Алгоритми та структури даних. Основи алгоритмізації

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 28

Виконав студент: ІП-15 Рибаков Дмитро Вадимович

Перевірив: Вечерковська Анастасія Сергіївна

Київ 2021

Алгоритми та структури даних. Основи алгоритмізації

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

Постановка задачі

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

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом.

2. Ініціювання змінної, що описана в завданні.

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

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

Побудова математичної моделі

Змінна	Тип	Ім’я	Призначення
Розмірність матриці	Цілий	n	Початкові дані
Згенерований двовимірний масив(матриця)	Цілий	matrix	Початкові дані
Змінна зберігаюча масив як проміжні дані	Цілий	arr	Проміжні дані
Змінна зберігаюча найбільші значення одновимірного масиву	Цілий	a	Проміжні дані
Індекс нового масиву	Цілий	b	Проміжні дані
Крок циклу. Індекс числа у масиві	Цілий	i	Проміжні дані
Крок циклу. Індекс числа у масиві	Цілий	j	Проміжні дані
Відсортований обміном за спаданням масив. Результат	Цілий	array	Вихідні дані

За умовою розмірність двовимірного масиву 5*5, тому покладемо n = 5. Згенеруємо двовимірний масив випадковими числами за допомогою функції rand(). Створимо новий одновимірний масив із додатних значень елементів головної діагоналі двовимірного масиву. Відсортуємо обміном за спаданням та виведимо результат.

У роботі використовуються наступні дії:

«=» - операція присвоєння;

«>» - більше (більше ніж);

«<» - менше (менше ніж);

«&&» - - і (логічне множення);

«a[x]» - масив a, де x – кількість(або змінна) змінних у масиві;

«++» - збільшення значення на 1.

Функція rand() генерує випадкове число.

Розв’язання

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

Крок 1. Визначимо основні дії.

Крок 2. Опишемо змінної індексованого типу (двовимірний масив).

Крок 3. Деталізуємо дію створення нової змінної індексованого типу (одновимірний масив) та її ініціювання значеннями із додатних значень елементів головної діагоналі двовимірного масиву.

Крок 4. Деталізуємо дію сортування обміном за спаданням нової змінної та виведемо результат.

Псевдокод

Основна програма:

крок 1

початок

згенерація двовимірного масиву

створення нової змінної індексованого типу та ініціювання значеннями із додатних значень елементів головної діагоналі двовимірного масиву

сортування обміном за спаданням нової змінної

виведення результату

кінець

крок 2

початок

n = 5

matrix = generate_matrix(n)

створення нової змінної індексованого типу та ініціювання значеннями із додатних значень елементів головної діагоналі двовимірного масиву

сортування обміном за спаданням нової змінної

виведення результату

кінець

крок 3

початок

n = 5

matrix = generate_matrix(n)

array = bild_array(matrix, n)

сортування обміном за спаданням нової змінної

виведення результату

кінець

крок 4

початок

n = 5

matrix = generate_matrix(n)

array = bild_array(matrix, n)

decrease_array(array, n)

write_array(array)

кінець

Підпрограми

generate_matrix(n)

початок

для i від 0 до n повторити

для j від 0 до n повторити

matrix[i][j] = rand() **від -9 до 9**

все повторити

все повторити

```
    повернути matrix
кінець

bild_array(matrix, n)
початок
    b = 0
    для і від 0 до n повторити
        якщо matrix[i][i] > 0 то
            array[b] = matrix[i][i]
            b++
        все якщо
    все повторити
    повернути array
кінець

decrease_array(array, n)
початок
    a = 0
    i = 0
    поки i < n – 1 && arr[i] > 0
        j = 0
        поки j < n – 1 - i && arr[i] > 0
            якщо arr[j + 1] > arr[j] то
                a = arr[j]
                arr[j] = arr[j + 1]
                arr[j] = arr[j + 1] = a
            все якщо
            j++
        все повторити
        i++
    все повторити
кінець
```

write_array(array)

