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. 174. 319. 2 ЛР

Виконав студент гр. 319

22.02.2025 Дворнік І.П
(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 даних здійснити в командному вікні.

Matrix26. Дана матриця розміру M × N. Знайти номер її стовпця з найменшим добутком елементів і вивести даний номер, а також значення найменшого добутку.

Завдання 2. Перетворити матрицю відповідно до свого варіанту завдання(Matrix67) розмір матриці і його елементи ввести з консолі. Вивести результати у консоль.

Matrix67. Дана матриця розміру M × N, що містить як додатні, так і від'ємні елементи. Видалити всі її стовпці, що містять тільки додатні елементи. Якщо необхідних стовпців немає, то вивести матрицю без змін.

ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі Matrix26.

Вхідні дані:

Матриця M x N розміром 2-20 x 2-20

M – рядки, від 2 до 20.

N – стовпці, від 2 до 20.

Елементи матриці – цілі числа

Вихідні дані:

№ стовпця матриці з найменшим добутком – цілі числа

Значення найменшого добутку – цілі числа

Помилка при введені не коректних розмірів(M та N) матриці.

Алгоритм вирішення

1. Вивід меню вибору завдання.
2. Вводимо 1.

3. Вивід запрошення до вводу кількості рядків(M) матриці, вводимо число 2-20.
4. Вивід запрошення до вводу кількості стовпців(N) матриці, вводимо число 2-20.
5. Виведення запрошень до вводу елементів матриці(послідовно).
6. Введення елементів матриці у консоль(послідовно).
7. Вирахування добутку кожного стовпця.
8. Знахождення меншого добутку методом проходження по всім стовпцям, та заміною його на менший.
9. Виведення номеру стовпця з найменшим добутком.
10. Виведення найменшого добутку.

Завдання 2

Вирішення задачі Matrix67.

Вхідні дані:

Матриця M x N розміром 2-20 x 2-20

M – рядки, від 2 до 20.

N – стовпці, від 2 до 20.

Елементи матриці – цілі числа

Вихідні дані:

Матриця без стовпців з тільки додатніми числами або матриця(введена) без змін, розміри – 2-20 x 2-20, числа – цилі.

Спеціальне повідомлення при умові що жодного стовпця не було видалено. Помилка при введені не коректних розмірів(M та N) матриці.

Алгоритм вирішення

1. Вивід меню вибору завдання.
2. Вводимо 2.
3. Вивід запрошення до вводу кількості рядків(M) матриці, вводимо число.
4. Вивід запрошення до вводу кількості стовпців(N) матриці, вводимо число.
5. Виведення запрошень до вводу елементів матриці(послідовно).
6. Введення елементів матриці у консоль(послідовно).
7. Знаходження стовпців з тільки додатніми елементами.

8. Умовне розділення стовпців на 2 групи: видалені(якщо всі елементи додатні), не видалені (якщо хоча б 1 елемент не додатній).
9. Створення нової матриці без групи стовпців «видалені».
10. Онолення кількості стовпців та виведення нової мартиці.
11. Виведення спеціального повідомлення якщо жодного стовпця не будо видалено.

Лістинг коду вирішення задач Matrix 26 та Matrix 67 наведено в дод. А (стор. 5-7).

Екран роботи програм показаний у дод. Б(стор. 8-9)

ВИСНОВКИ

Було вивчено налагодження меню.

Закріплено на практиці команди для введення/виведення данних cin, cout.

Відпрацьовано в коді програми роботу з матрицями.

Отримано навички якось долі програміста(можливо).

Виникли труднощі з налагодженням коду.

ДОДАТОК А
Лістинг коду програми

```
  #include <iostream>
using namespace std;

const int limit_M = 20, limit_N = 20;

// Функція для знаходження номера стовпця з найменшим добутком елементів
void findMinProductColumn(int matrix[limit_M][limit_N], int M, int N) {
    int min_N = 0;  // індекс стовпця з мінімальним добутком
    long long min_save_n = LLONG_MAX;  // мінімальне значення добутку

    // Проходимо по кожному стовпцю
    for (int j = 0; j < N; ++j) {
        long long product = 1;  // початковий добуток
        for (int i = 0; i < M; ++i) {
            product *= matrix[i][j];  // множимо всі елементи стовпця
        }
        if (product < min_save_n) {  // якщо добуток менший за мінімальний
                                     // оновлюємо
            min_save_n = product;
            min_N = j;
        }
    }

    // Виводимо номер стовпця (з урахуванням індексації від 1) та його добуток
    cout << "\nColumn number with the smallest product: " << (min_N + 1)
         << endl;
    cout << "Least product : " << min_save_n << endl;
}

// Функція для видалення всіх стовпців, у яких тільки додатні значення
void removePositiveColumns(int matrix[limit_M][limit_N], int M, int& N) {
    bool toDelete[limit_N] = { false };  // масив для позначення стовпців, які
                                         // потрібно видалити
    int newN = 0;

    // Визначаємо які стовпці містять тільки додатні елементи
    for (int j = 0; j < N; ++j) {
        bool allPositive = true;
        for (int i = 0; i < M; ++i) {
            if (matrix[i][j] <= 0) {  // якщо хоча б один елемент не додатній —
                                      // не видаляємо
                allPositive = false;
                break;
            }
```

