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 структури даних»

«Дослідження алгоритмів обходу масивів»

Варіант 34

Виконав студент ІП-1134 Шамков Іван Дмитрович
(прізвище, ім'я, по батькові)

Перевірив викладач Мартинова Оксана Петрівна
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота №9

Дослідження алгоритмів обходу масивів

Лабораторна робота 9

Дослідження алгоритмів обходу масивів

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

Варіант: 34

Умова задачі:

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).

34	Задано матрицю дійсних чисел A[m,n]. При обході матриці по рядках знайти в ній останній мінімальний елемент X і його місцезнаходження. Порівняти значення X із середньоарифметичним значенням елементів під побічною діагоналлю.

Математична модель:

Змінна	Тип	Ім’я	Призначення
Кількість рядків	Цілий	n	Початкове дане
Кількість стовпців	Цілий	m	Початкове дане
Двовимірний масив	Дійсний	A	Проміжне значення
Лічильник	Цілий	i	Проміжне значення
Лічильник	Цілий	j	Проміжне значення
Останній мінімальний елемент матриці	Дійсний	min	Результат
Індекс мінімального елементу	Цілий	row	Результат

Індекс мінімального елементу	Цілий	col	Результат
Середнє арифметичне елементів під побічною діагоналлю	Дійсний	avg	Результат

Постановка задачі:

Отже, математичне формулювання нашої задачі полягає в тому, щоб отримати від значення розмірів двовимірного масиву, створити його, який наповнюємо випадковими дійсними числами. Після цього обходимо масив змійкою та шукаємо останній мінімальний елемент масиву. Виводимо його та його розташування. Потім обходимо елементи, що знаходяться під побічною діагоналлю, та шукаємо їхнє середнє арифметичне. Потім порівнюємо значення avg та min.

(rand()%X - B) – генерація випадкового числа від з діапазону

[-B; X-B)

Наступні функції є створеними власноруч:

CreateArray( )

CoutArray( )

SnakeChase( )

DeleteArray( )

AvgDiagonal( )

Check( )

Псевдокод:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо значення A

Крок 3. Виведення A

Крок 4. Пошук min, avg

Крок 5. Порівняння min та avg

Крок 1:

Start

Деталізуємо значення n, m та A

Виведення A

Пошук min, avg

Порівняння min та avg

End

Крок 2:

Start

input n, m

A=CreateArray(n, m)

Виведення A

Пошук min, avg

Порівняння min та avg

End

Крок 3:

Start

input n, m

A=CreateArray(n, m)

CoutArray(A, n, m)

Пошук min, avg

Порівняння min та avg

End

Крок 4:

Start

input n, m

A=CreateArray(n, m)

CoutArray(A, n, m)

min=SnakeChase(A, n, m)

avg = AvgDiagonal(A, n, m)

Порівняння min та avg

End

Крок 5:

Start

input n, m

A=CreateArray(n, m)

CoutArray(A, n, m)

min=SnakeChase(A, n, m)

avg = AvgDiagonal(A, n, m)

Check(min, avg)

End

Підпрограми

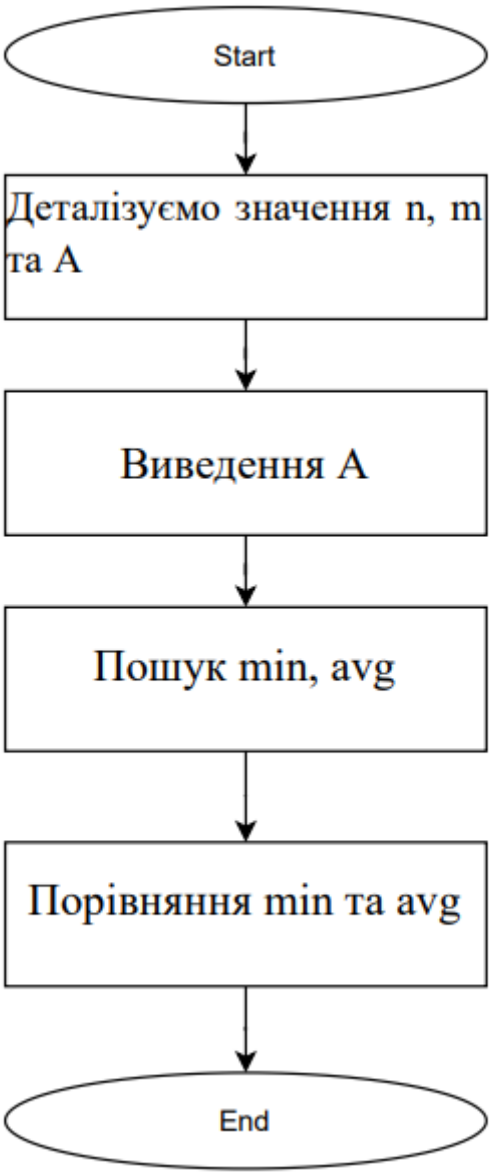
```
CreateArray(n, m)
        for i from 0 to n
           repeat
                for j from 0 to m
                   repeat
                        A[i][j] = (rand()%1000)/10
                end for
        end for
        return A

CoutArray(A, n, m)
        for i from 0 to n
           repeat
                for j from 0 to m
                   repeat
                        output A[i][j]
                end for
        end for
        return A


SnakeChase(A, n, m)
        min=A[0][0]
        row=0
        col=0
        j=0
        for i from 0 to n
```

```
        repeat
            while (j<m and i%2==0)
                repeat
                    if (min>= A[i][j])
                            min=A[i][j]
                            row=i
                            col=j
                    end if
                j=j+1
            end while

            while (j!=0 and i%2!=0)
                repeat
                    j=j-1
                    if (min>= A[i][j])
                            min=A[i][j]
                            row=i
                            col=j
                    end if
            end while
        return min
        end for

AvgDiagonal(A, n, m)
        if (n != m or n==1)
                output "The matrix is not square or it's size is [1x1]"
        else
                for i from 0 to n
```

