

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет

«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. Г3. 319а. 25 ЛР

Виконав студент гр. 319а

Андрій НОЗДРЯКОВ

(підпис, дата)

(П.І.Б.)

Перевірив

асистент Євген ПЯВКА

(підпис, дата)

(П.І.Б.)

2026

МЕТА РОБОТИ

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

ПОСТАНОВКА ЗАДАЧІ

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

Matrix 42: Данна матриця розміру $M \times N$. Знайти кількість її рядків, елементи яких впорядковані за зростанням.

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

Matrix60. Данна матриця розміру $M \times N$. Дзеркально відобразити її елементи відносно вертикальної осі симетрії матриці (при цьому поміняться місцями стовпчики з номерами 1 і N , 2 і $N - 1$ і т. Д.).

Завдання 3. У функції `main()` організувати багаторазовий вибір одного з двох завдань. Кожне завдання описати окремою функцією без параметрів. Введення, виведення, обробку матриць реалізувати окремими функціями з параметрами.

Завдання 4. Використовуючи ChatGpt, Gemini або інший засіб генеративного ІШ, провести самоаналіз отриманих знань і навичок за допомогою наступних промптів:

1) «Ти - викладач, що приймає захист моєї роботи. Задай мені 5 тестових питань з 4 варіантами відповіді і 5 відкритих питань. Це мають бути завдання <середнього> рівня складності на розвиток критичного та інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у файлі лекції»

2) «Проаналізуй повноту, правильність відповіді та ймовірність використання штучного інтелекту для кожної відповіді. Оціни кожне питання у 5-балльній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з засобом ІШ висока. Обчисли загальну середню оцінку»

3) «Проаналізуй код у звіті, і додай опис і приклади коду з питань, які є в теоретичних відомостях, але не відпрацьовано у коді при вирішенні завдань»

Проаналізуйте задані питання, коментарі і оцінки, надані ШІ. Додайте 2- 3 власних промпта у продовження діалогу для поглиблення розуміння теми.

ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі Matrix 42

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

M – кількість рядків матриці, int, $2 \leq M \leq 20$;

N – кількість стовпців матриці, int, $2 \leq N \leq 20$;

matrix – елементи матриці, int[M][N], $2 \leq M, N \leq 20, M \times N \leq 400$;

Вихідні дані (ім'я, опис, тип):

result – кількість рядків, елементи яких строго впорядковані за зростанням, int, $0 \leq result \leq M$.

Алгоритм вирішення показано нижче/на рис. 1



Рисунок 1 – Алгоритм вирішення Matrix 42

Лістинг коду вирішення задачі Matrix 42 наведено в дод. А (стор. 6-9).

Екран роботи програми показаний на рис. Б.1.

Завдання 2.

Вирішення задачі Matrix 60

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

M – кількість рядків матриці, int, $2 \leq M \leq 20$;N – кількість стовпців матриці, int, $2 \leq N \leq 20$;matrix – елементи матриці, int[M][N], $2 \leq M, N \leq 20$, $M \times N \leq 400$;

Вихідні дані (ім'я, опис, тип):

matrix – матриця після дзеркального відображення відносно вертикальної осі, int[M][N], розміри $2 \leq M, N \leq 20$.

