

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 28

Виконав студент ПІ-11 Сідак Кирил Ігорович  
(шифр, прізвище, ім'я, по батькові)

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

## 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:

##### Варіант 28

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).

28	5 x 5	Цілий	Із додатних значень елементів головної діагоналі двовимірного масиву. Відсортувати обміном за спаданням.
----	-------	-------	--

#### Постановка задачі

За допомогою підпрограми треба створити двовимірний масив цілих чисел заданої розмірності (5 x 5) 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 за спаданням. Отриманий масив і буде шуканим.

## Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Ім'я	Призначення
Задана розмірність двовимірного масиву	цілий	n	Вхідне дане
Заданий двовимірний масив	цілий	matrix	Вхідне дане
Шуканий одновимірний масив	цілий	array	Результат
Індекс одновимірного, на який треба поставити поточний додатній елемент головної діагоналі двовимірного масиву	цілий	k	Проміжне дане(змінна процедури)
Зміна для зберігання поточного найбільшого елемента одновимірного масиву	цілий	temp_num	Проміжне дане(змінна процедури)

Таким чином, формування задачі зводиться до створення динамічного двовимірного масиву `matrix` заданої розмірності(5x5) у підпрограмі `generate_matrix` та заповнення його випадковими елементами з діапазону від -9 до 9. За допомогою підпрограми `display_matrix` виведемо цей масив. Потім у підпрограмі `create_array` створимо одновимірний динамічний масив `array` та, використавши 2 арифметичні цикли та умовну форму оператора вибору, заповнимо його додатними елементами головної діагоналі `matrix`. У підпрограмі `sort_descending` відсортуємо масив `array` за спаданням елементів, використовуючи алгоритм сортування обміном. За допомогою підпрограми `display_array` виведемо `array` до сортування та після нього.

## Розв'язання

Програмні специфікації запишемо у псевдокодi та графічній формi 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 спаданням 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 i 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 i 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 i 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 i arr[i] != 0 повторити**

**виведення** arr[i]

i += 1

**все повторити**

**Кінець display\_array**

**sort\_descending(arr, length)**

i := 0

**поки i < length - 1 i arr[i] != 0 повторити**

j := 0

**поки j < length - 1 - i i arr[i] != 0 повторити**

**якщо** arr[j+1] > arr[j]

**то**

temp\_num := arr[j]

arr[j] := arr[j+1]

