# 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»

Варіант 13

Виконав студент ІП-11 Калашніков Андрій Євгенович

(шифр, прізвище, ім'я, по батькові)

Перевірив Мартинова О.П.

( прізвище, ім'я, по батькові)

Київ 2021

**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.

**Індивідуальне завдання:**

13. 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).



**Постановка задачі**

За допомогою підпрограми треба створити двовимірний масив цілих чисел заданої розмірності(6 х 6) за допомогою двох арифметичних циклів(один вкладений в інший) та генерації випадкових чисел з певного проміжку. Потім, використовуючи іншу підпрограму, створити одновимірний масив та заповнити його додатними елементами головної діагоналі. За допомогою ще однієї підпрограми, треба відсортувати цей масив методом бульбашки за зростанням. Отриманий масив і буде шуканим.

**Побудова математичної моделі**

Складемо таблицю змінних

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Задана розмірність двовимірного масиву | Цілий | n | Початкове дане |
| Заданий двовимірний масив | Цілий | matrix | Початкове дане |
| Шуканий одновимірний масив | Цілий | array | Результат |
| Індекс одновимірного, (поточного додатнього елемента головної діагоналі двовим. масиву) | Цілий | k | Проміжне дане |
| Поточноий найбільший елемент одновимірного масиву | Цілий | temp | Проміжне дане |

**Розв’язання**

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

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію створення та заповнення двовимірного масиву.

Крок 3. Деталізуємо дію виведення двовимірного масиву.

Крок 4. Деталізуємо дію створення та заповнення одновимірного масиву.

Крок 5. Деталізуємо дію виведення одновимірного масиву.

Крок 6. Деталізуємо сортування за зростанням елементів одновимірного масиву.

**Псевдокод**

Основна програма

**Початок**

n := 5

matrix := generate\_matrix(n, n)

**виведення** “Matrix:\n”

display\_matrix(matrix, n)

array := create\_array(matrix, n)

**виведення** “Array:\n”

display\_array(array, n)

sort\_descending(array, n)

**виведення** “Sorted array:\n”

display\_array(array, n)

**Кінець**

Підпрограми

**generate\_matrix(rows, columns)**

**повторити для** і **від** 0 **до** rows

**повторити для** j **від** 0 **до** columns

arr[i][j] := rand(-9, 9)

**все повторити**

**все повторити**

**повернути** arr

**Кінець generate\_matrix**

**display\_matrix(matrix, size)**

**повторити для і від 0** до size

**повторити для j від 0 до** size

**виведення** matrix[i][j]

**все повторити**

**все повторити**

**Кінець display\_matrix**

**create\_array(matrix, size)**

k := 0

**повторити для і від 0 до** size

**якщо** matrix[i][i] > 0

**то**

array[k] := matrix[i][i]

k += 1

**все якщо**

**все повторити**

**повернути** array

**Кінець create\_array**

**display\_array(arr, size)**

i := 0

**поки** i < length **і** arr[i] != 0 **повторити**

**виведення** arr[i]

i += 1

**все повторити**

**Кінець display\_array**

**sort(arr, length)**

i := 0

**повторити для** і **від** 0 **до** length-1

**повторити для** j **від** 0 **до** length-1

**якщо** arr[j+1] <arr[j]

**то**

temp= arr[j+1]

arr[j+1]=arr[j]

