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 № 8 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження алгоритмів пошуку та сортування»

Варіант\_**20**\_\_

Виконав студент: **ІП-11 Лошак Віктор Іванович**

(шифр, прізвище, ім'я, по батькові)

Перевірив: **Мартинова О.П.**

Київ 2021

**Лабораторна робота**

**Дослідження алгоритмів пошуку та сортування**

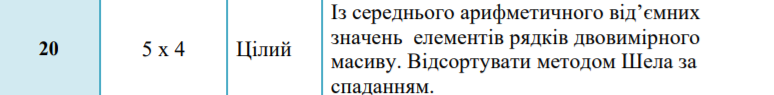
**Мета:** дослідити алгоритми пошуку та сортування, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

**Варіант №20**

**Задача:**

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. Створення нової змінної індексованого типу (одновимірний масив) та її ініціювання значеннями, що обчислюються згідно з варіантом (табл. 1).

**Варіант**:



**Розв’язок:**

1)Постановка задачі

Результатом розв’язку є масив елементи якого отримуються

шляхом знаходження середнього арифметичного елементів матриці що згенеровані випадковим чином. Для створення та ініціалізації матриці значеннями, ініціалізації результуючого масиву та здійснення процесу його сортування використаємо підпрограми що будуть реалізувати присвоєння значень елементам структур даних всередині арифметичного циклу. Для правильного сортування результуючої змінної індексованого типу використаємо алгоритм Шелла, що сортуватиме елементи за спаданням. Даний алгоритм можна реалізувати використовуючи цикли, з глибиною вкладеності 2, де зовнішній встановлює початкове значення gap що є ключовим значенням для цього алгоритму а внутрішні реалізують перестановку елементів в залежності від їх значень. Знаходження середнього арифметичного від’ємних значень рядків виконується за допомогою проходження по матриці змійкою. Такий спосіб обходу реалізовується за допомогою вкладених арифметичних циклів де зовнішній здійснює обхід по рядках а внутрішній використовує тернерний оператор для визначення напрямку проходження по рядку на даній ітерації зовнішнього циклу. Окремо реалізуємо функцію для заміни значень елементів що використовує проміжну змінну. Для використання випадкових згенерованих значень використаємо сторонні бібліотеки. Для перевірки значень матриці до та після сортування використаємо додаткофі функції для виводу значень індексованих змінних. Встановимо початкове значення ГПСЧ як початок відліку часу в системі.

2)Побудова математичної моделі

Складемо таблицю імен змінних

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Внутрішня змінна функції main та newMatrix/ параметр функції resultArray та printMatrix | індексований тип елементами якого є також індексовані типи з цілочисельних значень розміром 5\*4/ подвійний покажчик | matrix | Початкове дане |
| Внутрішня змінна функції main, resultArray/ параметр функцій printArray, swapElements, shellSort | індексований тип з дійсних значень/ покажчик | arr | результат обчислень |
| Параметр newMatrix, resultArray | Цілочисельний тип | rowAmount | Проміжний результат |
| Параметр newMatrix, resultArray та | Цілочисельний тип | columnAmount | Проміжний результат |
| Параметр printArray | Цілочисельний тип | columnLength | Проміжний результат |
| Параметр printArray | Цілочисельний тип | columnLength | Проміжний результат |
| Рахівник зовнішнього/ внутрішнього/простого арифметичного циклу | Цілочисельний тип | і, j | Проміжний результат |
| Внутрішня змінна resultArray | Цілочисельний тип | counter | Проміжний результат сума відємних елементів рядка |
| Внутрішня змінна resultArray | Цілочисельний тип | sum | Проміжний результат сума відємних елементів рядка |
| Внутрішня змінна функції printArray, shellSort | цілочисельний | length | Проміжний результат\ довжина arr |
| Параметри swap | цілочисельний | Ind1, ind2 | Проміжний результат обчислень/ значення індексів на яких відбувається заміна |

3)Програмні специфікації напишемо у псевдокоді та графічній формі у вигляді блок-схеми.

*Крок 1:* Визначимо основні дії.