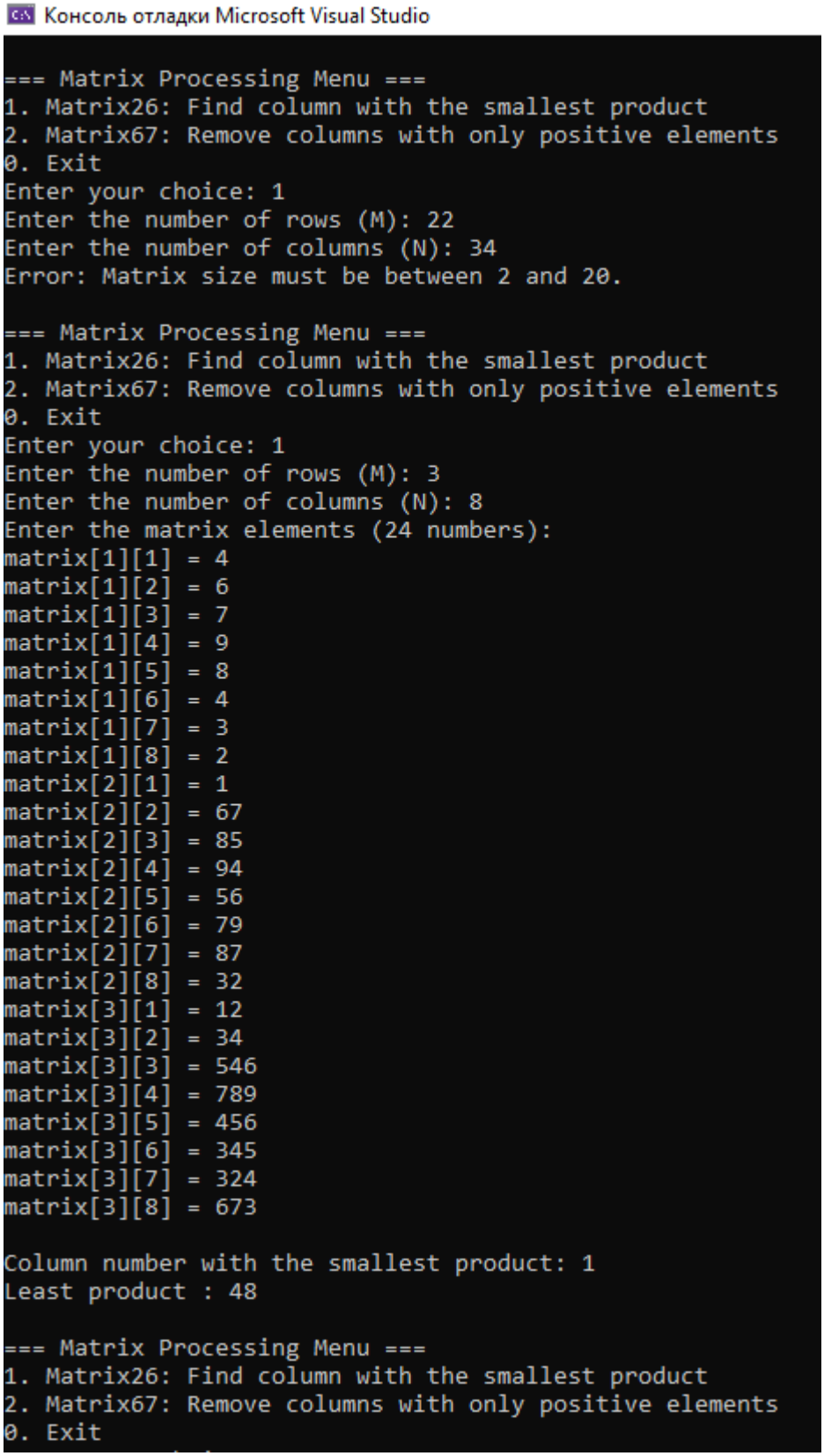
```
        }
        toDelete[j] = allPositive;  // якщо всі елементи додатні — помічаємо на
                                    // видалення
    }

    // Створюємо нову матрицю без стовпців, які потрібно видалити
    bool anyDeleted = false;  // прапор — чи були видалені стовпці
    for (int j = 0; j < N; ++j) {
        if (!toDelete[j]) {
            for (int i = 0; i < M; ++i) {
                matrix[i][newN] = matrix[i][j];  // копіюємо стовпець у нову
                                                 // позицію
            }
            newN++;
        }
        else {
            anyDeleted = true;  // хоча б один стовпець був видалений
        }
    }

    N = newN; // Оновлюємо кількість стовпців після видалення

    // Виводимо оновлену матрицю
    cout << "\nResulting matrix:" << endl;
    for (int i = 0; i < M; ++i) {
        for (int j = 0; j < N; ++j) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }

    if (!anyDeleted) {
        cout << "(No columns removed)" << endl;  // якщо жоден стовпець не було
                                                 // видалено
    }
}

int main() {
    int choice;
    do {
        // Меню вибору дії користувача
        cout << "\n=== Matrix Processing Menu ===" << endl;
        cout << "1. Matrix26: Find column with the smallest product" << endl;
        cout << "2. Matrix67: Remove columns with only positive elements"
             << endl;
        cout << "0. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        if (choice == 1 || choice == 2) {
```

```
            int M, N;
            int matrix[limit_M][limit_N];

            // Введення розміру матриці
            cout << "Enter the number of rows (M): ";
            cin >> M;
            cout << "Enter the number of columns (N): ";
            cin >> N;

            // Перевірка розмірів матриці
            if (M < 2 || M > limit_M || N < 2 || N > limit_N) {
                cout << "Error: Matrix size must be between 2 and 20.\n";
                continue;
            }

            // Введення елементів матриці
            cout << "Enter the matrix elements (" << M * N << " numbers):\n";
            for (int i = 0; i < M; ++i) {
                for (int j = 0; j < N; ++j) {
                    cout << "matrix[" << i + 1 << "][" << j + 1 << "] = ";
                    cin >> matrix[i][j];
                }
            }

            // Виклик відповідної функції в залежності від вибору
            if (choice == 1) {
                findMinProductColumn(matrix, M, N);
            }
            else if (choice == 2) {
                removePositiveColumns(matrix, M, N);
            }
        }
        else if (choice != 0) {
            cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 0);

    cout << "Exiting..." << endl;
    return 0;
}
```

