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.301.175.318.13 ЛР

Виконав студент гр. № 319а

Михайло ТЮТЮННИК

(підпис, дата) (П.І.Б.)

Перевірив к.т.н., доцент

( вчена ступінь, вчене звання)

Олена ГАВРИЛЕНКО (підпис,дата) (П.І.Б.)

2025

Мета роботи

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

Постановка задачі

Завдання 1 (Matrix19): Для кожного рядка матриці знайти суму її елементів.

Завдання 2 (Matrix78): Для квадратної матриці обнулити елементи, що лежать на побічній діагоналі та нижче неї, без використання умовних операторів.

Виконання роботи

Завдання 1. Matrix19

Вхідні дані:

* Матриця розміру M × N (цілі числа, обмеження: 2 ≤ M, N ≤ 20).

Вихідні дані:

* Сума елементів кожного рядка матриці.

Алгоритм:

1. Ввести розміри матриці (M, N).
2. Ввести елементи матриці.
3. Для кожного рядка обчислити суму елементів.
4. Вивести результати.

Лістинг коду до завдання 1 (Matrix19) наведено в дод. А (стор. 4).

Зображеня вікна виконання роботи наведено в дод. Б(стор. 8, рис. 1)

Завдання 2. Matrix78

Вхідні дані:

* Квадратна матриця порядку M (цілі числа, обмеження: 2 ≤ M ≤ 20).

Вихідні дані:

* Матриця з обнуленими елементами на побічній діагоналі та нижче неї.

Алгоритм:

1. Ввести розмір матриці M.
2. Ввести елементи матриці.
3. Обнулити елементи, для яких i + j >= M - 1.
4. Вивести змінену матрицю.

Лістинг коду до завдання 1 (Matrix19) наведено в дод. А (стор. 4).

Зображеня вікна виконання роботи наведено в дод. Б(стор. 8, рис. 2)

Висновки

У ході роботи було вивчено роботу з двовимірними масивами (матрицями) у мові С++. Реалізовано:

* Обчислення сум елементів кожного рядка матриці.
* Перетворення квадратної матриці шляхом обнулення елементів на побічній діагоналі та нижче неї.

Робота демонструє вміння працювати з матрицями, використовувати цикли та логічні операції для обробки даних.

ДОДАТОК А

Лістинг коду програми

#include <iostream> // Підключення бібліотеки для введення/виведення даних

using namespace std; // Використання стандартного простору імен

/\* Глобальні константи для обмеження розмірів матриць \*/

const int MAX\_SIZE = 20; // Максимальний розмір квадратної матриці

const int MAX\_ROWS = 20; // Максимальна кількість рядків для прямокутної матриці

const int MAX\_COLS = 20; // Максимальна кількість стовпців для прямокутної матриці

/\*\*

\* Функція для введення довільної прямокутної матриці

\* @param matrix - двовимірний масив для зберігання матриці

\* @param rows - посилання на змінну для зберігання кількості рядків

\* @param cols - посилання на змінну для зберігання кількості стовпців

\*

\* Деталі реалізації:

\* 1. Запитує у користувача кількість рядків і стовпців

\* 2. Перевіряє, чи значення знаходяться в допустимому діапазоні (2-20)

\* 3. У разі некоректного вводу завершує програму з кодом помилки

\* 4. Запитує введення кожного елемента матриці по порядку

\*/

void inputRectangularMatrix(int matrix[MAX\_ROWS][MAX\_COLS], int& rows, int& cols) {

// Введення кількості рядків

cout << "Enter number of rows (2-" << MAX\_ROWS << "): ";

cin >> rows;

// Введення кількості стовпців

cout << "Enter number of columns (2-" << MAX\_COLS << "): ";

cin >> cols;

