Лабораторна робота №8.

**Масив** - це структура даних у багатьох мовах програмування, призначена для зберігання послідовності елементів одного типу даних. Елементи масиву ідентифікуються індексами (порядковими номерами), і можна легко звертатися до кожного окремого елемента за допомогою індексу.

**Типи даних, які може зберігати масив**:

* Цілі числа (int, long, short, тощо)
* Дійсні числа (float, double)
* Символи (char)
* Булеві значення (bool)
* Вказівники (pointer) на інші об'єкти

**Масиви поділяються на кілька типів:**

* Одновимірні масиви - це масиви з одним індексом. Наприклад, масив цілих чисел int arr[10].
* Двовимірні масиви - це масиви з двома індексами і виглядають як таблиці, представлені у вигляді рядків і стовпців.
* Багатовимірні масиви - це масиви з більш ніж двома індексами, такі як три- та чотири-вимірні масиви, і так далі.

**Форма многомірного масиву**: Многомірні масиви подаються як вкладені одновимірні масиви. Наприклад, двовимірний масив можна представити як масив одновимірних масивів. Для двовимірного масиву int matrix[3][4] можна уявити як 3 окремі одновимірні масиви, кожен з яких має 4 елементи.

Завдання 1: Сортування одновимірного масиву.

1.

#include <iostream>

#include <algorithm> // Для std::sort

**int** main() {

**int** m[] = {3, 1, 4, 1, 5, 9, 2, 6, 5, 3};

**int** n = **sizeof**(m) / **sizeof**(m[0]); // Розмір масиву

std::sort(m, m + n); // Використовуємо std::sort для сортування

std::cout << "Відсортований масив: ";

**for** (**int** i = 0; i < n; ++i) {

std::cout << m[i] << " ";

}

std::cout << std::endl;

**return** 0;

}

2.

#include <iostream>

#include <algorithm> // Для std::sort

#include <vector>

**bool** compareDescending(**int** a, **int** b) {

**return** a > b;

}

**int** main() {

**int** m[] = {3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 8, 7};

**int** n = **sizeof**(m) / **sizeof**(m[0]); // Розмір масиву

std::sort(m, m + n, compareDescending); // Використовуємо std::sort з функцією compareDescending

std::cout << "Відсортований масив за спаданням: ";

**for** (**int** i = 0; i < n; ++i) {

std::cout << m[i] << " ";

}

std::cout << std::endl;

**return** 0;

}

Завдання 2: Одновимірний масив з умовним оператором.

1. Напишіть програму, що заміняє нулями всі від’ємні елементи заданого(переліком елементів) не порожнього масиву.
2. #include <iostream>
3. **int** main() {
4. **int** arr[] = {1, -2, 3, -4, 5, -6};
5. **int** n = **sizeof**(arr) / **sizeof**(arr[0]);
6. **for** (**int** i = 0; i < n; i++) {
7. **if** (arr[i] < 0) {
8. arr[i] = 0;
9. }
10. }
11. std::cout << "Замінено від'ємні елементи нулями: ";
12. **for** (**int** i = 0; i < n; i++) {
13. std::cout << arr[i] << " ";
14. }
15. **return** 0;
16. }

2)  Напишіть програму, що заміняє нулями всі додатні елементи заданого(переліком елементів) не порожнього масиву.

#include <iostream>

**int** main() {

**int** arr[] = {1, -2, 3, -4, 5, -6};

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

**for** (**int** i = 0; i < n; i++) {

**if** (arr[i] > 0) {

arr[i] = 0;

}

}

std::cout << "Замінено додатні елементи нулями: ";

**for** (**int** i = 0; i < n; i++) {

std::cout << arr[i] << " ";

}

**return** 0;

}

3)  Напишіть програму, що заміняє нулями всі елементи, що мають значення більше 10, заданого(переліком елементів) не порожнього масиву.

