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.

Основи алгоритмізації»

«Дослідження алгоритмів пошуку та сортування»

Варіант 10

Виконав студент ІП-12 Горобець Олексій Сергійович

(шифр, прізвище, ім'я, по батькові)

Перевірив

( прізвище, ім'я, по батькові)

Київ 202 1

**Лабораторна робота 8**

**Дослідження алгоритмів пошуку та сортування**

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

**Варіант 10**

***Задача 1.10***.

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

1. Опису змінної індексованого типу двовимірного масиву n x m

2. Ініціювання змінної, що описана в п.1 даного завдання.

3. Створення нової змінної індексованого типу (одновимірний масив) та її ініціювання мінімальними значеннями елементів стовпців двовимірного масиву.

***Постановка задачі*.** Результатом є відсортований масив з мінімальних елементів стовпців заданої матриці. Для визначення цього масиву нам необхідно задані розміри матриці, сама матриця. Інших початкових даних для розв’язку не потрібно.

***Математична побудова.*** Складемо таблицю змінних.

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| **Кількість рядків матриці n** | Цілий | n | Початкове дане |
| **Кількість стовпців матриці m** | Цілий | m | Початкове дане |
| **Двовимірний масив matrix** | Індексований | matrix | Початкове дане |
| **Мінімальний елемент у стовпці матриці min** | Цілий | min | Проміжне значення |
| **Проміжок для сортування Шелла** | Цілий | gap | Проміжне значення |
| **Одновимірний масив res\_arr** | Індексований | res\_arr | Результат |

Таким чином, математичне формулювання задачі зводиться до розрахування мінімальних елементів у стовпців матриці, запису їх у одновимірний масив. Далі необхідно відсортувати масив алгоритмом Шелла.

***Розв’язання:***

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

***Крок 1.*** Визначимо основні дії.

***Крок 2.*** Деталізуємо ініціалізації початкових змінних ***n, m, matrix***

***Крок 3.*** Деталізуємо перебір стовпців матриці

***Крок 4.*** Деталізуємо пошук та заповнення масиву мінімальними елементами зі стовпців матриці

***Крок 5.*** Деталізуємо сортування масиву алгоритмом Шелла

***Псевдокод***

*крок 1*

**початок**

Ініціалізації початкових змінних ***n, m, matrix***

Деталізуємо дію перебору стовпців матриці

Деталізуємо дію пошуку та заповнення масиву мінімальними елементами зі стовпців матриці

Деталізуємо дію сортування масиву алгоритмом Шелла

**Кінець**

*крок 2*

**початок**

**Ввід** n, m, matrix

**getColMinArr(matrix, n , m)**

Деталізуємо сортування масиву алгоритмом Шелла

**Кінець**

**початок функції getColMinArr(matrix, n , m)**

Деталізуємо дію перебору стовпців матриці

Деталізуємо дію пошуку та заповнення масиву мінімальними елементами зі стовпців матриці

**кінець функції getColMinArr**

*крок 3*

**початок**

**Ввід** n, m, matrix

**getColMinArr(matrix, n , m)**

Деталізуємо сортування масиву алгоритмом Шелла

**Кінець**

**початок функції getColMinArr(matrix, n , m)**

**повторити**

**для j від 0 до m**

**повторити**

**для і від 0 до n**

Деталізуємо дію пошук та заповнення масиву мінімальними елементами зі стовпців матриці

**все повторити**

**все повторити**

**кінець функції getColMinArr**

Крок 4

**початок**

**Ввід** n, m, matrix

**getColMinArr(matrix, n , m)**

Деталізуємо дію сортування масиву алгоритмом Шелла

**Кінець**

**початок функції getColMinArr(matrix, n , m)**

**повторити**

**для j від 0 до m**

min:=matrix[0][j]

**повторити**

**для і від 0 до n**

**якщо** matrix[i][j]<min

**то**

min:=matrix[i][j]