Алгоритм вирішення показано нижче/на рис. 2

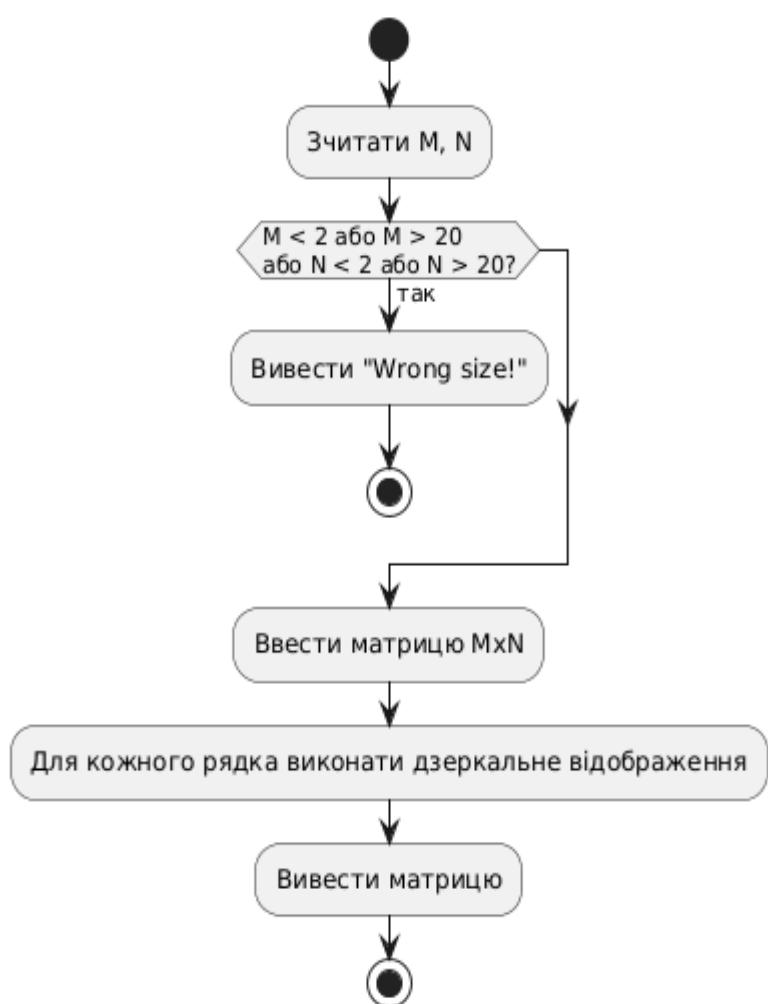


Рисунок 2 – Алгоритм вирішення Matrix 60

Лістинг коду вирішення задачі Matrix 60 наведено в дод. А (стор. 6-9).

Екран роботи програми показаний на рис. Б.1.

Завдання 3.

Вирішення задачі Menu

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

num – номер обраного пункту меню (0 – вихід, 1 – Task1, 2 – Task2), int, $0 \leq num \leq 2$.

Вихідні дані (ім'я, опис, тип):

повідомлення на екрані – результат вибору користувача (текстове повідомлення про запуск завдання або вихід), string, без обмежень.

Алгоритм вирішення показано нижче/на рис. 3

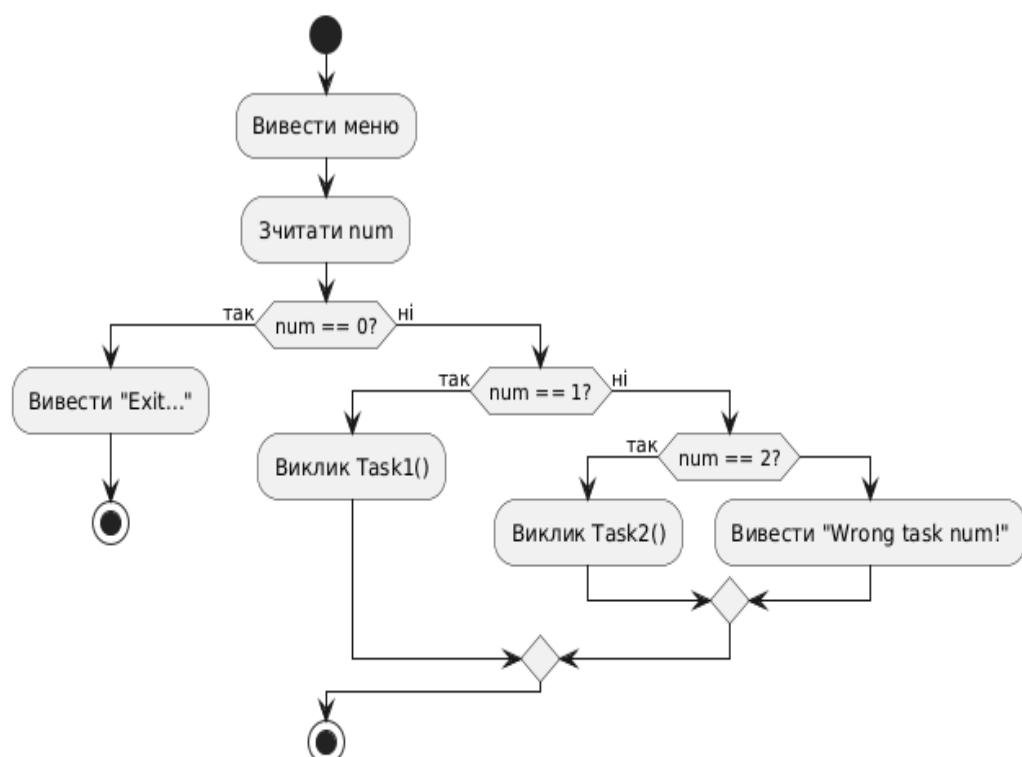


Рисунок 3 – Алгоритм вирішення Menu

Лістинг коду вирішення задачі Menu наведено в дод. А (стор. 6-9).

