Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра ПЗ

# Лабораторна робота №5, варіант № 19
з дисципліни «Основи програмування»

Виконала: ст. 1ПІ-25Б Семенов В.О. .
Перевірив: доцент Решетнік О.О.

Вінниця – 2025

**Тема**: Робота з багатовимірними масивами

**Мета**: Ініціалізація багатовимірних масивів, виконання операцій з масивами.

**Завдання 19.** Серед стовпців заданої цілочислової матриці, що містять тільки такі елементи, що за модулем не більше 10, знайти стовпець з мінімальним добутком елементів.

Декомпозиція завдання

1. Введення даних користувачем
    - Вводяться кількість рядків і стовпців матриці.
    - Виконується перевірка, щоб значення були позитивними.
2. Генерація матриці
    - Створюється матриця заданого розміру.
    - Кожен елемент заповнюється випадковим числом у діапазоні [-10; 10].
3. Виведення матриці
    - Матриця виводиться на екран для наочності.
4. Обчислення добутку елементів стовпців
    - Для кожного стовпця обчислюється добуток усіх його елементів.
5. Пошук стовпця з мінімальним добутком
    - Порівнюються добутки всіх стовпців.
    - Визначається стовпець з мінімальним добутком.
6. Виведення результату
    - Виводиться номер стовпця з мінімальним добутком.
    - Виводиться значення цього мінімального добутку.

Код програми показано на лістингу.

Приклади роботи програми можна побачити на рисунку 1

Лістинг 1 – Програма для знаходження рядка з найменшим добутком

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>


int inputPositiveInt(const std::string& prompt)
{
    int value;
    do
    {
        std::cout << prompt;
        std::cin >> value;
        if (value <= 0)
        {
            std::cout << "Invalid input. Please enter a positive
integer.\n";
        }
    } while (value <= 0);
    return value;
}

void generateMatrix(std::vector<std::vector<int>>& a, int n, int
m)
{
    a.resize(n, std::vector<int>(m));
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            a[i][j] = std::rand() % 21 - 10;
        }
    }
}

void printMatrix(const std::vector<std::vector<int>>& a)
{
    for (const auto& row : a)
    {
        for (int x : row)
        {
            std::cout << x << "\t";
        }
        std::cout << std::endl;
    }
}

long long columnProduct(const std::vector<std::vector<int>>& a,
int col)
{
    long long product = 1;
```

Продовження лістингу 1

```
    for (const auto& row : a)
    {
        product *= row[col];
    }
    return product;
}

int findMinProductColumn(const std::vector<std::vector<int>>& a)
{
    int m = static_cast<int>(a[0].size());
    int minCol = 0;
    long long minProduct = columnProduct(a, 0);

    for (int j = 1; j < m; j++)
    {
        long long currentProduct = columnProduct(a, j);
        if (currentProduct < minProduct)
        {
            minProduct = currentProduct;
            minCol = j;
        }
    }
    return minCol;
}

int main()
{
    int n = inputPositiveInt("Enter number of rows (positive
integer): ");
    int m = inputPositiveInt("Enter number of columns (positive
integer): ");

    std::vector<std::vector<int>> matrix;

    std::srand(static_cast<unsigned>(std::time(nullptr)));

    generateMatrix(matrix, n, m);

    std::cout << "\nGenerated matrix:\n";
    printMatrix(matrix);

    int minCol = findMinProductColumn(matrix);

    std::cout << "\nColumn with minimum product: " << minCol + 1
<< std::endl;
    std::cout << "Minimum product value: " <<
columnProduct(matrix, minCol) << std::endl;

    return 0;
}
```

```
Enter number of rows (positive integer): 10 10
Enter number of columns (positive integer):
Generated matrix:
2       -5      -6      2       7       4       1       1       4       10
1       9       -3      -7      4       9       -10     0       -5      3
-9      -9      1       0       -8      1       -2      5       10      -1
-10     -8      10      -9      -9      -2      7       -3      -7      10
-8      5       6       -10     5       -10     0       -2      -4      3
7       2       3       5       2       -3      8       -6      10      -3
-6      2       -6      3       6       -6      -2      -7      1       -6
-8      10      -1      -2      10      7       -9      1       3       -7
-1      -2      5       -8      -2      -4      -3      0       1       3
3       8       -2      -9      0       10      6       -10     1       2

Column with minimum product: 6
Minimum product value: -3628800
```

Рисунок 1 – робота програми при правильному введені

```
Enter number of rows (positive integer): -5
Invalid input. Please enter a positive integer.
Enter number of rows (positive integer): 5
Enter number of columns (positive integer): 8

Generated matrix:
1       -4      -9      0       -8      -7      6       -6
1       2       -7      -7      -1      8       -7      -6
-5      7       -9      -3      -2      3       2       10
-3      -8      -8      -7      -9      1       -1      -5
6       -9      -5      -5      -1      -6      7       -9

Column with minimum product: 3
Minimum product value: -22680
```

Рисунок 2 – робота програми при неправильному введені

Контрольні запитання:

19. Що таке операція взяття адреси?
Операція взяття адреси (оператор &) повертає адресу змінної в пам’яті.

20. Що таке константний покажчик?
Константний покажчик (type * const ptr) — це покажчик, який завжди вказує на один і той же об’єкт, але значення цього об’єкта можна змінювати.

21. Які арифметичні дії можна проводити над покажчиками? Для чого вони використовуються?

- Додавання та віднімання цілих чисел (ptr + n, ptr - n) — пересування по масиву.
- Віднімання одного покажчика від іншого (ptr1 - ptr2) — обчислення кількості елементів між ними.
- Порівняння (==, !=, <, >) — визначення порядку або рівності адрес. Використовуються для роботи з масивами і динамічною пам’яттю, обходу елементів і визначення відстані між ними.

Висновок:

У процесі роботи з багатовимірними масивами ми навчились ініціалізувати матриці, заповнювати їх значеннями, виконувати операції над елементами та знаходити потрібні результати. Це дозволяє ефективно обробляти структуровані дані та готує до роботи з більш складними структурами даних.