

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 C ++»

XAI.305. G7. 319. 16 ЛР

Виконав студент гр. 319

Павло Сафонов

(підпис, дата)

(П.І.Б.)

Перевірив

к.т.н., доц. Олена ГАВРИЛЕНКО

(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. Варіанти завдань представлено в табл.1.

Matrix22. Данна матриця розміру $M \times N$. Для кожного стовпця матриці з парним номером (2, 4, ...) знайти суму його елементів. Умовний оператор не використовувати.

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

Matrix54. Данна матриця розміру $M \times N$. Поміняти місцями стовпець з номером N і перший з стовпців, що містять тільки від'ємні елементи. Якщо необхідних стовпців немає, то вивести матрицю без змін.

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 ШІ, провести самоаналіз отриманих знань і навичок за

ВИКОНАННЯ РОБОТИ

```
#include <iostream>
```

```
using namespace std;
```

```
const int M = 20; // Розмір матриці
const int N = 20;
```

```
void Matrix22(); // Завдання 1
```

```
int diag_sum22(int matrix[M][N], const int row, const int col); // Сумує
елементи які стоять на парних стовпцях
```

```
void show_matrix22(int matrix[M][N], const int row, const int col, const int sum); // Виводить матрицю та суму
```

```
void Matrix54(); // Завдання 2
```

```
void diag_sum54(int (&matrix)[M][N], const int row, const int col); // Змінює розташування від'ємного елемента на перше а перше на від'ємне
```

```
void show_matrix54(int matrix[M][N], const int row, const int col); // виводить матрицю завдання 2
```

```
void get_matrix(int matrix[M][N], int &row, int &col); // цей метод дозволяє вводити користувачу розмір та елементи матриці
```

```
int main()
```

```
{
```

```
    int menu = 1; // змінна для вибора завдання
```

```
    while (menu != 0) // це цикл який дозволяє обирати завдання багато разів, 0 це вихід
```

```
{
```

```
    cout << "1. Matrix22" << endl; // виводимо просте меню
```

```
    cout << "2. Matrix54" << endl;
```

```
    cout << "0. Exit" << endl;
```

```
    cout << "Enter menu: ";
```

```
    cin >> menu; // дозволяємо користувачу вводити номер завдання
```

```
    switch (menu) // що обрав користувач?
```

```
{
```

```
    case 1:
```

```
        Matrix22();
```

```
        break;
```

```
    case 2:
```

```
        Matrix54();
```

```
        break;
```

```

        case 0:
            break;
        default:
            cout << "Invalid menu!" << endl; // помилка
            break;
    }
}

system("pause"); // закінчення
return 0;
}

void Matrix22()
{
    // просто завдання 1

    int matrix[M][N]; // матриця яка обмежена полями 20 на 20

    int row, col; // змінні за допомогою яких користувач буде обмежувати
    розмір своєї матриці

    get_matrix(matrix, row, col); // користувач вводить необхідні дані
    int sum = diag_sum22(matrix, row, col); // сума всіх парних стовпців
    show_matrix22(matrix, row, col, sum); // виводимо матрицю та суму
}

int diag_sum22(int matrix[M][N], const int row, const int col)
{
    int sum = 0; // змінна в якій зберігається сума

    for (int i = 0; i < row; i++) // лічильник рядків
        for (int j = 2; j < col; j+=2) // лічильник стовпців, кожне парне
            sum += matrix[i][j]; // сумуємо

    return sum; // повертаємо
}

