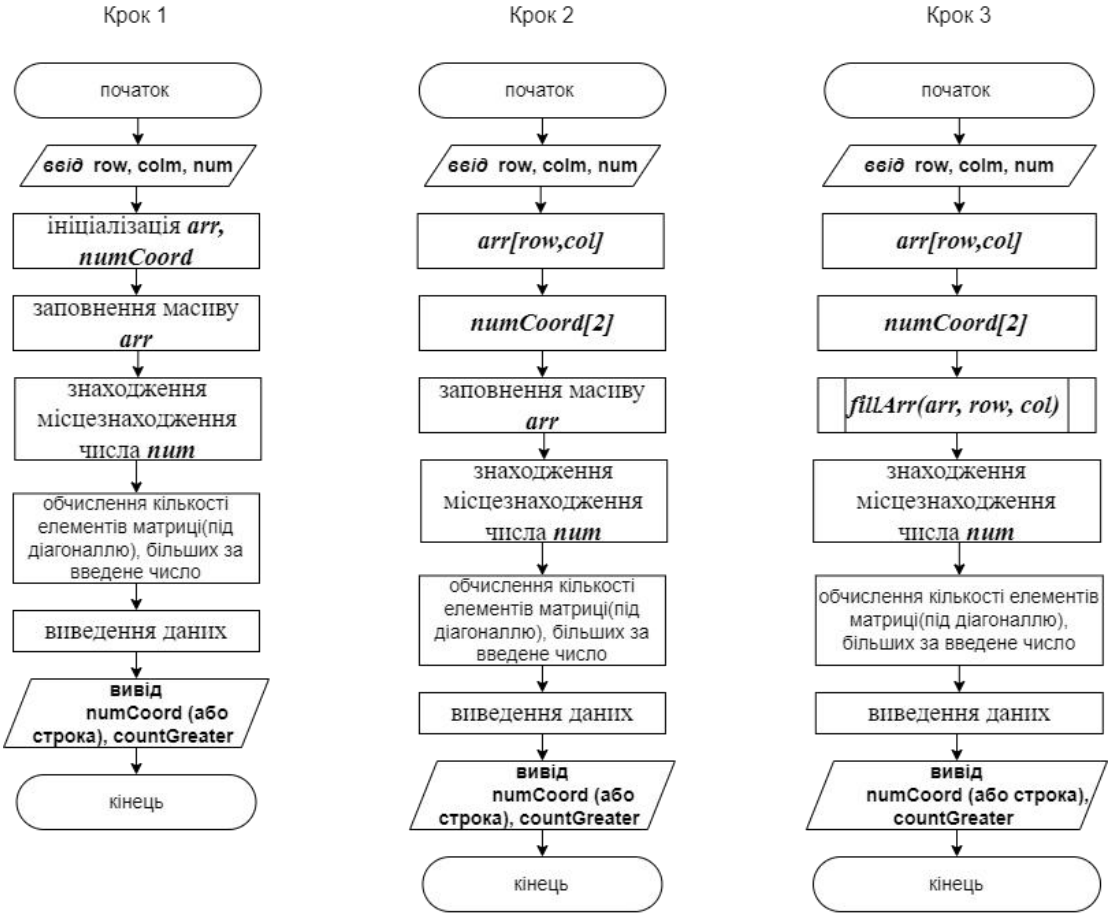
Крок 4.

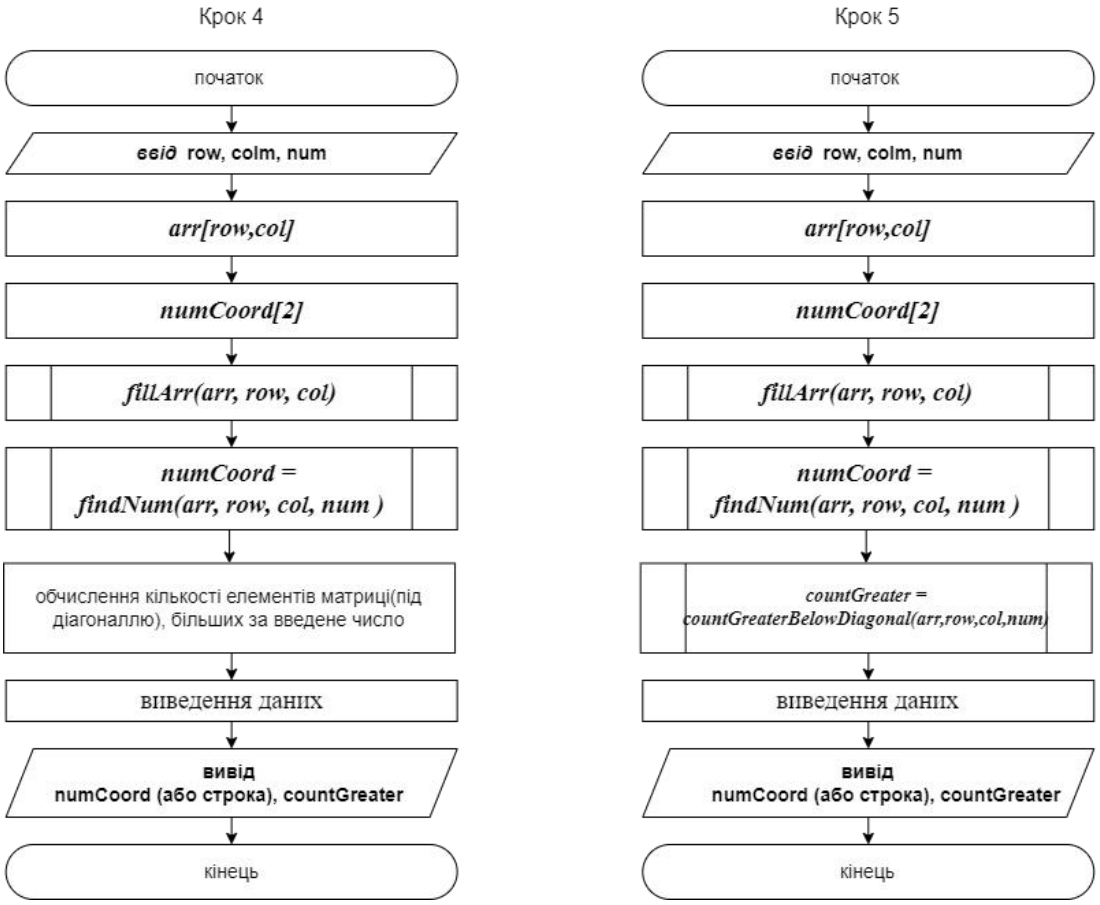
функція ***countGreaterBelowDiagonal(arr[][],row,col,num)***

```
 counter = 0
 повторити
       для і від 0 до row із кроком 1
             j =0
             повторити
                поки j < i && j < col
                      якщо arr[i][j] > num
                            то
                             counter++
                      все якщо
             все повторити
 все повторити
кінець
```