**все повторити**

res\_arr[j]:=min

**все повторити**

**Повернути** res\_arr

**кінець функції getColMinArr**

**Крок 5. Деталізуємо сортування масиву за зростанням алгоритмом Шелла**

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

***Крок 1.*** Визначимо основні дії.

***Крок 2.*** Деталізуємо ініціалізацію початкової змінної ***gap***

***Крок 3.*** Деталізуємо розбиття масиву на списки елементів, що відстають один від одного на gap

***Крок 4.*** Деталізуємо сортування елементів кожного списку сортуванням вставками

***Крок 5.*** Деталізуємо дію зміни проміжку між елементами масиву, що вибираються

***Крок 1***

**початок функції arrShellsort(arr\_res, m)**

Деталізуємо дію ініціалізації початкової змінної ***gap***

Деталізуємо дію розбиття масиву на списки елементів, що відстають один від одного на **gap**

Деталізуємо дію сортування елементів кожного списку сортуванням вставками

Деталізуємо дію зміни проміжку між елементами масиву, що вибираються

**кінець функції arrShellsort**

***Крок 2***

**початок функції arrShellsort(arr\_res, m)**

gap:=m/2

Деталізуємо дію розбиття масиву на списки елементів, що відстають один від одного на **gap**

Деталізуємо дію сортування елементів кожного списку сортуванням вставками

Деталізуємо дію зміни проміжку між елементами масиву, що вибираються

**кінець функції arrShellsort**

***Крок 3***

**початок функції arrShellsort(arr\_res, m)**

gap:=m/2

**поки gap>0**

**повторити**

**повторити**

**для і від gap до m**

temp:=arr\_res[i]

Деталізуємо дію сортування елементів кожного списку сортуванням вставками

**все повторити**

Деталізуємо дію зміни проміжку між елементами масиву, що вибираються

**все повторити**

**кінець функції arrShellsort**

***Крок 4***

**початок функції arrShellsort(arr\_res, m)**

gap:=m/2

**поки**  gap>0

**повторити**

**повторити**

**для і від gap до m**

temp:=arr\_res[i]

j:=i

**поки j>=gap and arr\_res[j-gap]<temp**

**поторити**

arr[j]:= arr\_res[j-gap]

j:=j-gap

**все повторити**

arr\_res[j]:=temp

**все повторити**

Деталізуємо дію зміни проміжку між елементами масиву, що вибираються

**все повторити**

**кінець функції arrShellsort**

***Крок 5***

**початок функції arrShellsort(arr\_res, m)**

**gap:=m/2**

**поки gap>0**

**повторити**

**повторити**

**для і від gap до m**

temp:=arr\_res[i]

j:=i

**поки j>=gap and arr\_res[j-gap]<temp**

**поторити**

arr[j]:= arr\_res[j-gap]

j:=j-gap

**все повторити**

arr\_res[j]:=temp

**все повторити**

gap:=gap/2

**все повторити**

**кінець функції arrShellsort**

Об’єднуємо псевдокод основної програми з функцією arrShellsort

Крок 5

**початок**

**Ввід** n, m, matrix

**getColMinArr(matrix, n , m)**

**arrShellsort(res\_arr, m)**

**Кінець**

**початок функції getColMinArr(matrix, n , m)**

**повторити**

**для j від 0 до m**

min:=matrix[0][j]

**повторити**

**для і від 0 до n**

**якщо** matrix[i][j]<min

**то**

min:=matrix[i][j]

**все повторити**

res\_arr[j]:=min

**все повторити**

**Повернути** res\_arr

**кінець функції getColMinArr**

**початок функції arrShellsort(arr\_res, m)**

**gap:=m/2**

**поки gap>0**

**повторити**

**повторити**

**для і від gap до m**

temp:=arr\_res[i]

j:=i

**поки j>=gap and arr\_res[j-gap]<temp**

**поторити**

arr[j]:= arr\_res[j-gap]

