МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра систем управління літальних апаратів

Лабораторна робота № 7

з дисципліни «Алгоритмізація та програмування»
на тему «Реалізація алгоритмів обробки двовимірних масивів мовою С ++»

ХАІ.301. 175. 318. 08 ЛР

Виконав студент гр. 318

Каріна ГЛЄБОВА
(підпис, дата) (П.І.Б.)

Перевірив
к.т.н., доц. Олена ГАВРИЛЕНКО
(підпис, дата) (П.І.Б.)

2025

МЕТА РОБОТИ

Вивчити теоретичний матеріал з основ представлення двовимірних масивів (матриць) у мові С ++ і реалізувати декларацію, введення з консолі, обробку і виведення в консоль матриць мовою С ++ в середовищі Visual Studio.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Вирішити завдання на аналіз і виведення елементів матриці. Введення і виведення здійснити в командному вікні.

Matrix 44. Дана матриця розміру M * N. Знайти мінімальний серед елементів тих рядків, які впорядковані або по зростанню, або по спадаючій. Якщо впорядковані рядки в матриці відсутні, то вивести 0.

Завдання 2. Перетворити матрицю відповідно до свого варіанту завдання, розмір матриці і її елементи ввести з консолі. Вивести результати у консоль.

Matrix 71. Дана матриця розміру M * N. Продублювати стовпець матриці, що містить її мінімальний елемент.

ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі Matrix 44.

Вхідні дані (ім’я, опис, тип, обмеження):

const M = N = 20 – максимальний розмір матриці, ціле, константа.

row – кількість рядків, ціле, 2-20.

col – кількість стовпців, ціле, 2-20

matr1 – цільночисельний двовимірний масив.

Вихідні дані (ім’я, опис, тип):

isSortedAscending – логічний тип, впорядкований за зростанням рядок.

isSortedDescending – логічний тип, впорядкований за спаданням рядок.

min_research – ціле число, мінімальний елемент серед впорядкованих по зростанню та спаданню рядків.

Лістинг коду вирішення задачі Matrix 44 наведено в дод. А (стор. 5-9).

Екран роботи програми показаний на рис. Б.1. (дод. Б, стор. 10-13)

Приклад діаграми для завдання Matrix 44 наведено на рис. Б.3. (дод. Б, стор. 10-13)

Завдання 2.

Вирішення задачі Matrix 71.

Вхідні дані (ім’я, опис, тип, обмеження):

const M = N = 20 – максимальний розмір матриці, ціле, константа.

row – кількість рядків, ціле, 2-20.

col – кількість стовпців, ціле, 2-20

matr2 – цільночисельний двовимірний масив.

Вихідні дані (ім’я, опис, тип):

duplicate_min_column – ціле число, стовпець з найменшим елементам.

Лістинг коду вирішення задачі Matrix 71 наведено в дод. А (стор. 5-9).

Екран роботи програми показаний на рис. Б.2. (дод. Б, стор. 10-13)

Приклад діаграми для завдання Matrix 71 наведено на рис. Б.4. (дод. Б, стор. 10-13)

Завдання 3.

Організація меню.

Вхідні дані (ім’я, опис, тип, обмеження):

«Номер завдання» – введення номеру завдання.

Вихідні дані (ім’я, опис, тип):

task_matrix44 – якщо ввели число «1», виводяться розрахунки задачі Matrix 44.

task_matrix71 – якщо ввели число «2», виводяться розрахунки задачі Matrix 71.

Лістинг коду вирішення завдання 3 наведено в дод. А (стор. 5-9)

Приклад діаграми для завдання 3 наведено на рис. Б.5. (дод. Б, стор. 10-13)

ВИСНОВКИ

Було вивчено теоретичний матеріал з основ представлення одновимірних і масивів на мові С ++ і було закріплено на практиці реалізація декларації, введення з консолі, обробка і виведення в консоль одновимірних масивів на мові С ++ в середовищі Visual Studio.