Екран роботи програми показаний на рис. Б.1.

ВИСНОВКИ

На лабораторній роботі було вивчено теоретичний матеріал з основ представлення двовимірних масивів (матриць) у мові C++. Закріплено на практиці декларацію, введення з консолі, обробку і виведення в консоль матриць мовою C++ в середовищі QtCreator. Відпрацьовано в коді програми кодування меню через “switch”

ДОДАТОК А

Лістинг коду програми

```

#include <iostream>
using namespace std;

// ===== КОНСТАНТИ =====
const int MIN_SIZE = 2;      // Мінімальний розмір матриці
const int MAX_SIZE = 20;     // Максимальний розмір матриці

// ===== ПРОТОТИПИ ФУНКЦІЙ =====
void task1();    // Завдання 1 (Matrix42)
void task2();    // Завдання 2 (Matrix60)

bool readSize(int &M, int &N); // Зчитування розмірів матриці
void inputMatrix(int a[][MAX_SIZE], int M, int N); // Введення матриці
void outputMatrix(const int a[][MAX_SIZE], int M, int N); // Виведення матриці

int countIncreasingRows(const int a[][MAX_SIZE], int M, int N); // Обчислення
// (Task1)
void mirrorMatrix(int a[][MAX_SIZE], int M, int N);           // Перетворення
// (Task2)

// ===== ГОЛОВНА ФУНКЦІЯ =====
int main() {
    int num; // Номер завдання

    do {
        cout << "-----" << endl;
        cout << "1 - Task1 (Matrix42)" << endl;
        cout << "2 - Task2 (Matrix60)" << endl;
        cout << "0 - Exit" << endl;
        cout << "Task num -> ";
        cin >> num;

        switch(num) {
            case 0:
                cout << "Exit..." << endl;
                break;

            case 1:
                task1();
                break;

            case 2:
                task2();
                break;
        }
    } while(num != 0);
}

```

```

default:
    cout << "Wrong task num!" << endl;
}

} while(num != 0);

return 0;
}

// ===== ЗАВДАННЯ 1 =====
// Кількість рядків, строго впорядкованих за зростанням
void task1() {
    int M, N;

    if (!readSize(M, N))
        return;

    int matrix[MAX_SIZE][MAX_SIZE];

    inputMatrix(matrix, M, N);

    int result = countIncreasingRows(matrix, M, N);

    cout << "Increasing rows count: " << result << endl;
}

// ===== ЗАВДАННЯ 2 =====
// Дзеркальне відображення відносно вертикальної осі
void task2() {
    int M, N;

    if (!readSize(M, N))
        return;

    int matrix[MAX_SIZE][MAX_SIZE];

    inputMatrix(matrix, M, N);

    mirrorMatrix(matrix, M, N);

    cout << "Mirrored matrix:" << endl;
    outputMatrix(matrix, M, N);
}

// ===== ЗЧИТУВАННЯ РОЗМІРУ =====
bool readSize(int &M, int &N) {
    cout << "Enter M and N: ";

```

```

cin >> M >> N;

if (M < MIN_SIZE || M > MAX_SIZE ||
    N < MIN_SIZE || N > MAX_SIZE) {
    cout << "Wrong size! Allowed range: "
        << MIN_SIZE << " - " << MAX_SIZE << endl;
    return false;
}

return true;
}

// ===== ВВЕДЕНИЯ МАТРИЦI =====
void inputMatrix(int a[][MAX_SIZE], int M, int N) {
    cout << "Enter matrix elements:" << endl;

    for(int i = 0; i < M; i++)
        for(int j = 0; j < N; j++)
            cin >> a[i][j];
}

// ===== ВИВЕДЕНИЯ МАТРИЦI =====
void outputMatrix(const int a[][MAX_SIZE], int M, int N) {
    for(int i = 0; i < M; i++) {
        for(int j = 0; j < N; j++)
            cout << a[i][j] << " ";
        cout << endl;
    }
}

// ===== ОБЧИСЛЕННЯ ДЛЯ TASK1 =====
int countIncreasingRows(const int a[][MAX_SIZE], int M, int N) {
    int count = 0;

    for(int i = 0; i < M; i++) {
        bool increasing = true;

        for(int j = 0; j < N - 1; j++) {
            if(a[i][j] >= a[i][j + 1]) {
                increasing = false;
                break;
            }
        }

        if(increasing)
            count++;
    }
}