j:=j-gap

**все повторити**

arr\_res[j]:=temp

**все повторити**

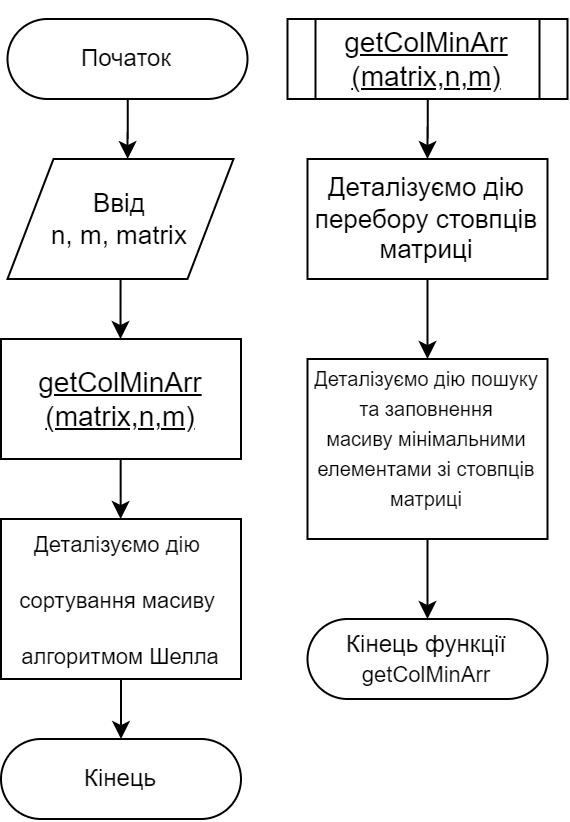
gap:=gap/2

