Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ОТЧЁТ

к лабораторной работе №4

на тему

МНОГОМЕРНЫЕ МАССИВЫ

ВАРИАНТ 1

Проверил Выполнил

асс. каф. ЭВМ студент гр. 318302 П.С. Жуковец В.С.Бахур

МИНСК 2023

**1** **ЗАДАНИЕ №1**

**1.1 Условие**

Двумерный массив, содержащий равное число строк и столбцов, называется магическим квадратом, если суммы чисел, записанных в каждой строке, каждом столбце и каждой из двух больших диагоналей, равны одному и тому же числу. Определить, является ли данный массив А из n строк и n столбцов магическим квадратом.

**1.2 Алгоритм решения**

Алгоритм предоставлен в «Приложении A»; Рисунок 1.1 (стр.7)

**1.3 Листинг программы**

/\* Двумерный массив, содержащий равное число строк и столбцов, называется магическим квадратом,

если суммы чисел, записанных в каждой строке, каждом столбце и каждой из двух больших диагоналей,

равны одному и тому же числу.

Определить, является ли данный массив А из n строк и n столбцов магическим квадратом.\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#define MAX\_SIZE 100

// Функция для проверки, является ли массив магическим квадратом

int isMagicSquare(int arr[MAX\_SIZE][MAX\_SIZE], int elQuantity) {

// Вычисление суммы для первой строки

int sum = 0;

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

sum += arr[0][j];

}

// Проверка суммы по строкам

for (int i = 1; i < elQuantity; ++i) {

int rowSum = 0;

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

rowSum += arr[i][j];

}

if (rowSum != sum) {

return 0; // Не является магическим квадратом

}

}

// Проверка суммы по столбцам

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

int colSum = 0;

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

colSum += arr[i][j];

}

if (colSum != sum) {

return 0; // Не является магическим квадратом

}

}

// Проверка суммы по главной диагонали

int mainDiagonalSum = 0;

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

mainDiagonalSum += arr[i][i];

}

if (mainDiagonalSum != sum) {

return 0; // Не является магическим квадратом

}

// Проверка суммы по побочной диагонали

int secondaryDiagonalSum = 0;

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

secondaryDiagonalSum += arr[i][elQuantity - i - 1];

}

if (secondaryDiagonalSum != sum) {

return 0; // Не является магическим квадратом

}

return 1; // Является магическим квадратом

}

int main() {

setlocale(LC\_ALL, "Rus");

int elQuantity;

// Ввод размеров массива с клавиатуры

printf("Введите размер матрицы (<= %d): ", MAX\_SIZE);

scanf\_s("%d", &elQuantity);

if (elQuantity > MAX\_SIZE) {

printf("Неверный ввод!\n");

return 1;

}

int array[MAX\_SIZE][MAX\_SIZE];

// Ввод элементов массива с клавиатуры

printf("Введите элементы матрицы:\n");

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

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

scanf\_s("%d", &array[i][j]);

}

}

// Проверка является ли массив магическим квадратом

if (isMagicSquare(array, elQuantity)) {

printf("Эта матрица является магическим квадратом!.\n");

}

else {

printf("Эта матрица не является магическим квадратом.\n");

}

return 0;

}

**2 ЗАДАНИЕ №2**

**2.1 Условие**

Найти в матрице первую строку, все элементы которой положительны, и сумму этих элементов. Уменьшить все элементы матрицы на эту сумму.

**2.2 Алгоритм решения**

Алгоритм предоставлен в «Приложении Б»; Рисунок 1.2 (стр.8-9)

**2.3 Листинг программы**

/\*Найти в матрице первую строку, все элементы которой положительны,

и сумму этих элементов. Уменьшить все элементы матрицы на эту сумму.\*/

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

void initializeMatrix(int\*\* matrix, int rows, int cols) {

printf("Введите значения для матрицы %d x %d:\n", rows, cols);

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

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

printf("Элемент [%d][%d]: ", i + 1, j + 1);

// Проверка на ввод символов

if (scanf\_s("%d", &matrix[i][j]) != 1) {

printf("Ошибка ввода. Пожалуйста, введите целое число.\n");

exit(EXIT\_FAILURE);

}

}

}

}

void findPositiveRowAndReduceMatrix(int\*\* matrix, int rows, int cols) {

int positiveRow = -1;

int sum = 0;

// Находим первую строку, все элементы которой положительны

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

int allPositive = 1;

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

if (matrix[i][j] <= 0) {

allPositive = 0;

break; //прерывания выполнения цикла

}

}

if (allPositive) {

positiveRow = i;

break; //прерывания выполнения цикла

}

}

// Если такая строка найдена, вычисляем сумму элементов этой строки

if (positiveRow != -1) {

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

sum += matrix[positiveRow][j];

}

// Уменьшаем все элементы матрицы на сумму

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

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

matrix[i][j] -= sum;

}

}

// Выводим результат

printf("Найдена строка с положительными элементами (строка %d)\n", positiveRow + 1);

printf("Сумма элементов этой строки: %d\n", sum);

printf("Матрица после уменьшения:\n");

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

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

printf("%d\t", matrix[i][j]);

}

printf("\n");

}

}

else {

printf("В матрице нет строки, все элементы которой положительны.\n");

}

}

int main() {

setlocale(LC\_ALL, "Rus");

int rows, cols;

printf("Введите количество строк матрицы: ");

scanf\_s("%d", &rows);

printf("Введите количество столбцов матрицы: ");

scanf\_s("%d", &cols);

// Выделяем память для матрицы

int\*\* matrix = (int\*\*)malloc(rows \* sizeof(int\*));

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

matrix[i] = (int\*)malloc(cols \* sizeof(int));

}

initializeMatrix(matrix, rows, cols);

printf("Исходная матрица:\n");

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

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

printf("%d\t", matrix[i][j]);

}

printf("\n");

}

findPositiveRowAndReduceMatrix(matrix, rows, cols);

// Освобождаем выделенную память

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

free(matrix[i]);

}

free(matrix);

return 0;

}

**3 ЗАДАНИЕ №3**

* 1. **Условие**

В квадратной матрице размером NxN найти сумму элементов в 1-ой области.