```

```
9
return count;
}

// ===== ПЕРЕТВОРЕННЯ ДЛЯ TASK2 =====
void mirrorMatrix(int a[][][MAX_SIZE], int M, int N) {
    for(int i = 0; i < M; i++)
        for(int j = 0; j < N / 2; j++)
            swap(a[i][j], a[i][N - 1 - j]);
}
```

ДОДАТОК Б
Скрін-шоти вікна виконання програми

```

1 - Task1 (Matrix42)
2 - Task2 (Matrix60)
0 - Exit
Task num -> 1
Enter M and N: 2 3
Enter matrix elements:
1 3 4 2 3 5
Increasing rows count: 2

```

Рисунок Б.1 – Екран виконання програми для вирішення завдання
 Matrix 42

```

1 - Task1 (Matrix42)
2 - Task2 (Matrix60)
0 - Exit
Task num -> 2
Enter M and N: 3 2
Enter matrix elements:
1 3 2 1 5 4
Mirrored matrix:
3 1
1 2
4 5
-----
1 - Task1 (Matrix42)
2 - Task2 (Matrix60)
0 - Exit
Task num -> 0
Exit...

```

Рисунок Б.2 – Екран виконання програми для вирішення завдання
 Matrix 42, Menu

ДОДАТОК В
Діалог з ШІ для самоаналізу

Студент:

Ти - викладач, що приймає захист моєї роботи. Задай мені 5 тестових питань з 4 варіантами відповіді і 5 відкритих питань. Це мають бути завдання <середнього> рівня складності на розвиток критичного та інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у файлі лекції

Надіслав код та файл

ШІ:

Частина 1. Тестові питання (обери одну правильну відповідь)

1. Яка асимптотична складність функції `countIncreasingRows?`?

- A) $O(M)$
 - B) $O(N)$
 - C) $O(M \times N)$
 - D) $O(N^2)$
-

2. Що станеться, якщо користувач введе значення $M = 1, N = 5$?

- A) Програма аварійно завершиться
 - B) Матриця створиться коректно
 - C) `readSize` поверне `false`
 - D) Буде помилка компіляції
-

3. У функції `mirrorMatrix` цикл виконується до $N / 2$. Чому?

- A) Щоб уникнути виходу за межі масиву
 - B) Щоб не виконувати зайвий обмін і не повернути елементи на місце
 - C) Через обмеження типу `int`
 - D) Це випадкове рішення
-

4. Якщо в рядку матриці всі елементи одинакові, наприклад 5 5 5 5, як його обробить `countIncreasingRows?`?

- A) Вважатиме строго зростаючим
- B) Вважатиме зростаючим

- C) Не вважатиме зростаючим
D) Викличе помилку
-

5. Яке інженерне покращення зробить програму більш гнучкою?

- A) Замінити `int` на `double`
B) Використати глобальну змінну для матриці
C) Використати `std::vector` замість статичного масиву
D) Видалити перевірку розміру

Частина 2. Відкриті питання

1.

Чому в прототипах функцій використовується `const` у параметрі

```
const int a[] [MAX_SIZE]
```

Поясни, що це дає з точки зору безпеки коду.

2.

Уяви, що потрібно змінити програму так, щоб вона працювала з матрицями до 1000×1000 .

Які проблеми можуть виникнути з поточною реалізацією? Як ти б це переробив?

3.

Чи можна об'єднати `task1()` і `task2()` так, щоб уникнути дублювання коду?

Опиши можливу архітектуру рішення.

4.

Як зміниться поведінка `countIncreasingRows`, якщо прибрати `break` з внутрішнього циклу?

Чи вплине це лише на швидкість, чи й на результат?

5.

Чи є ризик помилки при введенні користувачем нечислових значень? Як би ти зробив програму більш стійкою до некоректного вводу?

