

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 32

Виконав студент ІП-14, Шляхтун Денис Михайлович
(шифр, прізвище, ім'я, по батькові)

Перевірів доц. кафедри ІІІ Мартинова Оксана Петрівна
(прізвище, ім'я, по батькові)

Лабораторна робота 9

Дослідження алгоритмів обходу масивів

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

Задача: Задано матрицю дійсних чисел $A[m,n]$. При обході матриці змійкою по стовбцям знайти в ній перший додатний елемент X . Порівняти X з середньоарифметичним значенням елементів над побічною діагоналлю

Постановка задачі. Результатом розв'язку є порівняння елементу X з середньоарифметичним значенням елементів над побічною діагоналлю матриці. X – перше додатне число, що знаходиться обходом матриці змійкою по стовбцям. Початкові дані – матриця, що генерується випадковим чином, та розмірність матриці, що вводиться користувачем.

Побудова математичної моделі. Складемо таблицю імен змінних

<i>Змінна</i>	<i>Тип</i>	<i>Ім'я</i>	<i>Призначення</i>
<i>Основні змінні</i>			
Кількість рядків	Цілий	m	Початкове дане
Кількість стовбців	Цілий	n	Початкове дане
Двовимірний масив	Дійсний	arr[m][n]	Початкове дане
Середнє значення	Дійсний	av	Результат
Перший додатний елемент	Дійсний	X	Результат
<i>Змінні, що використовуються у підпрограмах</i>			
Лічильники	Цілий	i, k	Проміжне значення
Сума елементів над діагоналлю	Дійсний	sum	Проміжне значення
К-сть елементів над діагоналлю	Цілий	num	Проміжне значення

Враховуючи те, що для виконання алгоритму потрібна діагональ матриці, сама матриця повинна бути квадратною, отже m і n співпадають.

Для обходу матриці змійкою будемо знаходити остачу від ділення лічильника на 2, з парним значенням прохід відбувається вниз по стовпчику, з непарним значенням – вгору.

Для виконання алгоритму були використані наступні функції зі стандартних бібліотек:

- `srand(time(NULL))` та `rand()` – генерація випадкових чисел для масиву

Програмні специфікації запишемо у псевдокодi та графічній формi у вигляді блок-схеми.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію генерації масиву.

Крок 3. Деталізуємо дію визначення середнього значення елементів над побічною діагоналлю.

Крок 4. Деталізуємо дію визначення першого додатного елемента обходом зміійкою по стовбцям.

Крок 5. Деталізуємо дію порівняння додатного елемента та середнього значення.

Псевдокод.

Крок 1.

початок

генерація першого масиву

знаходження середнього значення елементів

знаходження першого додатного числа

порівняння середнього значення та першого додатного числа

кінець

Крок 2.

початок

`arr = arrayRand(arr, m, n)`

знаходження середнього значення елементів

знаходження першого додатного числа

порівняння середнього значення та першого додатного числа

кінець

Крок 3.

початок

arr = arrayRand(arr, m, n)

av = average(arr, m, n)

знаходження першого додатного числа

порівняння середнього значення та першого додатного числа

кінець

Крок 4.

початок

arr = arrayRand(arr, m, n)

av = average(arr, m, n)

X = findX(arr, m, n)

порівняння середнього значення та першого додатного числа

кінець

Крок 5.

початок

arr = arrayRand(arr, m, n)

av = average(arr, m, n)

X = findX(arr, m, n)

check(X, av)

кінець

функція arrayRand(arr[[]], m, n)

повторити для i = 0; i < m; i++

повторити для k = 0; k < n; k++

arr[i][k] = (rand() % 199 - 99) / 10

все повторити

все повторити

повернути arr

кінець функції

функція average(arr[[]], m, n)

sum = 0

num = 0

повторити для i = 0; i < m - 1; i++

повторити для k = 0; k < n - i - 1; k++

sum += arr[i][k]

num++

все повторити

все повторити

повернути sum / num

кінець функції

функція findX(arr[[]], m, n)

повторити для i = 0; i < n; i++

якщо i % 2 = 0

повторити для k = 0; k < m; k++

якщо arr[k][i] > 0

повернути arr[k][i]

все якщо

все повторити

інакше

повторити для k = m - 1; k > -1; k--

якщо arr[k][i] > 0

повернути arr[k][i]