*Крок 2:* Деталізуємо дії викликів функцій всередині main та оголошення змінних

5)Псевдокод

*Крок 1*

**Початок**

**-** створення нової матриці заповненої згенерованими випадково значеннями

-створення результуючого масиву та заповнення його середніми арифметичними негативних елементів рядків матриці

-сортування результуючого масиву використовуючи алгоритм шелла(за спаданням)

- виведення результуючого масиву значень на екран

**Кінець**

*Крок 2*

**Початок**

**- оголошення і присвоєння** matrix=newMatrix(5,4);

- **оголошення і присвоєння** arr=resultArray(matrix,5,4)

- **виклик** shellSort(arr, 5)

- **виклик** printArray(arr, 5)

**Кінець**

*Підпрограми:*

**Початок** newMatrix(rowAmount, columnAmount)

**- ініціалізація** arr[rowAmount] **динамічно**

**- повторити для** і=0 **з кроком** 1 **поки** і<rowAmount

**ініціалізація** arr[i]:=new int [columnAmount] **динамічно**

**-повторити для** j=0 **з кроком** 1 **поки** j<columnAmount

arr [i][j] **:** **=** (rand( ) % (rowAmount \* columnAmount)) - (rowAmount \* columnAmount / 2 - 1)

**все повторити**

**все повторити**

**-повернути** arr

**Кінець** newMatrix(rowAmount, columnAmount)

**Початок** resultArray(matrix,rowAmount, columnAmount)

**- ініціалізація** res[rowAmount] **динамічно**

**-** sum = 0, counter = 0, j=0

**- повторити для** і=0 **з кроком** 1 **поки** і<rowAmount

**- повторити для** j **з кроком**

**Якщо** (i % 2 == 0) **то** 1

**Інакше** -1

**Все якщо**

**поки**

**якщо** (i % 2 == 0) **то** j<columnAmount

**інакще** j>=0

**все якщо**

**якщо** matrix[i][j] < 0 **то**

sum= sum+matrix[i][j]

**інкремент** counter

**все якщо**

**все повторити**

**- Якщо** (i % 2 == 0) **то декремент j**

**Інакше інкремент j**

**Все якщо**

**-**

**все повторити**

**-повернути** arr

**Кінець** newMatrix(rowAmount, columnAmount)

**Початок** shellSort( arr, length)

**- повторити для** gap=[length/2] **з кроком** [gap/2] **поки** gap > 0

**- повторити для** j=0 **з кроком** 1 **поки** j<length

**-повторити для** i=j-gap **з кроком** -gap **поки** i>=0

**-якщо** arr[i] > arr[i + gap] **то все повторити**

**Інакше якщо** arr[i] < arr[i + gap] **то виклик** swapElements(arr, i, i + gap)

**все повторити**

**все повторити**

**все повторити**

**Кінець** shellSort( arr, length)

**Початок** swapElements( arr, ind1, ind2)

**- ініціалізація** temp = arr[ind1]

- **присвоєння** arr[ind1] = arr[ind2]