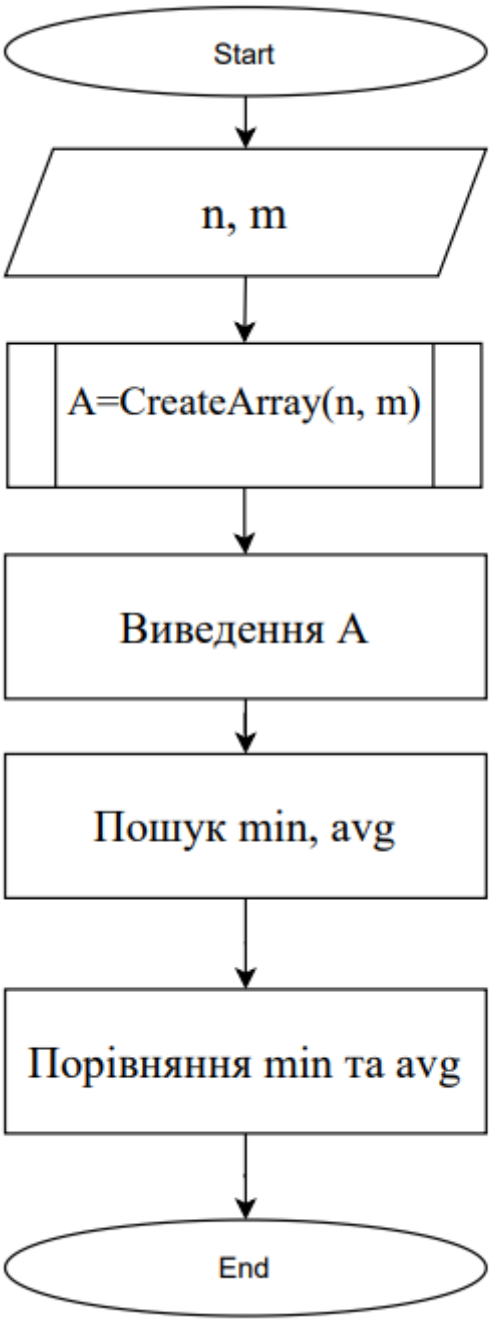
```
            repeat
                avg += A[i][m-i]
        end for
    end if
    return avg/(m-1)


Check(min, avg)
    if (min > avg)
        output "Minimal element is bigger than average"

    else if (avg > min)
        output "Average element is bigger than minimal"

    else
        output "Average and minimal element are equal"
    end if
```

Блок схема:

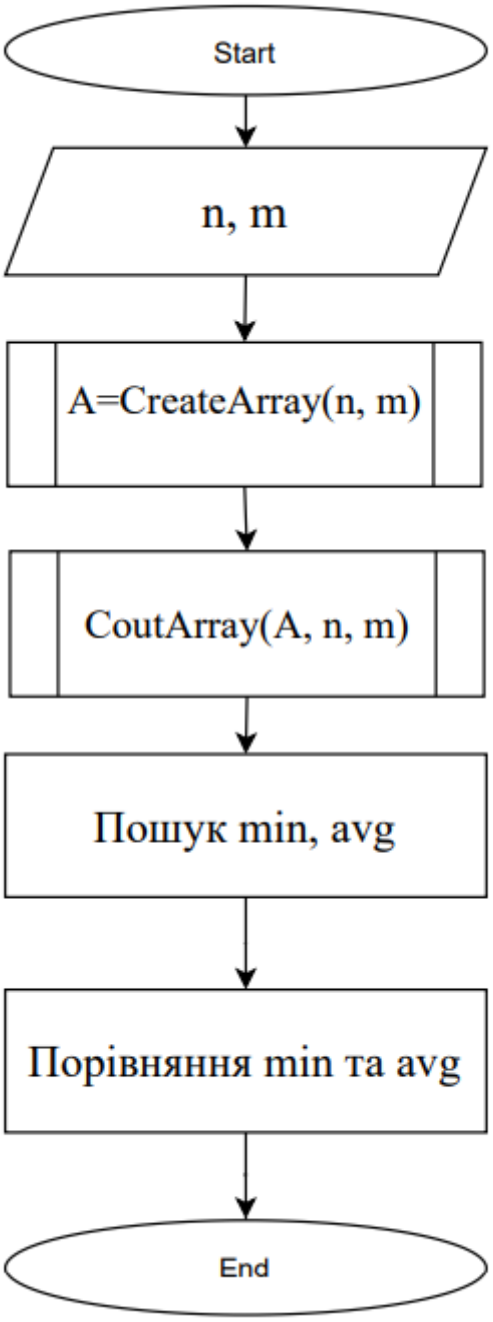
Крок 1



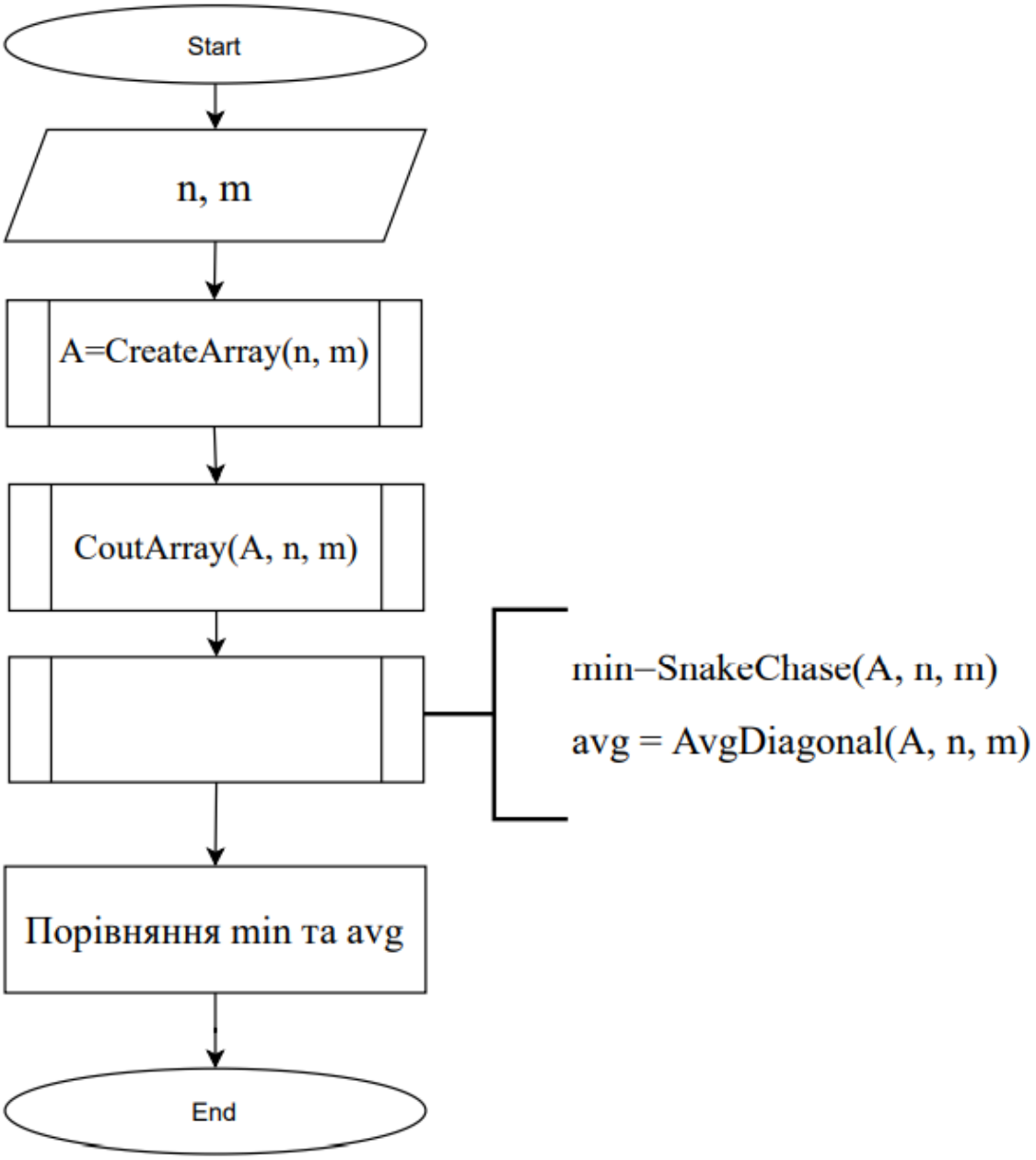
Крок 2


Start
n, m
A=CreateArray(n, m)
Виведення A
Пошук min, avg