**все повторити**

**кінець функції arrShellsort**

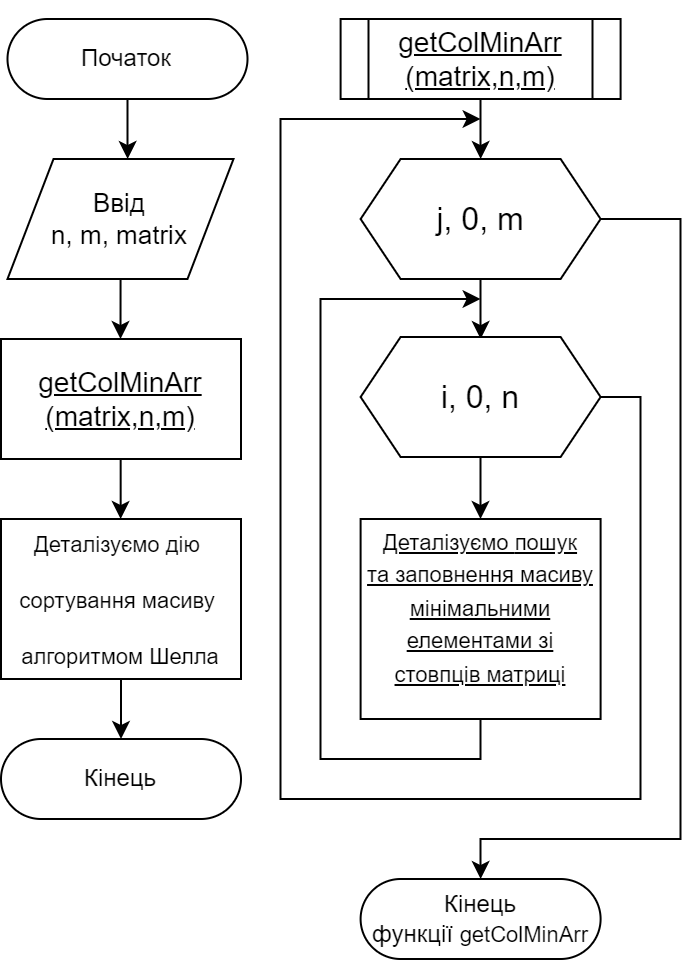
***Блок-схема***