- **присвоєння** arr[ind2] = temp

**Кінець** swapElements( arr, ind1, ind2)

**Початок** printArray( arr, length)

**- повторити для** і=0 **з кроком** 1 **поки** і<length

**Вивід** arr[i]

**Все повторити**

**Кінець** printArray( arr, length)

**Початок** printMatrix( matrix, rowLength, columnLength)

**- повторити для** і=0 **з кроком** 1 **поки** і<rowAmount

**-повторити для** j=0 **з кроком** 1 **поки** j<columnAmount

**Вивід** matrix[i][j]

**все повторити**

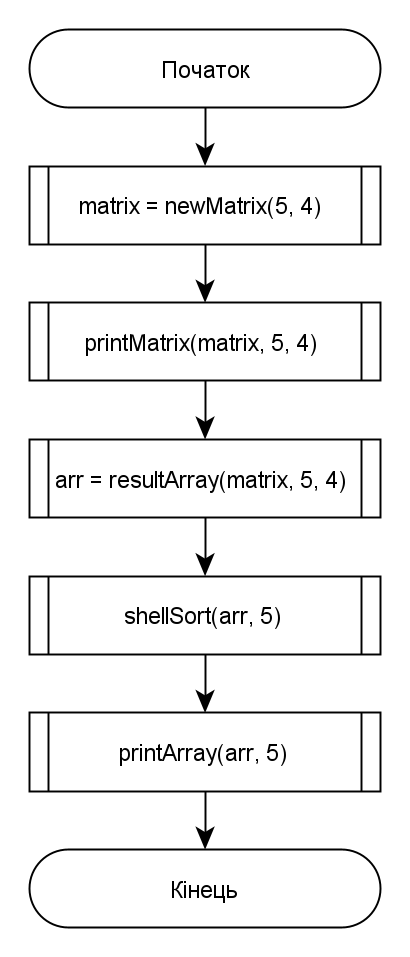
**все повторити**

**-повернути** arr

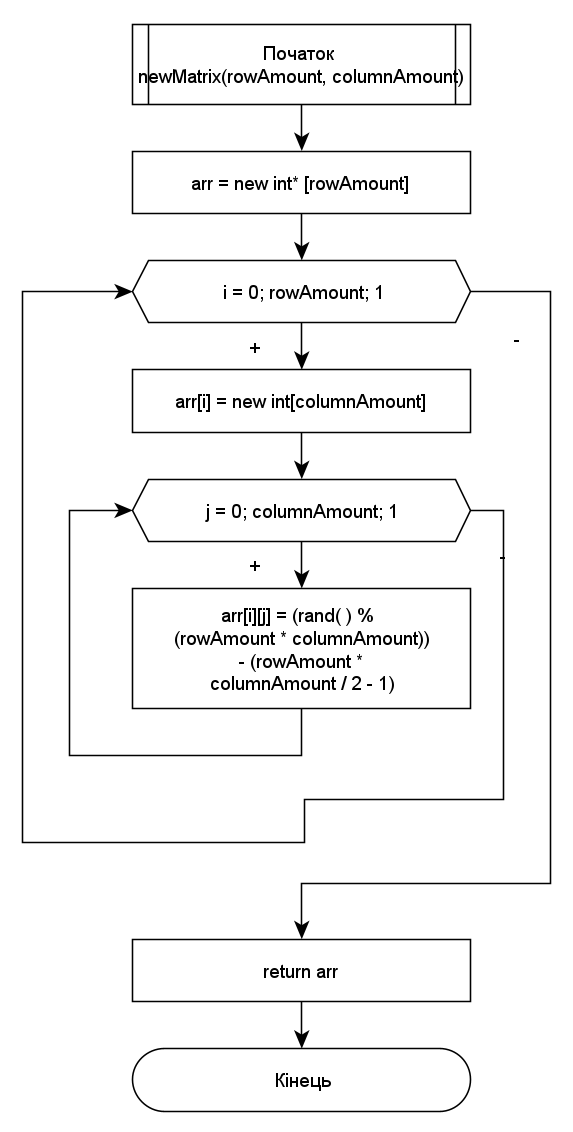
**Кінець** printMatrix( matrix, rowLength, columnLength)