#include <iostream>

**int** main() {

**int** arr[] = {8, 15, 3, 12, 7, 18};

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

**for** (**int** i = 0; i < n; i++) {

**if** (arr[i] > 10) {

arr[i] = 0;

}

}

std::cout << "Замінено елементи, що більше 10, на нулі: ";

**for** (**int** i = 0; i < n; i++) {

std::cout << arr[i] << " ";

}

**return** 0;

}

4)  Напишіть програму, що заміняє нулями всі елементи, що мають значення менше 10, заданого(переліком елементів) не порожнього масиву.

#include <iostream>

**int** main() {

**int** arr[] = {8, 15, 3, 12, 7, 18};

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

**for** (**int** i = 0; i < n; i++) {

**if** (arr[i] < 10) {

arr[i] = 0;

}

}

std::cout << "Замінено елементи, що менше 10, на нулі: ";

**for** (**int** i = 0; i < n; i++) {

std::cout << arr[i] << " ";

}

**return** 0;

}

5)  Напишіть програму, що заміняє нулями всі елементи, значення яких виражене парним числом, заданого(переліком елементів) не порожнього масиву.

#include <iostream>

**int** main() {

**int** arr[] = {8, 15, 3, 12, 7, 18};

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

**for** (**int** i = 0; i < n; i++) {

**if** (arr[i] % 2 == 0) {

arr[i] = 0;

}

}

std::cout << "Замінено елементи, значення яких парні, на нулі: ";

**for** (**int** i = 0; i < n; i++) {

std::cout << arr[i] << " ";

}

**return** 0;

}

6)  Напишіть програму, що заміняє нулями всі елементи, значення яких виражене непарним числом, заданого(переліком елементів) не порожнього масиву.

#include <iostream>

**int** main() {

**int** arr[] = {8, 15, 3, 12, 7, 18};

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

**for** (**int** i = 0; i < n; i++) {

**if** (arr[i] % 2 != 0) {

arr[i] = 0;

}

}

std::cout << "Замінено елементи, значення яких непарні, на нулі: ";

**for** (**int** i = 0; i < n; i++) {

std::cout << arr[i] << " ";

}

**return** 0;

}

7)  Напишіть програму, що заміняє нулями всі елементи, що розташовуються на парних позиціях, заданого(переліком елементів) не порожнього масиву.

#include <iostream>

**int** main() {

**int** arr[] = {10, 20, 30, 40, 50, 60};

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

**for** (**int** i = 0; i < n; i += 2) {

arr[i] = 0;

}

std::cout << "Замінено елементи на парних позиціях на нулі: ";

**for** (**int** i = 0; i < n; i++) {

std::cout << arr[i] << " ";

}

**return** 0;

}

8)  Напишіть програму, що заміняє нулями всі елементи, що розташовуються на непарних позиціях, заданого(переліком елементів) не порожнього масиву.

#include <iostream>

**int** main() {

**int** arr[] = {10, 20, 30, 40, 50, 60};

**int** n = **sizeof**(arr) / **sizeof**(arr[0]);

**for** (**int** i = 1; i < n; i += 2) { // Починаючи з індексу 1 (непарні позиції)

arr[i] = 0;

}

std::cout << "Замінено елементи на непарних позиціях на нулі: ";

**for** (**int** i = 0; i < n; i++) {

std::cout << arr[i] << " ";

}

**return** 0;

}

Завдання 3: Заповнення масиву випадковими значеннями. Відшукання максимального та мінімального елемента.

1. Заповніть одновимірний масив розміру 10 випадковими значеннями з відрізка [−30; 35]. Виведіть максимальний елемент.