початок

i = 0

поки arr[i] > 0 && i < n

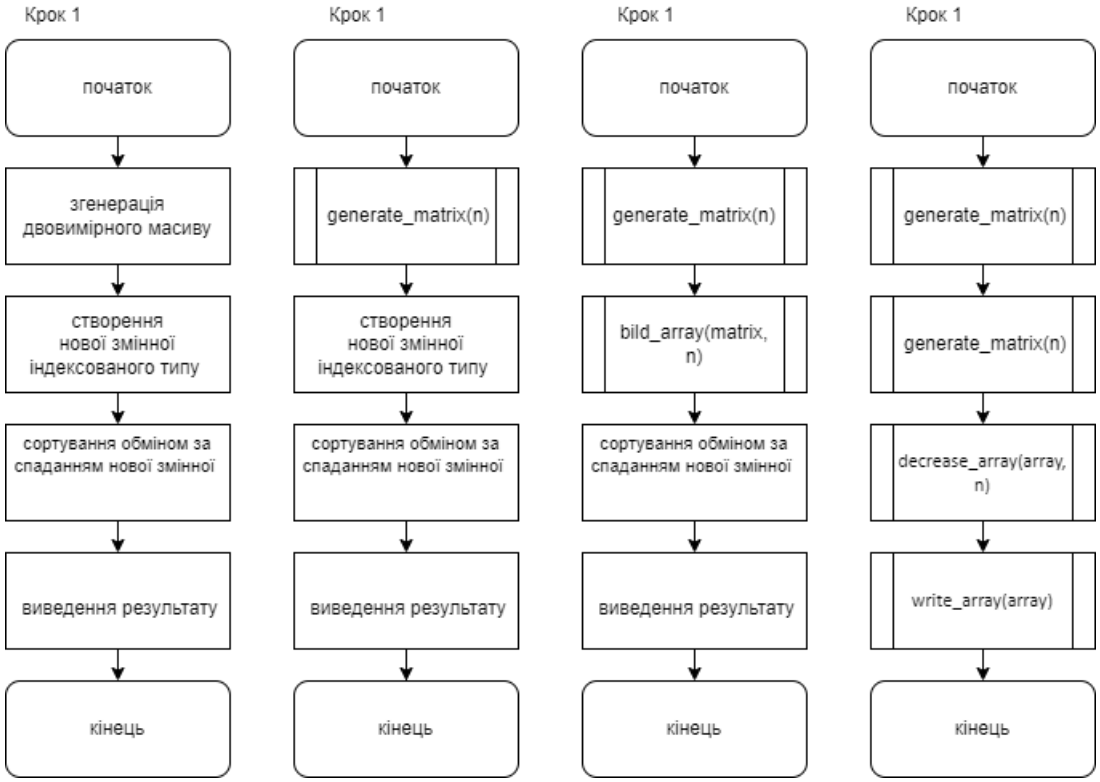
вивід arr[i]