***Крок 1 Крок 2***

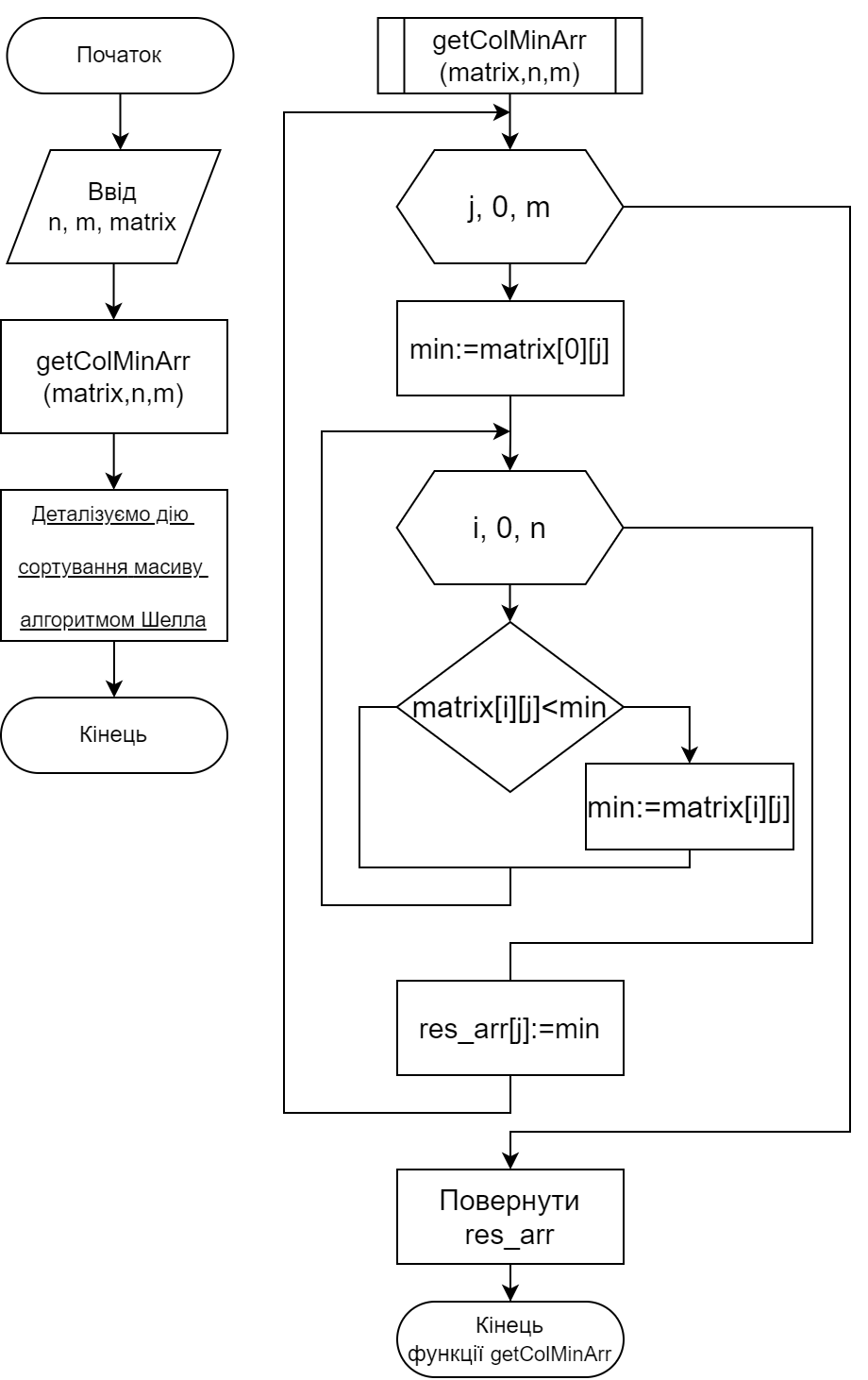
**

**

***Крок 3***

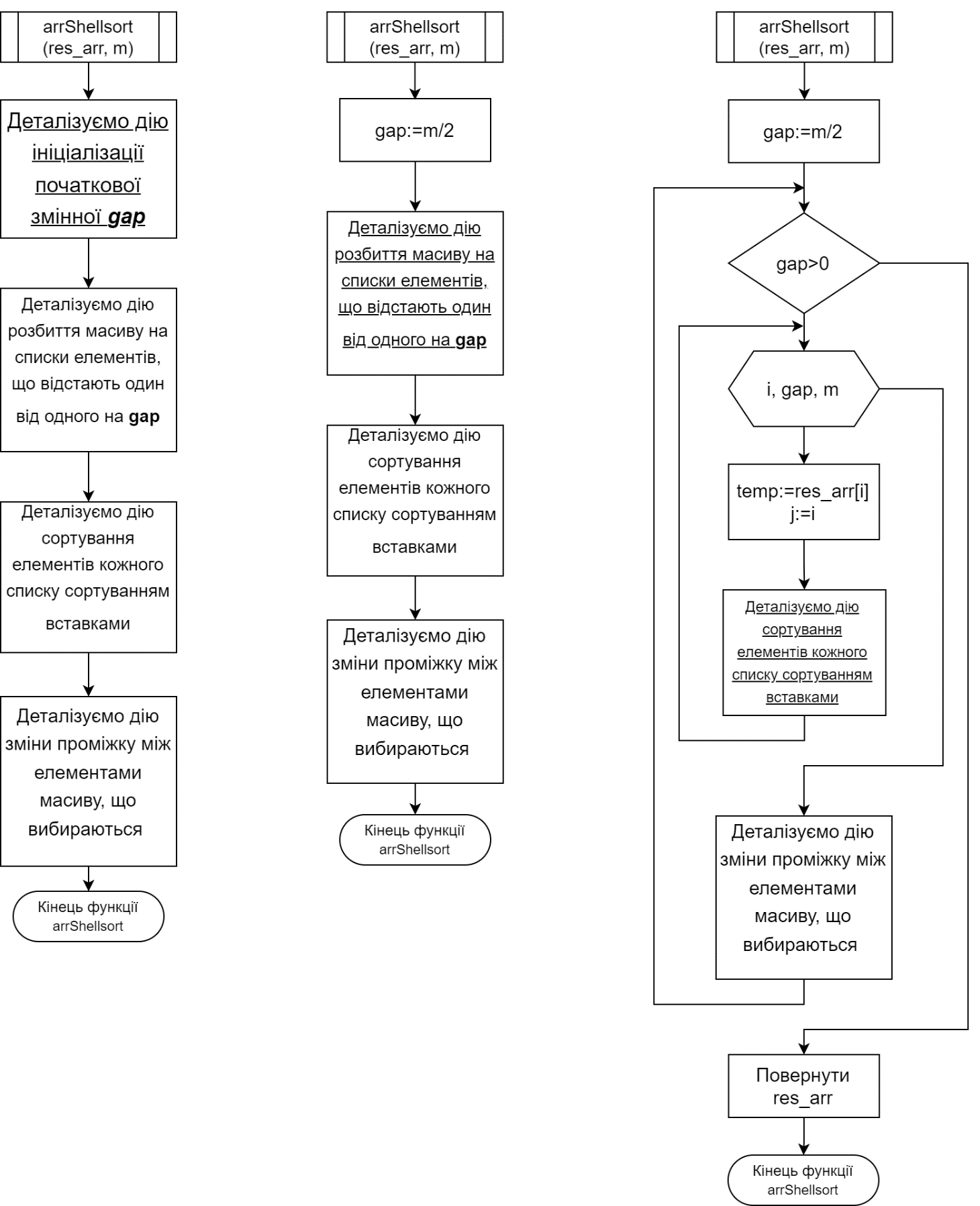