все якщо

все повторити

все повторити

кінець функції

функція `check(X, av)`

якщо $X > av$

виведення « $X > av$ »

інакше якщо $X = av$

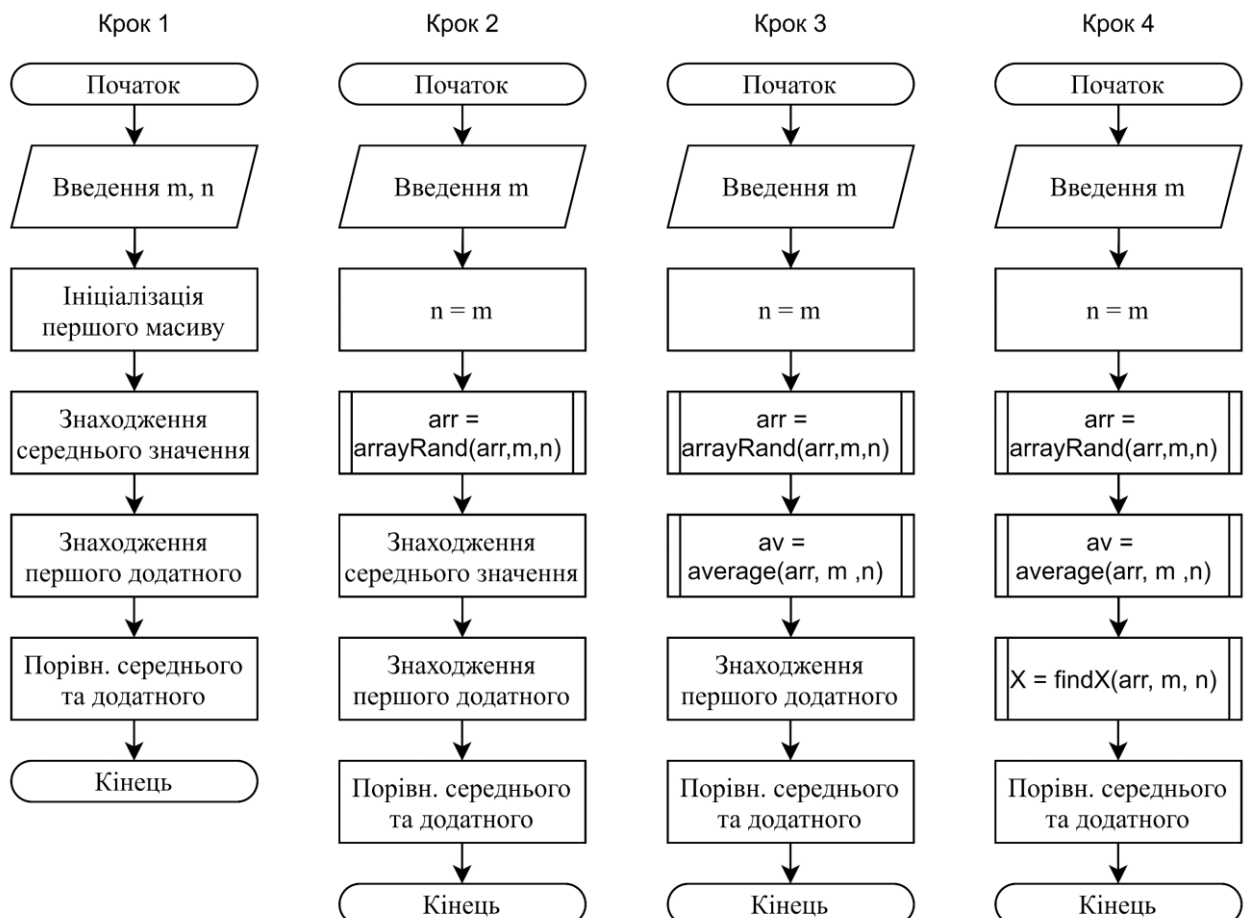
виведення « $X = av$ »