i++

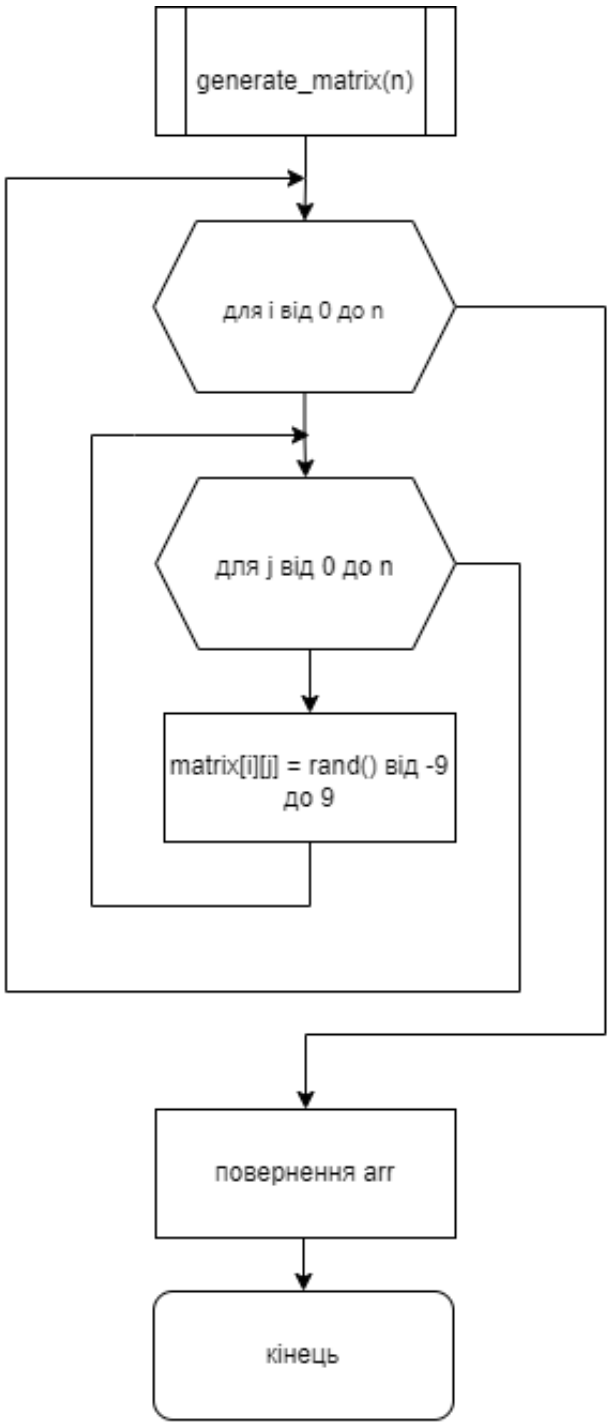
все повторити

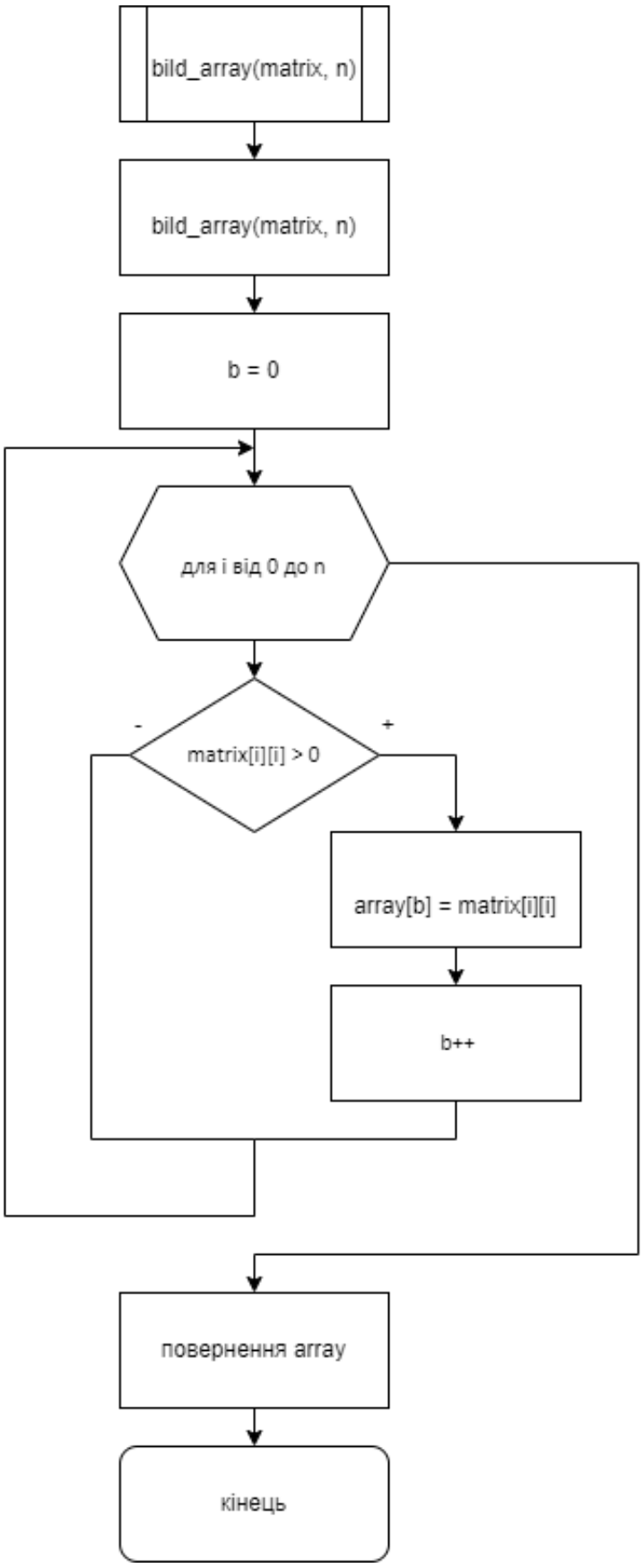
кінець

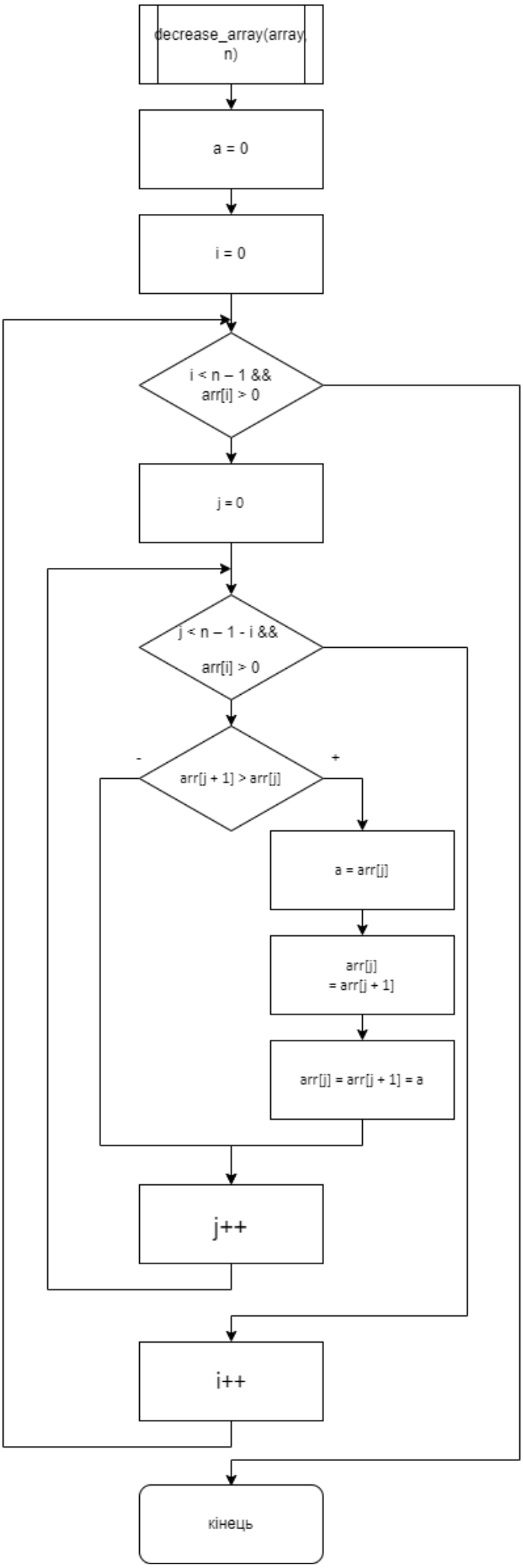
Блок-схема

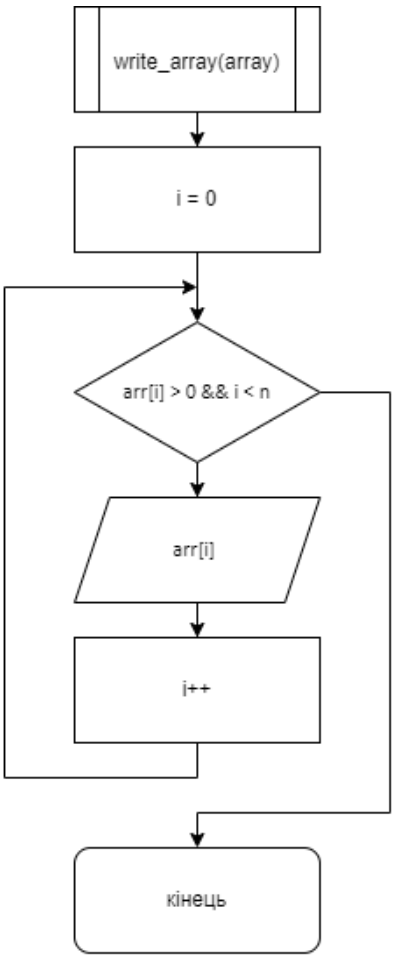


Підпрограми






decrease_array(array, n)
a = 0
i = 0
i < n – 1 && arr[i] > 0
j = 0
j < n – 1 - i && arr[i] > 0
-
+
arr[j + 1] > arr[j]
a = arr[j]
arr[j] = arr[j + 1]
arr[j] = arr[j + 1] = a
j++
i++
кінець