/\* Перевірка коректності введених розмірів \*/

if (rows < 2 || rows > MAX\_ROWS || cols < 2 || cols > MAX\_COLS) {

cout << "Error: Matrix dimensions must be between 2 and 20.\n";

exit(1); // Завершення програми з кодом помилки

}

/\* Введення елементів матриці \*/

cout << "Enter matrix elements (" << rows << "x" << cols << "):\n";

for (int i = 0; i < rows; i++) { // Цикл по рядках

for (int j = 0; j < cols; j++) { // Цикл по стовпцях

cin >> matrix[i][j]; // Зчитування значення елемента

}

}

}

/\*\*

\* Функція для введення квадратної матриці

\* @param matrix - двовимірний масив для зберігання матриці

\* @param size - посилання на змінну для зберігання розміру матриці

\*

\* Деталі реалізації:

\* 1. Запитує у користувача розмір квадратної матриці

\* 2. Перевіряє коректність введеного значення (2-20)

\* 3. Заповнює матрицю елементами по рядках

\*/

void inputSquareMatrix(int matrix[MAX\_SIZE][MAX\_SIZE], int& size) {

// Введення розміру матриці

cout << "Enter matrix size (2-" << MAX\_SIZE << "): ";

cin >> size;

/\* Перевірка коректності розміру \*/

if (size < 2 || size > MAX\_SIZE) {

cout << "Error: Matrix size must be between 2 and 20.\n";

exit(1); // Завершення програми з кодом помилки

}

/\* Введення елементів матриці \*/

cout << "Enter matrix elements (" << size << "x" << size << "):\n";

for (int i = 0; i < size; i++) { // Цикл по рядках

for (int j = 0; j < size; j++) { // Цикл по стовпцях

cin >> matrix[i][j]; // Зчитування значення елемента

}

}

}

/\*\*

\* Функція для виведення довільної матриці

\* @param matrix - двовимірний масив з матрицею

\* @param rows - кількість рядків матриці

\* @param cols - кількість стовпців матриці

\*

\* Деталі реалізації:

\* 1. Виводить матрицю у вигляді таблиці

\* 2. Елементи розділяються табуляцією для вирівнювання

\* 3. Після кожного рядка матриці перехід на новий рядок

\*/

void printRectangularMatrix(int matrix[MAX\_ROWS][MAX\_COLS], int rows, int cols) {

cout << "\nMatrix:\n";

for (int i = 0; i < rows; i++) { // Цикл по рядках

for (int j = 0; j < cols; j++) { // Цикл по стовпцях

cout << matrix[i][j] << "\t"; // Виведення елемента з табуляцією

}

cout << endl; // Перехід на новий рядок після кожного рядка матриці

}

}

/\*\*

\* Функція для виведення квадратної матриці

\* @param matrix - двовимірний масив з матрицею

\* @param size - розмір квадратної матриці

\*/

void printSquareMatrix(int matrix[MAX\_SIZE][MAX\_SIZE], int size) {

cout << "\nMatrix:\n";

for (int i = 0; i < size; i++) { // Цикл по рядках

for (int j = 0; j < size; j++) { // Цикл по стовпцях

cout << matrix[i][j] << "\t"; // Виведення елемента з табуляцією

}

cout << endl; // Перехід на новий рядок

}

}

/\*\*

\* Функція для обчислення сум елементів у кожному рядку матриці

\* @param matrix - двовимірний масив з матрицею

\* @param rows - кількість рядків матриці

\* @param cols - кількість стовпців матриці

\*

\* Деталі реалізації:

\* 1. Для кожного рядка обчислює суму його елементів

\* 2. Виводить результати у форматі "Row X: Sum"

\*/

void calculateRowSums(int matrix[MAX\_ROWS][MAX\_COLS], int rows, int cols) {

cout << "\nSum of elements in each row:\n";

for (int i = 0; i < rows; i++) { // Цикл по рядках

int sum = 0; // Змінна для накопичення суми

for (int j = 0; j < cols; j++) { // Цикл по стовпцях

sum += matrix[i][j]; // Додавання поточного елемента до суми

}

// Виведення результату для поточного рядка

// i+1 використовується для нумерації рядків з 1 (замість 0)

cout << "Row " << i + 1 << ": " << sum << endl;