інакше

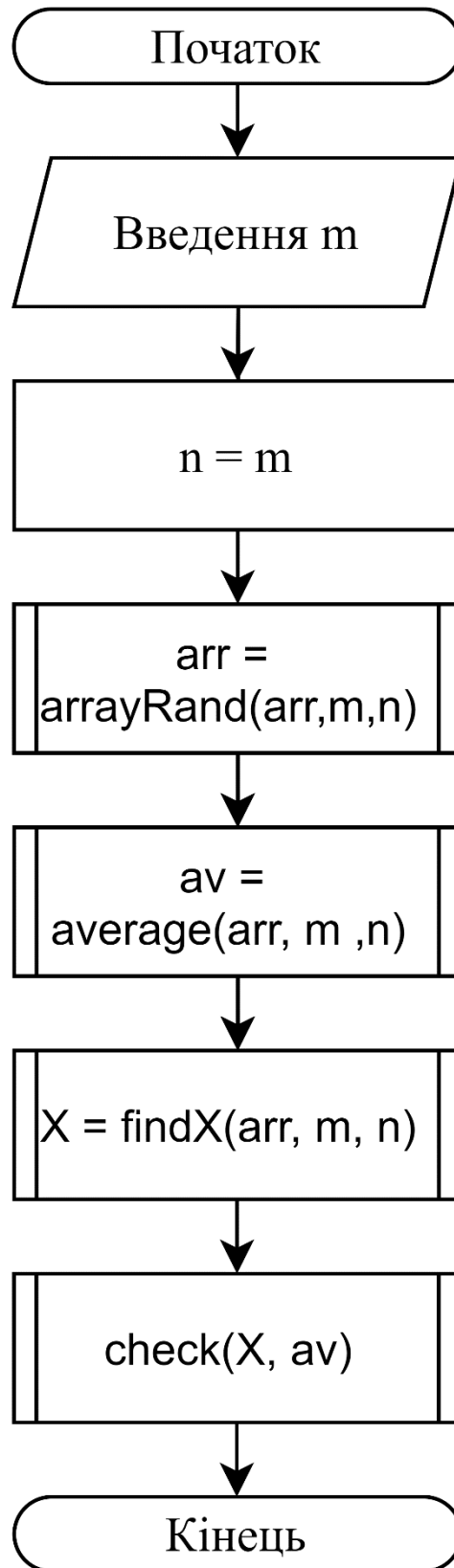
виведення « $X < av$ »