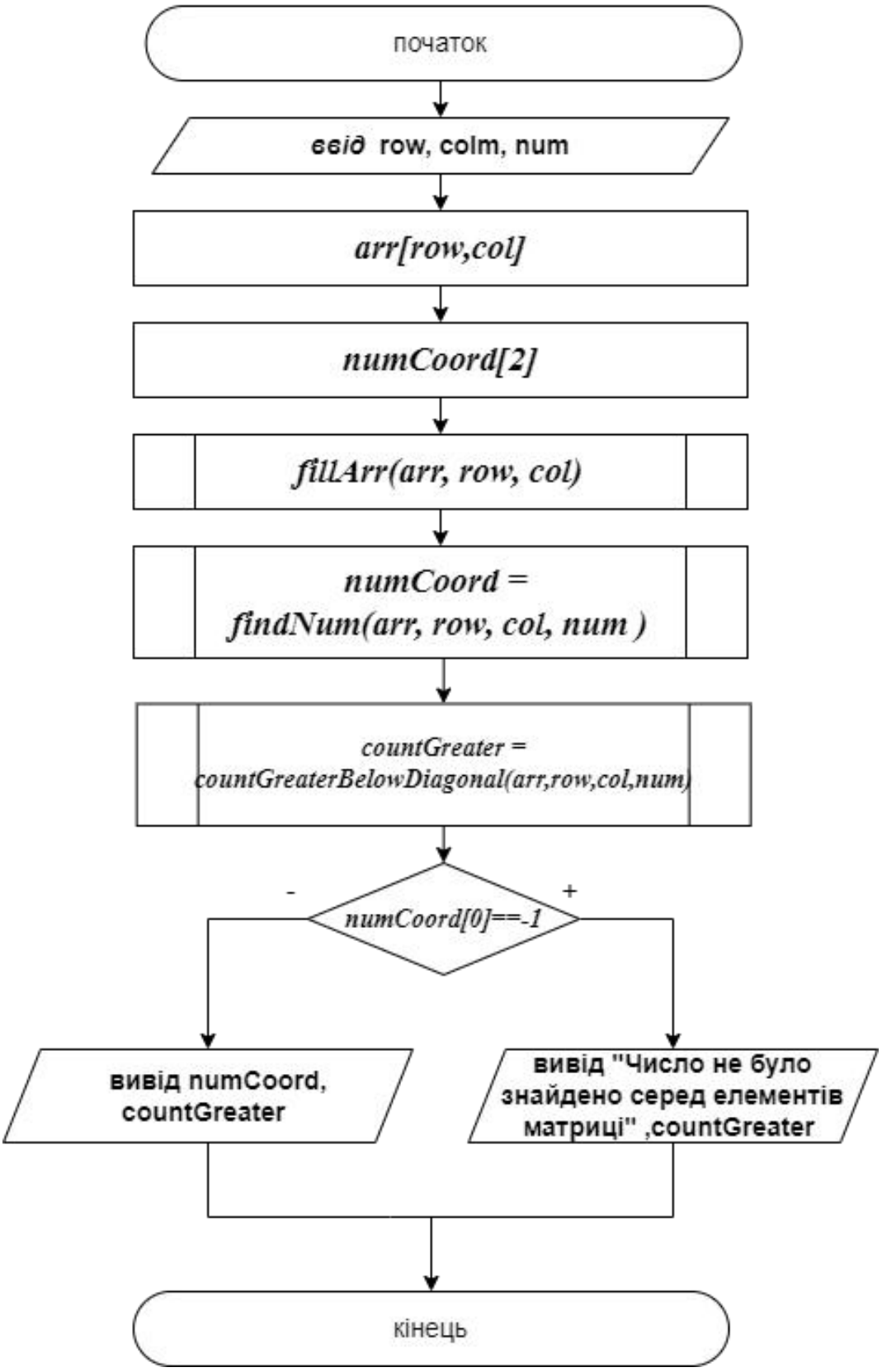
Блок схема:

Основна програма



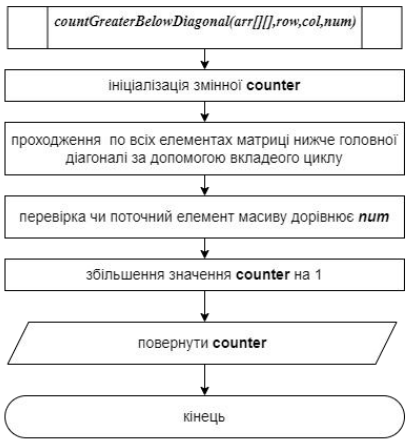


Крок 6

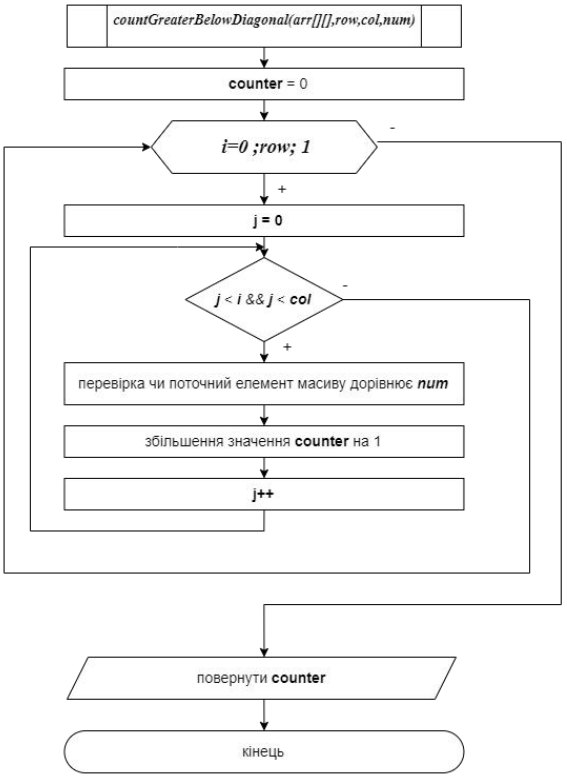

початок
ввід row, colm, num
arr[row,col]
numCoord[2]
fillArr(arr, row, col)
numCoord =
findNum(arr, row, col, num)
countGreater =
countGreaterBelowDiagonal(arr,row,col,num)
-
numCoord[0]==-1
+
вивід numCoord,
countGreater
вивід "Число не було
знайдено серед елементів
матриці" ,countGreater
кінець

