

# А л г о р и т м и   т а   с т р у к т у р и   д а н и х .

---

## О с н о в и   а л г о р и т м і з а ц і ї

*Додаток 1*

Міністерство освіти і науки України  
Національний технічний університет України «Київський політехнічний  
інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації  
і управління

Звіт

з лабораторної роботи №9  
з дисципліни «Алгоритми та структури даних-1.  
Основи алгоритмізації»  
«Дослідження алгоритмів обходу масивів»  
Варіант 6

Виконав  
студент

ІП-15, Волинець Кирило Михайлович

(шифр, прізвище, ім'я, по батькові)

Перевірів

Вечерковська Анастасія Сергіївна

( прізвище, ім'я, по батькові)

# А л г о р и т м и   т а   с т р у к т у р и   д а н и х .

## О с н о в и   а л г о р и т м і з а ц і ї

### Лабораторна робота 9

#### Дослідження алгоритмів обходу масивів

**Мета** - дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

### Індивідуальне завдання

Варіант 6

|   |  |
|---|--|
| 6 | Задано матрицю дійсних чисел $A[m,n]$ . При обході матриці по стовпчиках знайти в ній перший мінімальний елемент і його місцезнаходження. Обміняти знайдене значення $X$ з елементом побічної діагоналі. |
|---|--|

### Побудова математичної моделі.

Складемо таблицю імен змінних.

| Змінна                                 | Тип                     | Ім'я                     | Призначення                 |
|--|-------------------------|--------------------------|-----------------------------|
| Кількість рядків і стовбців            | Цілочисельний           | size                     | Вхідні                      |
| Вхідна матриця                         | Дійсний[size][size]     | matrix                   | Вхідні, проміжні, результат |
| Стовбець                               | Дійсний[size]           | column                   | Проміжні                    |
| Позиція мінімального                   | Цілочисельний           | min_index                | Проміжні                    |
| Ітератор                               | Цілочисельний           | i                        | Проміжні                    |
| Тимчасова змінна                       | Дійсний                 | temp                     | Проміжні                    |
| Знаходження позиції мінімального       | Цілочисельний (функція) | min_pos(double*, int)    | Обчислення                  |
| Визначення стовбцю за індексом         | Дійсний[] (функція)     | get_column(double*, int) | Обчислення                  |
| Поміняти значення двох змінних місцями | Пустий (функція)        | swap(double*, double*)   | Обчислення                  |

Створимо квадратну матрицю розміром size та заповнимо її числами (у програмі генеруються випадкові числа). Потім перебираємо кожний стовбець, знаходячи мінімальний його елемент міняємо його місцями з елементом головної діагоналі цього стовбцю.

### Розв'язання

Програмні специфікації запишемо у псевдокодi та графічній формi у вигляді блок-схеми.

### Псевдокод

#### Основна програма:

**початок**

**ввести** size

**ввести** matrix

**повторити** для i від 0 до i < size:

        column = **get\_column**(matrix, size, i)

        min\_index = **min\_pos**(column, size)

**swap**(&matrix[min\_index][i], &matrix[i][i])

**все повторити**

**вивести** matrix

**кінець**

#### Підпрограми:

**get\_column**(double\*\* matrix, int y\_size, int column\_index)

**повторити** для i від 0 до i < y\_size:

        column[i] = matrix[i][column\_index]