```

```
}
```

```
void show_matrix22(int matrix[M][N], const int row, const int col, const int
sum)
```

```
{
```

```
    for (int i = 0; i < row; i++) // лільник рядків
```

```
{
```

```
        for (int j = 0; j < col; j++) // лічильник стовпців
```

```
            cout << matrix[i][j] << " "; // виводимо елемент
```

```
        cout << endl; // новий рядок
```

```
}
```

```
    cout << "Sum: " << sum << endl; // виводимо суму
```

```
}
```

```
void Matrix54()
```

```
{
```

```
    // просто завдання 2
```

```
    int matrix[M][N]; // матриця яка обмежена полями 20 на 20
```

```
    int row, col; // змінні за допомогою яких користувач буде обмежувати
    розмір своєї матриці
```

```
    get_matrix(matrix, row, col); // користувач вводить дані
```

```
    diag_sum54(matrix, row, col); // змінюємо місцями
```

```
    show_matrix54(matrix, row, col); // виводимо
```

```
}
```

```
void diag_sum54(int (&matrix)[M][N], const int row, const int col)
```

```
{
```

```
    for (int i = 0; i < row; i++) // лічильник рядків
```

```
        for (int j = 0; j < col; j++) // лічильник стовпців
```

```
            if (matrix[i][j] < 0) // умова якщо від'ємне
```

```

{
    int element = matrix[i][0]; // змінна яка хранить перший елемент

    matrix[i][0] = matrix[i][j]; // змінюємо місцями
    matrix[i][j] = element;
}

void show_matrix54(int matrix[M][N], const int row, const int col)
{
    for (int i = 0; i < row; i++) // лічильник рядків
    {
        for (int j = 0; j < col; j++) // лічильник стовпців
            cout << matrix[i][j] << " "; // виводимо елемент

        cout << endl; // новий рядок
    }
}

void get_matrix(int matrix[M][N], int &row, int &col)
{
    do
    {
        cout << "Enter row and column counts (1-20): "; // просимо ввести
огранічення
        cin >> row >> col;
    } while (row < 2 || row > M || col < 2 || col > N || row != col); // доки рядок
або стовпець не будуть більше 2 та буде більше 20 фбо якщо вони співпадають
тоді коректно

    for (int i = 0; i < row; i++) // лічильник рядків
        for (int j = 0; j < col; j++) // лічильник стовпців
            cin >> matrix[i][j]; // користувач вводить елемент
}

```

Результат:

```
D:\Learn>a.exe
1. Matrix22
2. Matrix54
0. Exit
Enter menu: 3
Invalid menu!
1. Matrix22
2. Matrix54
0. Exit
Enter menu: 1
Enter row and column counts (1-20): 5
5
2
4
3
2
4
5
4
3
2
4
5
3
2
4
5
4
2 4 3 2 4
5 4 3 2 4
5 3 2 4 5
4 3 4 5 4
3 2 4 5 4
Sum: 37
```

```
1. Matrix22
2. Matrix54
0. Exit
Enter menu: 2
Enter row and column counts (1-20): 3
3
3
-5
20
3
4
5
-5
4
3
-5 3 20
3 4 5
-5 4 3
1. Matrix22
2. Matrix54
0. Exit
Enter menu: 0
Press any key to continue . . .
D:\Learn>
```

III

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»

2) «Проаналізуї 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 ШІ висока. Обчисли загальну середню оцінку»

2

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

Добре. Я — викладач, що приймає захист. Працюємо поетапно.

✓ 1. Тестові питання (середній рівень)

Тест 1

Яке фактичне значення мають «парні стовпці» у функції `diag_sum22()`?

«» C++ □

```
for (int j = 2; j < col; j += 2)
```

- A) Стовпці з індексами 1, 3, 5...
- B) Стовпці з індексами 0, 2, 4...
- C) Стовпці з індексами 2, 4, 6...
- D) Усі стовпці

Тест 2

Що станеться, якщо в рядку є декілька від'ємних елементів у `diag_sum54()`?

- A) Заміниться тільки перший від'ємний
- B) Заміниться тільки останній від'ємний
- C) Перший елемент рядка буде послідовно мінятись з кожним від'ємним
- D) Код викличе помилку

Тест 3

Чому параметр у `diag_sum54` передається так:

«» C++



```
int (&matrix)[M][N]
```

- A) Для копіювання матриці
- B) Для передачі за посиланням без копіювання
- C) Для передачі тільки константної матриці
- D) Це нічим не відрізняється від звичайного масиву

Тест 4

Яке обмеження накладає функція `get_matrix()` ?

«» C++



```
while (row < 2 || row > M || col < 2 || col > N || row != col);
```

- A) Матриця має бути прямокутною
- B) Матриця має бути квадратною
- C) Дозволяється розмір 1×1
- D) Будь-який розмір до 20

Тест 5

Що є потенційною проблемою в коді?