#include <iostream>

#include <ctime> // Для генерації випадкових чисел

#include <cstdlib> // Для функцій rand() і srand()

**int** main() {

**const** **int** arraySize = 10;

**int** array[arraySize];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо масив випадковими числами з відрізка [-30; 35]

**for** (**int** i = 0; i < arraySize; i++) {

array[i] = std::rand() % 66 - 30; // Генеруємо від -30 до 35

}

// Знайдемо максимальний елемент у масиві

**int** maxElement = array[0];

**for** (**int** i = 1; i < arraySize; i++) {

**if** (array[i] > maxElement) {

maxElement = array[i];

}

}

// Виводимо масив та максимальний елемент

std::cout << "Елементи масиву: ";

**for** (**int** i = 0; i < arraySize; i++) {

std::cout << array[i] << " ";

}

std::cout << "\nМаксимальний елемент: " << maxElement << std::endl;

**return** 0;

}

1. Заповніть одновимірний масив розміру 10 випадковими значеннями з відрізка [−10; 13]. Виведіть максимальний елемент.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** arraySize = 10;

**int** array[arraySize];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо масив випадковими числами з відрізка [-10; 13]

**for** (**int** i = 0; i < arraySize; i++) {

array[i] = std::rand() % 24 - 10; // Генеруємо від -10 до 13

}

// Знайдемо максимальний елемент у масиві

**int** maxElement = array[0];

**for** (**int** i = 1; i < arraySize; i++) {

**if** (array[i] > maxElement) {

maxElement = array[i];

}

}

// Виведемо масив та максимальний елемент

std::cout << "Елементи масиву: ";

**for** (**int** i = 0; i < arraySize; i++) {

std::cout << array[i] << " ";

}

std::cout << "\nМаксимальний елемент: " << maxElement << std::endl;

**return** 0;

}

3)  Заповніть одновимірний масив розміру 10 випадковими значеннями з відрізка [−15; 13]. Виведіть максимальний елемент.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** arraySize = 10;

**int** array[arraySize];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо масив випадковими числами з відрізка [-15; 13]

**for** (**int** i = 0; i < arraySize; i++) {

array[i] = std::rand() % 29 - 15; // Генеруємо від -15 до 13

}

// Знайдемо максимальний елемент у масиві

**int** maxElement = array[0];

**for** (**int** i = 1; i < arraySize; i++) {

**if** (array[i] > maxElement) {

maxElement = array[i];

}

}

// Виведемо масив та максимальний елемент

std::cout << "Елементи масиву: ";

**for** (**int** i = 0; i < arraySize; i++) {

std::cout << array[i] << " ";

}

std::cout << "\nМаксимальний елемент: " << maxElement << std::endl;

**return** 0;

}

1. Заповніть одновимірний масив розміру 10 випадковими значеннями з відрізка [−20; 15]. Виведіть максимальний елемент.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** arraySize = 10;

**int** array[arraySize];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо масив випадковими числами з відрізка [-20; 15]

**for** (**int** i = 0; i < arraySize; i++) {

array[i] = std::rand() % 36 - 20; // Генеруємо від -20 до 15

}

// Знайдемо максимальний елемент у масиві

**int** maxElement = array[0];

**for** (**int** i = 1; i < arraySize; i++) {

**if** (array[i] > maxElement) {

maxElement = array[i];

}

}

// Виведемо масив та максимальний елемент

std::cout << "Елементи масиву: ";

**for** (**int** i = 0; i < arraySize; i++) {

std::cout << array[i] << " ";

}

std::cout << "\nМаксимальний елемент: " << maxElement << std::endl;

**return** 0;

}

5)  Заповніть одновимірний масив розміру 10 випадковими значеннями з відрізка [−30; 35]. Виведіть мінімальний елемент.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** arraySize = 10;

**int** array[arraySize];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо масив випадковими числами з відрізка [-30; 35]

**for** (**int** i = 0; i < arraySize; i++) {

array[i] = std::rand() % 66 - 30; // Генеруємо від -30 до 35

}

// Знайдемо мінімальний елемент у масиві

**int** minElement = array[0];

**for** (**int** i = 1; i < arraySize; i++) {

**if** (array[i] < minElement) {

minElement = array[i];

}

}

// Виведемо масив та мінімальний елемент

std::cout << "Елементи масиву: ";

**for** (**int** i = 0; i < arraySize; i++) {

std::cout << array[i] << " ";

}

std::cout << "\nМінімальний елемент: " << minElement << std::endl;