arr[j]=temp

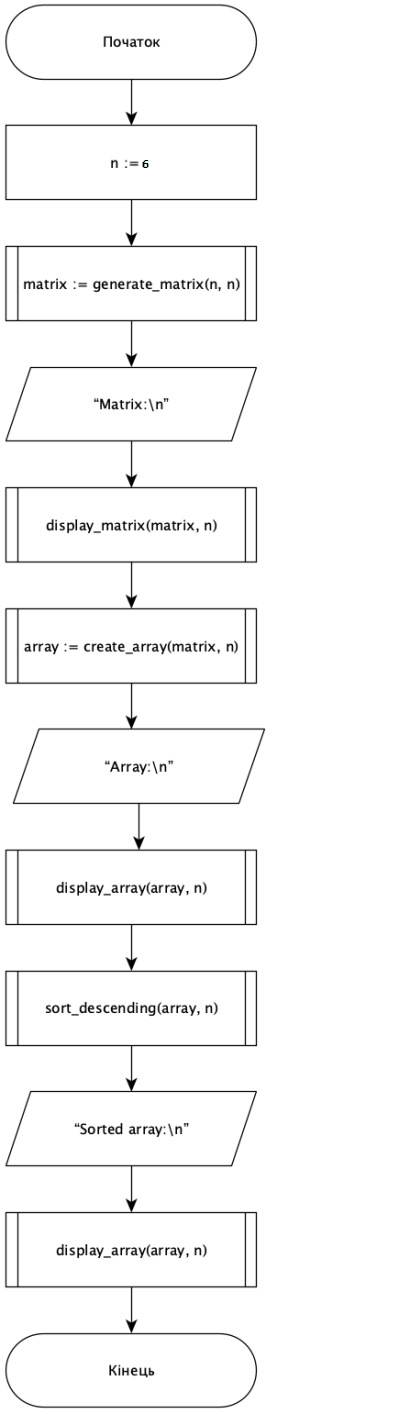
**все якщо**

**все повторити**

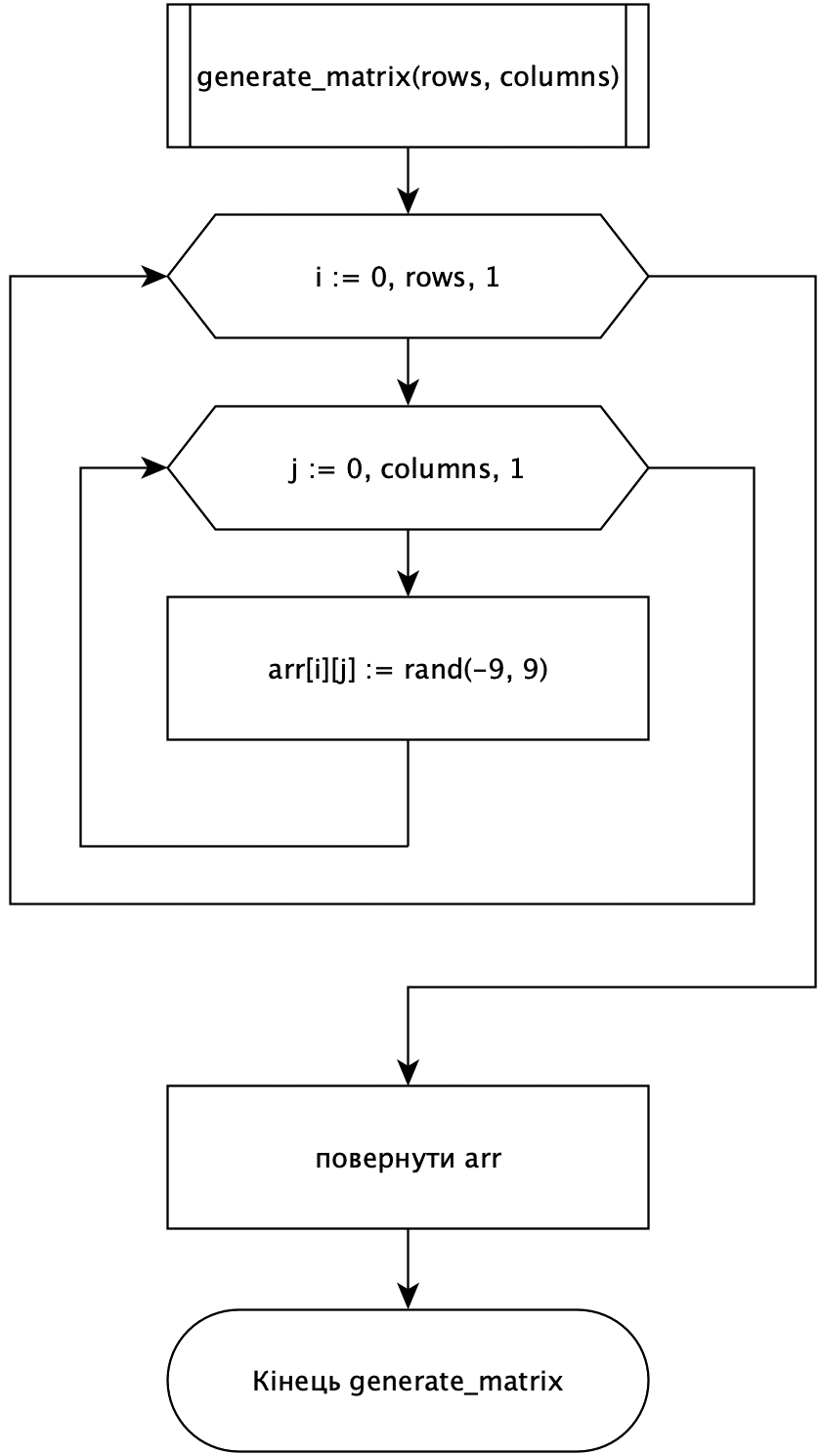
**все повторити**

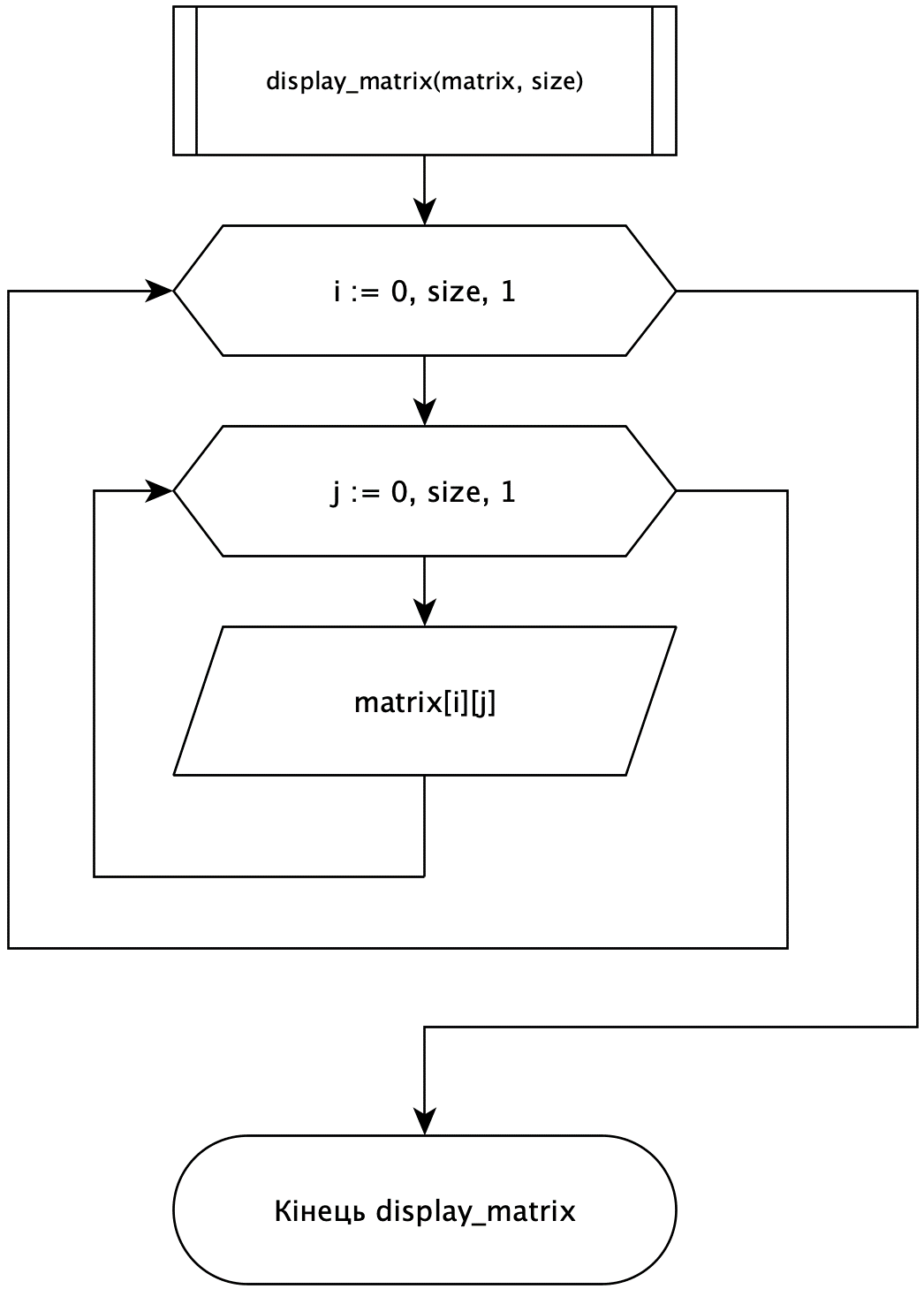
**Кінець sort**

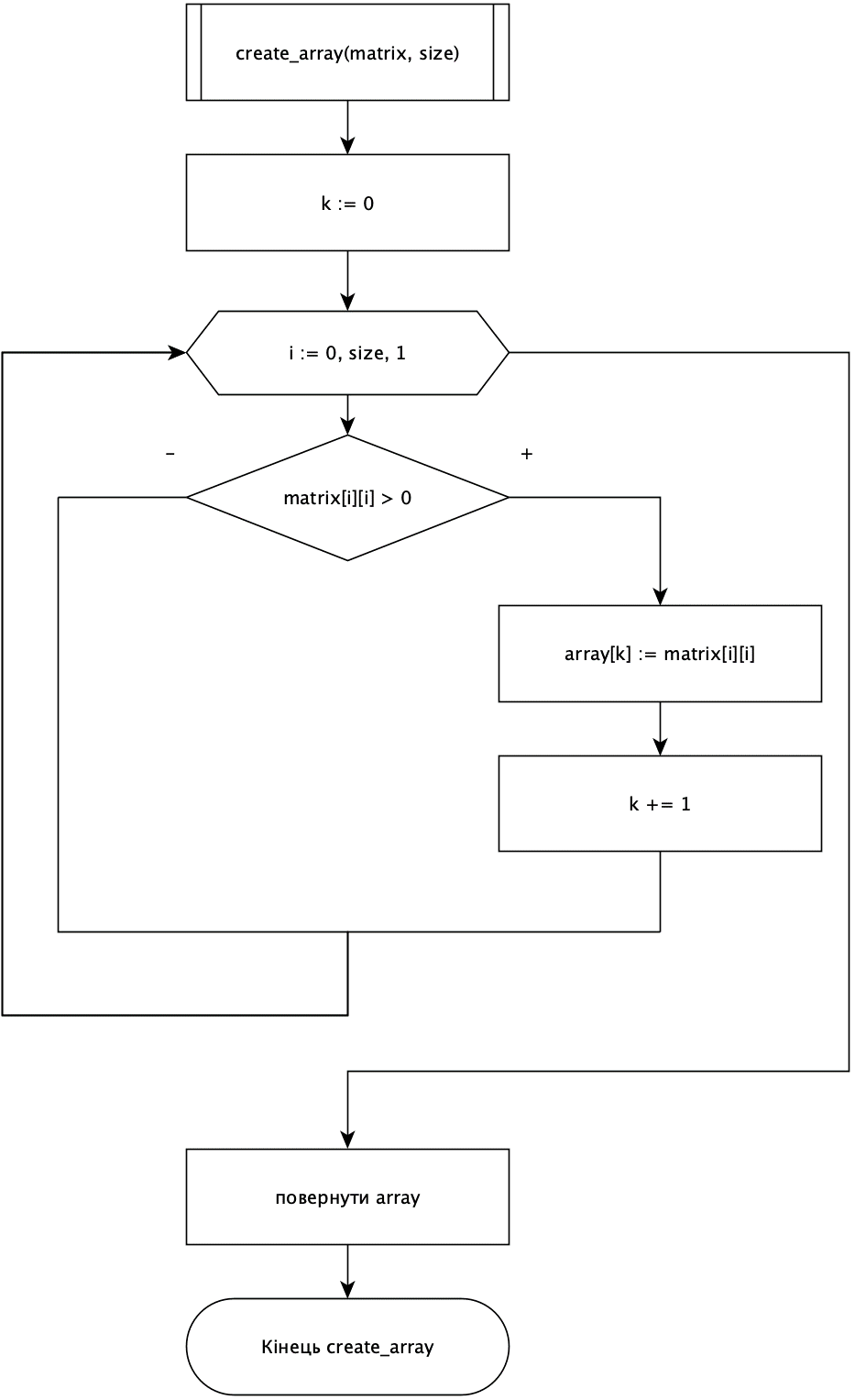
**Блок-схема**

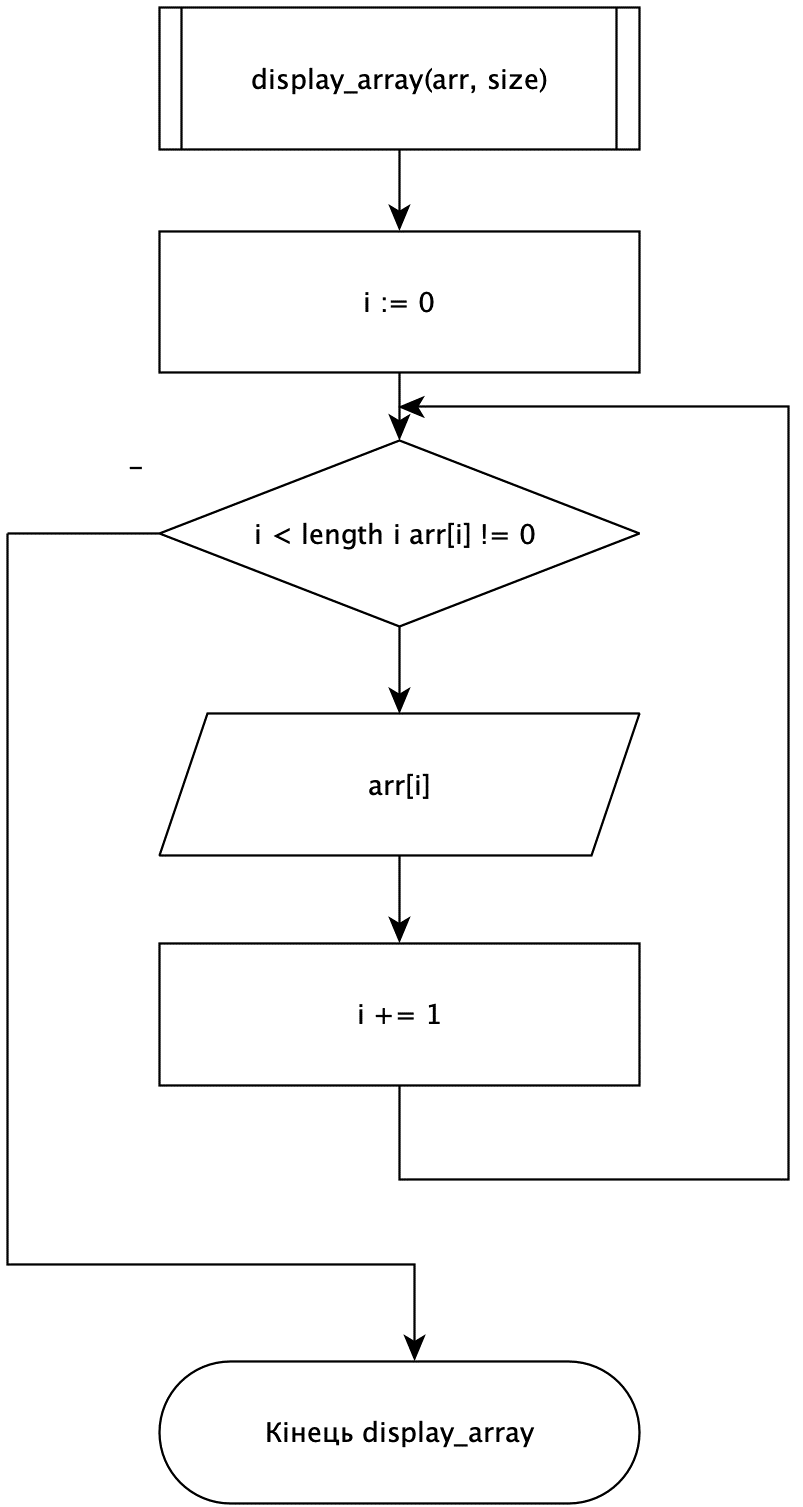
****

**2)**











**Код на С++ :**

#include <iostream>

using namespace std;

int\*\* generate\_matrix(int, int);

void display\_matrix(int\*\*, int);

int\* create\_array(int\*\*, int);

void display\_array(int\*, int);

void sort(int\*, int);

void delete\_matrix(int\*\*, int);

int main() {

int n = 6;

