Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 14

По дисциплине «Основы алгоритмизации и программирования»

На тему «Многомерные массивы»

Выполнила:

Студентка 1 курса 7 группы

Бабич Виолетта Станиславовна

Преподаватель: асс. Андронова М.В.

2023, Минск

|  |  |
| --- | --- |
| **2** | 1. В каждой строке заданной матрицы **A(n, m)** вычислить сумму, количество и среднее арифметическое положительных элементов.  2. Определить количество различных элементов матрицы размером **n**x**m** (повторяющиеся элементы считать один раз). |

№1

#include <iostream>

#include <iomanip>

using namespace std;

// Функция calculateRowStats принимает указатель на массив row и его размер size

void calculateRowStats(int\* row, int size) {

int sum = 0;

int count = 0;

// Проходим по каждому элементу массива

for (int i = 0; i < size; ++i) {

// Если текущий элемент положителен

if (row[i] > 0) {

sum += row[i];

++count;

}

}

// Если есть хотя бы один положительный элемент

if (count > 0) {

// Вычисляем среднее арифметическое

double average = static\_cast<double>(sum) / count;

// Выводим результаты на экран

cout << "Сумма положительных элементов: " << sum << endl;

cout << "Количество положительных элементов: " << count << endl;

cout << "Среднее арифметическое положительных элементов: " << fixed << setprecision(2) << average << endl;

}

else {

// Если положительных элементов нет

cout << "В данной строке нет положительных элементов." << endl;

}

}

// Функция main - точка входа в программу

int main() {

setlocale(LC\_ALL, "rus");

int n, m;

// Вводим количество строк матрицы

cout << "Введите количество строк матрицы: ";

cin >> n;

// Вводим количество столбцов матрицы

cout << "Введите количество столбцов матрицы: ";

cin >> m;

// Выделяем память под двумерный массив (матрицу)

int\*\* matrix = new int\* [n];

// Заполняем матрицу значениями, вводимыми с клавиатуры

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

matrix[i] = new int[m];

cout << "Введите элементы строки " << i + 1 << " (разделители - пробелы): ";

for (int j = 0; j < m; ++j) {

cin >> matrix[i][j];

}

}

// Для каждой строки матрицы вызываем функцию calculateRowStats

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

cout << "Строка " << i + 1 << ": ";

calculateRowStats(matrix[i], m);

cout << endl;

}

// Освобождение выделенной памяти

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

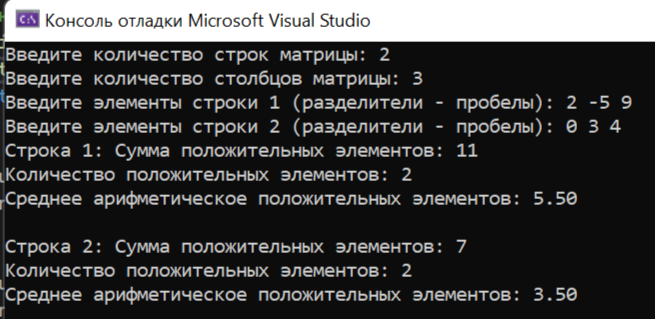
delete[] matrix[i];

}

delete[] matrix;

return 0;

}



№2

#include <iostream>

using namespace std;

int countDistinctElements(int matrix[][100], int n, int m) {

// создаем флаг для того, чтобы отмечать посещённые элементы матрицы

bool visited[100][100] = { false };

int distinctCount = 0;

// проходим по всем строкам и столбцам матрицы

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

for (int j = 0; j < m; ++j) {

// если элемен еще не посещен

if (!visited[i][j]) {

// отмечаем, что он посещён

visited[i][j] = true;

// уведичиваем значение не повторяющихся элементов

++distinctCount;

// если этот эелемент уже был посещен, то мы его прпускаем

for (int x = i; x < n; ++x) {

for (int y = j; y < m; ++y) {

if (matrix[x][y] == matrix[i][j]) {

visited[x][y] = true;

}

}

}

}

}

}

return distinctCount;

}

int main(){

setlocale(LC\_ALL, "rus");

int n, m;

cout << "Введите кол-во строк в массиве (n): ";

cin >> n;

cout << "Введите кол-во столбцов в массиве (m): ";

cin >> m;

// вводим элементы массива

int matrix[100][100];

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

for (int j = 0; j < m; ++j) {

cin >> matrix[i][j];

}

}

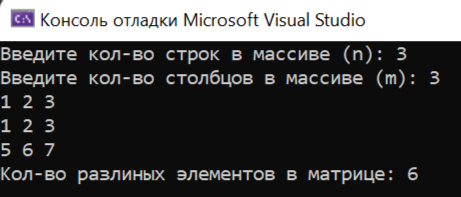
// вычисляем и выводим результат

int distinctCount = countDistinctElements(matrix, n, m);

cout << "Кол-во разлиных элементов в матрице: " << distinctCount << endl;

return 0;

}



**Дополнительные задания**

№1Дана квадратная матрица порядка **2n**, элементы которой формируются случайным образом и находятся в пределах от −10 до 10. Получить новую матрицу, переставляя ее блоки размера **n×n** в соответствии со схемой.