**return** 0;

}

6)  Заповніть одновимірний масив розміру 10 випадковими значеннями з відрізка [−10; 13]. Виведіть мінімальний елемент.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** arraySize = 10;

**int** array[arraySize];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо масив випадковими числами з відрізка [-10; 13]

**for** (**int** i = 0; i < arraySize; i++) {

array[i] = std::rand() % 24 - 10; // Генеруємо від -10 до 13

}

// Знайдемо мінімальний елемент у масиві

**int** minElement = array[0];

**for** (**int** i = 1; i < arraySize; i++) {

**if** (array[i] < minElement) {

minElement = array[i];

}

}

// Виведемо масив та мінімальний елемент

std::cout << "Елементи масиву: ";

**for** (**int** i = 0; i < arraySize; i++) {

std::cout << array[i] << " ";

}

std::cout << "\nМінімальний елемент: " << minElement << std::endl;

**return** 0;

}

7)  Заповніть одновимірний масив розміру 10 випадковими значеннями з відрізка [−15; 13]. Виведіть мінімальний елемент.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** arraySize = 10;

**int** array[arraySize];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо масив випадковими числами з відрізка [-15; 13]

**for** (**int** i = 0; i < arraySize; i++) {

array[i] = std::rand() % 29 - 15; // Генеруємо від -15 до 13

}

// Знайдемо мінімальний елемент у масиві

**int** minElement = array[0];

**for** (**int** i = 1; i < arraySize; i++) {

**if** (array[i] < minElement) {

minElement = array[i];

}

}

// Виведемо масив та мінімальний елемент

std::cout << "Елементи масиву: ";

**for** (**int** i = 0; i < arraySize; i++) {

std::cout << array[i] << " ";

}

std::cout << "\nМінімальний елемент: " << minElement << std::endl;

**return** 0;

}

8)  Заповніть одновимірний масив розміру 10 випадковими значеннями з відрізка [−20; 15]. Виведіть мінімальний елемент.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** arraySize = 10;

**int** array[arraySize];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо масив випадковими числами з відрізка [-20; 15]

**for** (**int** i = 0; i < arraySize; i++) {

array[i] = std::rand() % 36 - 20; // Генеруємо від -20 до 15

}

// Знайдемо мінімальний елемент у масиві

**int** minElement = array[0];

**for** (**int** i = 1; i < arraySize; i++) {

**if** (array[i] < minElement) {

minElement = array[i];

}

}

// Виведемо масив та мінімальний елемент

std::cout << "Елементи масиву: ";

**for** (**int** i = 0; i < arraySize; i++) {

std::cout << array[i] << " ";

}

std::cout << "\nМінімальний елемент: " << minElement << std::endl;

**return** 0;

}

Завдання 4: Задання двовимірних масивів. Знаходження кількості елементів по рядках та стовпцях.

1)  Задати двомірний масив 𝑚 × 𝑛 цілих чисел 𝑚 = 4, 𝑛 = 3випадковими значеннями. Визначити кількість парних чисел у кожному рядку.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** m = 4; // Кількість рядків

**const** **int** n = 3; // Кількість стовпців

**int** matrix[m][n];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо двовимірний масив випадковими числами

**for** (**int** i = 0; i < m; i++) {

**for** (**int** j = 0; j < n; j++) {

matrix[i][j] = std::rand() % 101; // Генеруємо випадкове число від 0 до 100

}

}

// Виведемо двовимірний масив та кількість парних чисел у кожному рядку

**for** (**int** i = 0; i < m; i++) {

std::cout << "Рядок " << i << ": ";

**int** evenCount = 0; // Лічильник парних чисел у рядку

**for** (**int** j = 0; j < n; j++) {

std::cout << matrix[i][j] << " ";

**if** (matrix[i][j] % 2 == 0) {

evenCount++;

}

}

std::cout << " (Парних чисел: " << evenCount << ")\n";