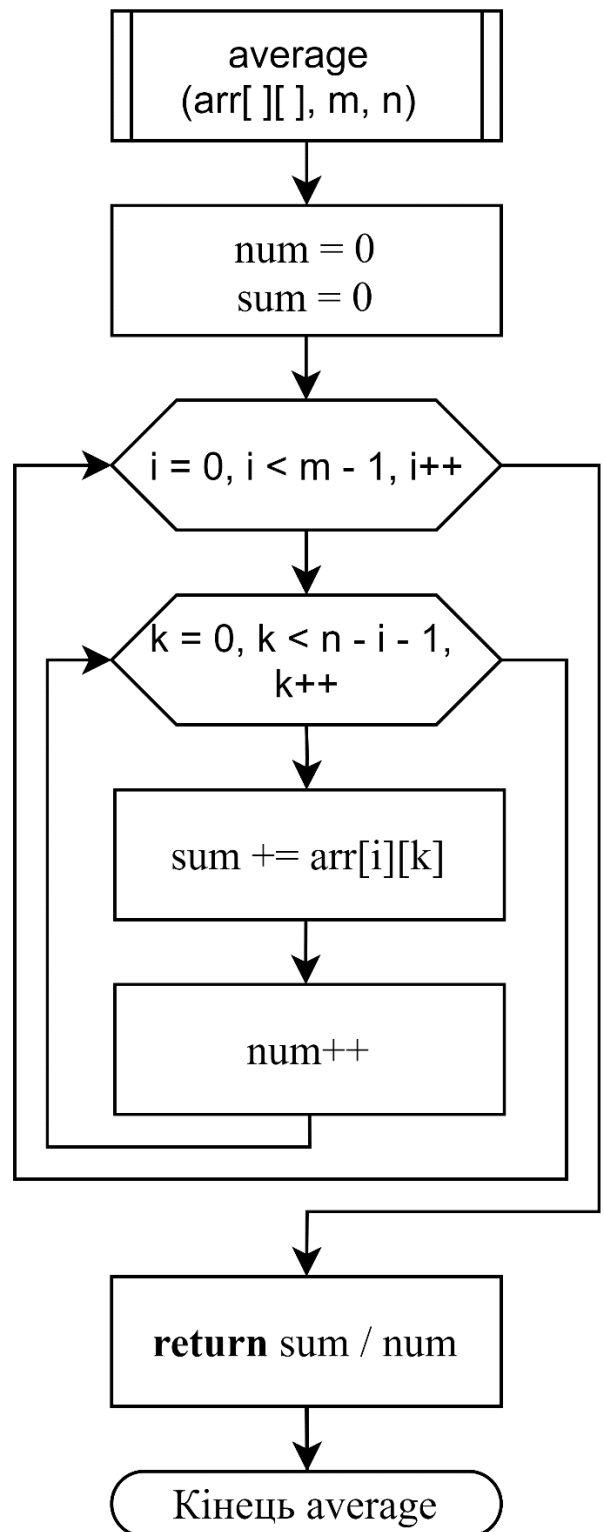
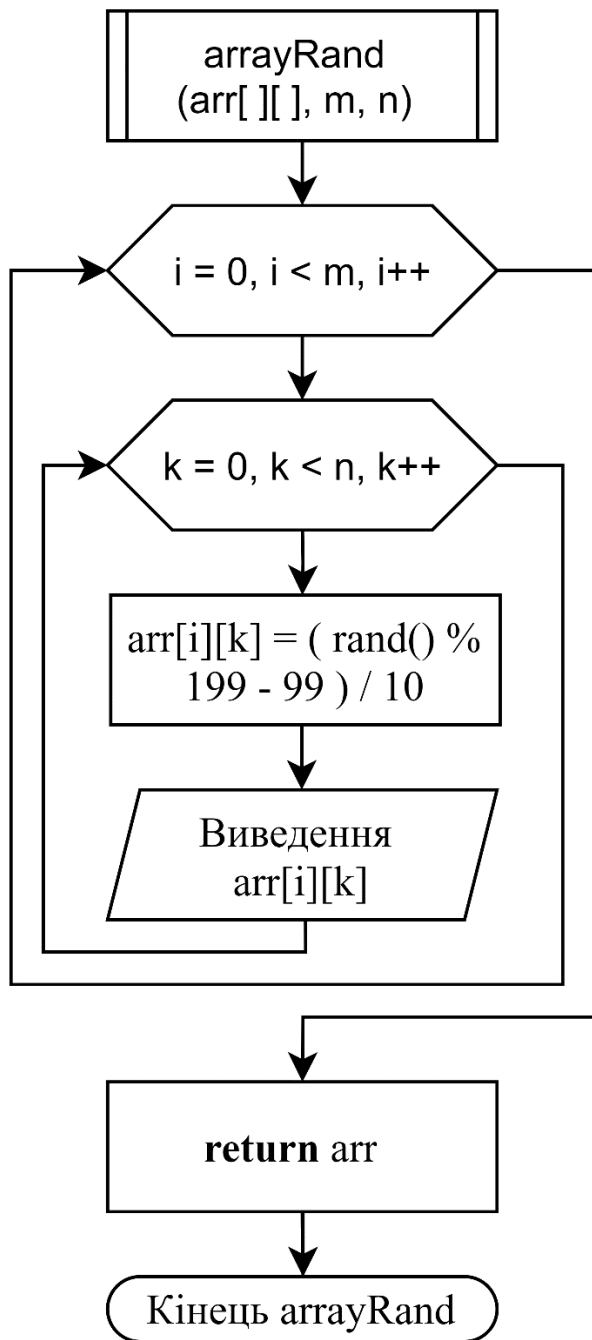
кінець функції