******

***Крок 4***

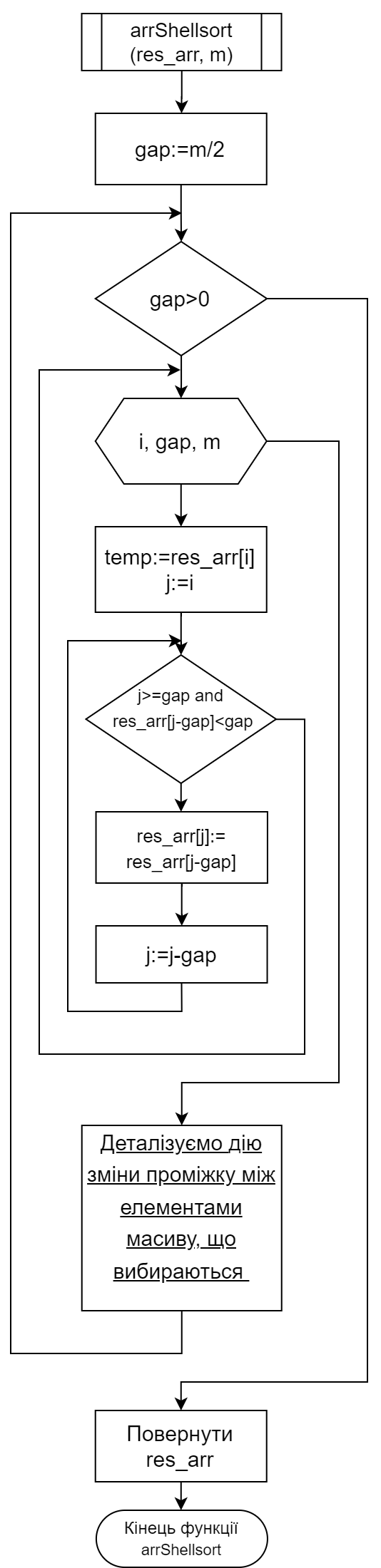
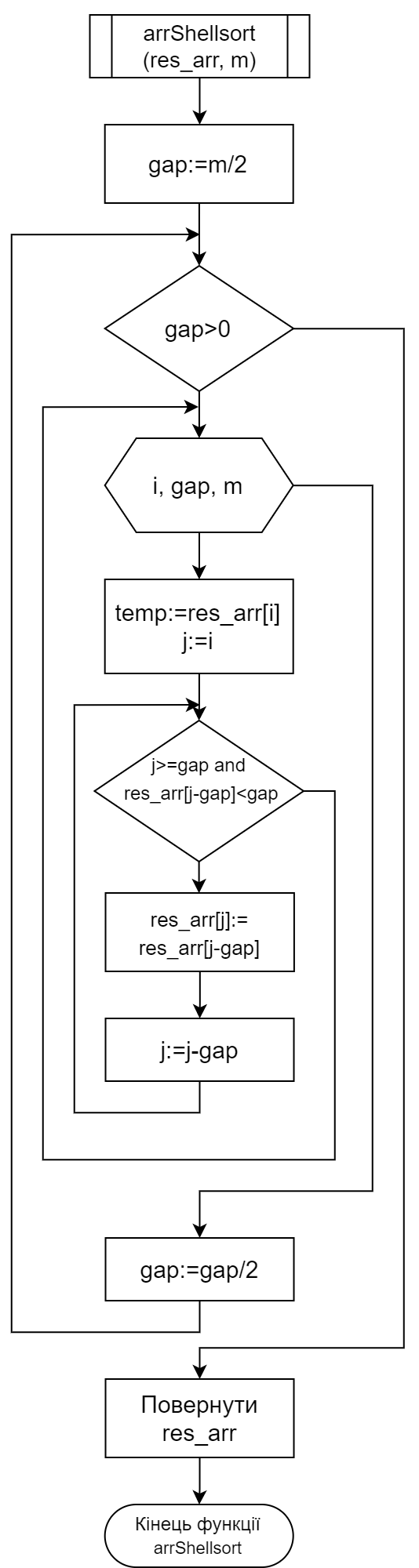
******