Підпрограма ***countGreaterBelowDiagonal***

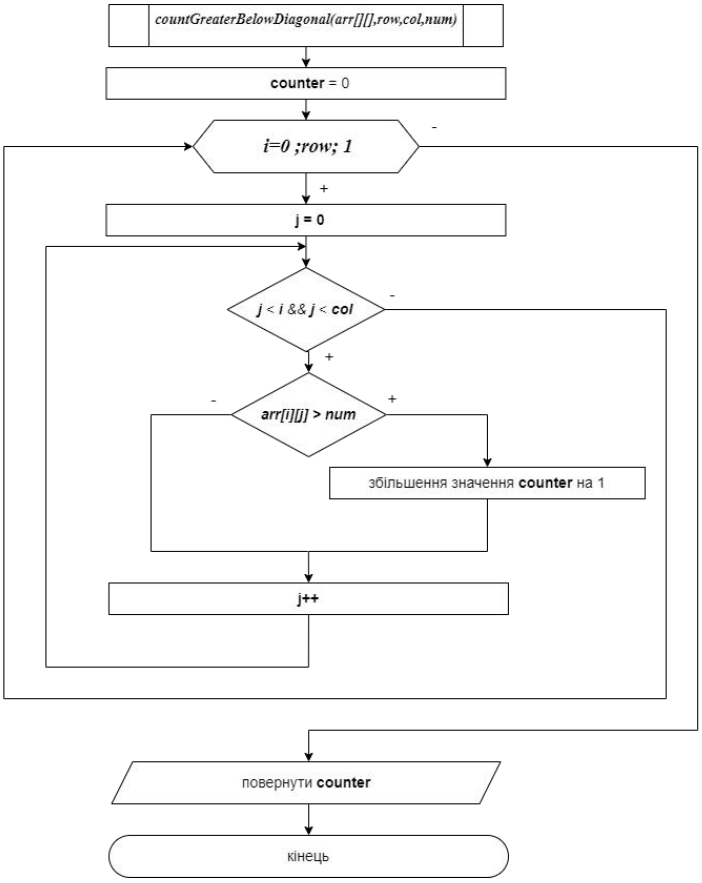
Крок 1



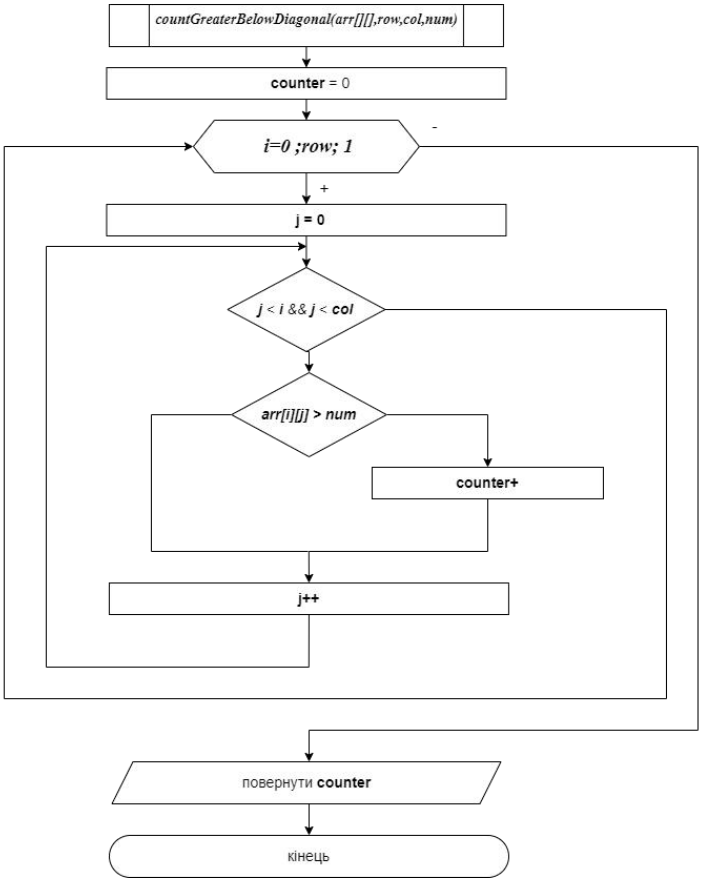
Крок 2



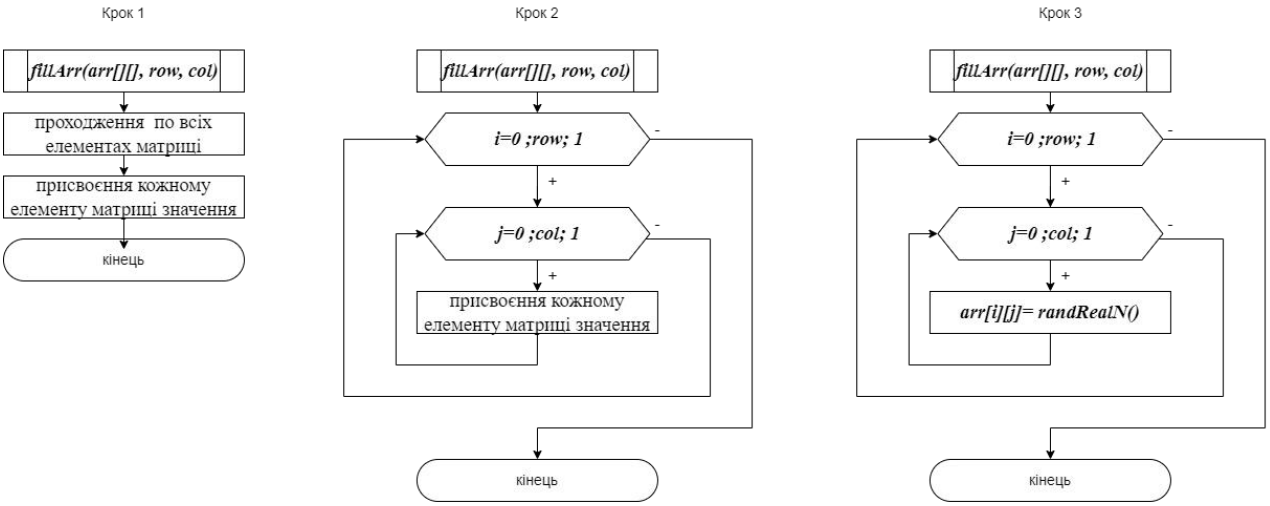
Крок 3



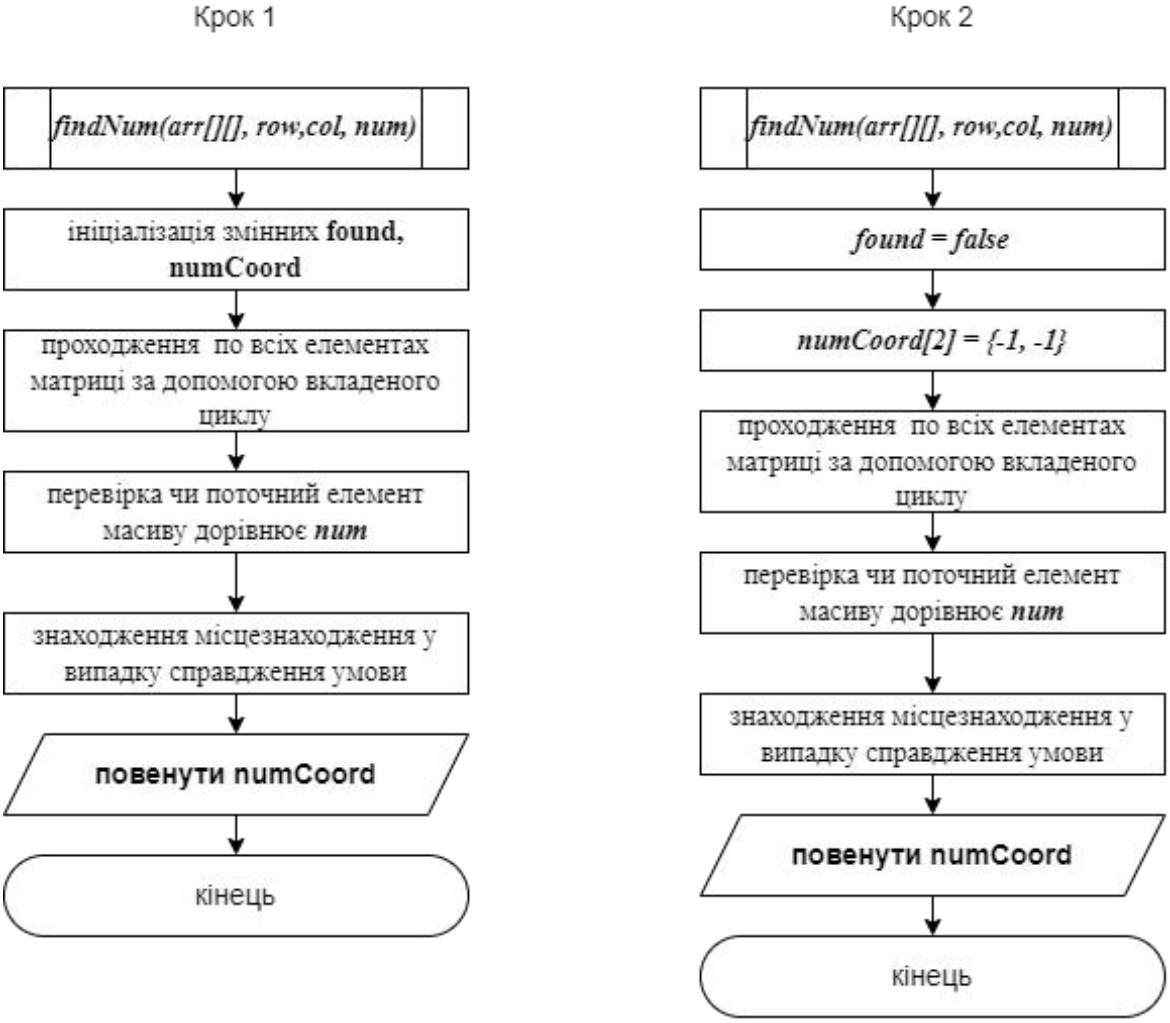
Крок 4



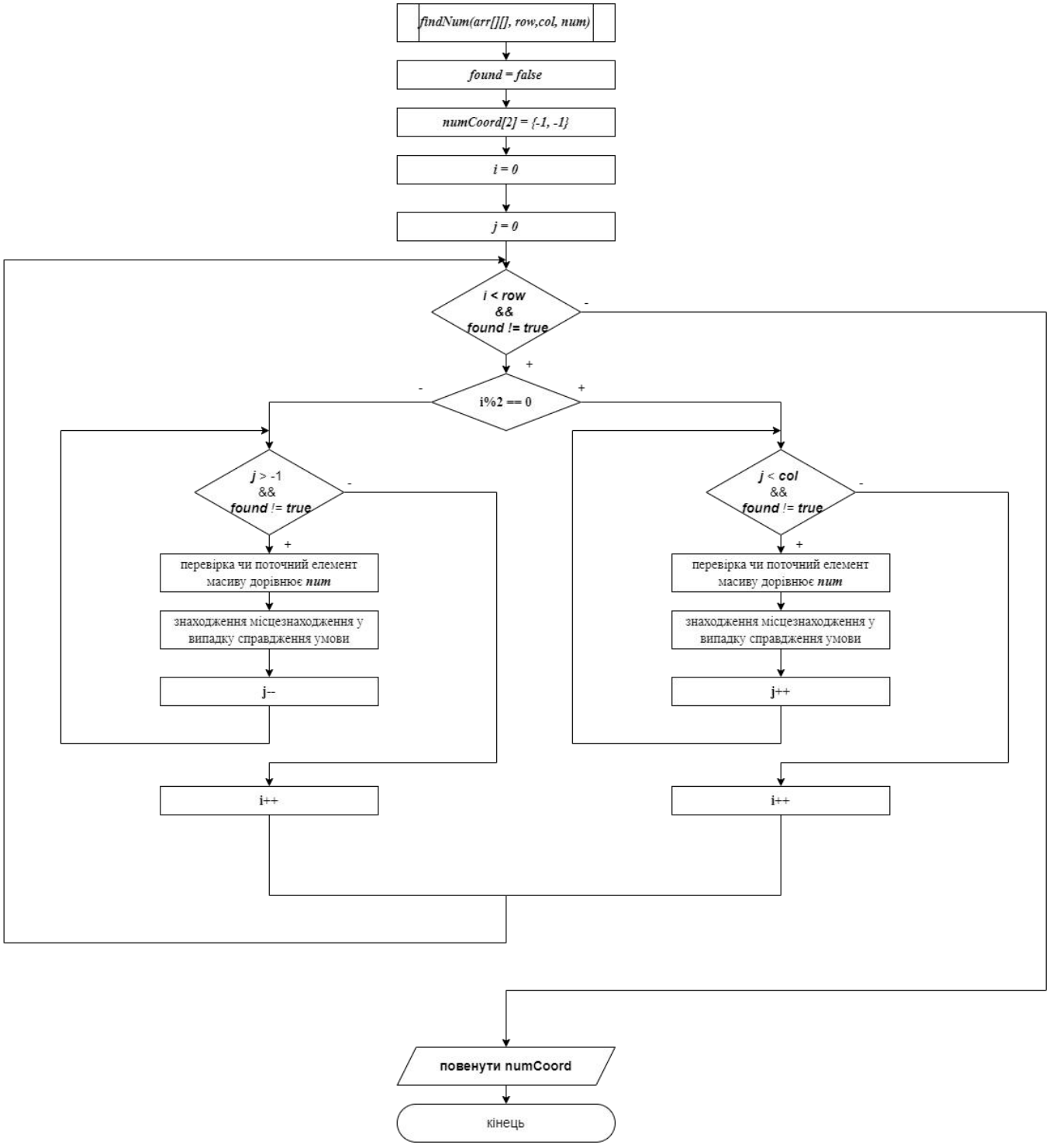
*Підпрограма **fillArr***



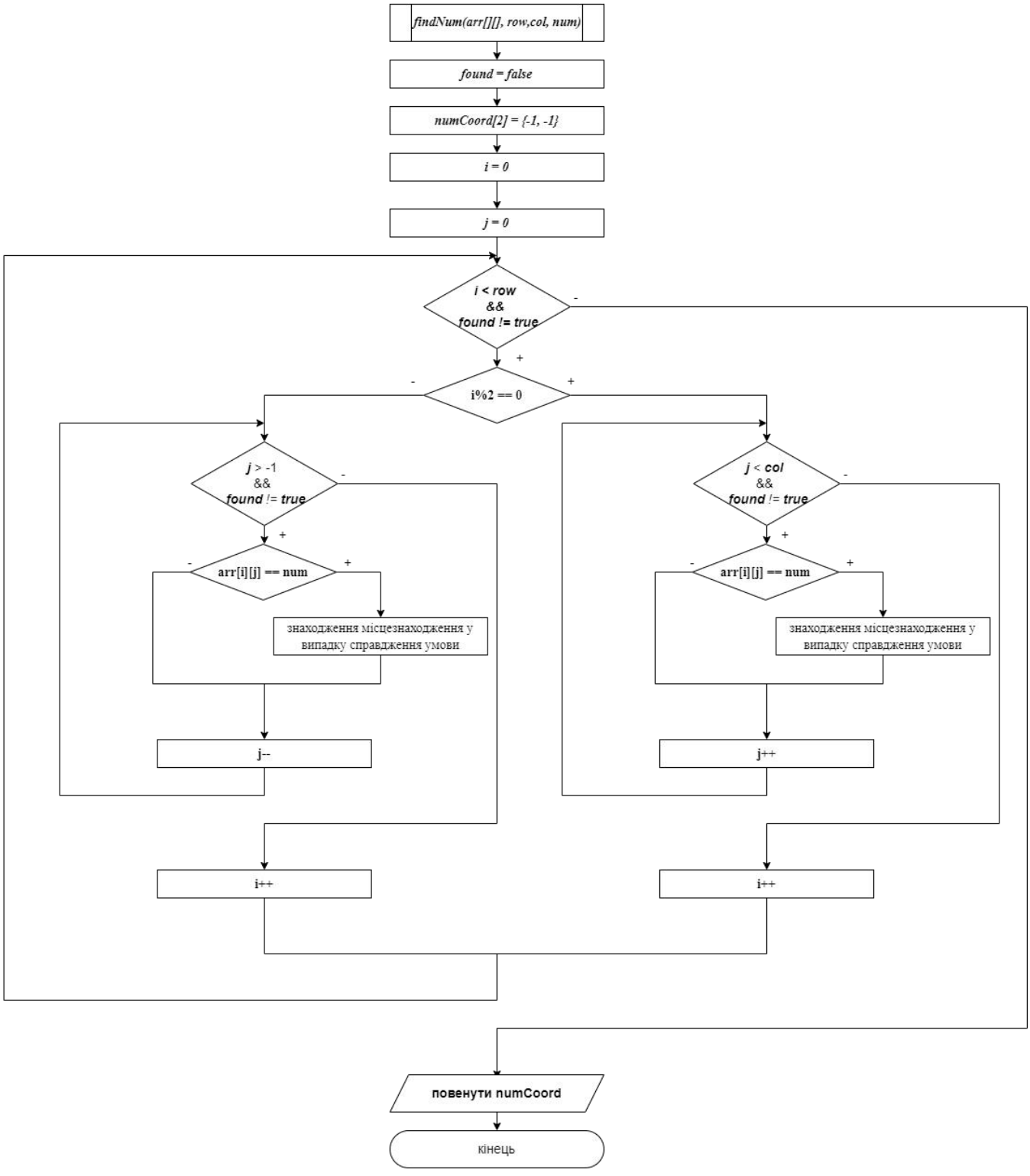
*Підпрограма **findNum***



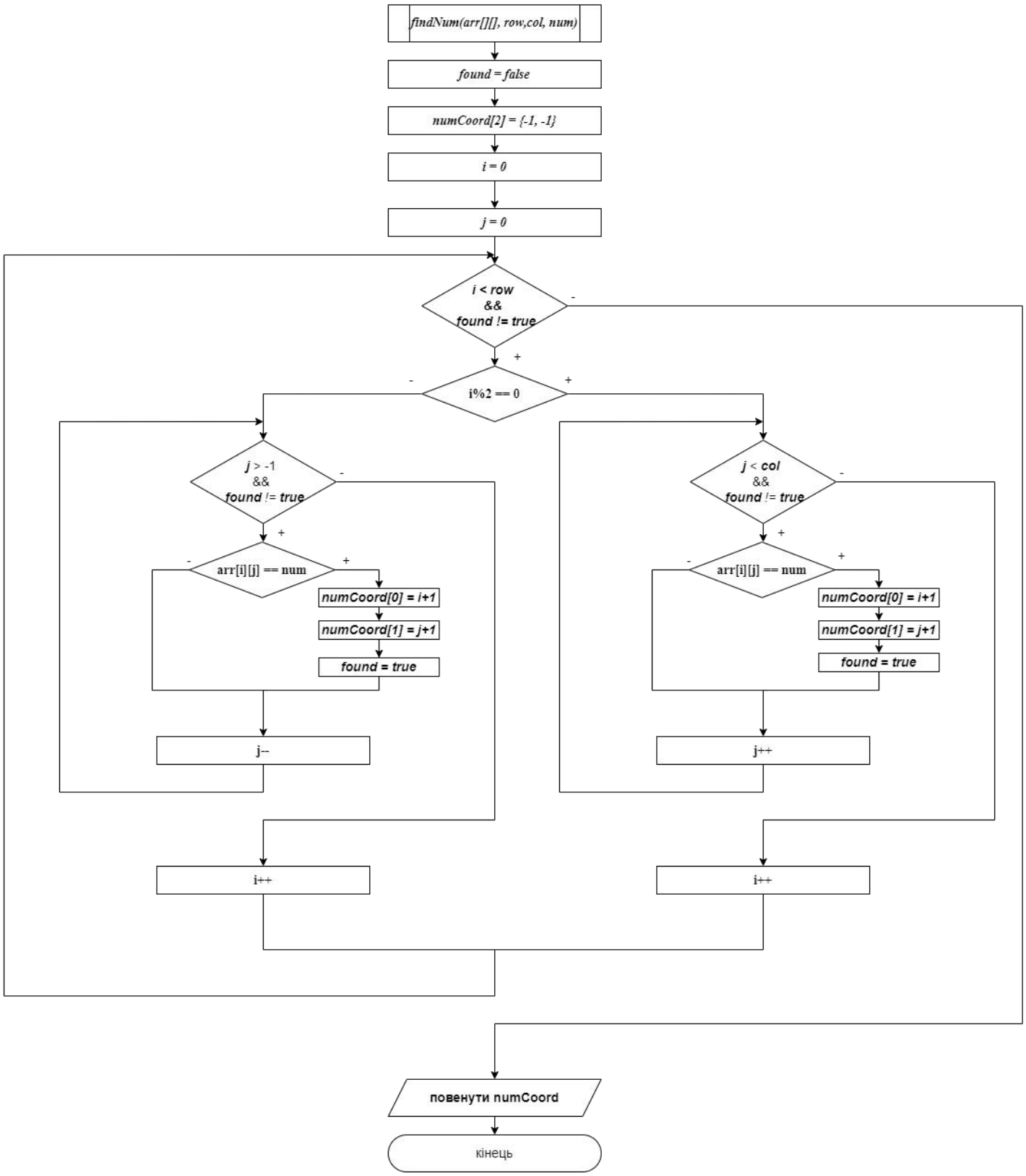
Крок 3


findNum(arr[][], row,col, num)
found = false
numCoord[2] = {-1, -1}
i = 0
j = 0
i < row
&&
found != true
-
+
i%2 == 0
-
+
j > -1
&&
found != true
-
+
перевірка чи поточний елемент масиву дорівнює num
знаходження місцезнаходження у випадку справдження умови
j--
i++
j < col
&&
found != true
-
+
перевірка чи поточний елемент масиву дорівнює num
знаходження місцезнаходження у випадку справдження умови
j++