ДОДАТОК Б

Скрін-шоти вікна виконання програми

Matrix 26



Matrix 67

```
Column number with the smallest product: 1
Least product : 48

=== Matrix Processing Menu ===
1. Matrix26: Find column with the smallest product
2. Matrix67: Remove columns with only positive elements
0. Exit
Enter your choice: 2
Enter the number of rows (M): 43
Enter the number of columns (N): 21
Error: Matrix size must be between 2 and 20.

=== Matrix Processing Menu ===
1. Matrix26: Find column with the smallest product
2. Matrix67: Remove columns with only positive elements
0. Exit
Enter your choice: 2
Enter the number of rows (M): 5
Enter the number of columns (N): 4
Enter the matrix elements (20 numbers):
matrix[1][1] = 4
matrix[1][2] = -7
matrix[1][3] = 5
matrix[1][4] = -9
matrix[2][1] = 3
matrix[2][2] = -8
matrix[2][3] = 7
matrix[2][4] = -3
matrix[3][1] = 2
matrix[3][2] = -8
matrix[3][3] = 65
matrix[3][4] = 78
matrix[4][1] = 21
matrix[4][2] = 56
matrix[4][3] = 32
matrix[4][4] = -48
matrix[5][1] = 51
matrix[5][2] = 28
matrix[5][3] = 91
matrix[5][4] = -75

Resulting matrix:
-7 -9
-8 -3
-8 78
56 -48
28 -75

=== Matrix Processing Menu ===
1. Matrix26: Find column with the smallest product
2. Matrix67: Remove columns with only positive elements
0. Exit
Enter your choice: 0
Exiting...
```