***Блок схема підпрограми arrShellsort***

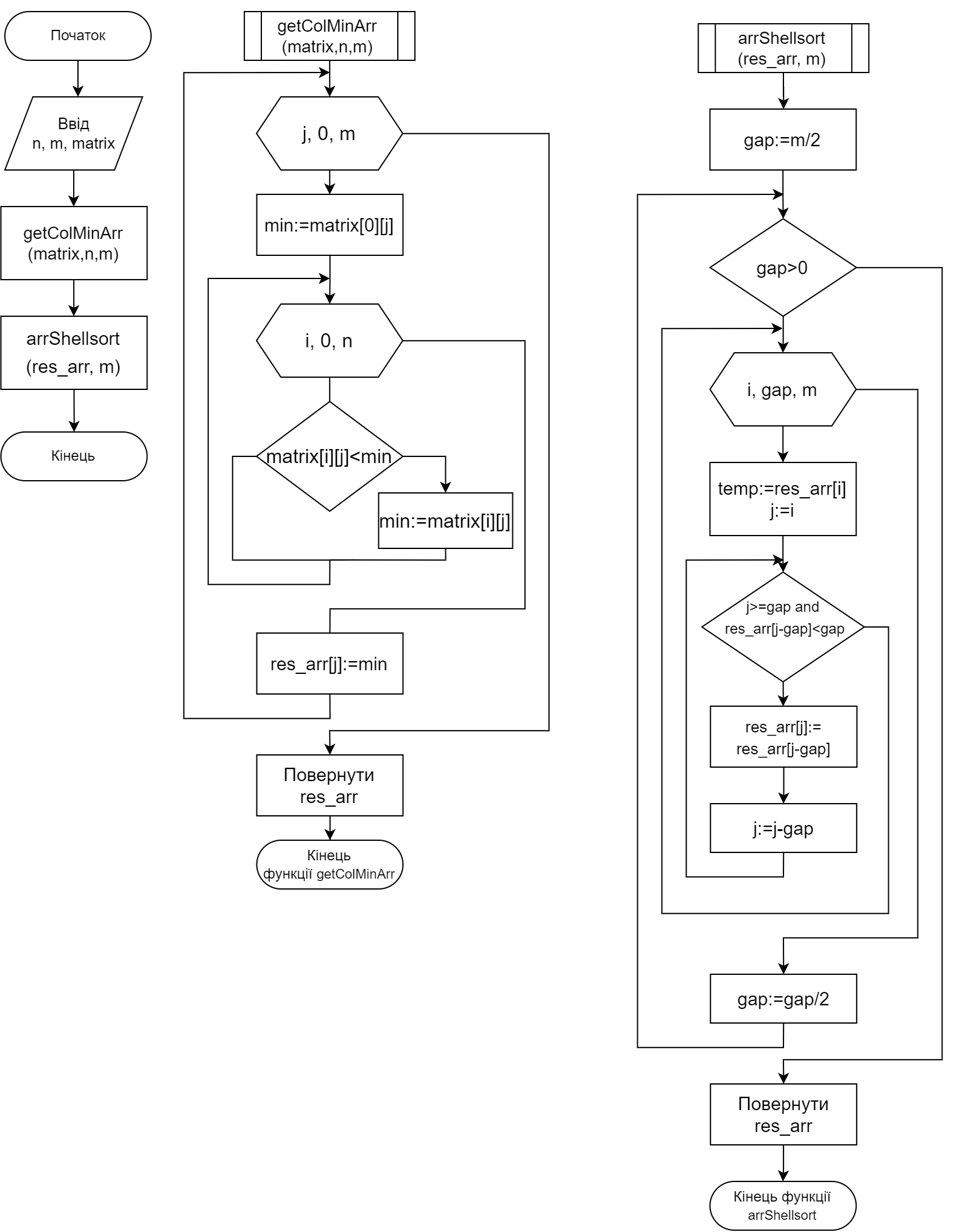
***Крок 1 Крок 2 Крок 3***

******

***Крок 4 Крок 5***

******

***Крок 5 кінцева блоксхема***

******

***Код програми c++***

#include <iostream>

#include <ctime>

#include <iomanip>

using namespace std;

int intInput(string);

double\*\* createDoubleMatrix(int, int);

void randMatrixFill(double\*\*, int, int);

double\* getColMinArr(double\*\*, int, int);

void arrOutput(double\*\*, int, int, string);

void arrOutput(double\*, int, string);

void arrShellsort(double\*, int);

void arrDelete(double\*);

void arrDelete(double\*\* matrix, int n);

int main()