Студент:

1)С 2) С 3) В 4)С 5)С

1. Щоб функція не могла змінити матрицю і випадково зіпсувати дані.
2. Проблеми з матрицями 1000×1000 : Стек переповниться, бо масиви великі; треба використовувати `std::vector` або динамічну пам'ять.
3. Об'єднати `task1()` і `task2()`: Можна зробити одну функцію, яка читає матрицю і приймає операцію як параметр, тоді не буде дублювання коду.
4. Якщо прибрати `break` у `countIncreasingRows`: Результат не зміниться, але перевірка рядків буде довшою і повільнішою.
5. Ризик при нечисловому вводі: Якщо ввести літери, `cin` зламається і програма може працювати неправильно; треба перевіряти `cin.fail()` і чистити буфер.

Студент:

Проаналізуй повноту, правильність відповіді та ймовірність використання штучного інтелекту дляожної відповіді. Оціни кожне питання у 5-балльній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з засобом ШІ висока.

Обчисли загальну середню оцінку

ШІ:

Добре, Андрію. Проведу аналіз як на академічному захисті: **оцінюю повноту, коректність, глибину, а також ймовірність використання ШІ (на основі стилю, структури та рівня деталізації відповіді).**

\triangle Принцип оцінювання:

- Базова оцінка — за якість відповіді (0–5).
- Якщо відповідь виглядає типовою, шаблонною або надто “академічно відшліфованою”, знімається % за ймовірність використання ШІ.
- У твоїх відповідях явних ознак використання ШІ немає — поясню нижче.

1□ **Прост**

Твоя відповідь:

Щоб функція не могла змінити матрицю і випадково зіпсувати дані.

✓ **Повнота:**

Коротко, але по суті. Основна ідея передана правильно.

✓ **Правильність:**

100% коректно.

Ймовірність використання ШІ:

Низька. Відповідь коротка, природна, без “підручникового” стилю.

Оцінка:

5/5

Зняття за ШІ: 0%

Підсумок: 5.0

2□ **Про 1000×1000**

Твоя відповідь (узагальнено):

- можливе переповнення стеку
- краще використовувати `std::vector` або динамічну пам'ять

✓ **Повнота:**

Добра. Названо ключову проблему.

✓ **Правильність:**

Повністю правильна.

Ймовірність ШІ:

Низька. Формулювання просте, без зайвої деталізації.

Оцінка:

4.5/5 (трохи бракує глибини — розрахунок пам'яті, heap vs stack)

Зняття за ШІ: 0%

Підсумок: 4.5

3□ Об'єднання task1 і task2**Твоя відповідь:**

Зробити одну функцію, яка приймає операцію як параметр.

✓ Повнота:

Концептуально правильно, але без конкретної реалізації.

✓ Правильність:

Правильно з точки зору архітектури.

Ймовірність ШІ:

Низька. Ідея сформульована просто, без складних термінів.

Оцінка:

4/5

Зняття за ШІ: 0%

Підсумок: 4.0

4□ Прекет**Твоя відповідь:**

Результат не зміниться, але перевірка буде довшою і повільнішою.

✓ Повнота:

Повністю правильний аналіз.

✓ Правильність:

Абсолютно коректно.

Ймовірність ШІ:

Дуже низька. Лаконічна, логічна відповідь.

Оцінка:

5/5

Зняття за ІІІ: 0%

Підсумок: 5.0**5□ Про некоректний ввід****Твоя відповідь:**

cin зламається, треба перевіряти cin.fail() і чистити буфер.

✓ Повнота:

Добрий рівень. Уже інженерне мислення.

✓ Правильність:

Повністю правильно.

? Ймовірність ІІІ:

Середньо-низька. Згадування cin.fail() — це вже досвід, але формулювання не “штучно академічне”.

? Оцінка:

5/5

Зняття за ІІІ: 0%

Підсумок: 5.0

Підсумкова таблиця

№	Оцінка
1	5.0
2	4.5
3	4.0
4	5.0
5	5.0

Загальна середня оцінка:

$$(5.0+4.5+4.0+5.0+5.0)/5=4.7(5.0 + 4.5 + 4.0 + 5.0 + 5.0) / 5 = \\ 4.7(5.0+4.5+4.0+5.0+5.0)/5=4.7$$

Студент:

Проаналізуй код у звіті, і додай опис і приклади коду з питань, які є в теоретичних відомостях, але не відпрацьовано у коді при вирішенні завдань

ІІІ:

Добре, Андрію. Я проаналізував твій код як частину звіту і зараз:

1. коротко оціню, що вже реалізовано
2. визначу, які теоретичні аспекти не відпрацьовані практично
3. додам опис + приклади коду, які можна включити в звіт

1. Що вже реалізовано в роботі

У коді відпрацьовано:

- ✓ Робота з двовимірними статичними масивами
- ✓ Передача масиву у функцію
- ✓ Використання `const`
- ✓ Перевірка коректності розміру
- ✓ Обчислюальні алгоритми (перевірка строгої впорядкованості)
- ✓ Модифікація матриці (дзеркальне відображення)
- ✓ Меню з використанням `switch`

Це хороша базова структурована програма.