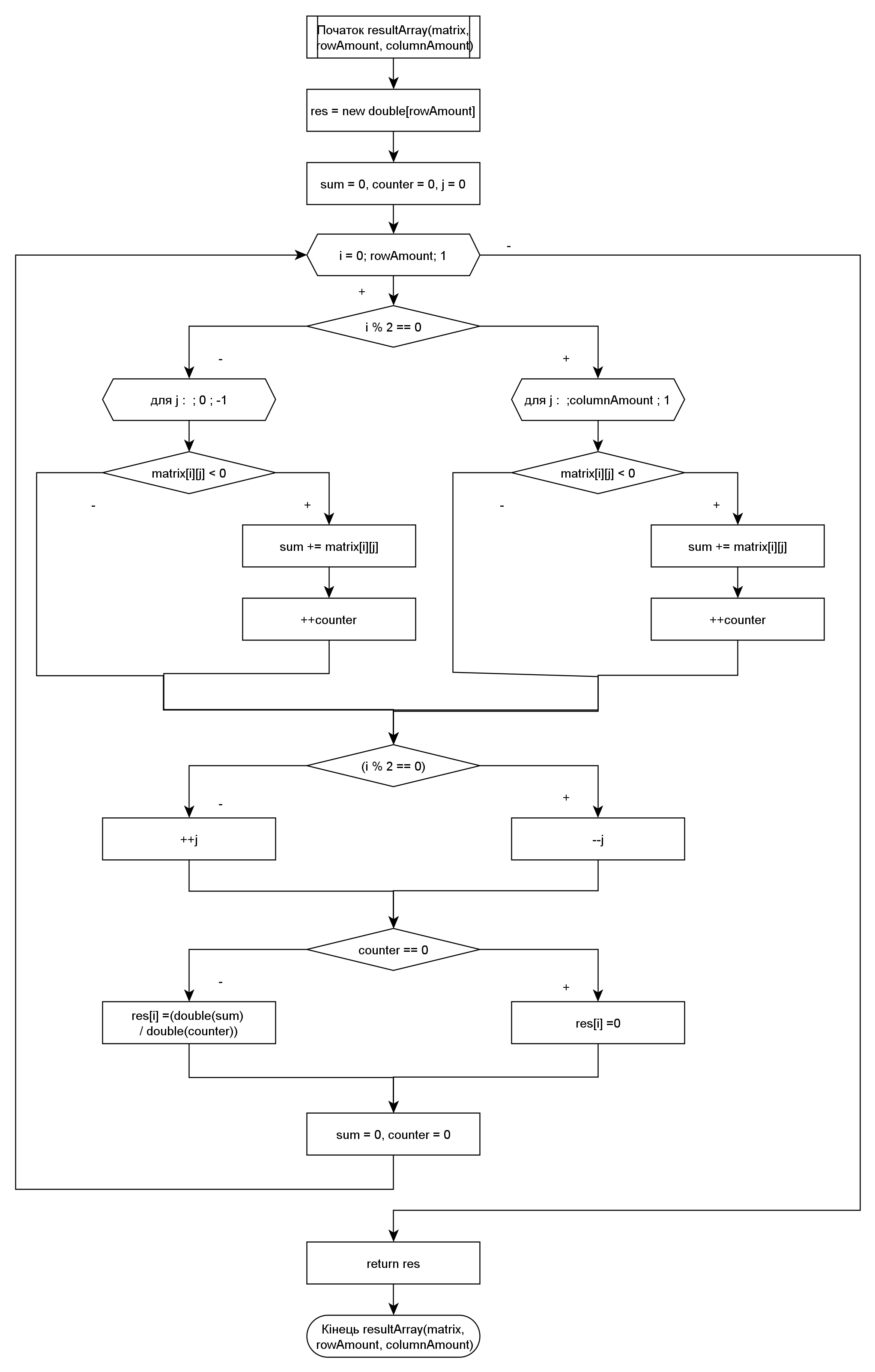
**Блок-Схема**

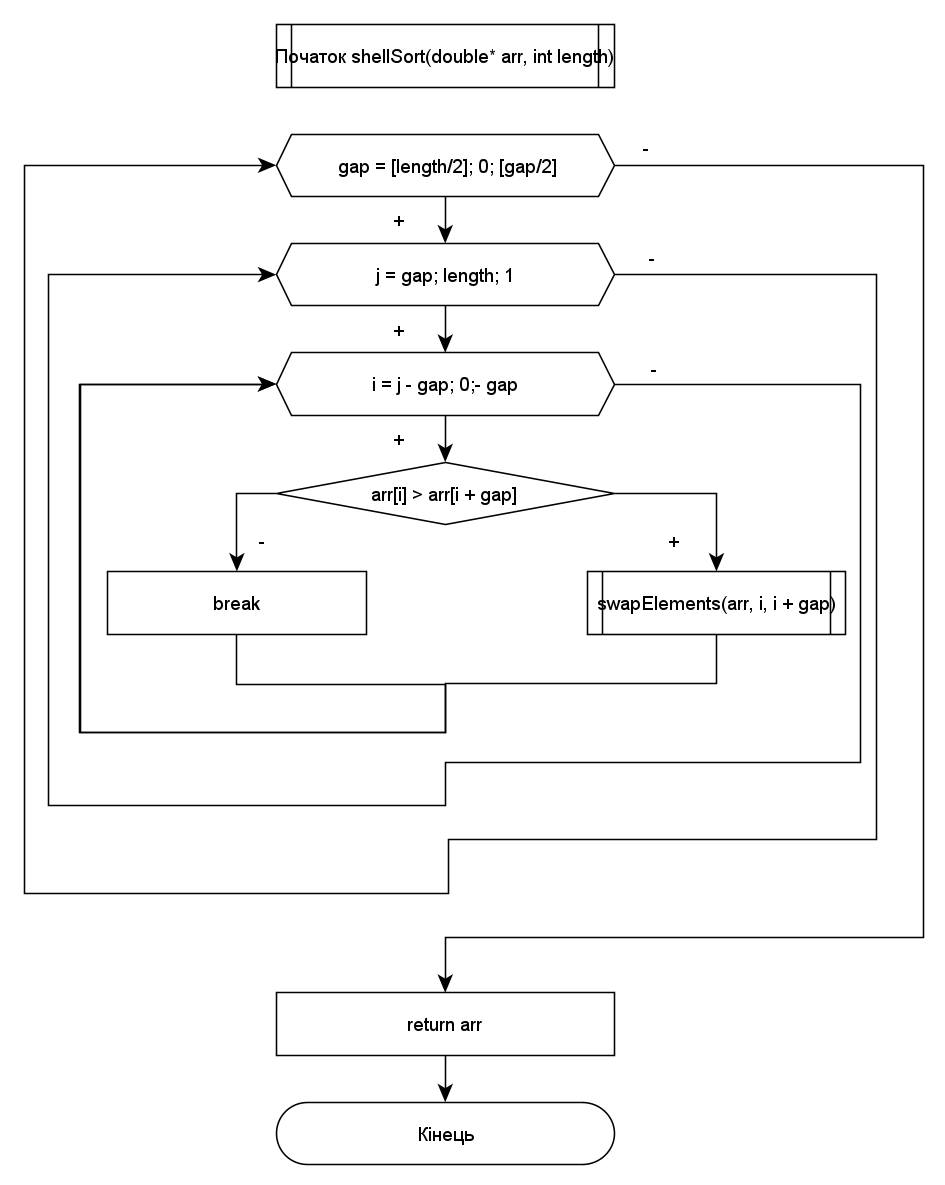
**Крок 2**

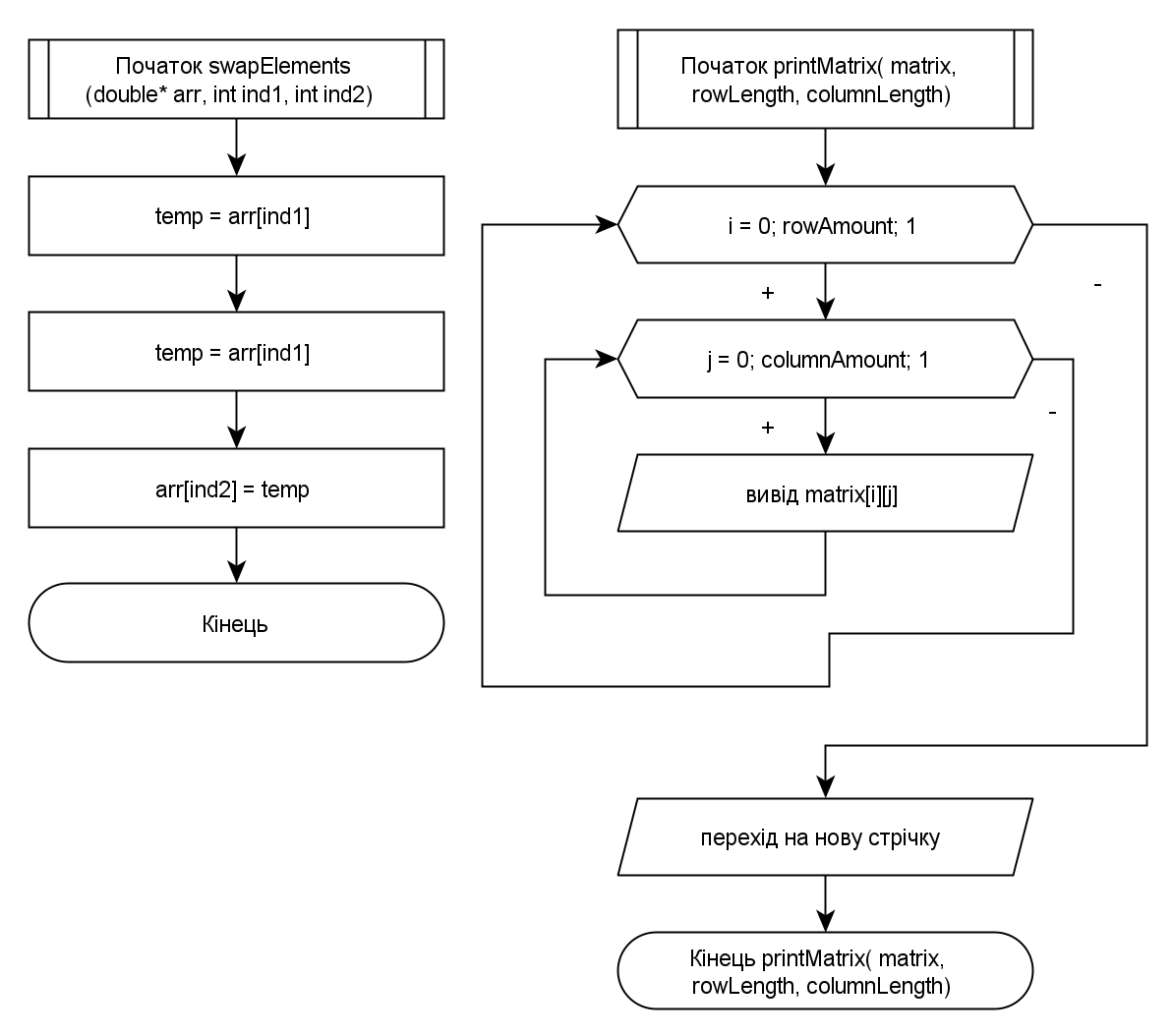
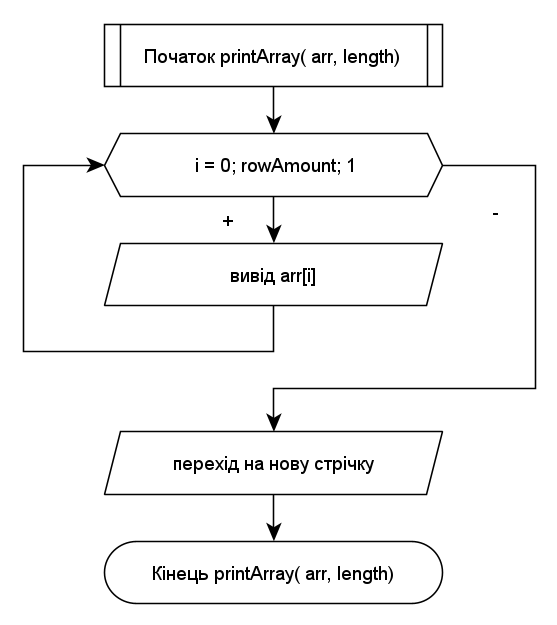
****

**Підпрограми:**

****



****

**** ****

**Код програми:**

К

#include <iostream>

#include <ctime>

#include <cmath>

#include <iomanip>

using namespace std;

int\*\* newMatrix(int rowAmount, int columnAmount);

double\* resultArray(int\*\* matrix, int rowAmount, int columnAmount);

void printMatrix(int\*\* arr, int rowLength, int columnLength);

void printArray(double\* arr, int length);

void shellSort(double\* array, int length);

void swapElements(double\* array, int ind1, int ind2);

int main( )

{

srand(time(0));

rand( );

//using magical numbers cause its easy to track what matrix youre creating and what youre working with

int\*\* matrix = newMatrix(5, 4);

printMatrix(matrix, 5, 4);

double\* arr = resultArray(matrix, 5, 4);

printArray(arr, 5);

shellSort(arr, 5);

printArray(arr, 5);

return 0;

}

int\*\* newMatrix(int rowAmount, int columnAmount)

{

//creating an empty matrix of given parameters

int\*\* arr = new int\* [rowAmount];

for (int i = 0; i < rowAmount; ++i)

{

arr[i] = new int[columnAmount];

for (int j = 0; j < columnAmount; ++j)

{

arr[i][j] = (rand( ) % (rowAmount \* columnAmount)) - (rowAmount \* columnAmount / 2 - 1);