ДОДАТОК А
Лістинг коду програми

```
#include <iostream>
#include "windows.h"
#include <climits>
using namespace std;

const int M = 20, N = 20; // максимальна допустима кількість рядків та стовпців

void get_matr1(int in_matr[M][N], int& in_m, int& in_n); // функція для
зчитування матриці першого завдання

void show_matr1(const int out_matr[M][N], const int n, const int m); // функція
для виведення матриці на екран

bool isSortedAscending(const int out_matr[M], int N); // функція перевіряє, чи
рядок впорядкований по зростанню

bool isSortedDescending(const int out_matr[M], int N); // функція перевіряє, чи
рядок впорядкований по спаданню

int min_research(const int out_matr[M][N], const int m, const int n); // функція
знаходить мінімальний елемент у матриці

void get_matr2(int in_matr[M][N], int& in_m, int& in_n); // функція для
зчитування матриці другого завдання

void show_matr2(const int out_matr[M][N], const int m, const int n); // функція
для виведення матриці на екран

int find_min_column(const int matr[M][N], const int m, const int n); // функція
пошуку мінімального стовпця матриці

// Matrix44. Дана матриця розміру M × N. Знайти мінімальний серед елементів тих
// рядків, які впорядковані або по зростанню, або по спадаючій. Якщо
впорядковані рядки в матриці
// відсутні, то вивести 0.
void task_matrix44(); //завдання 1, декларація функції

// Matrix71. Дана матриця розміру M × N. Продублювати стовпець матриці, що
містить її мінімальний
// елемент.
void task_matrix71(); //завдання 2, декларація функції

int main()
{
    SetConsoleOutputCP(1251);
    int menu; // Зміна для номеру завдання
    do
    { // початок циклу
        cout << "Номер завдання:"; //введення номеру завдання
        cin >> menu; // обирання номеру завдання
        cout << endl; //вільна строка
        switch (menu) {
        case 1: task_matrix44(); break; // 1 - завдання 1
        case 2: task_matrix71(); break; // 2 - завдання 2
        case -1: cout << "Вихід..." << endl; break; // -1 - вихід
        default: cout << "Помилка! Лише 1, 2!" << endl; // інший номер -
повторити
```

```
        }
        cout << endl; // вільна строка
        cout << "+---------------------+" << endl; // строка задля полегшення
візуального сприймання тексту
        cout << endl; // вільна строка
    } // кінець циклу
    while (menu != -1); // умова виконання циклу
    return 0;
}

// функція введення матриці
void get_matr1(int in_matr[M][N], int& in_m, int& in_n) {
    int a;
    do {
        cout << "Введіть кількість рядків та стовбців (2-20): "; // обмеження на
кількість рядків/стовбців: від 2 до 20
        cin >> a; // введення кількості
        in_m = a; in_n = a;
    } while (in_n < 2 || in_n > N || in_m < 2 || in_m > M); // перевірка на
коректність введених значень
    cout << "Введіть елементи: " << endl; // введення елеменів матриці
    for (int i = 0; i < in_m; i++)
        for (int j = 0; j < in_n; j++)
            cin >> in_matr[i][j]; // вводимо елементи по черзі
}

//функція виведення масиву
void show_matr1(const int out_matr[M][N], const int m, const int n) {
    cout << endl << "Матриця: " << endl;
    // перевіряємо кожний рядок по черзі
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            cout << out_matr[i][j] << "\t"; // виведення елементів з табуляцією
        }
        cout << endl; // перехід на новий рядок після виведення всіх елементів
поточного рядка
    }
}

// функція перевірки, чи рядок впорядкований по зростанню
bool isSortedAscending(const int row[M], int N) {
    for (int i = 1; i < N; i++) {
        if (row[i] < row[i - 1]) { // якщо знайшли зменшення - рядок не
впорядкований
            return false; // якщо всі елементи йдуть по зростанню, рядок
впорядкований
        }
    }
    return true; // Рядок впорядкований за зростанням
}

// функція перевірки, чи рядок впорядкований по спаданню
bool isSortedDescending(const int row[M], int N) {
    for (int i = 0; i < N - 1; ++i) {
        if (row[i] < row[i + 1]) {
            return false; // Якщо поточний елемент менший за наступний, рядок не
впорядкований по спаданню
        }
    }
    return true; // Рядок впорядкований за спаданню
}
```

```
// Функція для пошуку мінімального елемента в певних рядках матриці, які відсортовані за зростанням або спаданням
int min_research(const int out_matr[M][N], const int m, const int n) { // Ініціалізація змінної для зберігання мінімального значення.
    int minValue = INT_MAX; // початкове задання найбільш можливого значення
    for (int i = 0; i < m; i++) { // проходження всіх рядків матриці
        // Перевірка, чи є рядок відсортованим або за зростанням, або за спаданням
        if (isSortedAscending(out_matr[i], n) || isSortedDescending(out_matr[i], n)) {
            for (int j = 0; j < n; j++) { // проходження по елементах рядка, якщо він відсортований