int k = 0;

int\*\* matrix;

int\* array;

matrix = generate\_matrix(n, n);

cout << "Matrix:\n";

display\_matrix(matrix, n);

array = create\_array(matrix, n);

cout << "Array:\n";

display\_array(array, n);

sort(array, n);

cout << "Sorted array:\n";

display\_array(array, n);

delete\_matrix(matrix, n);

delete[] array;

return 0;

}

int\*\* generate\_matrix(int rows, int columns) {

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

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

arr[i] = new int[columns];

}

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

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

arr[i][j] = rand() % 19 - 9;

}

}

return arr;

}

void display\_matrix(int\*\* matrix, int size) {

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

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

cout << matrix[i][j]<<"\t";

}

cout << "\n";

}

}

int\* create\_array(int\*\* matrix, int size) {

int k = 0;

int\* array = new int[size];

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

if (matrix[i][i] > 0) {

array[k] = matrix[i][i];

k++;

}

}

return array;

}

void display\_array(int\* arr, int length) {

int i = 0;

while (arr[i] != 0 && i < length) {

cout << arr[i] << " ";

i++;

}

cout << "\n";

}

void sort(int\* arr, int length) {

int temp;

int i = 0, j;

for (i = 0; i < length - 1; ++i) {

for (j = 0; j < length - 1; ++j) {

if (arr[j + 1] < arr[j]) {

temp = arr[j + 1];

arr[j + 1] = arr[j];

arr[j] = temp;

}

}

}

}

void delete\_matrix(int\*\* matrix, int size) {

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

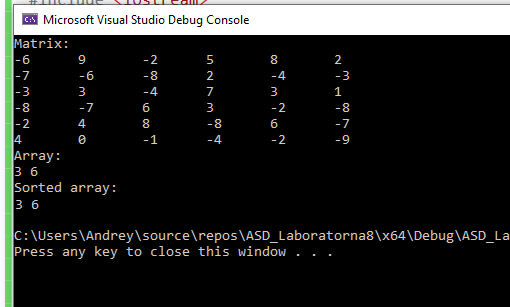
delete[] matrix[i];

}

delete[] matrix;

}

**Скріншот результатів C++ :**

****

**Випробування**

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Виведення: Matrix:  -6 9 -2 5 8 2  -7 -6 -8 2 -4 -3  -3 3 -4 7 3 1  -8 -7 6 3 -2 -8  -2 4 8 -8 6 -7  4 0 -1 -4 -2 -9 |
| 2 | Виведення: Array:  3 6 |
| 3 | Виведення: Sorted array:  3 6 |
|  | Кінець |

**Висновок**

Під час лабораторної роботи я дослідив алгоритми пошуку та сортування й набув практичних навичок їх використання, застосувавши алгоритм пошуку для знаходження всіх додатних елементів головної діагоналі заданої квадратної матриці та алгоритм сортування бульбашкою за зростанням в одновимірному масиві, заповненого цими додатними елементами головної діагоналі, та отримав коректний результат, отже алгоритм працює правильно.