write_array(array)
i = 0
arr[i] > 0 && i < n
arr[i]
i++
кінець

Код програми

```cpp
#include <iostream>
#include <iomanip>
#include <ctime>
using namespace std;
int** generate_matrix(int n) {
    int** matrix = new int* [n];
    for (int i = 0; i < n; i++) {
        matrix[i] = new int[n];
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            matrix[i][j] = rand() % 19 - 9;
        }
    }
    return matrix;
}
void write_matrix(int** matrix, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << setw(4) << matrix[i][j];
        }
        cout << endl;
    }
}
int* bild_array(int** matrix, int n) {
    int b = 0;
    int* array = new int[n];
    for (int i = 0; i < n; i++) {
        if (matrix[i][i] > 0) {
            array[b] = matrix[i][i];
            b++;
        }
    }
    return array;
}
void  write_array(int* arr, int n) {
    int i = 0;
    while (arr[i] > 0 && i < n) {
        cout << arr[i] << " ";
        i++;
    }
    cout << endl;
}
void decrease_array(int* arr, int n) {
```

```cpp
void decrease_array(int* arr, int n) {
    int a;
    int i = 0, j;
    while (i < n - 1 && arr[i] > 0) {
        j = 0;
        while (j < n - 1 - i && arr[j] > 0) {
            if (arr[j + 1] > arr[j]) {
                a = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = a;
            }
            j++;
        }
        i++;
    }
}
void delete_matrix(int** matrix, int n) {
    for (int i = 0; i < n; i++) {
        delete[] matrix[i];
    }
    delete[] matrix;
}

int main() {

    setlocale(LC_ALL, "Ukrainian");
    srand(time(NULL));
    int n = 5;
    int* array;
    int** matrix;
    matrix = generate_matrix(n);
    cout << "Початкова матриця:" << endl;
    write_matrix(matrix, n);
    array = bild_array(matrix, n);
    cout << "Масив додатних значень елементів головної діагоналі матриці:" << endl;
    write_array(array, n);
    decrease_array(array, n);
    cout << "Відсортований обміном за спаданням масив:" << endl;
    write_array(array, n);
    delete_matrix(matrix, n);
    delete[] array;

    return 0;
}
```


```cpp
#include <iostream>
#include <iomanip>
#include <ctime>
using namespace std;
int** generate_matrix(int n) {
    int** matrix = new int* [n];
    for (int i = 0; i < n; i++) {
        matrix[i] = new int[n];
```

```
	}
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < n; j++) {
			matrix[i][j] = rand() % 19 - 9;
		}
	}
	return matrix;
}
void write_matrix(int** matrix, int n) {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < n; j++) {
			cout << setw(4) << matrix[i][j];
		}
		cout << endl;
	}
}
int* bild_array(int** matrix, int n) {
	int b = 0;
	int* array = new int[n];
	for (int i = 0; i < n; i++) {
		if (matrix[i][i] > 0) {
			array[b] = matrix[i][i];
			b++;
		}
	}
	return array;
}
void  write_array(int* arr, int n) {
	int i = 0;
	while (arr[i] > 0 && i < n) {
		cout << arr[i] << " ";
		i++;
	}
	cout << endl;
}
void decrease_array(int* arr, int n) {
	int a;
	int i = 0, j;
	while (i < n - 1 && arr[i] > 0) {
		j = 0;
		while (j < n - 1 - i && arr[j] > 0) {
			if (arr[j + 1] > arr[j]) {
				a = arr[j];
				arr[j] = arr[j + 1];
				arr[j + 1] = a;
			}
			j++;
		}
		i++;
	}
}
void delete_matrix(int** matrix, int n) {
	for (int i = 0; i < n; i++) {
		delete[] matrix[i];
	}
	delete[] matrix;
}

int main() {

	setlocale(LC_ALL, "Ukrainian");
	srand(time(NULL));
	int n = 5;
	int* array;
	int** matrix;
	matrix = generate_matrix(n);
	cout << "Початкова матриця:" << endl;
	write_matrix(matrix, n);
	array = bild_array(matrix, n);
	cout << "Масив додатних значень елементів головної діагоналі матриці:" << endl;
```

```
        write_array(array, n);
        decrease_array(array, n);
        cout << "Відсортований обміном за спаданням масив:" << endl;
        write_array(array, n);
        delete_matrix(matrix, n);
        delete[] array;

        return 0;
}
```

Тестування програми

Висновки

На цій лабораторній роботі ми дослідили алгоритми пошуку та сортування, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій. В результаті виконання лабораторної роботи ми отримали алгоритм сортування обміном за спаданням, при цьому розділили виконання задачи на 4 кроки. В процесі випробування ми розглянули декілька випадків результатом яких отримали масиви зі спаданням чисел.