- A) Використання `const`
 - B) `system("pause")`
 - C) Передача параметрів за значенням
 - D) Немає жодних проблем
-

5 відкритих питань

1.

Поясніть різницю між передачею масиву як:

 C++ 

```
int matrix[M][N]
```

та

 C++ 

```
int (&matrix)[M][N]
```

у контексті цієї програми.

2.

Чому назва `diag_sum22()` не відповідає її реальному функціоналу?

3.

Як змінити код `diag_sum54()`, щоб обмін відбувався лише з першим знайденим від'ємним елементом у кожному рядку?

4.

Які недоліки має жорстко зафікований розмір `20x20`? Як це можна покращити інженерно?

5.

Як змінити програму, щоб вона працювала з неквадратними матрицями?

2. Інструкція для наступного етапу

Тепер:

1. Дайте відповіді на всі 10 питань.
2. Після цього я:
 - Проаналізую правильність
 - Оціню повноту
 - Оціню ймовірність використання ШІ
 - Зніму 60% балів, якщо відповідь має ознаки ШІ
 - Виведу середній бал

3. Аналіз коду та доповнення теоретичних відомостей

Тепер аналізує сам код.

1. У коді НЕ відпрацьовано такі теоретичні аспекти:

1 Динамічні масиви

У теорії зазвичай розглядається:

- `new` / `delete`
- `vector`
- виділення пам'яті під час виконання

У вас — статичний масив:

```
// C++
```

```
int matrix[M][N];
```



Приклад покращення:

```
// C++
```

```
int** matrix = new int*[row];
for(int i = 0; i < row; i++)
    matrix[i] = new int[col];
```



Або сучасніше:

```
// C++
```

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

vector<vector<int>> matrix(row, vector<int>(col));
```



💡 2. Не використано алгоритмічну оптимізацію

У `diag_sum54()` цикл не зупиняється після першої заміни.

Покращена версія:

```
↪ C++
void diag_sum54(int (&matrix)[M][N], const int row, const int col)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            if (matrix[i][j] < 0)
            {
                swap(matrix[i][0], matrix[i][j]);
                break; // зупинка після першого
            }
        }
    }
}
```

➊ 3. Не розглянуто принципи кодування

У теоретичних відомостях зазвичай є:

- DRY
- Іменування функцій
- Розділення відповідальності

Проблема:

- `diag_sum22` — не має відношення до діагоналі
- `diag_sum54` — не рахує суму

```
// C++  
  
int sum_even_columns(...)  
void swap_first_negative(...)
```

🔍 4. Не використано перевірку введення

Якщо користувач введе символ — програма впаде.

Покращення:

```
// C++  
  
if(!(cin >> row >> col))  
{  
    cin.clear();  
    cin.ignore(1000, '\n');  
}
```

🔍 5. Не розглянуто складність алгоритму

Обидві функції мають:

- Часова складність: $O(n^2)$
- Просторова складність: $O(1)$



Загальний інженерний висновок

- ✓ Програма логічно структурована
- ✓ Є декомпозиція на функції
- ✓ Є передача по посиланню

! Ale:

- Назви функцій не відповідають змісту
- Є зайві ітерації
- Обмеження на квадратність без потреби
- Відсутня обробка помилок
- Статична пам'ять

Тепер ваша черга:

👉 Дайте відповіді на 10 питань — і я проведу повне оцінювання захисту.

- 1) С
- 2) С
- 3) В
- 4) В
- 5) В

Відкриті питання

- 1) переше це звичайний масив у параметрі методу який не бере посилання на пам'ять вже існуючого масиву але створює новий, другий навпаки бере посилання на вже існуючу матрицю але не створює а просто бере, це потрібно для того щоб зберегти зміни у оригінальній матриці одразу
- 2) скоріж за все я спутав рядки та строчки
- 3) поставити `break` у `if`, це зупине вложений цикл тим самим програма перейде на наступну строчку
- 4) це не є гнучкістю та може зробити оскладнення для користувача, якщоб не було умови яка предотвращає вводити користувачу більше 20 то це була би помилка, це можно було б упростити агтай `vector` або застаре **
- 5) прибрати умову ду вайл а саме `col != row` тим самим матриця може бути не тільки квадратною

Добре. Проводжу повний аналіз як на захисті.

ЧАСТИНА 1 — ТЕСТОВІ ПИТАННЯ

1) С — правильно

- ✓ Ви правильно зрозуміли, що цикл починається з `j = 2`, тому беруться індекси 2,4,6...

Оцінка: 5/5

Ймовірність ШІ: низька

Після корекції: 5/5

2) С — правильно

- ✓ Дійсно, перший елемент буде послідовно мінятись з кожним від'ємним.

Оцінка: 5/5

Ймовірність ШІ: низька

Після корекції: 5/5

3) В — правильно

- ✓ Передача по посиланню без копіювання.

Оцінка: 5/5

Ймовірність ШІ: низька

Після корекції: 5/5

4) В — правильно

- ✓ Умова `row != col` примушує матрицю бути квадратною.

Оцінка: 5/5

Ймовірність ШІ: низька

Після корекції: 5/5

5) В — правильно

- ✓ `system("pause")` — непереносимий виклик (Windows-only).

Оцінка: 5/5

Ймовірність ШІ: низька

Після корекції: 5/5



Результат тестової частини:

$5/5 \times 5 = 25$ балів

✓ ЧАСТИНА 2 — ВІДКРИТИ ПИТАННЯ

1 Передача масиву

Ваше твердження:

"передє це звичайний масив ... створює новий"

! Тут є концептуальна помилка.

У C++ масив не копіюється повністю при передачі в параметр функції.

Фактично:

C++ □

```
int matrix[M][N]
```

деградує до вказівника на перший рядок.

Тобто **новий масив не створюється**.

- ✓ Ви правильно пояснили роль посилання.
- ✗ Але неправильно пояснили перший випадок.

Повнота: середня

Правильність: частково правильна

Оцінка: 3/5

Ймовірність ШІ: дуже низька (видно природну помилку)

Після корекції: 3/5

2 Назва функції

Ви написали:

"скоріж за все я сплутав рядки та строчки"

! Відповідь неповна.

Проблема не в рядках/стовпцях, а в тому, що:

- `diag_sum` означає "сума діагоналі"
- але функція сумує **парні стовпці**

Тут порушена семантика іменування.

Повнота: низька

Оцінка: 2/5

Ймовірність ШІ: низька

Після корекції: 2/5

3 break у циклі

- ✓ Правильно.
- ✓ Чітко по суті.

Оцінка: 5/5

Ймовірність ШІ: низька

Після корекції: 5/5

4 Недоліки 20×20

Ви написали загальну ідею про негнучкість — це правильно.

Але:

- Не згадано про пам'ять
- Не згадано про масштабованість
- Не згадано про обмеження стеку
- Не пояснено чому `vector` кращий

Оцінка: 3/5

Ймовірність ШІ: низька

Після корекції: 3/5

5 Неквадратна матриця

✓ Правильно визначили, що треба прибрати `row != col`.

Оцінка: 5/5

Ймовірність ШІ: низька

Після корекції: 5/5



Результат відкритих питань:

№	Бал
1	3
2	2
3	5
4	3
5	5

Сума: 18/25



ЧАСТИНА 3 — Перевірка на використання ШІ

Ознаки:

- Є мовні неточності
- Є логічні неточності
- Є природні помилки
- Відсутні академічні формулювання

👉 Ймовірність використання ШІ: низька (~15%)

⚠ Тому зняття 60% НЕ застосовується.



ПІДСУМКОВИЙ РОЗРАХУНОК

Тестові: 25/25

Відкриті: 18/25

Загальна сума: 43/50

Середній бал:

$$43 \div 10 = 4.3$$



ПІДСУМКОВА ОЦІНКА: 4.3 / 5

(міцна 4+, близько до відмінно)



Коментар викладача

- ✓ Ви добре орієнтуетесь у коді
- ✓ Розумієте передачу по посиланню
- ✓ Розумієте логіку циклів

! Треба підтягнути:

- Теорію передачі масивів
- Семантику іменування
- Інженерну аргументацію