**все повторити**

**повернути** column

**кінець** get\_column

**min\_pos**(double\* array, int size)

    min\_index = 0

**повторити** для i від 1 до i < size:

**якщо** array[i] < array[min\_index] **то** min\_index = i

**все повторити**

**повернути** min\_index

**кінець** min\_pos

**swap** (double\* element1, double\* element2)

    temp = \*element1

    \*element1 = \*element2

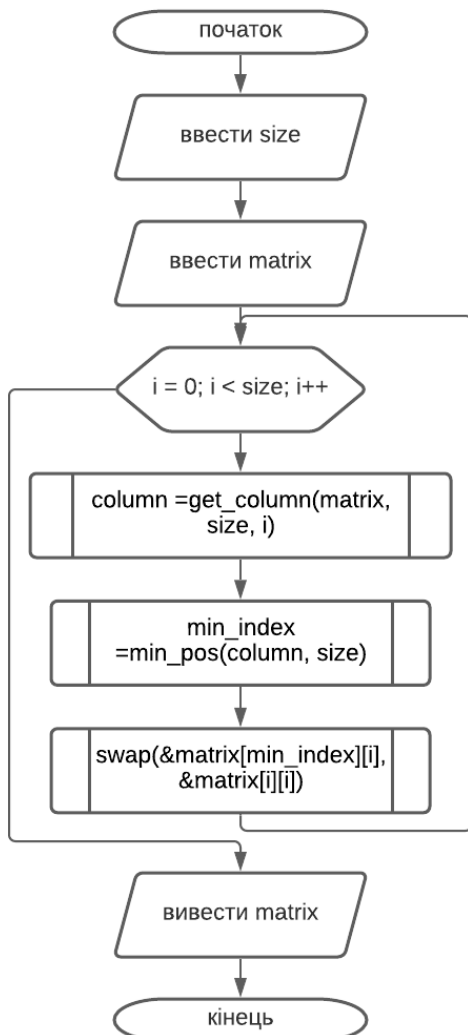
    \*element2 = temp

**кінець** swap

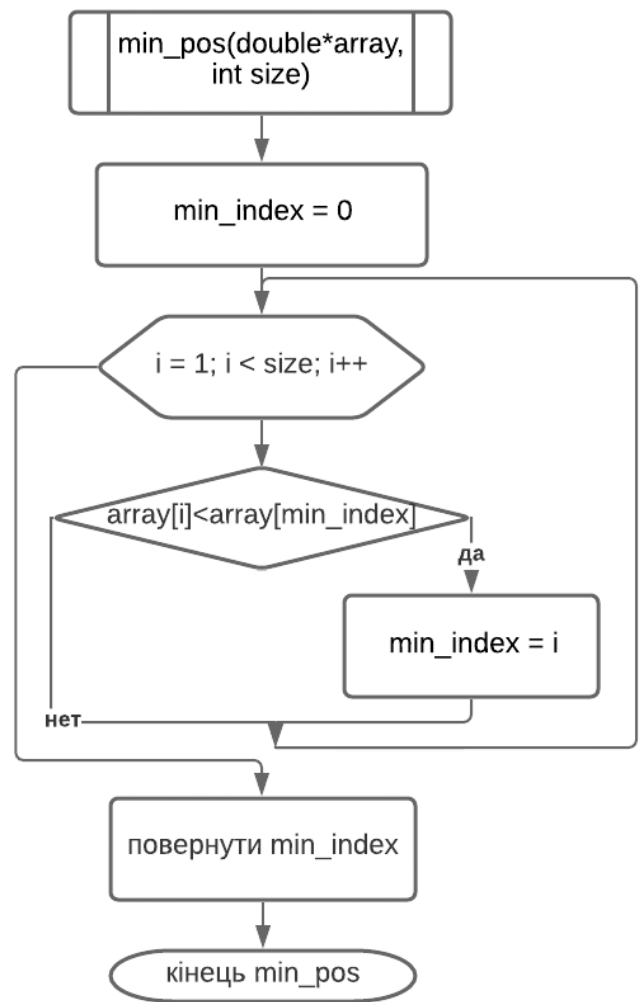
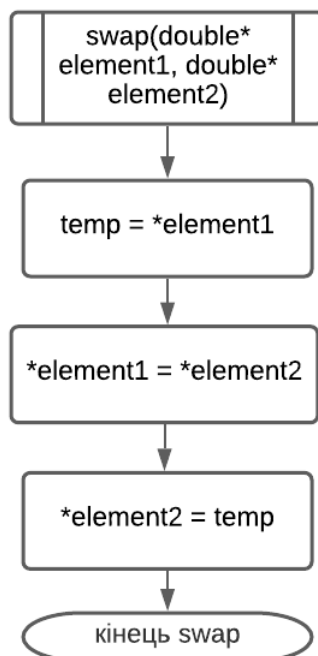
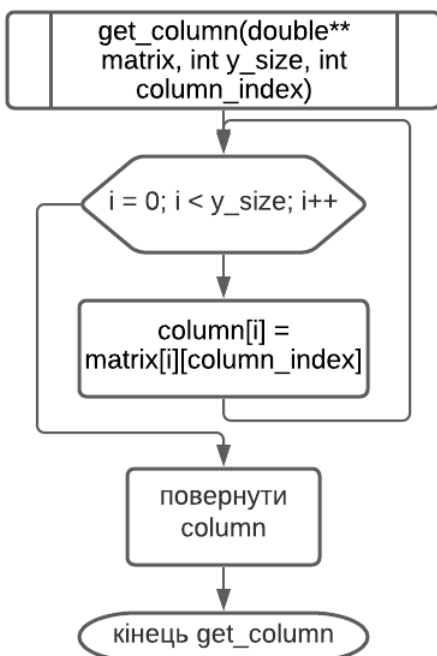
# Алгоритми та структури даних.