                if (out_matr[i][j] < minValue) { // порівняння поточного елемента з мінімальним значенням
                    minValue = out_matr[i][j]; // оновлення мінімума, якщо знайдено менше значення
                }
            }
        }
    }
    return (minValue == INT_MAX) ? 0 : minValue; // якщо жоден з рядків не впорядкований, виводимо 0
}

// Завдання 1 реалізація
void task_matrix44()
{
    int matr1[M][N]; // оголошення матриці
    int row, col; // реальні розміри матриці
    get_matr1(matr1, row, col); // виклик функції введення
    show_matr1(matr1, row, col); // виклик функції виведення
    min_research(matr1, row, col); // пошук мінімального елемента серед впорядкованих рядків

    int i, a=0, b=0;

    // Перевіряємо кожний рядок
    for (i = 0; i < row; i++) { // проходимо по кожному рядку матриці
        // якщо рядок впорядкований по зростанню, виводимо відповідне повідомлення
        if (isSortedAscending(matr1[i], col)) {
            cout << "Рядок " << i + 1 << " впорядкований за зростанням." << endl; // виводимо номер рядка
            a++;
        }
    }

    for (i = 0; i < row; ++i) { // проходимо по кожному рядку матриці
        // якщо рядок впорядкований по спаданню, виводимо відповідне повідомлення
        if (isSortedDescending(matr1[i], col)) {
            cout << "Рядок " << i + 1 << " впорядкований за спаданням." << endl; // виводимо номер рядка
            b++;
        }
    }

    if (a == 0 && b == 0) { // при умові, що немає впорядкованих рядків
        cout << "Зростання: 0" << endl; // виводимо 0
        cout << "Спадання: 0" << endl; // виводимо 0
    }
```

```
    cout << "Найменший елемент: " << min_research(matr1, row, col) << endl; //
виведення найменшого елемента
}

// Завдання 2.
// Функція для введення матриці
void get_matr2(int in_matr[M][N], int& in_m, int& in_n) {
    int a;
    do {
        cout << "Введіть кількість рядків та стовпців (2-20): "; // введення
кількості рядків та стовпців
        cin >> a;
        in_m = a; in_n = a; // присвоєння однакових значень для рядків і
стовпців
    } while (in_n < 2 || in_n > N || in_m < 2 || in_m > M); // перевірка, чи
значення в допустимих межах

    cout << "Введіть елементи матриці: " << endl; // введення елементів матриці
    for (int i = 0; i < in_m; i++) {
        for (int j = 0; j < in_n; j++) {
            cin >> in_matr[i][j]; // введення елементів матриці по черзі
        }
    }
}

// функція для виведення матриці на екран
void show_matr2(const int out_matr[M][N], const int m, const int n) {
    cout << "Матриця: " << endl;
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            cout << out_matr[i][j] << "\t"; // виведення елементу матриці з
табуляцією
        }
        cout << endl; // перехід на новий рядок після виведення елементів
поточного рядка
    }
}

// функція для знаходження стовпця з мінімальним елементом
int find_min_column(const int matr[M][N], const int m, const int n) {
    int minVal = INT_MAX; // Ініціалізація мінімального елемента максимально
можливим значенням
    int minCol = -1; // Індекс стовпця з мінімальним елементом

    for (int j = 0; j < n; j++) { // перевірка всіх стовпців
        for (int i = 0; i < m; i++) { // перевірка всіх елементів стовпця
            if (matr[i][j] < minVal) { // при знаходженні нового мінімума,
мінімальне значення та його індекс оновлюється
                minVal = matr[i][j];
                minCol = j;
            }
        }
    }
    return minCol;  // Повернення індекса стовпця з мінімальним елементом
}

// Функція для дублювання стовпця, що містить мінімальний елемент, в кінці
матриці
void duplicate_min_column(int matr[M][N], int& m, int& n) {
    // Знаходження стовпець з мінімальним елементом
    int minCol = find_min_column(matr, m, n);
```

```
    if (minCol == -1) { // Якщо не вдалося знайти мінімальний стовпець,
виводиться помилка
        cout << "Помилка: не вдалося знайти мінімальний елемент!" << endl;
        return;
    }

    // Дублювання стовпця, що містить мінімальний елемент, у новий стовпець в
кінці
    for (int i = 0; i < m; i++) {
        matr[i][n] = matr[i][minCol]; // Копіювання елементу з мінімального
стовпця в новий стовпець
    }
    n++; // збільшення кількості стовпців
}

//завдання 2, реалізація
void task_matrix71()
{
    int matr1[M][N]; // оголошення матриці
    int row, col; // реальні розміри матриці

    get_matr2(matr1, row, col); // введення матриці
    show_matr2(matr1, row, col); // введення початкової матриці
    duplicate_min_column(matr1, row, col); // дублювання стовпця з мінімальним
елементом

    cout << "Матриця після дублювання стовпця з мінімальним елементом:" << endl;
// виведення матриці після дублювання
    show_matr2(matr1, row, col);
}
```