i++
повенути numCoord
кінець

Крок 4


findNum(arr[][], row,col, num)
found = false
numCoord[2] = {-1, -1}
i = 0
j = 0
i < row
&&
found != true
-
+
i%2 == 0
-
+
j > -1
&&
found != true
-
+
arr[i][j] == num
-
+
знаходження місцезнаходження у випадку справдження умови
j--
i++
j < col
&&
found != true
-
+
arr[i][j] == num
-
+
знаходження місцезнаходження у випадку справдження умови
j++
i++
повенути numCoord
кінець

Крок 5


findNum(arr[][], row,col, num)
found = false
numCoord[2] = {-1, -1}
i = 0
j = 0
i < row
&&
found != true
-
+
i%2 == 0
-
+
j > -1
&&
found != true
-
+
arr[i][j] == num
-
+
numCoord[0] = i+1
numCoord[1] = j+1
found = true
j--
i++
j < col
&&
found != true
-
+
arr[i][j] == num
-
+
numCoord[0] = i+1
numCoord[1] = j+1
found = true
j++
i++
повенути numCoord
кінець

4. Код програми(C++)

```
#include <iostream>
#include <iomanip>
using namespace std;
void fillArr(double** ,int, int);
void displayArr(double**, string, int, int);
int* findNum(double** ,int, int, double);
int countGreaterBelowDiagonal(double** , int, int , double);

int main(){
    int row,col;
    double num;
    int* numCoord;
//
    cout << "Введіть кількість рядків матриці: ";
    cin >> row;
    cout << "Введіть кількість стовпців матриці: ";
    cin >> col;
    cout << "Введіть шукане число: ";
    cin >> num;
//
    double** arr = new double*[row];
//
    for (int i=0; i < row; i++) {
        arr[i] = new double[col]{};
    }
//

    fillArr(arr, row, col);
    arr[4][7] = 38;
    numCoord = findNum(arr, row, col, num );
    int countGreater = countGreaterBelowDiagonal(arr,row,col,num);

    if (numCoord[0]==-1) {
        cout << "Число " << num <<" не було знайдено серед елементів матриці";
    } else {
        cout << "Число " << num <<" було знайдено серед елементів матриці у " << numCoord[0] << "-му рядку, "<<
numCoord[1] << "-му стовпчику"<<endl;
    }
//
    displayArr(arr,"Array: ", row, col);

    cout << "Кількість чисел під основною діагоналлю, що перевищують "<< num << " : " << countGreater;
    delete[] numCoord;
    for (int i=0; i < row; i++) {
        delete[] arr[i];
    }
    delete[] arr;
//
    return 0;
}
void fillArr(double ** arr, int row,int col) {
    srand(time(NULL));
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
                arr[i][j] = rand()%201 -100 + double(rand()%100)/100;

            }
        }
}
void displayArr(double** arr2, string message, int row, int col){
    cout << endl << message << "\n\n";
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
```