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

// вводим константы для определения диапазона значений в матрице

const int MAX\_VALUE = 10;

const int MIN\_VALUE = -10;

// функция для генерации случайных элементов матрицы

void generateRandomMatrix(int matrix[][20], int n) {

srand(time(0)); // инициализация генератора случайных чисел

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

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

// генерация случайного числа в заданном диапазоне

matrix[i][j] = rand() % (MAX\_VALUE - MIN\_VALUE + 1) + MIN\_VALUE;

}

}

}

// функция для вывода матрицы на экран

void printMatrix(int matrix[][20], int rows, int cols) {

for (int i = 0; i < rows; ++i) {

for (int j = 0; j < cols; ++j) {

cout << matrix[i][j] << "\t"; // Вывод элемента матрицы с табуляцией

}

cout << endl; // Переход на новую строку после каждой строки матрицы

}

}

// Функция для перестановки блоков размером n×n

void swapBlocks(int matrix[][20], int n) {

// Меняем левый верхний блок с правым нижним

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

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

swap(matrix[i][j], matrix[i + n][j + n]); // Используем swap для обмена элементов

}

}

// Меняем левый нижний блок с правым верхним

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

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

swap(matrix[i][j], matrix[i - n][j + n]); // Используем swap для обмена элементов

}

}

}

int main() {

setlocale(LC\_ALL, "rus");

int n;

cout << "Введите значение n: ";

cin >> n;

int matrix[20][20];

generateRandomMatrix(matrix, n);

cout << "Исходная матрица:" << endl;

printMatrix(matrix, 2 \* n, 2 \* n); //вывод исходной матрицы ("2\*n" пишем для того, чтобы мы могли перемещать блоки размером n\*n)

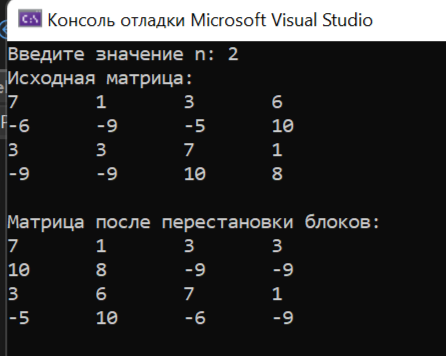
swapBlocks(matrix, n);//перемещаем блоки по диоганали

cout << "\nМатрица после перестановки блоков:" << endl;

printMatrix(matrix, 2 \* n, 2 \* n);

return 0;

}



№2 Латинским квадратом порядка **n** называется квадратная таблица размером **nхn**, каждая строка и каждый столбец которой содержат все числа от 1 до **n**. Для заданного **n** в матрице **L(n, n**) построить латинский квадрат порядка **n**.

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

int A[200][200];

int n, t;

cout << "Введите n: ";

cin >> n;

//создаем матрицу, в которой каждый следуйщий элемент будет на единицу больше предыдущего, но не будет превышать n

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

t = i + 1;

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

A[i][j] = t;

t++;

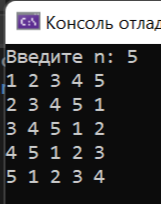
if (t > n) t = 1; //если t>n, то присваем значение 1

cout << A[i][j] << " ";

}

cout << endl;

}

}

№3 Путем перестановки элементов квадратной вещественной матрицы добиться того, чтобы ее максимальный элемент находился в левом верхнем углу, следующий по величине в позиции (2, 2), следующий в позиции (3, 3) и т. д., заполнив таким образом всю главную диагональ.

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

const int n = 5;

int A[n][n] = {

{1, 2, 4, 999, 6},

{111, 7, 8, 9, 1},

{8, 8, 555, 6, 5},

{666, 5, 3, 2, 1},

{888, 1, 2, 3, 4}

};

int max[n];

// Перебор строк для установки максимальных элементов на главной диагонали

for (int l = 0; l < n; l++) {

max[l] = A[l][l]; // Инициализация max[l] значением на главной диагонали

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

for (int s = 0; s < n; s++) {

// Поиск максимального элемента в матрице

if (A[i][s] > max[l]) {

bool isUnique = true;

// Проверка, что максимальное значение уникально в пределах строки

for (int m = 0; m < n; m++) {

if (max[m] == A[i][s]) {

isUnique = false;

}

}

//меняем местами переменные

if (isUnique) {

max[l] = A[i][s]; // Исправление: присваивание значения max[l]

swap(A[i][s], A[l][l]);

}

}

}

}

}

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

for (int s = 0; s < n; s++) {

cout << A[i][s] << ' ';

}

cout << '\n';

}

return 0;

}