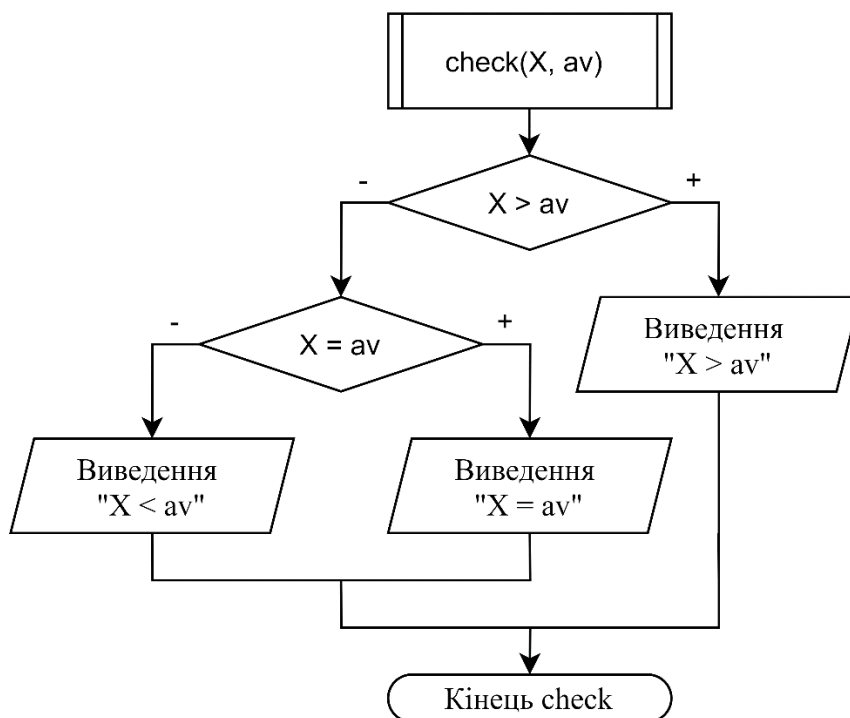
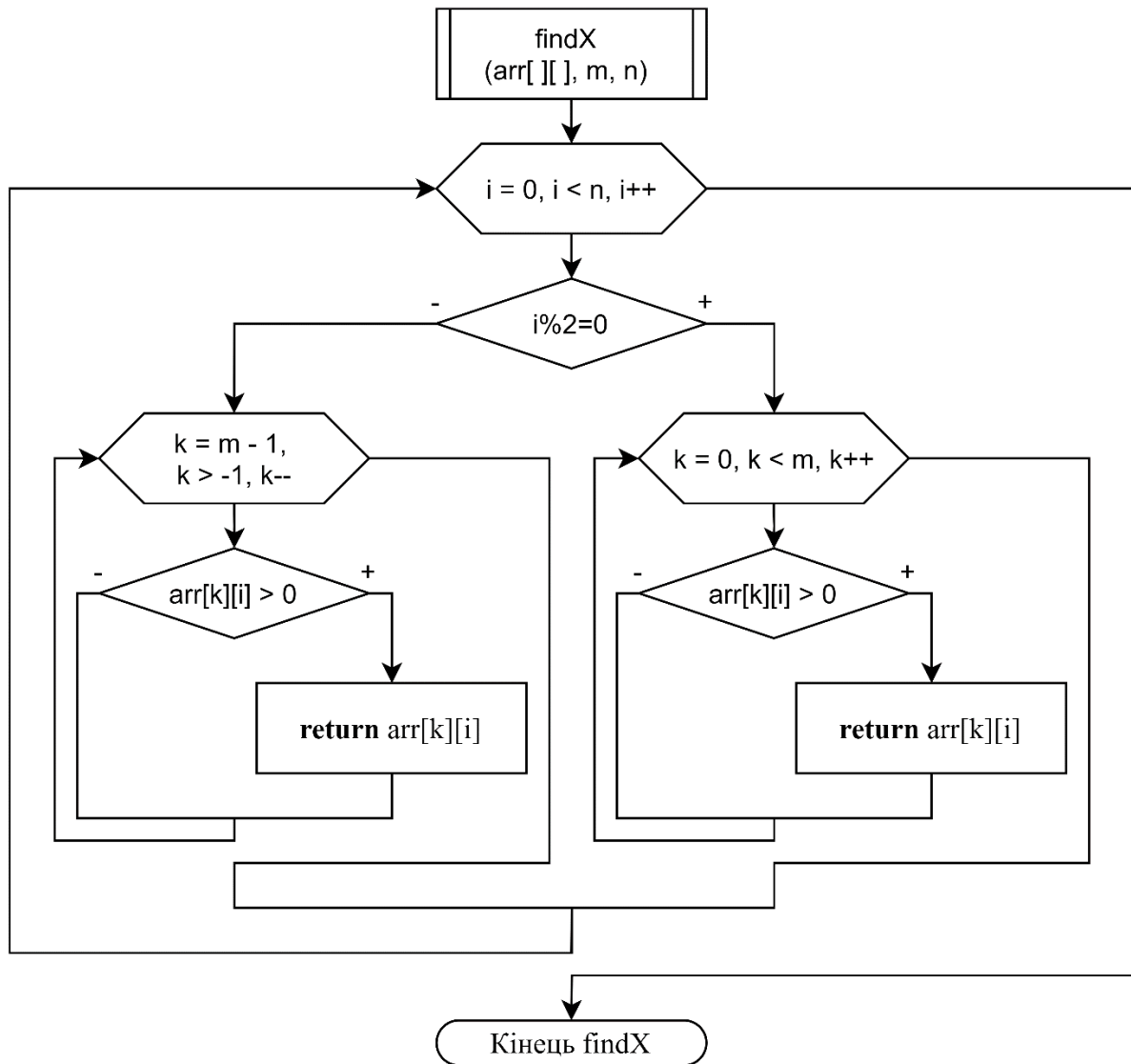
Блок-схема алгоритму