}

}

return arr;

}

double\* resultArray(int\*\* matrix, int rowAmount, int columnAmount)

{

//creating an array whic contains the result for every row in the matrix

double\* res = new double[rowAmount];

//opening the outer cycle that goes through every row

//for every row creating the values: sum stores the sum of negative elements and counter stores amount of them

int sum = 0, counter = 0;

//going through the matrix using "snake"

int j = 0;

for (int i = 0; i < rowAmount; ++i)

{

//depending on the row number we move through matrix row in different directions

for (; (i % 2 == 0) ? j < columnAmount : j >= 0; (i % 2 == 0) ? ++j : --j)

{

if (matrix[i][j] < 0)

{

sum += matrix[i][j];

++counter;

}

}

//at the end of the row we set j value back to normal(in the gap between row max length and row min length in order

//to enter the previous cycle on the next row iteration

(i % 2 == 0) ? --j : ++j;

//putting result for current row into array depending on if negative numbers vere ever found in the row

res[i] = (counter == 0) ? 0 : (double(sum) / double(counter));

//setting zero values of sum and counter for the next row

sum = 0, counter = 0;

}

return res;

}

void printMatrix(int\*\* matrix, int rowLength, int columnLength)

{

for (int i = 0; i < rowLength; ++i)

{

for (int j = 0; j < columnLength; ++j)

{

cout << setw(3) << matrix[i][j];

}

cout << '\n';

}

cout << '\n';

}

void printArray(double\* arr, int length)

{

cout.precision(2);

for (int i = 0; i < length; ++i) cout << setw(4) << arr[i] << ' ';

cout << '\n';

}

void shellSort(double\* arr, int length)

{

for (int gap = floor(length / 2); gap > 0; gap = floor(gap / 2))

{

for (int j = gap; j < length; ++j)

{

for (int i = j - gap; i >= 0; i -= gap)

{

if (arr[i] > arr[i + gap]) break;

else if (arr[i] < arr[i + gap]) swapElements(arr, i, i + gap);

}

}

}

}

void swapElements(double\* arr, int ind1, int ind2)

{

double temp = arr[ind1];

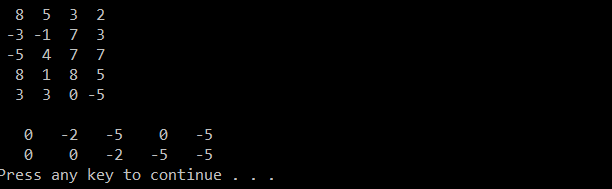
arr[ind1] = arr[ind2];

arr[ind2] = temp;

}

Для перевірки правильності роботи алгоритму додамо в код проміжні перевірки поточних значень змінних за допомогою функцій виведення.

Результат запуску програми для перевірки:



|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Ініціалізація матриці 5\*4 випадково згенерованими значеннями що містять від’ємні елементи |
| 2 | Перший рядок, середнє арифметичне від’ємних елементів =0, запис значення в перший елемент array |
| 3 | Другий рядок, середнє арифметичне від’ємних елементів =-4/2=-2, запис значення в 2 елемент array |
| 4 | 3 рядок, середнє арифметичне від’ємних елементів =-5/1=-5, запис значення в 3 елемент array |
| 5 | 4 рядок, середнє арифметичне від’ємних елементів =0, запис значення в 4 елемент array |
| 6 | 5 рядок, середнє арифметичне від’ємних елементів =-5/1, запис значення в 5 елемент array |
| 7 | Сортування масиву array методом шела за спаданням і вивід на екран |
|  | Кінець |

**Висновок:**

В ході виконання даної лабораторної роботи я навчився використовувати алгоритми пошуку та сортування(а саме алгоритм Шелла), набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій , а саме, використав алгоритми пошуку для знаходження від’ємних елементів у масиві та дослідив методи обходу матриці . Навчився створювати блок-схеми алгоритмів що містять алгоритм лінійного пошуку та алгоритмів сортування та описувати програму за допомогою псевдокоду. Використовуючи отримані знання я успішно обрахував і відобразив на екрані відсортований масив з середніх арифметичних значень від’ємних елементів кожного рядка заданої матриці, на компільованій мові с++.