{

setlocale(LC\_ALL, "rus");

int n, m;

n = intInput("Enter n: ");

m = intInput("Enter m: ");

double\*\* matrix = createDoubleMatrix(n, m);

randMatrixFill(matrix, n, m);

double\* res\_arr = new double[m];

res\_arr = getColMinArr(matrix, n, m);

arrOutput(matrix, n, m, "\nInitial matrix\n");

arrOutput(res\_arr, m, "\nArray of min elements:\n");

arrShellsort(res\_arr, m);

arrOutput(res\_arr, m, "\n\nArray after Shell sort:\n");

arrDelete(res\_arr);

arrDelete(matrix, n);

}

int intInput(string msg) {

int n;

cout << msg;

cin >> n;

return n;

}

double\*\* createDoubleMatrix(int n, int m) {

double\*\* matrix = new double\* [n];

for (int i = 0; i < n; i++) {

matrix[i] = new double[m];

}

return matrix;

}

void randMatrixFill(double\*\* matrix, int n, int m) {

srand(time(NULL));

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

matrix[i][j] = rand() % 201 - 100;

}

}

}

double\* getColMinArr(double\*\* matrix, int n, int m) {

double min;

double\* res\_arr = new double[m];

for (int j = 0; j < m; j++) {

min = matrix[0][j];

for (int i = 0; i < n; i++) {

if (min > matrix[i][j]) {

min = matrix[i][j];

}

}

res\_arr[j] = min;

}

return res\_arr;

}

void arrOutput(double\*\* matrix, int n, int m, string msg) {

cout << msg << endl;

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

cout << setw(4) << matrix[i][j];

}cout << endl;

}

}

void arrOutput(double\* arr, int n, string msg){

cout<<msg<< endl;

for (int i = 0; i < n; i++) {

cout << setw(4) << arr[i];

}

cout << endl;

}

void arrShellsort(double\* arr, int n) {

for (int gap = n / 2; gap > 0; gap /= 2) {

for (int i = gap; i < n; i ++) {

int temp = arr[i];

int j;

for (j = i; j >= gap && arr[j - gap] < temp; j -= gap)

arr[j] = arr[j - gap];

arr[j] = temp;

}

}

}

void arrDelete(double\* arr) {

delete[] arr;

}

void arrDelete(double\*\* matrix, int n) {

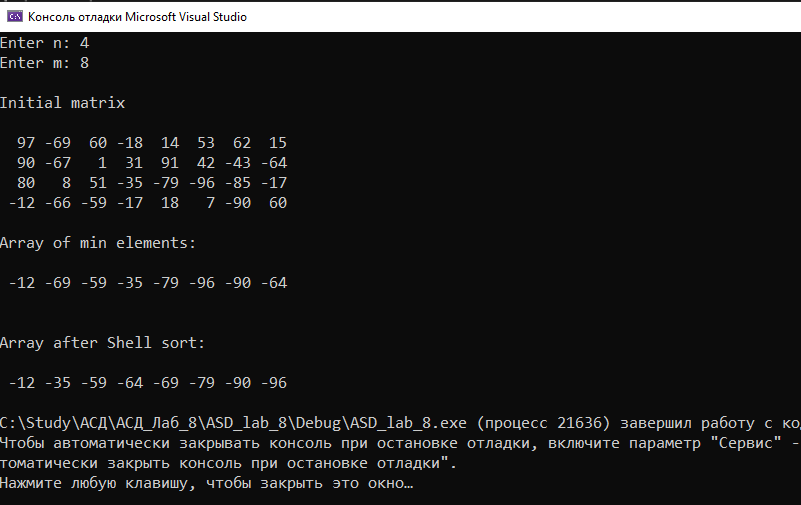
for (int i = 0; i < n; i++) {

delete[] matrix[i];

}

delete[] matrix;

}



***Висновок***

Протягом лабораторної роботи було досліджено алгоритми пошуку та сортування та набуто практичних навичок їх використання під час складання програмних специфікацій. Особливістю роботи було:

1. Розробка функції пошуку мінімальних елементів у колонках матриці та запису їх у окремий масив для подальшої обробки
2. Розробка підпрограми для сортування масиву за спаданням користуючись алгоритмом Шелла.
3. Опис алгоритму сортування Шелла, за допомогою чіткої, покрокової блок-схеми та псевдокоду.