Крок 5







Код програми (C++).

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  //основні функції
5  void arrayRand(float**, int, int);
6  float average(float**, int, int);
7  float findX(float**, int, int);
8  void check(float, float);
9  //допоміжні функції
10 int inputSize();
11 float** arrayDecl(int, int);
12 void arrayDelete(float**, int, int);
13
14 int main()
15 {
16     //введення розмірності масиву
17     int m = inputSize();
18     int n = m;
19     //генерація масиву
20     float** arr = arrayDecl(m, n);
21     arrayRand(arr, m, n);
22     //визначення середнього значення
23     float av = average(arr, m, n);
24     cout << "Average: " << av << endl;
25     //визначення першого додатнього елементу
26     float X = findX(arr, m, n);
27     cout << "X: " << X << endl;
28     //порівняння X та av
29     check(X, av);
30     //допоміжні операції
31     arrayDelete(arr, m, n);
32     system("pause");
33 }
```

```

35 void arrayRand(float** arr, int m, int n)
36 {
37     srand(time(NULL));
38     cout << "Generated array:" << endl;
39     for (int i = 0; i < m; i++)
40     {
41         for (int k = 0; k < n; k++)
42         {
43             arr[i][k] = float(rand() % 199 - 99) / 10;
44             cout << setw(5) << arr[i][k];
45         }
46         cout << endl;
47     }
48 }
49
50 float average(float** arr, int m, int n)
51 {
52     int num = 0;
53     float sum = 0;
54     for (int i = 0; i < m - 1; i++)
55     {
56         for (int k = 0; k < n - i - 1; k++)
57         {
58             sum += arr[i][k];
59             num++;
60         }
61     }
62     return sum / num;
63 }

```

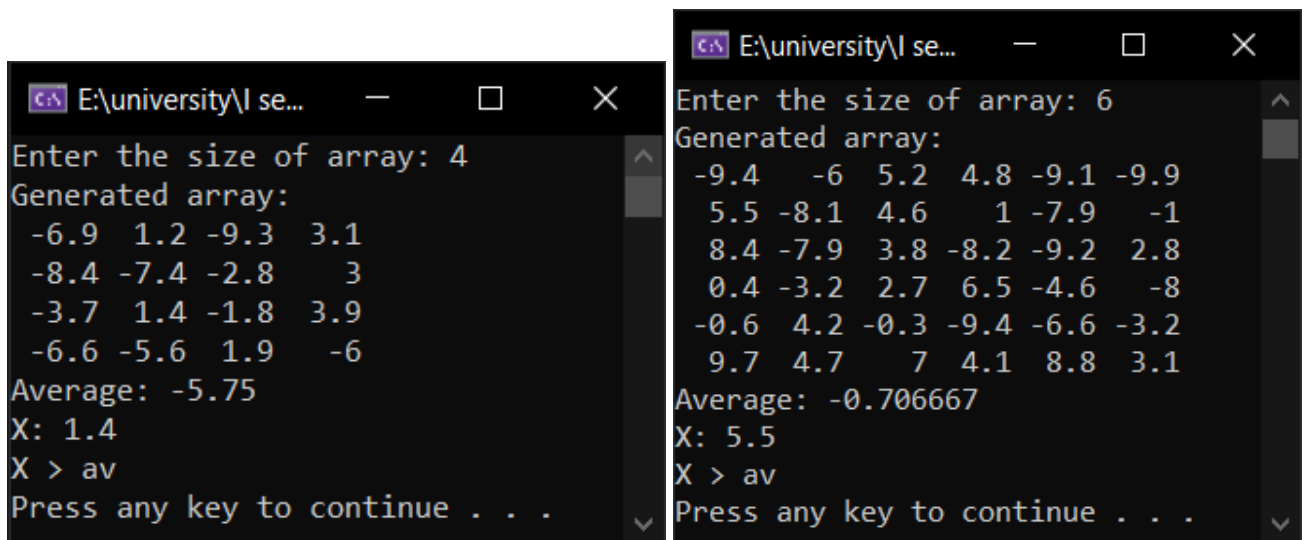
```
65 float findX(float** arr, int m, int n)
66 {
67     for (int i = 0; i < n; i++)
68     {
69         if(i % 2 == 0)
70         {
71             for (int k = 0; k < m; k++)
72             {
73                 if (arr[k][i] > 0)
74                 {
75                     return arr[k][i];
76                 }
77             }
78         }
79         else
80         {
81             for (int k = m - 1; k > -1; k--)
82             {
83                 if (arr[k][i] > 0)
84                 {
85                     return arr[k][i];
86                 }
87             }
88         }
89     }
90     return NULL;
91 }
```