Порівняння min та avg
End

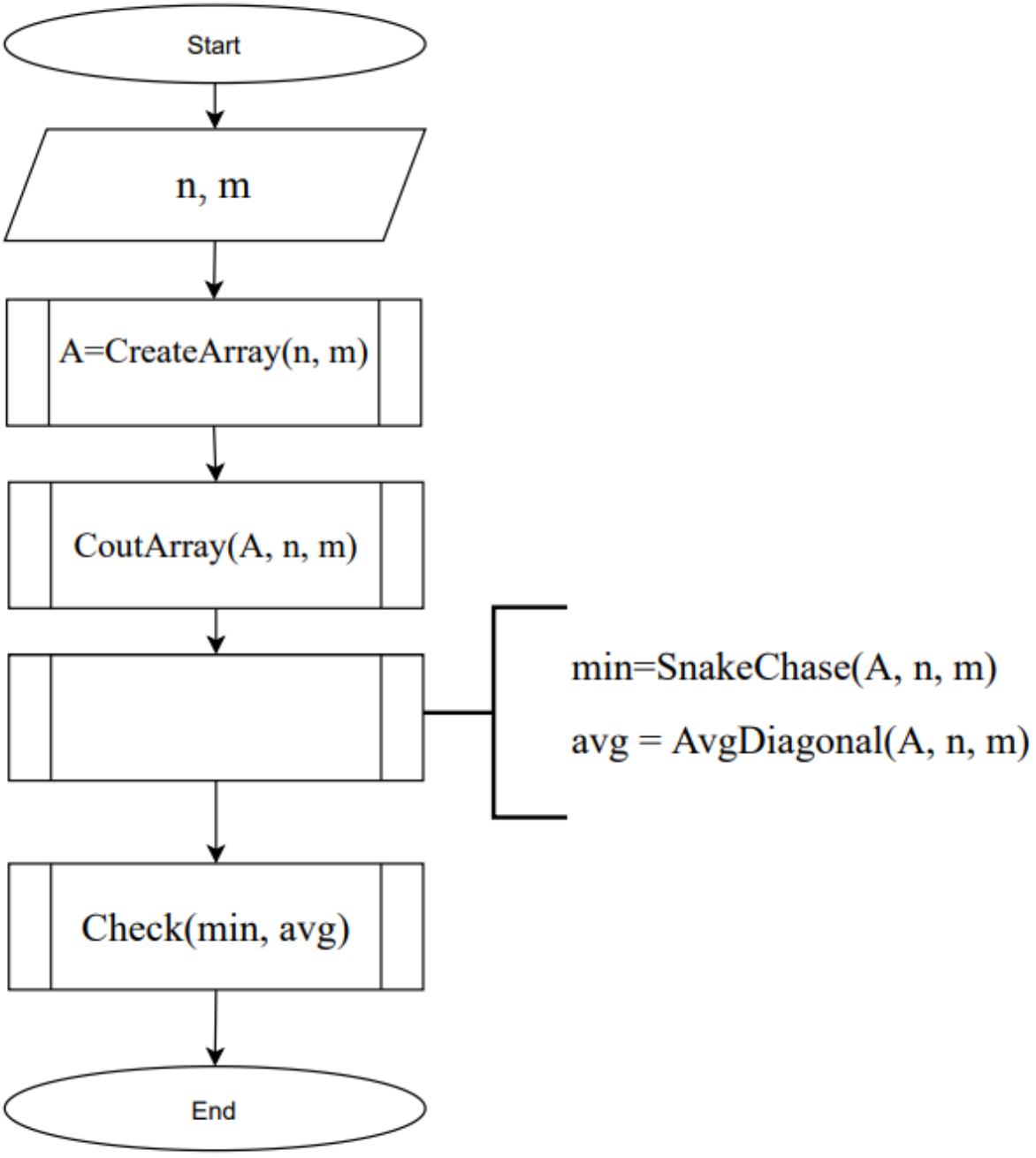
Крок 3


Start
n, m
A=CreateArray(n, m)
CoutArray(A, n, m)
Пошук min, avg
Порівняння min та avg
End