}

}

/\*\*

\* Функція для обнулення елементів на побічній діагоналі та нижче неї

\* @param matrix - двовимірний масив з квадратною матрицею

\* @param size - розмір квадратної матриці

\*

\* Деталі реалізації:

\* 1. Побічна діагональ визначається умовою i+j = size-1

\* 2. Область нижче діагоналі - i+j > size-1

\* 3. Використовує множення на 0 без явних умовних операторів

\*/

void zeroBelowAntiDiagonal(int matrix[MAX\_SIZE][MAX\_SIZE], int size) {

for (int i = 0; i < size; i++) { // Цикл по рядках

for (int j = 0; j < size; j++) { // Цикл по стовпцях

/\* Логіка обнулення:

- i+j >= size-1: елемент на діагоналі або нижче

- !(умова) повертає 0 для цих елементів, 1 - для інших

- Множення на цю величину або обнуляє, або зберігає значення \*/

matrix[i][j] \*= !(i + j >= size - 1);

}

}

}

/\*\*

\* Функція для відображення головного меню

\*

\* Деталі реалізації:

\* 1. Виводить доступні опції меню

\* 2. Нумерація опцій починається з 1

\*/

void displayMainMenu() {

cout << "\n=== Matrix Operations Menu ===";

cout << "\n1. Calculate sums of matrix rows (Task 19)";

cout << "\n2. Zero elements below anti-diagonal (Task 78)";

cout << "\n3. Exit program";

cout << "\nPlease select an option (1-3): ";

}

/\*\*

\* Головна функція програми

\*

\* Деталі роботи:

\* 1. Виводить меню з опціями

\* 2. Обробляє вибір користувача

\* 3. Виконує відповідні операції з матрицями

\* 4. Продовжує роботу до вибору опції виходу

\*/

int main() {

int userChoice; // Змінна для зберігання вибору користувача

/\* Головний цикл меню \*/

do {

displayMainMenu(); // Відображення меню

cin >> userChoice; // Очікування вводу користувача

/\* Обробка вибору користувача \*/

switch (userChoice) {

case 1: { // Опція для обчислення сум рядків

int matrix[MAX\_ROWS][MAX\_COLS]; // Оголошення матриці

int rowCount, colCount; // Змінні для розмірів

// Введення та виведення матриці

inputRectangularMatrix(matrix, rowCount, colCount);

printRectangularMatrix(matrix, rowCount, colCount);

// Обчислення та виведення сум

calculateRowSums(matrix, rowCount, colCount);

break;

}

case 2: { // Опція для роботи з побічною діагоналлю

int matrix[MAX\_SIZE][MAX\_SIZE]; // Оголошення матриці

int matrixSize; // Змінна для розміру

// Введення та виведення початкової матриці

inputSquareMatrix(matrix, matrixSize);

cout << "\nOriginal matrix:";

printSquareMatrix(matrix, matrixSize);

// Виконання перетворення матриці

zeroBelowAntiDiagonal(matrix, matrixSize);

// Виведення результату

cout << "\nModified matrix:";

printSquareMatrix(matrix, matrixSize);

break;

}

case 3: { // Опція виходу

cout << "Exiting the program. Goodbye!\n";

break;

}

default: { // Обробка некоректного вводу

cout << "Invalid selection. Please try again.\n";

}

}

} while (userChoice != 3); // Умова продовження циклу

return 0; // Успішне завершення програми

}

ДОДАТОК Б

Скрін-шоти вікна виконання програми

На рисунку 1 зображена робота програми до завдання Matrix19

Изображение выглядит как текст, компьютер, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 1 — Результат роботи програми

На рисунку 2 зображена робота програми до завдання Matrix 78

Изображение выглядит как текст, электроника, компьютер, снимок экрана

Автоматически созданное описание

Рисунок 2 — Результат роботи програми

ДОДАТОК В

Діаграми активності програм

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Діаграма активності 1

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Діаграма активності 2