```

93 void check(float x, float av)
94 {
95     if (x > av)
96         cout << "X > av" << endl;
97     else if (x == av)
98         cout << "X = av" << endl;
99     else
100         cout << "X < av" << endl;
101 }
102
103 int inputSize()
104 {
105     int n;
106     cout << "Enter the size of array: ";
107     cin >> n;
108     return n;
109 }
110
111 float** arrayDecl(int m, int n)
112 {
113     float** arr = new float* [m];
114     for (int i = 0; i < m; i++)
115         arr[i] = new float[n];
116     return arr;
117 }
118
119 void arrayDelete(float** arr, int m, int n)
120 {
121     for (int i = 0; i < m; i++)
122         delete[] arr[i];
123     delete[] arr;
124 }

```

Результат виконання програми.



The image shows two side-by-side screenshots of a C# console application window titled "E:\university\I se...".

The left screenshot shows the program with an array size of 4. The generated array is:

-6.9	1.2	-9.3	3.1
-8.4	-7.4	-2.8	3
-3.7	1.4	-1.8	3.9
-6.6	-5.6	1.9	-6

The average is calculated as -5.75. The user input X is 1.4. The comparison result is "X > av".

The right screenshot shows the program with an array size of 6. The generated array is:

-9.4	-6	5.2	4.8	-9.1	-9.9
5.5	-8.1	4.6	1	-7.9	-1
8.4	-7.9	3.8	-8.2	-9.2	2.8
0.4	-3.2	2.7	6.5	-4.6	-8
-0.6	4.2	-0.3	-9.4	-6.6	-3.2
9.7	4.7	7	4.1	8.8	3.1

The average is calculated as -0.706667. The user input X is 5.5. The comparison result is "X > av".

Перевірка виконання.

Перше виконання:

1. $(-6.9+1.2-9.3-8.4-7.4-3.7)/6 = -5.75$ – середнє значення визначено правильно
2. Оскільки прохід відбувається змійкою по стовбцям, то X визначено правильно
3. $1.4 > -5.75$ – визначено правильно

Перше виконання:

1. $(-9.4-6+5.2+4.8-9.1+5.5-8.1+4.6+1+8.4-7.9+3.8+0.4-3.2-0.6)/15 = -0.7067$ – середнє значення визначено правильно
2. Прохід відбувається по стовбцям, X визначено правильно
3. $5.5 > -0.7$ – визначено правильно

Висновок.

При виконанні лабораторної роботи було використано матрицю – іменовану сукупність послідовностей значень одного типу, де кожен елемент має два порядкові номери. Для виконання задачі був складений алгоритм для обходу матриці змійкою

по стовбцям. Також був досліджений алгоритм пошуку елементів над побічною діагоналлю матриці. В результаті роботи складено математичну модель, покрокові псевдокод та блок-схему, написано програму на мові C++ та успішно перевірено правильність її виконання відносно поставленої задачі.