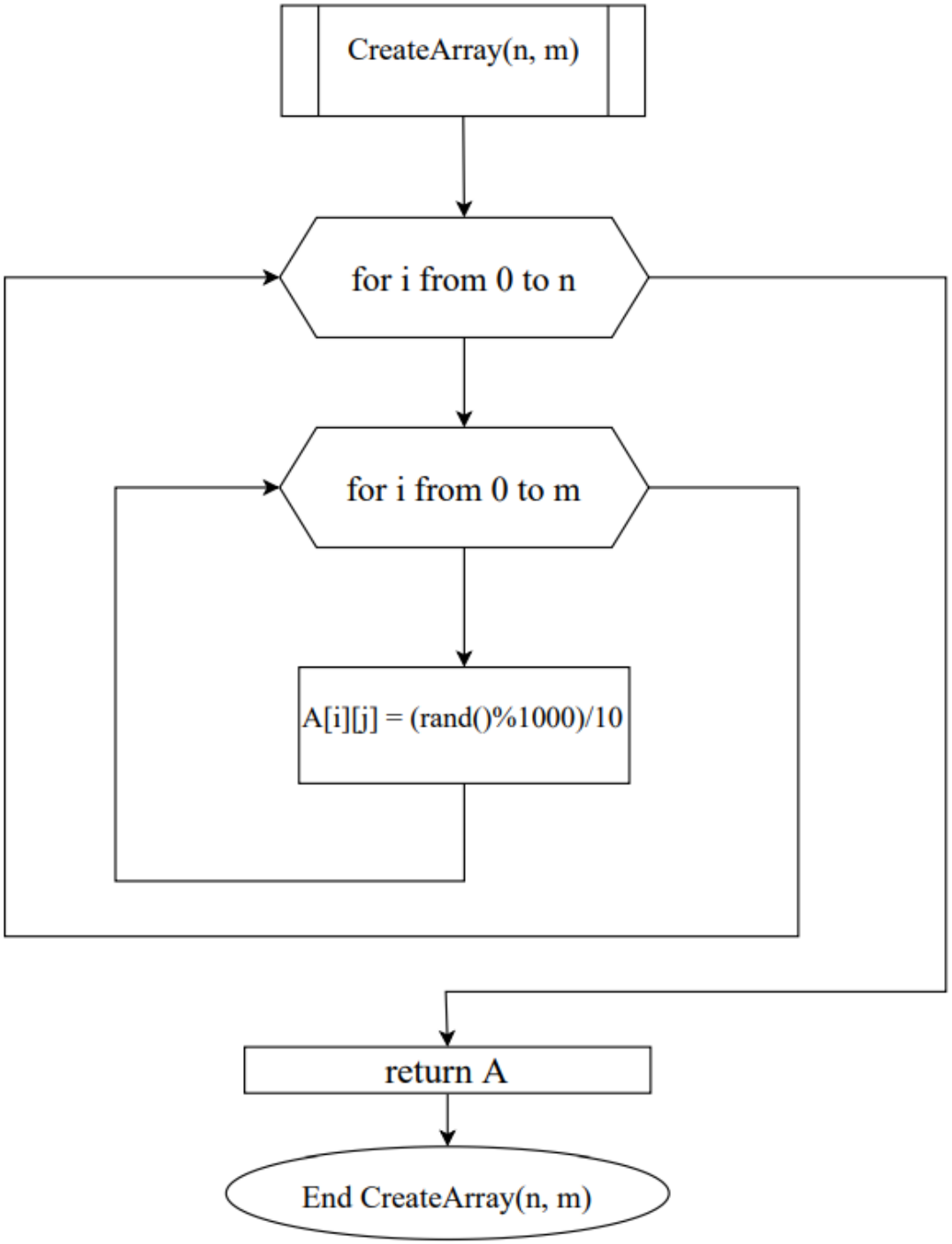
Крок 4

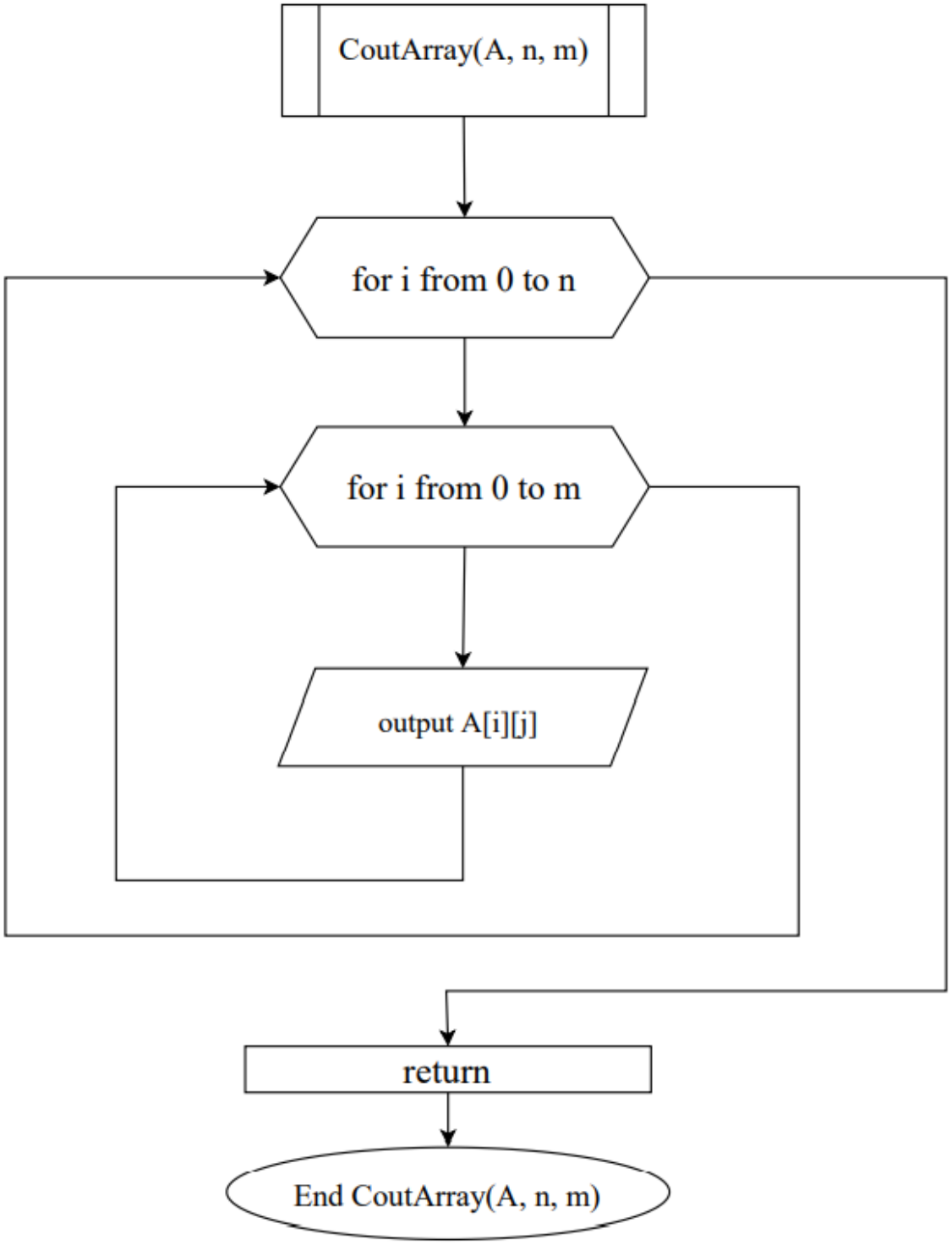

Start
n, m
A=CreateArray(n, m)
CoutArray(A, n, m)
min=SnakeChase(A, n, m)
avg = AvgDiagonal(A, n, m)
Порівняння min та avg
End