arr[j+1] := temp\_num

**все якщо**

j += 1

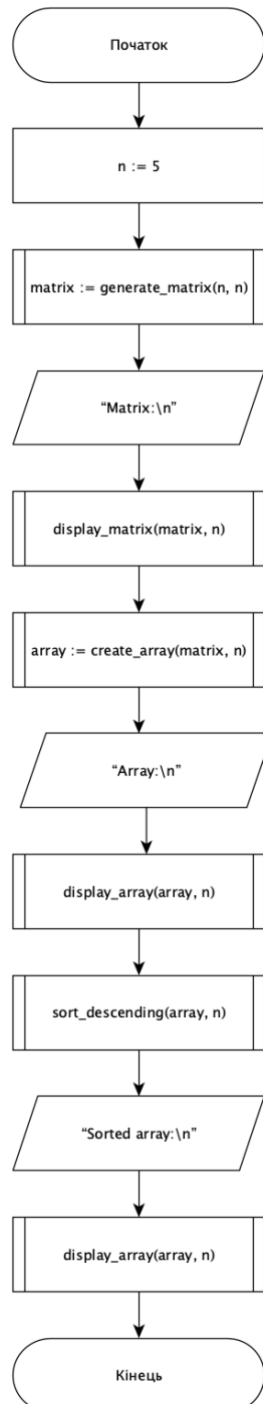
**все повторити**

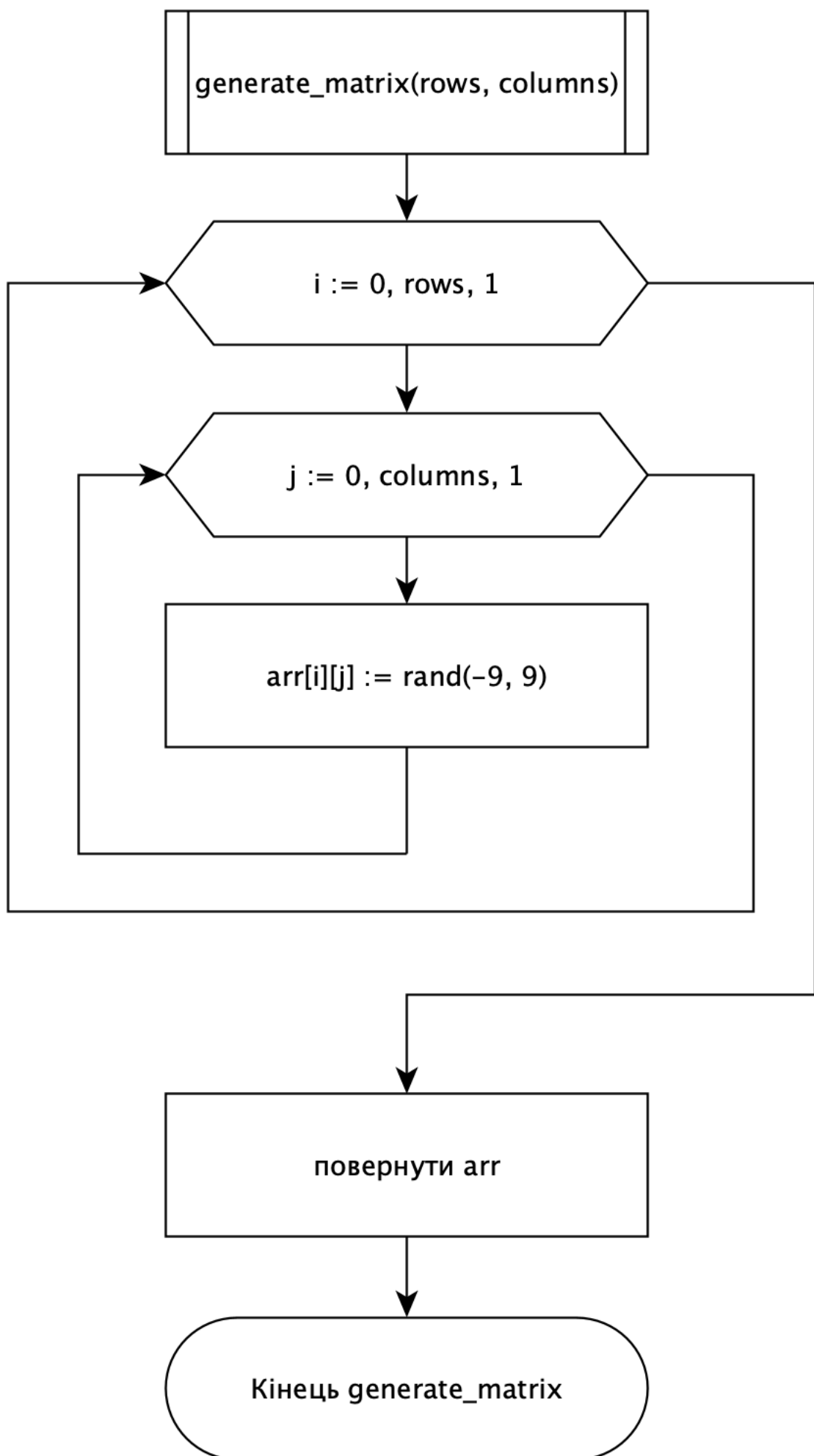
i += 1

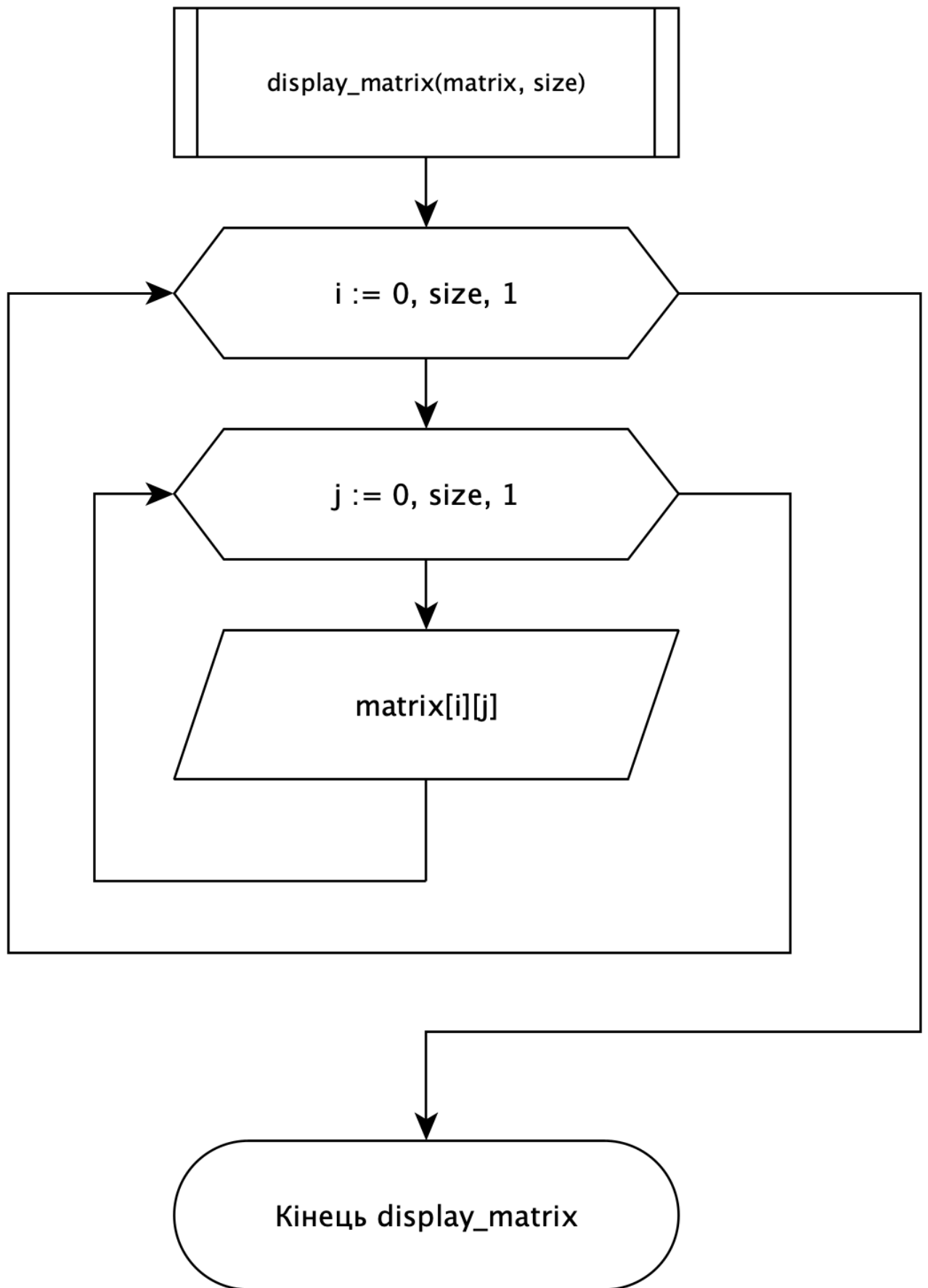
**все повторити**

**Кінець sort\_descending**

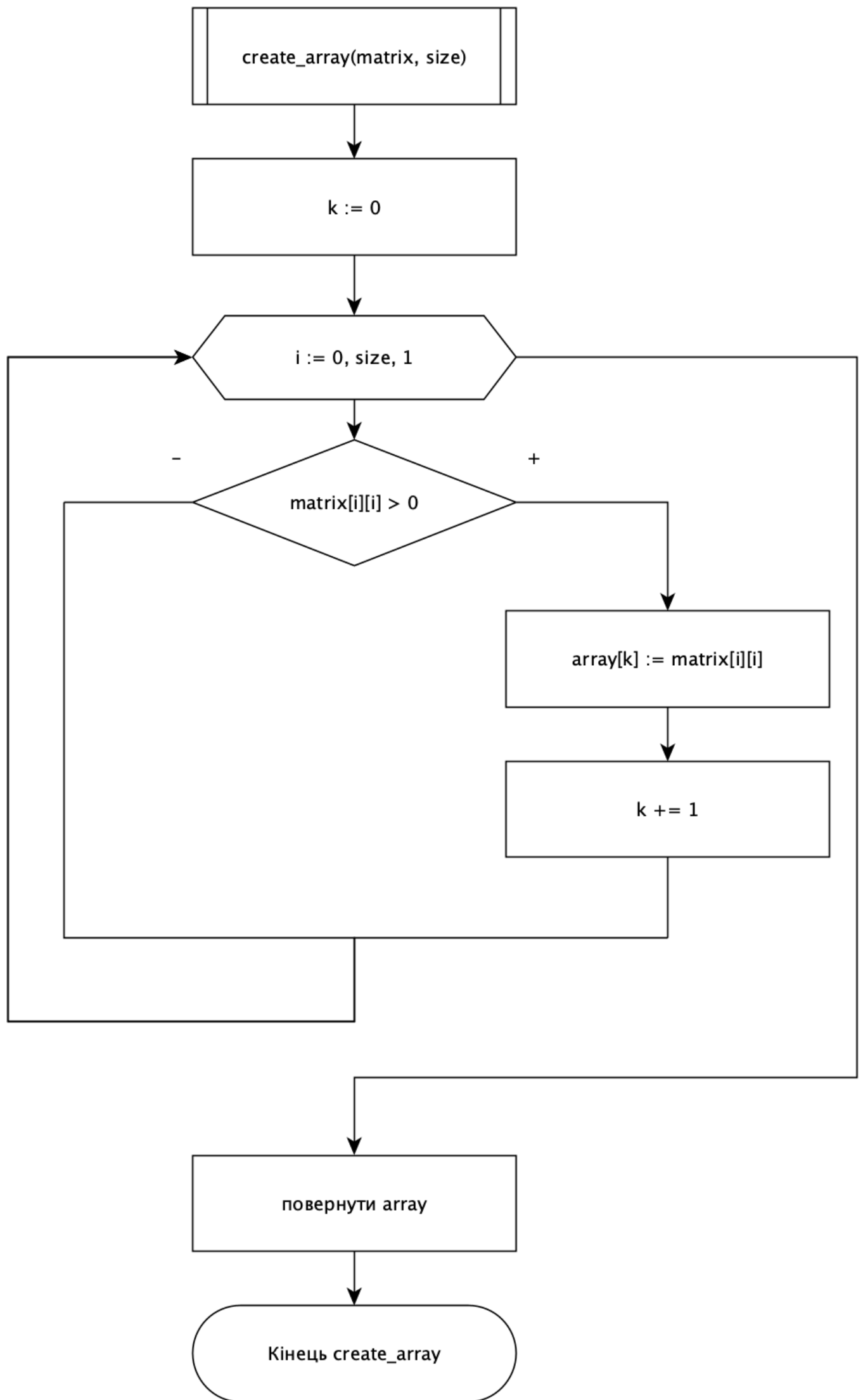
## Блок-схема

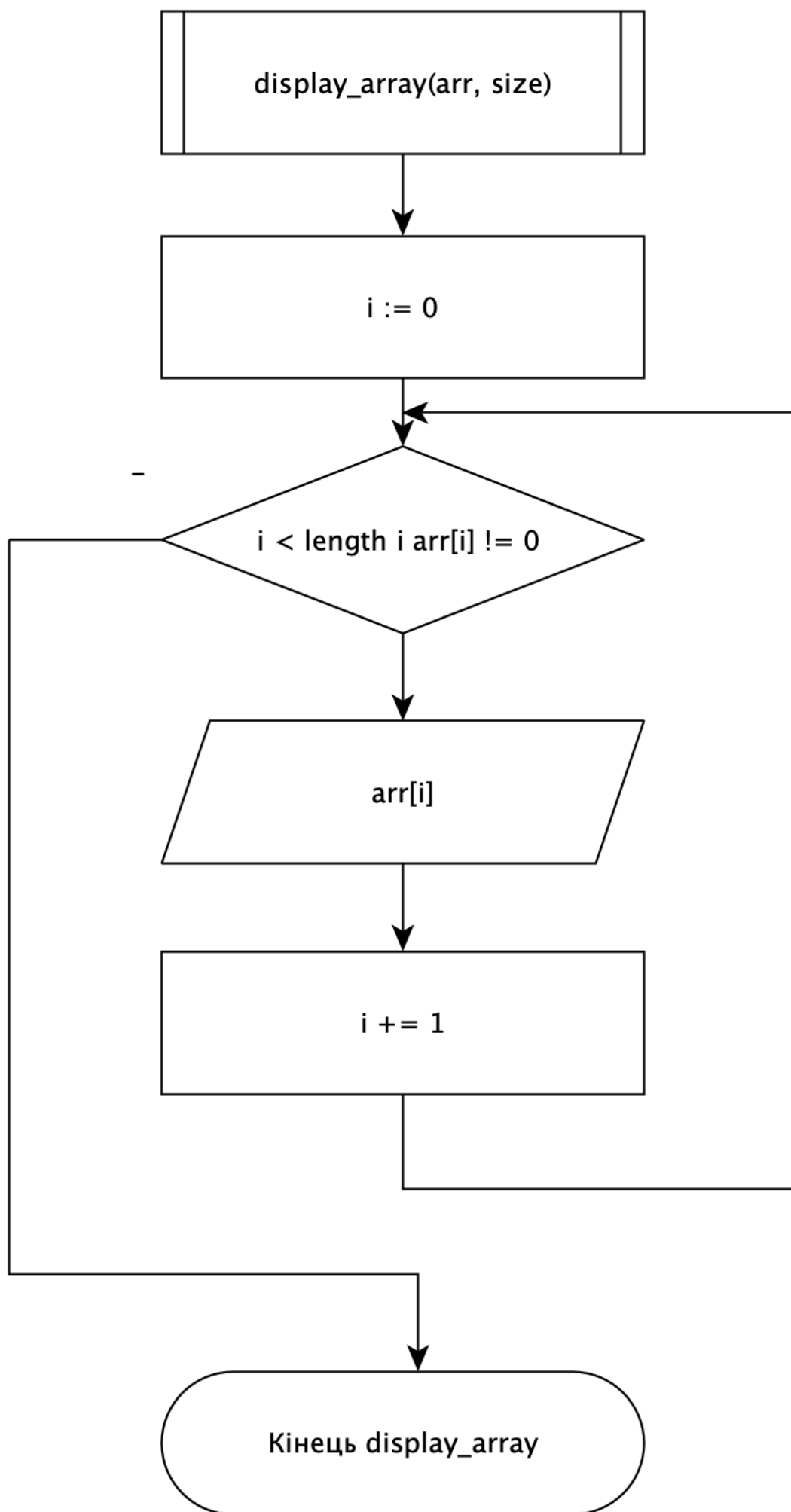


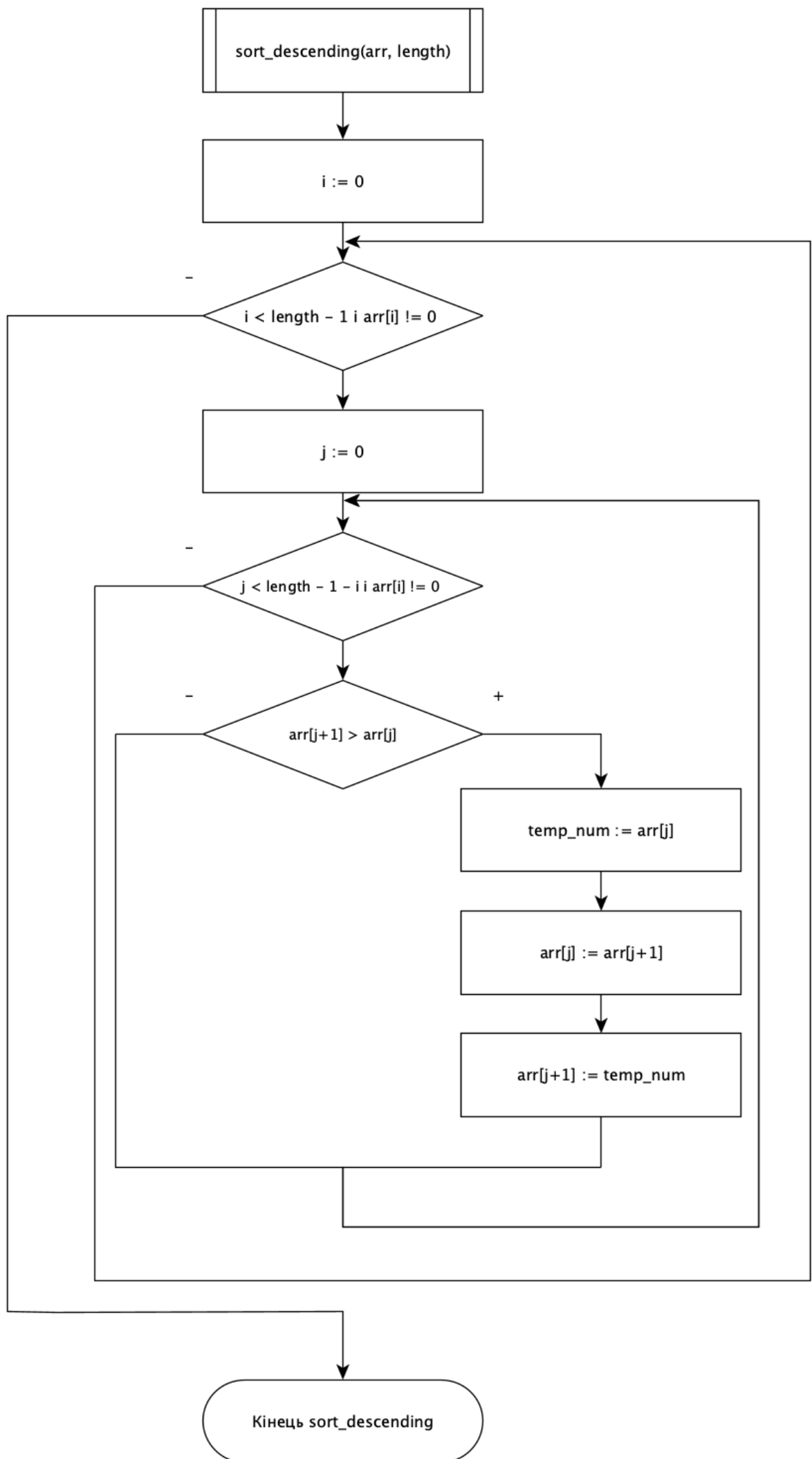