## Основи алгоритмізації

### Блок-схема:



### Підпрограми:



# Алгоритми та структури даних.

## Основи алгоритмізації

### Код

```
#include <iostream>
#include <random>
#include <ctime>
#include <cmath>
#include <iomanip>

using namespace std;

const int base_y_size = 7, base_x_size = 5;

double max(double* array, int size);
void inverted_bubble_sort(double* array, int size);

void generate_random_values(double** matrix, int y_size, int x_size, double min_value, double max_value);
void print(double** matrix, int y_size, int x_size);
void print(double* array, int size);

int main()
{
    double** matrix = new double* [base_y_size];
    for (int i = 0; i < base_y_size; i++) {
        matrix[i] = new double[base_x_size];
    }
    generate_random_values(matrix, base_y_size, base_x_size, -50, 110);
    print(matrix, base_y_size, base_x_size);
    double* result_array = new double[base_y_size];
    for (int i = 0; i < base_y_size; i++) {
        result_array[i] = max(matrix[i], base_x_size);
    }
    print(result_array, base_y_size);
    inverted_bubble_sort(result_array, base_y_size);
    print(result_array, base_y_size);
}

void generate_random_values(double** matrix, int y_size, int x_size, double min_value, double max_value) {
    srand(time(NULL) * 100);
    //srand(229);
    double random_zeroone_number;
    for (int i = 0; i < y_size; i++) {
        for (int j = 0; j < x_size; j++) {
            random_zeroone_number = double(rand()) / RAND_MAX;
            matrix[i][j] = random_zeroone_number * (max_value - min_value) + min_value;
        }
    }
}

double max(double* array, int size) {
    double maximum = array[0];
    for (int i = 1; i < size; i++) {
        if (array[i] > maximum) maximum = array[i];
    }
    return maximum;
}

void inverted_bubble_sort(double* array, int size) {
    double temp;
    int num_of_operations, pass = 1;
    do{
        num_of_operations = 0;
        for (int i = 0; i < size - pass; i++) {
            if (array[i] < array[i + 1]) {
                temp = array[i];
```