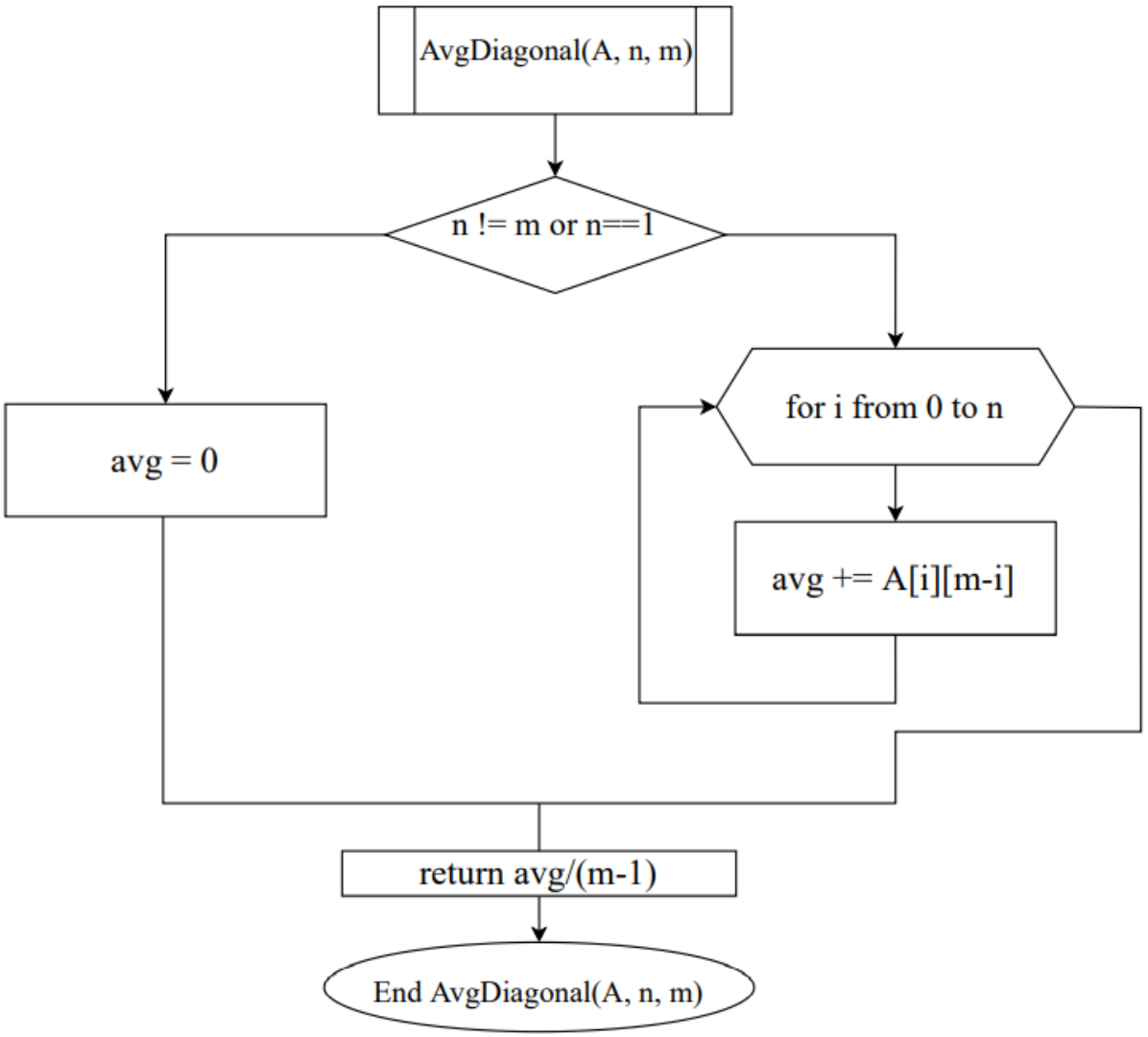
Крок 5

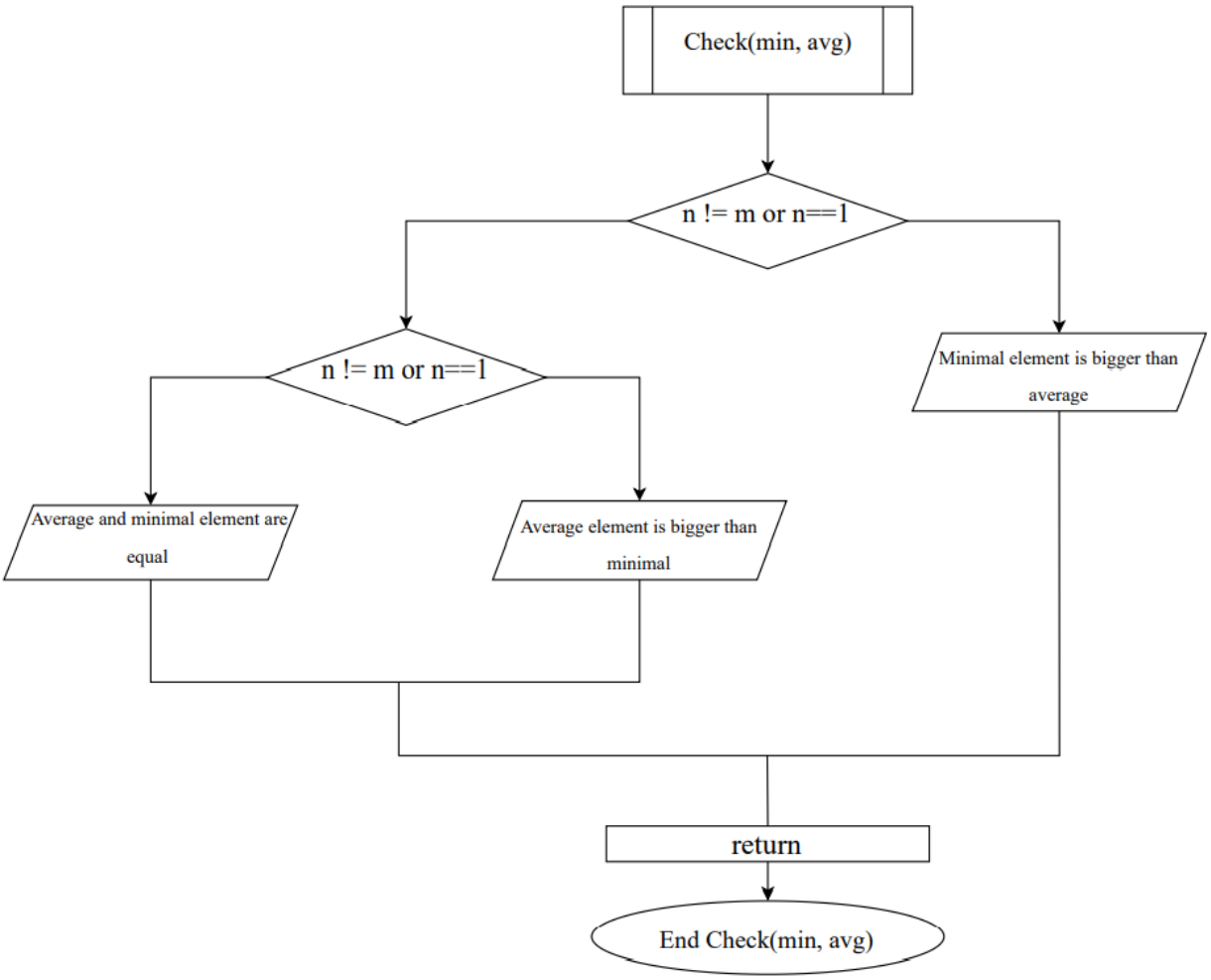

Start
n, m
A=CreateArray(n, m)
CoutArray(A, n, m)
min=SnakeChase(A, n, m)
avg = AvgDiagonal(A, n, m)
Check(min, avg)
End