## Програма на C++

```
#include <iostream>
#include <ctime>
#include <iomanip>
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_descending(int*, int);
void delete_matrix(int**, int);

int main() {
    int n = 5;
    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_descending(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];
    }
    srand(time(NULL));
    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 << setw(4) << matrix[i][j];
        }
        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_descending(int* arr, int length) {
    int temp_num;
    int i = 0, j;
    while(i < length - 1 && arr[i] != 0) {
        j = 0;
        while (j < length - 1 - i && arr[j] != 0) {
            if (arr[j+1] > arr[j]) {
                temp_num = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp_num;
            }
            j++;
        }
        i++;
    }
}

void delete_matrix(int** matrix, int size) {
    for (int i = 0; i < size; i++) {
        delete[] matrix[i];
    }
    delete[] matrix;
}

```

```

Run: Lab8_ASD x
/Users/kyryl/Desktop/Lab8_ASD/cmake-build-debug/Lab8_ASD
Matrix:
3 -1 -3 3 7
1 7 -3 6 -5
-1 0 6 -4 4
7 -4 8 8 8
2 4 -4 8 -7
Array:
3 7 6 8
Sorted array:
8 7 6 3
Process finished with exit code 0

```

## Випробування алгоритму

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

### Висновок

Отже, 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 обміном(бульбашкою) за спаданням в одновимірному масиві, заповненого цими додатними елементами головної діагоналі, та отримав коректний результат.