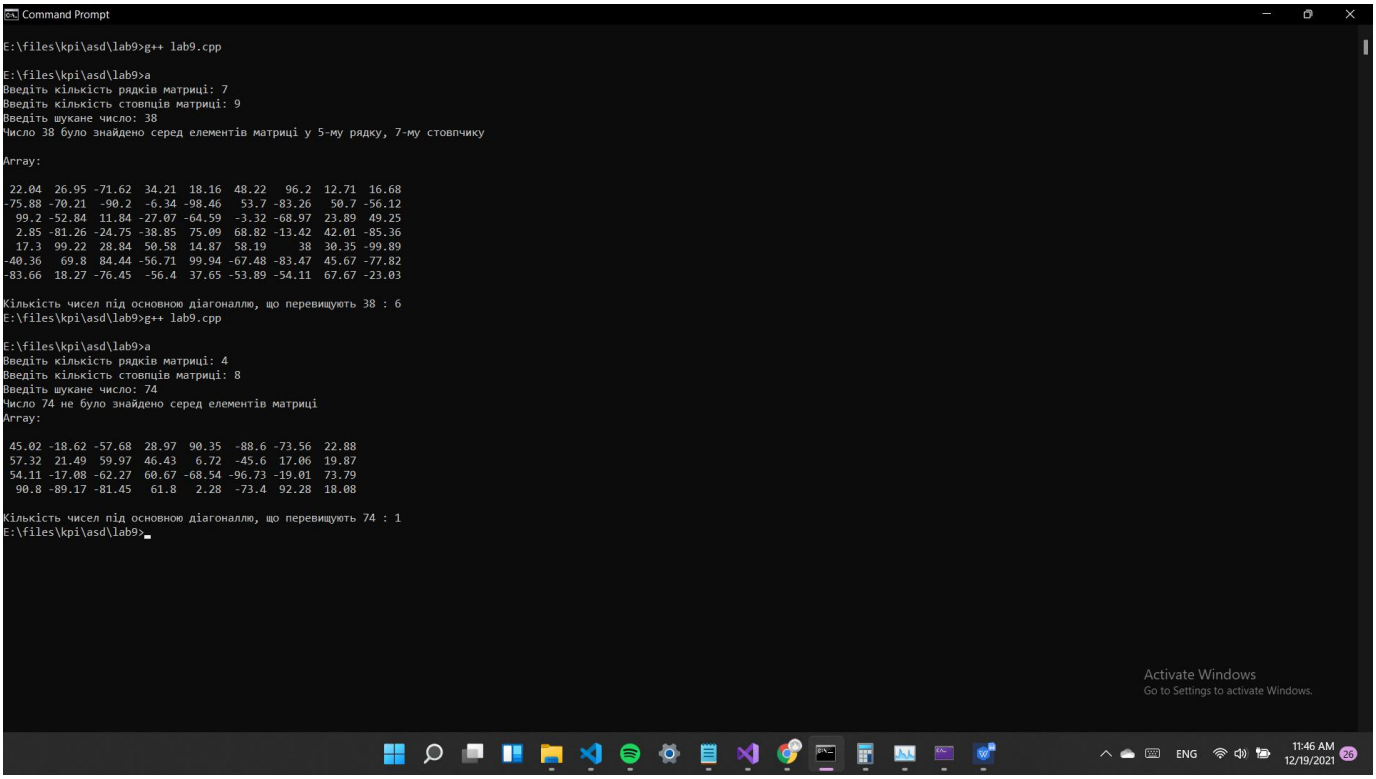
```
            cout <<setw(6)<< arr2[i][j] << " ";
        }
```

```
        cout << "\n";;
    }
    cout << "\n";
}
int* findNum(double** arr,int row, int col , double num) {
    int i = 0 ,j = 0;;
    bool found = false;
    int* numCoord = new int[2] {-1};
    while (i < row &&  found != true) {
        if (i%2==0) {

            while (j < col &&  found != true ) {
                if ( arr[i][j] == num ){
                    numCoord[0] = i+1;
                    numCoord[1] = j+1;
                    found = true;
                }
                j++;
            }
        } else {
            while (j > -1 &&  found != true ) {
                if ( arr[i][j] == num ){
                    numCoord[0] = i+1;
                    numCoord[1] = j+1;
                    found = true;
                }
                j--;
            }
        }

        i++;
    }
    return numCoord;
}
int countGreaterBelowDiagonal(double** arr, int row, int col, double num) {
    int counter = 0;
    for (int i = 0; i < row; i++) {
        int j = 0;
        while (j < i && j < col) {
            if (arr[i][j] > num) {
                counter++;
            }
            j++;
        }
    }
    return counter;
}
```

5. Тестування програми



Висновок - Було досліджено алгоритми обходу масивів, набуто практичних навичок використання цих алгоритмів під час складання програмних специфікацій

Декомпозовано задачу на 3 етапи:

1. Генерація матриці.
2. Знаходження заданого числа серед елементів масиву
3. Визначення кількості елементів під головною діагоналлю, що є більшими за дане число