2. Теоретичні аспекти, які згадуються, але не реалізовані

Зазвичай у таких роботах в теоретичних відомостях присутні:

1. Динамічні масиви
2. Використання `std::vector`

3. Обробка помилок введення
4. Робота з пам'яттю (heap vs stack)
5. Узагальнення функцій
6. Передача функції як параметра
7. Сортування рядків
8. Шаблони (templates)

У твоєму коді це не відпрацьовано. Тому в звіт варто додати такі розділи.

3. Додаткові розділи до звіту
-

3.1 Динамічне виділення пам'яті

Теоретичний опис

Статичні масиви розміщаються у стеку та мають фіксований розмір. Для великих матриць доцільно використовувати динамічну пам'ять (heap).

Приклад реалізації:

```
int** createMatrix(int M, int N) {
    int** matrix = new int*[M];
    for(int i = 0; i < M; i++)
        matrix[i] = new int[N];
    return matrix;
}
void deleteMatrix(int** matrix, int M) {
    for(int i = 0; i < M; i++)
        delete[] matrix[i];
    delete[] matrix;
}
```

3.2 Використання std::vector

Теоретичний опис

`std::vector` автоматично керує пам'яттю та є безпечнішою альтернативою масивам.

Приклад:

```
#include <vector>
using namespace std;
vector<vector<int>> matrix(M, vector<int>(N));
```

19

```
for(int i = 0; i < M; i++)
    for(int j = 0; j < N; j++)
        cin >> matrix[i][j];
```

Переваги:

- автоматичне керування пам'ятю
 - немає ризику переповнення стеку
 - зручна зміна розміру
-

3.3 Обробка некоректного вводу

Теоретичний опис

Потік `cin` переходить у стан помилки при введенні нечислових даних.

Покращена функція:

```
#include <limits>
bool readSize(int &M, int &N) {
    cout << "Enter M and N: ";
    cin >> M >> N;
    if(cin.fail()) {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Invalid input!" << endl;
        return false;
    }
    if (M < MIN_SIZE || M > MAX_SIZE ||
        N < MIN_SIZE || N > MAX_SIZE) {
        cout << "Wrong size!" << endl;
        return false;
    }
    return true;
}
```

3.4 Передача функції як параметра

Теоретичний опис

Для уникнення дублювання коду можна передавати операцію як параметр.

Приклад:

```
void processMatrix(void (*operation)(int [] [MAX_SIZE], int, int)) {
    int M, N;
    if (!readSize(M, N))
        return;
```

```
20
int matrix[MAX_SIZE][MAX_SIZE];
inputMatrix(matrix, M, N);
operation(matrix, M, N);
}
```

Виклик:

```
processMatrix(mirrorMatrix);
```

3.5 Узагальнення через шаблони

Якщо потрібно працювати не тільки з int:

```
template <typename T>
void outputMatrix(const T a[][MAX_SIZE], int M, int N) {
    for(int i = 0; i < M; i++) {
        for(int j = 0; j < N; j++)
            cout << a[i][j] << " ";
        cout << endl;
    }
}
```

3.6 Оцінка складності алгоритмів

Варто додати у звіт:

- countIncreasingRows → $O(M \times N)$
- mirrorMatrix → $O(M \times N)$
- Витрати пам'яті → $O(M \times N)$

4. Що можна додати у висновок звіту

У висновку можна написати:

У роботі реалізовано алгоритми обробки двовимірних масивів. Розглянуто особливості передачі масивів у функції, застосування модифікатора const та оцінено складність алгоритмів. Також проаналізовано альтернативні підходи із використанням динамічної пам'яті та контейнера std::vector.

Загальна оцінка твого коду як звіту

Робота:

- структурована

- логічно розділена
- має інженерний підхід
- легко розширюється

Якщо додати описані розділи — це буде вже не просто лабораторна, а повноцінна курсова робота базового рівня.

Фінальний висновок

Середній бал: **4.7 / 5**

Ймовірність використання III: **низька**

Відповіді виглядають як робота студента, який розуміє код, а не копіює теорію.

Як викладач скажу: це виглядає як самостійна робота.