ДОДАТОК Б
Скрін-шоти вікна виконання програми

```
Номер завдання:1

Введіть кількість рядків та стовбців (2-20): 4
Введіть елементи:
1 2 3 4
4 3 2 1
6 3 7 3
1 6 2 5

Матриця:
1       2       3       4
4       3       2       1
6       3       7       3
1       6       2       5
Рядок 1 впорядкований за зростанням.
Рядок 2 впорядкований за спаданням.
Найменший елемент: 1
```

Рисунок Б.1 – Екран виконання програми для вирішення завдання Matrix 44

```
Номер завдання:2

Введіть кількість рядків та стовпців (2-20): 4
Введіть елементи матриці:
1 2 3 4
4 3 2 1
6 7 3 9
7 4 2 0
Матриця:
1       2       3       4
4       3       2       1
6       7       3       9
7       4       2       0
Матриця після дублювання стовпця з мінімальним елементом:
Матриця:
1       2       3       4       4
4       3       2       1       1
6       7       3       9       9
7       4       2       0       0
```

Рисунок Б.2 – Екран виконання програми для вирішення завдання Matrix 71

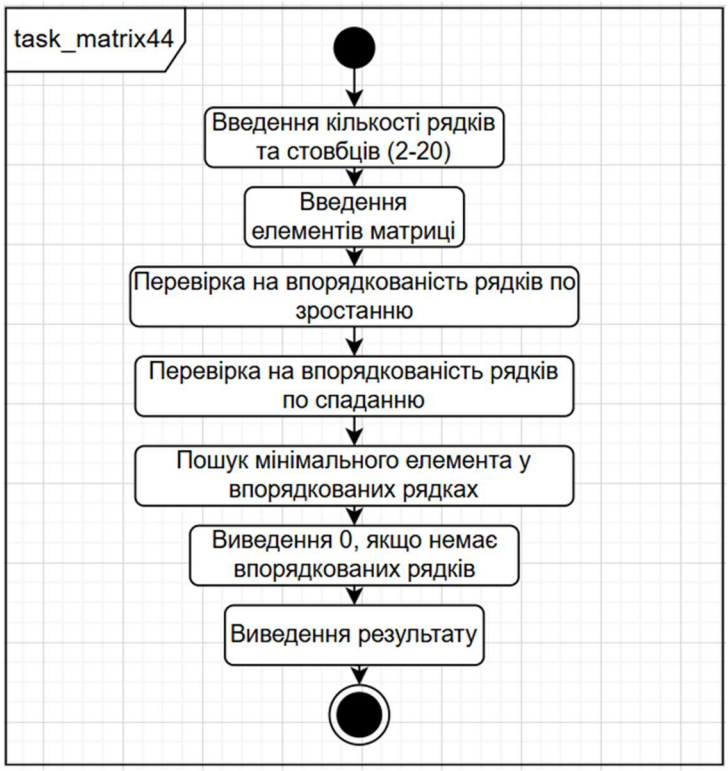

Рисунок Б.3 – Діаграма для завдання Matrix 44

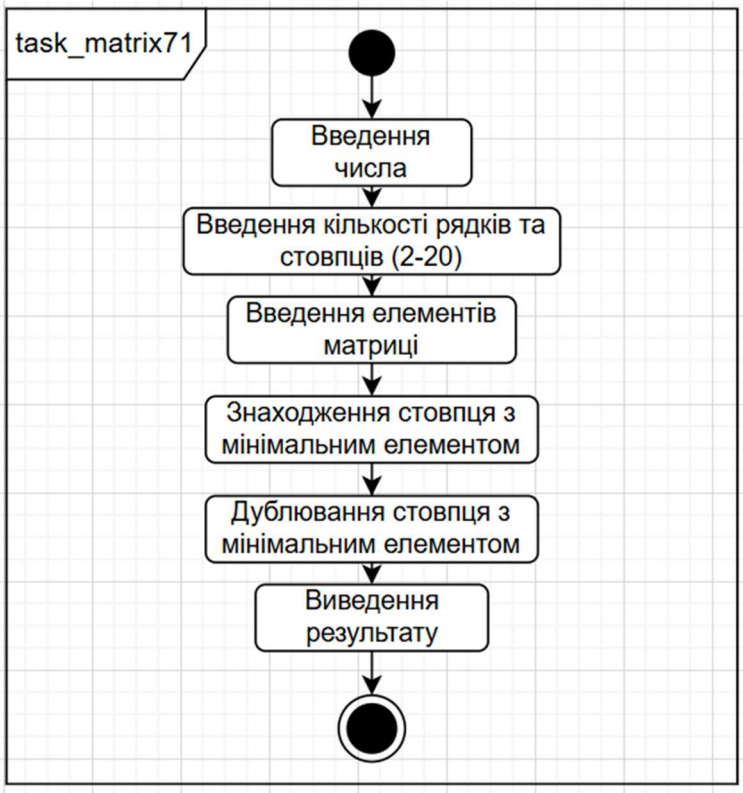

Рисунок Б.4 – Діаграма для завдання Matrix 71

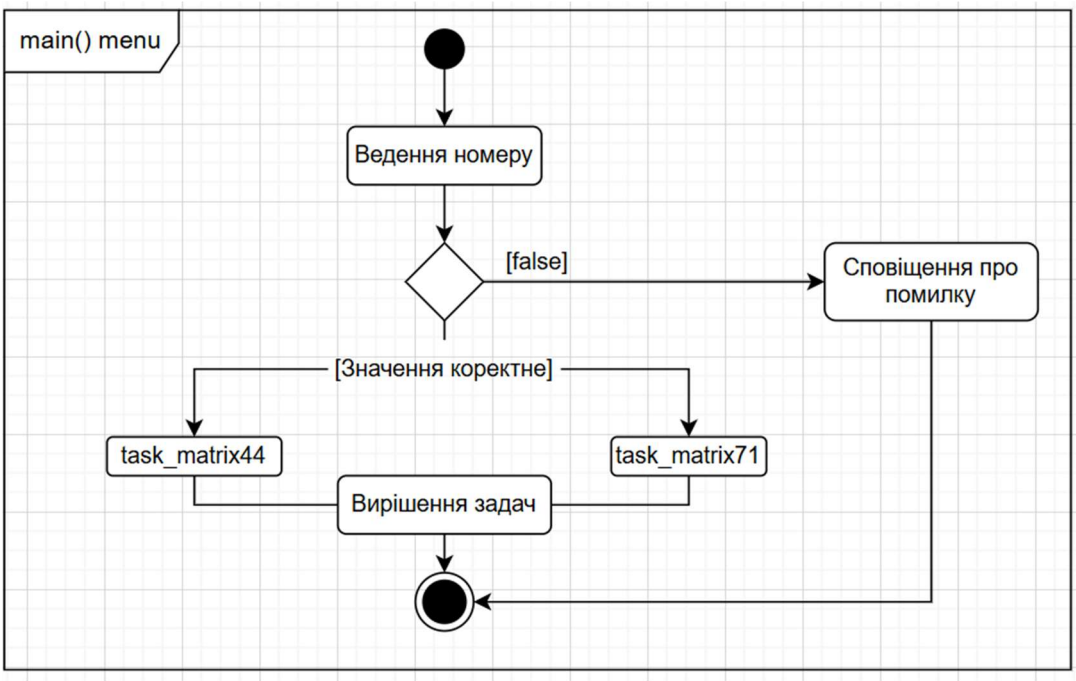

Рисунок Б.5 – Діаграма для завдання 3