}

**return** 0;

}

2)  Задати двомірний масив 𝑚 × 𝑛 цілих чисел 𝑚 = 6, 𝑛 = 4 випадковими значеннями. Визначити кількість парних чисел у кожному стовпчику.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** m = 6; // Кількість рядків

**const** **int** n = 4; // Кількість стовпців

**int** matrix[m][n];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо двовимірний масив випадковими числами

**for** (**int** i = 0; i < m; i++) {

**for** (**int** j = 0; j < n; j++) {

matrix[i][j] = std::rand() % 101; // Генеруємо випадкове число від 0 до 100

}

}

// Виведемо двовимірний масив та кількість парних чисел у кожному стовпчику

**for** (**int** j = 0; j < n; j++) {

std::cout << "Стовпчик " << j << ": ";

**int** evenCount = 0; // Лічильник парних чисел у стовпчику

**for** (**int** i = 0; i < m; i++) {

std::cout << matrix[i][j] << " ";

**if** (matrix[i][j] % 2 == 0) {

evenCount++;

}

}

std::cout << " (Парних чисел: " << evenCount << ")\n";

}

**return** 0;

}

3)  Задати двомірний масив 𝑚 × 𝑛 цілих чисел 𝑚 =4, 𝑛 =3 випадковими значеннями. Визначити кількість непарних чисел у кожному рядку.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** m = 4; // Кількість рядків

**const** **int** n = 3; // Кількість стовпців

**int** matrix[m][n];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо двовимірний масив випадковими числами

**for** (**int** i = 0; i < m; i++) {

**for** (**int** j = 0; j < n; j++) {

matrix[i][j] = std::rand() % 101; // Генеруємо випадкове число від 0 до 100

}

}

// Виведемо двовимірний масив та кількість непарних чисел у кожному рядку

**for** (**int** i = 0; i < m; i++) {

std::cout << "Рядок " << i << ": ";

**int** oddCount = 0; // Лічильник непарних чисел у рядку

**for** (**int** j = 0; j < n; j++) {

std::cout << matrix[i][j] << " ";

**if** (matrix[i][j] % 2 != 0) {

oddCount++;

}

}

std::cout << " (Непарних чисел: " << oddCount << ")\n";

}

**return** 0;

}

4)  Задати двомірний масив 𝑚 × 𝑛 цілих чисел 𝑚 = 6, 𝑛 = 4 випадковими значеннями. Визначити кількість непарних чисел у кожному стовпчику.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** m = 6; // Кількість рядків

**const** **int** n = 4; // Кількість стовпців

**int** matrix[m][n];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо двовимірний масив випадковими числами

**for** (**int** i = 0; i < m; i++) {

**for** (**int** j = 0; j < n; j++) {

matrix[i][j] = std::rand() % 101; // Генеруємо випадкове число від 0 до 100

}

}

// Виведемо двовимірний масив та кількість непарних чисел у кожному стовпчику

**for** (**int** j = 0; j < n; j++) {

std::cout << "Стовпчик " << j << ": ";

**int** oddCount = 0; // Лічильник непарних чисел у стовпчику

**for** (**int** i = 0; i < m; i++) {

std::cout << matrix[i][j] << " ";

**if** (matrix[i][j] % 2 != 0) {

oddCount++;

}

}

std::cout << " (Непарних чисел: " << oddCount << ")\n";

}

**return** 0;

}

5)  Задати двомірний масив 𝑚 × 𝑛 цілих чисел 𝑚 = 3, 𝑛 = 4випадковими значеннями. Визначити кількість парних чисел у кожному рядку.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** m = 3; // Кількість рядків

**const** **int** n = 4; // Кількість стовпців

**int** matrix[m][n];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо двовимірний масив випадковими числами

**for** (**int** i = 0; i < m; i++) {

**for** (**int** j = 0; j < n; j++) {

matrix[i][j] = std::rand() % 101; // Генеруємо випадкове число від 0 до 100

}

}

// Виведемо двовимірний масив та кількість парних чисел у кожному рядку

**for** (**int** i = 0; i < m; i++) {

std::cout << "Рядок " << i << ": ";