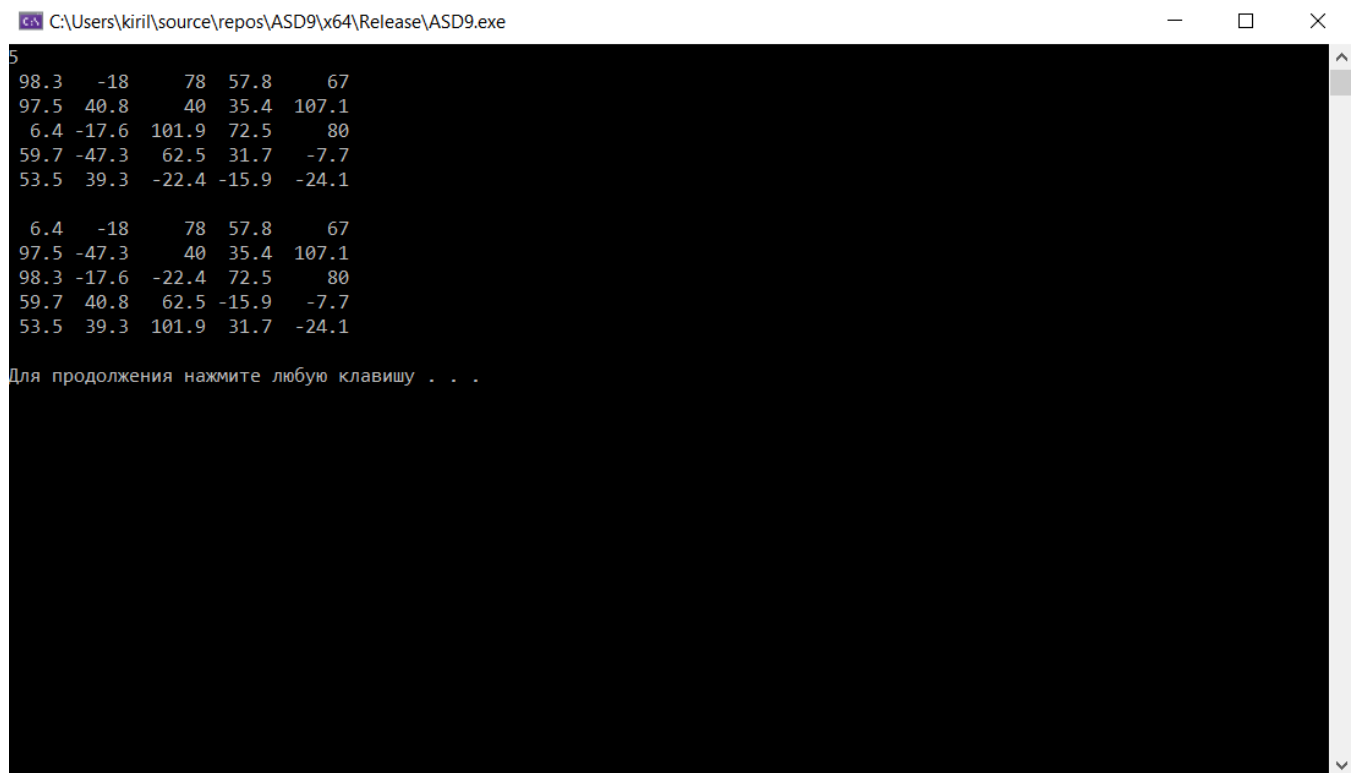
# Алгоритми та структури даних.

## Основи алгоритмізації

```
        array[i] = array[i + 1];
        array[i + 1] = temp;
        num_of_operations++;
    }
}
pass++;
} while (num_of_operations != 0);
}

void print(double* array, int size) {
    for (int i = 0; i < size; i++) {
        cout << round(array[i]*10)/10 << " ";
    }
    cout << endl;
}

void print(double** matrix, int y_size, int x_size) {
    int max_column_length = 0, curr_len;
    int* column_lengths = new int[x_size];
    for (int i = 0; i < x_size; i++) {
        max_column_length = 0;
        for (int j = 0; j < y_size; j++) {
            curr_len = ceil(log10(fabs(matrix[j][i])));
            if (curr_len > max_column_length) max_column_length = curr_len;
        }
        column_lengths[i] = max_column_length;
    }
    for (int i = 0; i < y_size; i++) {
        for (int j = 0; j < x_size; j++) {
            cout << setw(column_lengths[j]+3) << round(matrix[i][j]*10)/10 << " ";
        }
        cout << endl;
    }
    cout << endl;
}
```



```
C:\Users\kiril\source\repos\ASD9\x64\Release\ASD9.exe
5
98.3  -18   78  57.8   67
97.5  40.8  40  35.4  107.1
 6.4 -17.6 101.9  72.5   80
59.7 -47.3  62.5  31.7  -7.7
53.5  39.3 -22.4 -15.9 -24.1

 6.4  -18   78  57.8   67
97.5 -47.3  40  35.4  107.1
98.3 -17.6 -22.4  72.5   80
59.7  40.8  62.5 -15.9  -7.7
53.5  39.3 101.9  31.7 -24.1

Для продолжения нажмите любую клавишу . . .
```

### **Висновки**

Ми дослідили алгоритми обходу масивів, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій. В результаті виконання лабораторної роботи ми отримали алгоритм для знаходження мінімального значення стовбця та зміни місцями його значення з елементом головної діагоналі.