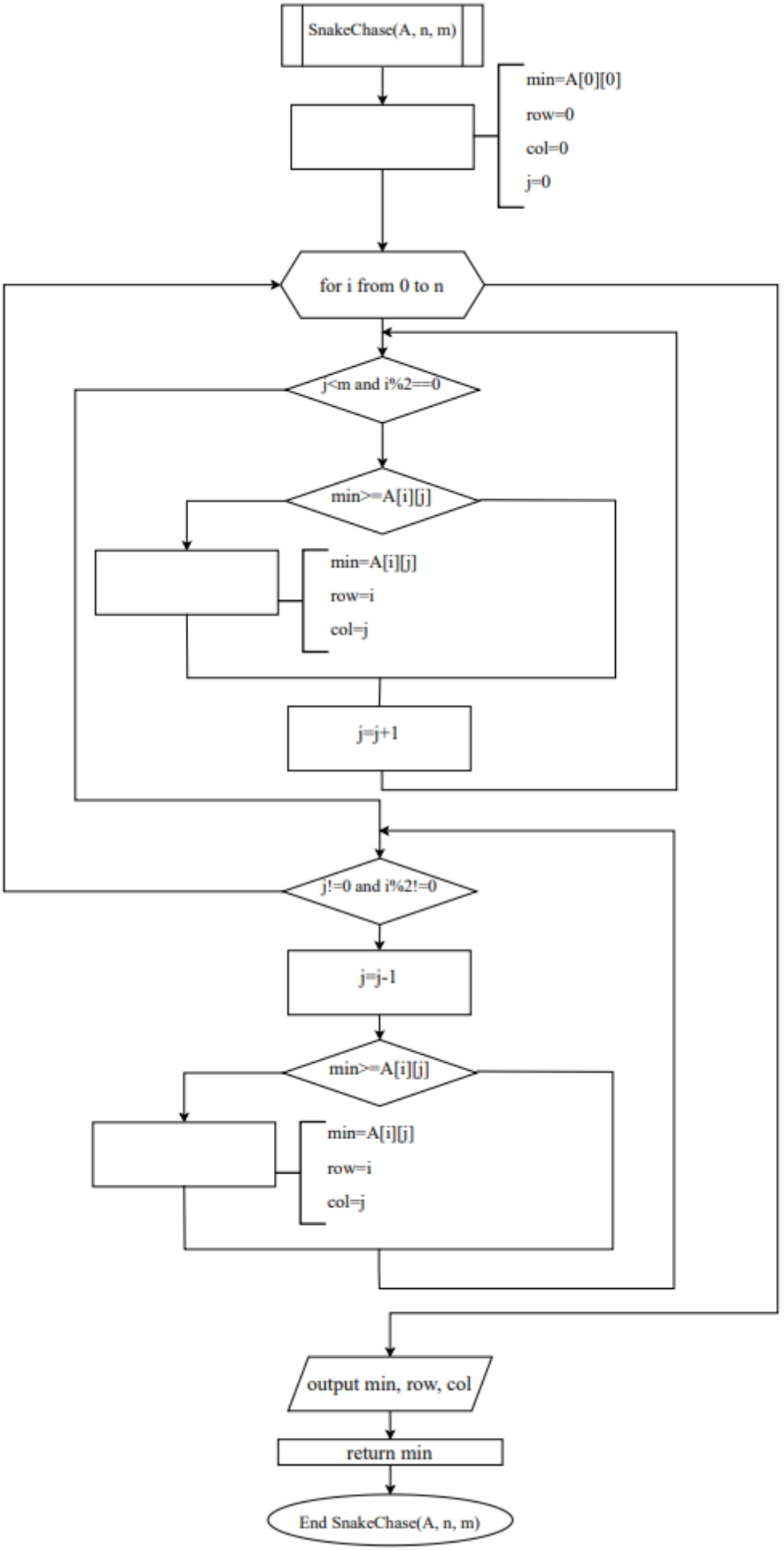
Підпрограми




CoutArray(A, n, m)
for i from 0 to n
for i from 0 to m
output A[i][j]
return
End CoutArray(A, n, m)


AvgDiagonal(A, n, m)
n != m or n==1
avg = 0
for i from 0 to n
avg += A[i][m-i]
return avg/(m-1)
End AvgDiagonal(A, n, m)


Check(min, avg)
n != m or n==1
n != m or n==1
Minimal element is bigger than average
Average and minimal element are equal
Average element is bigger than minimal
return
End Check(min, avg)


SnakeChase(A, n, m)
min=A[0][0]
row=0
col=0
j=0
for i from 0 to n
j<m and i%2==0
min>=A[i][j]
min=A[i][j]
row=i
col=j
j=j+1
j!=0 and i%2!=0
j=j-1
min>=A[i][j]
min=A[i][j]
row=i
col=j
output min, row, col
return min
End SnakeChase(A, n, m)

Код на C++:

```
#include <iostream>
#include <ctime>
#include <iomanip>
using namespace std;

float** CreateArray(int&, int&);
void CoutArray(float** ,int, int);
float SnakeChase(float**, int, int);
void DeleteArray(float**, int, int);
float AvgDiagonal(float**, int, int);
void Check(float, float);

int main(){

    int n, m;
    srand(time(NULL));
    float** A;
    A=CreateArray(n, m);
    CoutArray(A, n, m);
    float min=SnakeChase(A, n, m);
    float avg = AvgDiagonal(A, n, m);
    Check(min, avg);
```

```
        DeleteArray(A, n, m);
}

float** CreateArray(int &n, int &m) {

        cout << "Enter number of rows: ";
        cin >> n;
        float** A = new float* [n];

        cout << "\nEnter number of columns: ";
        cin >> m;

        for (int i=0; i < n; i++) {
                *(A+i) = new float [m];
                for (int j=0; j < m; j++) {
                        A[i][j] = double(rand()%1000)/10.;
                }
        }
        return A;
}

void DeleteArray(float** A, int n, int m) {
        for (int i = 0; i < n; i++) {
                delete A[i];
        }
        delete[] A;
```

```
}

void CoutArray(float** A, int n, int m) {
        cout << "\t\tTwo dimensional array [" << n << "x" << m <<
"]\n\n";
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < m; j++) {
                        cout << setw(8) << A[i][j];
                }
                cout << "\n\n";
        }
}

float SnakeChase(float** A, int n, int m) {
        float min=A[0][0], row=0, col=0;
        for (int i = 0, j=0; i < n; i++) {

                while (j < m && i%2==0) {

                        if (min >= A[i][j]) {

                                min = A[i][j];
                                row = i;
                                col = j;
                        }
                        ++j;
```

```
            }
            while (j!=0 && i%2!=0) {
                --j;
                if (min >= A[i][j]) {

                    min = A[i][j];
                    row = i;
                    col = j;
                }

            }

        }
        cout << "Last minimum element of the matrix is " << min << "
and it's id is [" << row << "][" << col << "]" << endl;
        return min;
}

float AvgDiagonal(float** A, int n, int m) {
        if (n != m || n==1) {
            cout << "The matrix is not square or it's size is [1x1]" <<
endl;
            return 0;
        }
        else {
            float avg=0;
```

```
            for (int i = 1; i < n; i++) {
                avg += A[i][m-i];
            }
            cout << "Average Diagonal number: " << avg / (m - 1) <<
endl;
            return avg/(m-1);
        }
}

void Check(float min, float avg) {
        cout << "\n\n";
        if (min > avg) {
            cout << "Minimal element is bigger than average" << endl;
        }
        else if (avg > min) {
            cout << "Average element is bigger than minimal" << endl;
        }

        else {
            cout << "Average and minimal element are equal" << endl;
        }
}
```