**int** evenCount = 0; // Лічильник парних чисел у рядку

**for** (**int** j = 0; j < n; j++) {

std::cout << matrix[i][j] << " ";

**if** (matrix[i][j] % 2 == 0) {

evenCount++;

}

}

std::cout << " (Парних чисел: " << evenCount << ")\n";

}

**return** 0;

}

6)  Задати двомірний масив 𝑚 × 𝑛 цілих чисел 𝑚 = 4, 𝑛 = 6 випадковими значеннями. Визначити кількість парних чисел у кожному стовпчику.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** m = 4; // Кількість рядків

**const** **int** n = 6; // Кількість стовпців

**int** matrix[m][n];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо двовимірний масив випадковими числами

**for** (**int** i = 0; i < m; i++) {

**for** (**int** j = 0; j < n; j++) {

matrix[i][j] = std::rand() % 101; // Генеруємо випадкове число від 0 до 100

}

}

// Виведемо двовимірний масив та кількість парних чисел у кожному стовпчику

**for** (**int** j = 0; j < n; j++) {

std::cout << "Стовпчик " << j << ": ";

**int** evenCount = 0; // Лічильник парних чисел у стовпчику

**for** (**int** i = 0; i < m; i++) {

std::cout << matrix[i][j] << " ";

**if** (matrix[i][j] % 2 == 0) {

evenCount++;

}

}

std::cout << " (Парних чисел: " << evenCount << ")\n";

}

**return** 0;

}

7)  Задати двомірний масив 𝑚 × 𝑛 цілих чисел 𝑚 = 3, 𝑛 = 4випадковими значеннями. Визначити кількість непарних чисел у кожному рядку.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** m = 3; // Кількість рядків

**const** **int** n = 4; // Кількість стовпців

**int** matrix[m][n];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо двовимірний масив випадковими числами

**for** (**int** i = 0; i < m; i++) {

**for** (**int** j = 0; j < n; j++) {

matrix[i][j] = std::rand() % 101; // Генеруємо випадкове число від 0 до 100

}

}

// Виведемо двовимірний масив та кількість непарних чисел у кожному рядку

**for** (**int** i = 0; i < m; i++) {

std::cout << "Рядок " << i << ": ";

**int** oddCount = 0; // Лічильник непарних чисел у рядку

**for** (**int** j = 0; j < n; j++) {

std::cout << matrix[i][j] << " ";

**if** (matrix[i][j] % 2 != 0) {

oddCount++;

}

}

std::cout << " (Непарних чисел: " << oddCount << ")\n";

}

**return** 0;

}

8)  Задати двомірний масив 𝑚 × 𝑛 цілих чисел 𝑚 = 4, 𝑛 = 6 випадковими значеннями. Визначити кількість непарних чисел у кожному стовпчику.

#include <iostream>

#include <ctime>

#include <cstdlib>

**int** main() {

**const** **int** m = 4; // Кількість рядків

**const** **int** n = 6; // Кількість стовпців

**int** matrix[m][n];

// Ініціалізуємо генератор випадкових чисел

std::srand(**static\_cast**<**unsigned**>(std::time(0)));

// Заповнюємо двовимірний масив випадковими числами

**for** (**int** i = 0; i < m; i++) {

**for** (**int** j = 0; j < n; j++) {

matrix[i][j] = std::rand() % 101; // Генеруємо випадкове число від 0 до 100

}

}

// Виведемо двовимірний масив та кількість непарних чисел у кожному стовпчику

**for** (**int** j = 0; j < n; j++) {

std::cout << "Стовпчик " << j << ": ";

**int** oddCount = 0; // Лічильник непарних чисел у стовпчику

**for** (**int** i = 0; i < m; i++) {

std::cout << matrix[i][j] << " ";

**if** (matrix[i][j] % 2 != 0) {

oddCount++;

}

}

std::cout << " (Непарних чисел: " << oddCount << ")\n";

}

**return** 0;

}