Копії екранних форм:

```
#include <iostream>
#include <ctime>
#include <iomanip>
using namespace std;

float** CreateArray(int&, int&);
void CoutArray(float** ,int, int);
float SnakeChase(float**, int, int);
void DeleteArray(float**, int, int);
float AvgDiagonal(float**, int, int);
void Check(float, float);

int main(){

    int n, m;
    srand(time(NULL));
    float** A;
    A=CreateArray(n, m);
    CoutArray(A, n, m);
    float min=SnakeChase(A, n, m);
    float avg = AvgDiagonal(A, n, m);
    Check(min, avg);

    DeleteArray(A, n, m);
}

```

```
26
27  float** CreateArray(int &n, int &m) {
28
29      cout << "Enter number of rows: ";
30      cin >> n;
31      float** A = new float* [n];
32
33      cout << "\nEnter number of columns: ";
34      cin >> m;
35
36      for (int i=0; i < n; i++) {
37          *(A+i) = new float [m];
38          for (int j=0; j < m; j++) {
39              A[i][j] = double(rand()%1000)/10.;
40          }
41      }
42      return A;
43  }
44
```

```
44
45  void DeleteArray(float** A, int n, int m) {
46      for (int i = 0; i < n; i++) {
47          delete A[i];
48      }
49      delete[] A;
50  }
51
52  void CoutArray(float** A, int n, int m) {
53      cout << "\t\tTwo dimensional array [" << n << "x" << m << "]\n\n";
54      for (int i = 0; i < n; i++) {
55          for (int j = 0; j < m; j++) {
56              cout << setw(8) << A[i][j];
57          }
58          cout << "\n\n";
59      }
60  }
61
```

```
61
62  float SnakeChase(float** A, int n, int m) {
63      float min=A[0][0], row=0, col=0;
64      for (int i = 0, j=0; i < n; i++) {
65
66          while (j < m && i%2==0) {
67
68              if (min >= A[i][j]) {
69
70                  min = A[i][j];
71                  row = i;
72                  col = j;
73              }
74              ++j;
75          }
76          while (j!=0 && i%2!=0) {
77              --j;
78              if (min >= A[i][j]) {
79
80                  min = A[i][j];
81                  row = i;
82                  col = j;
83              }
84
85          }
86
87      }
88      cout << "Last minimum element of the matrix is " << min << " and it's id is [" << row << "][" << col << "]" << endl;
89      return min;
90  }
```

```
91
92  float AvgDiagonal(float** A, int n, int m) {
93      if (n != m || n==1) {
94          cout << "The matrix is not square or it's size is [1x1]" << endl;
95          return 0;
96      }
97      else {
98          float avg=0;
99          for (int i = 1; i < n; i++) {
100             avg += A[i][m-i];
101         }
102         cout << "Average Diagonal number: " << avg / (m - 1) << endl;
103         return avg/(m-1);
104     }
105 }
106
107 void Check(float min, float avg) {
108     cout << "\n\n";
109     if (min > avg) {
110         cout << "Minimal element is bigger than average" << endl;
111     }
112     else if (avg > min) {
113         cout << "Average element is bigger than minimal" << endl;
114     }
115
116     else {
117         cout << "Average and minimal element are equal" << endl;
118     }
119 }
120
```

```
Enter number of rows: 8

Enter number of columns: 3
                Two dimensional array [8x3]

     7.9    55.5     6.2

    94.8    13.4      77

    24.5    29.7    59.1

    15.2     0.8    54.6

       8    55.4    79.8

    55.2    85.1    96.6

    72.3      91    38.4

      48    45.2    93.7

Last minimal element of the matrix is 0.8 and it's id is [3][1]
The matrix is not square or it's size is [1x1]


Minimal element is bigger than average
```

```
Enter number of rows: 4

Enter number of columns: 4
                Two dimensional array [4x4]

    95.5     9.9    35.1     4.6

    39.6    39.4    27.6    46.2

    74.3    40.2    62.9    87.2

    77.1    15.4    46.4     6.5

Last minimal element of the matrix is 4.6 and it's id is [0][3]
Average Diagonal number: 41.5


Average element is bigger than minimal
```

```
Enter number of rows: 5

Enter number of columns: 5
                Two dimensional array [5x5]

    89.3    48.8    61.2    59.5     2.8

    32.2    67.6    31.2    70.3    14.3

    43.9     3.5    11.7    68.6    79.7

    32.1    32.1    30.4    67.6    49.1

       6    80.2    34.3    32.1    79.3

Last minimal element of the matrix is 2.8 and it's id is [0][4]
Average Diagonal number: 48.375


Average element is bigger than minimal
```

Випробування алгоритму

Проведемо випробування на прикладі другого результату

Блок	Дія
	Початок
1	Пошук останнього мінімального елементу: A[0][4]=2.8
2	Розрахунок середнього арифметичного елементів під побічною діагоналлю: (80.2+30.4+68.6+14.3)/4=48.375
3	Перевірка розмірів: 48.375>2.8 Середнє значення більше за мінімальне
	Кінець

Висновок

Отже, виконавши цю лабораторну роботу, ми навчилися обходити двовимірний масив змійкою. Ідея цього обходу полягає в тому, щоб при закінченні обходу рядка з парним числом(рахуємо від нуля) переходити не в початок наступного рядка, а в кінець, і рухатися з кінця до його початку. Відповідно цей рядок буде мати непарний номер, при завершенні обходу треба перейти на початок наступного парного рядка, якщо такий існує. Цей алгоритм реалізуємо через арифметичний цикл, у якому знаходиться два ітераційні цикли: один для руху від початку рядка до його кінця, а інший для зворотнього руху. У кожному циклі відбувається зміна лічильника. При цьому варто зауважити, що для першого ітераційного циклу змінюємо лічильник після виконання умову пошуку мінімального елементу, а для другого перед. Усе для того, щоб не вилазити за межі масиву. Після цього проходимо по елементах під побічною діагоналлю та рахуємо середнє арифметичне цих чисел. Відбувається це лише у випадку, коли матриця квадратна. В іншій ситуації записуємо середнє значення значення як нуль. У процесі виконання ми сформулювали задачу, побудували математичну модель та псевдокод алгоритму, що допомогло нам краще її зрозуміти