**3.2 Алгоритм решения**

Алгоритм предоставлен в «Приложении В»; Рисунок 1.3

* 1. **Листинг программы**

include<stdio.h>

#include<conio.h>

#include<locale.h>

#include<stdlib.h>

int main()

{

int i, n, m, A, B, count = 0, max, choice, index, sum = 0, input;

setlocale(LC\_ALL, "Russian");

start:

printf("Введите кол-во строк и столбцов матрицы\n");

scanf\_s("%d\t%d", &n, &m);

if (n <= 0 || m <= 0) {

printf("Некорректный ввод\n");

goto start;

}

int mas[100][100];

srand(time(0));

Start:

printf("Выберите способ заполнения массива 1: вручную, 2: рандом\t");

scanf\_s("%d", &choice);

if (choice == 2) {

printf("Сгенерированный массив:\n");

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

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

mas[i][j] = rand() % (0 + 1 - (-10)) + (-10); // диапазон - от 0 до -10

}

}

else if (choice == 1) {

printf("Введите %d элемента(ов)", m \* n);

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

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

scanf\_s("%d", &mas[i][j]);

}

}

else {

printf("Некорректный ввод\n");

getchar() != "\n";

goto Start;

}

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

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

{

printf("[%d][%d]=%d \t", i, j, mas[i][j]);

}

printf("\n");

}

if (n % 2 == 0 && m % 2 == 0) {

for (int k = 0; k < n / 2 - 1; k++) {

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

sum += mas[k][cc];

}

}

for (i = n / 2 - 1; i <= n / 2 - 1; i++) {

for (int j = m / 2 - i; j < m / 2 + i; j++) {

sum += mas[i][j];

}

}

}

else if (n % 2 == 1 && m % 2 == 1) {

for (int k = 0; k < n / 2; k++) {

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

sum += mas[k][cc];

}

}

for (i = n / 2; i <= n / 2; i++) {

for (int j = m / 2 + 1 - i; j < m / 2 + i; j++) {

sum += mas[i][j];

}

}

}

else {

printf("Матрица не квадратная\n");

goto start;

}

printf("\nCумма элементов в первой области - %d", sum);

}

**Приложение А**

Блок-схема алгоритма 1

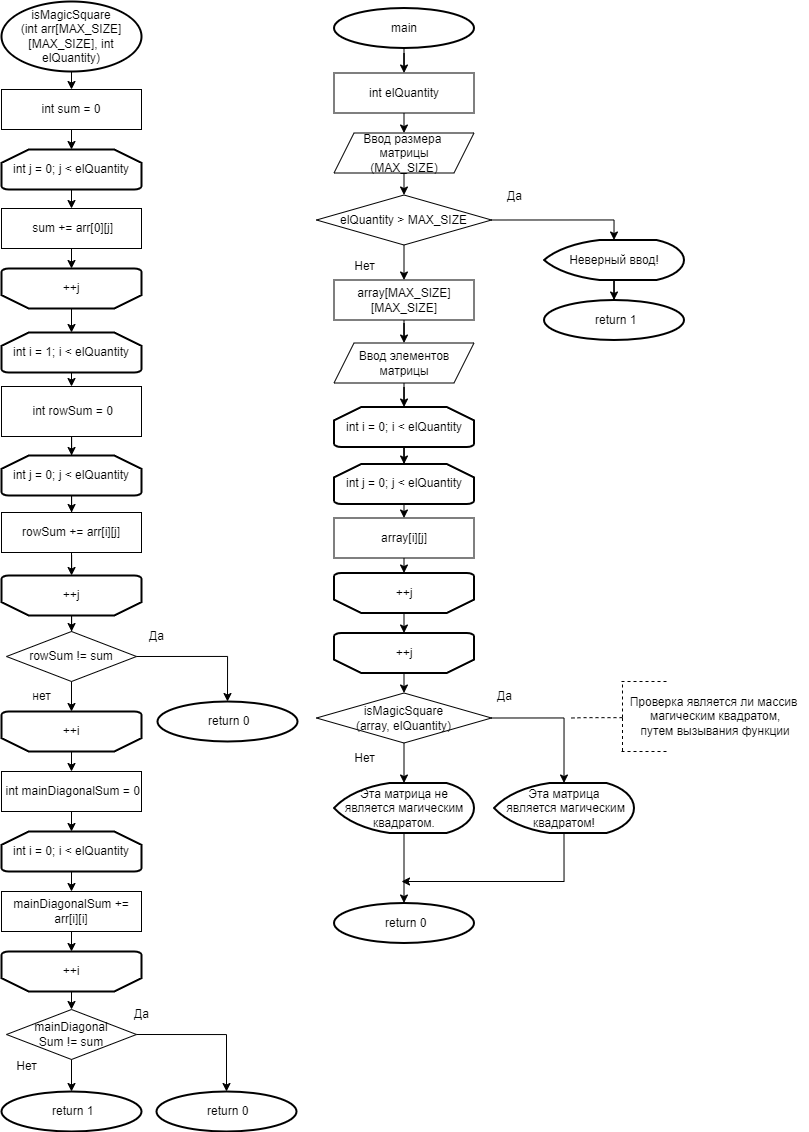
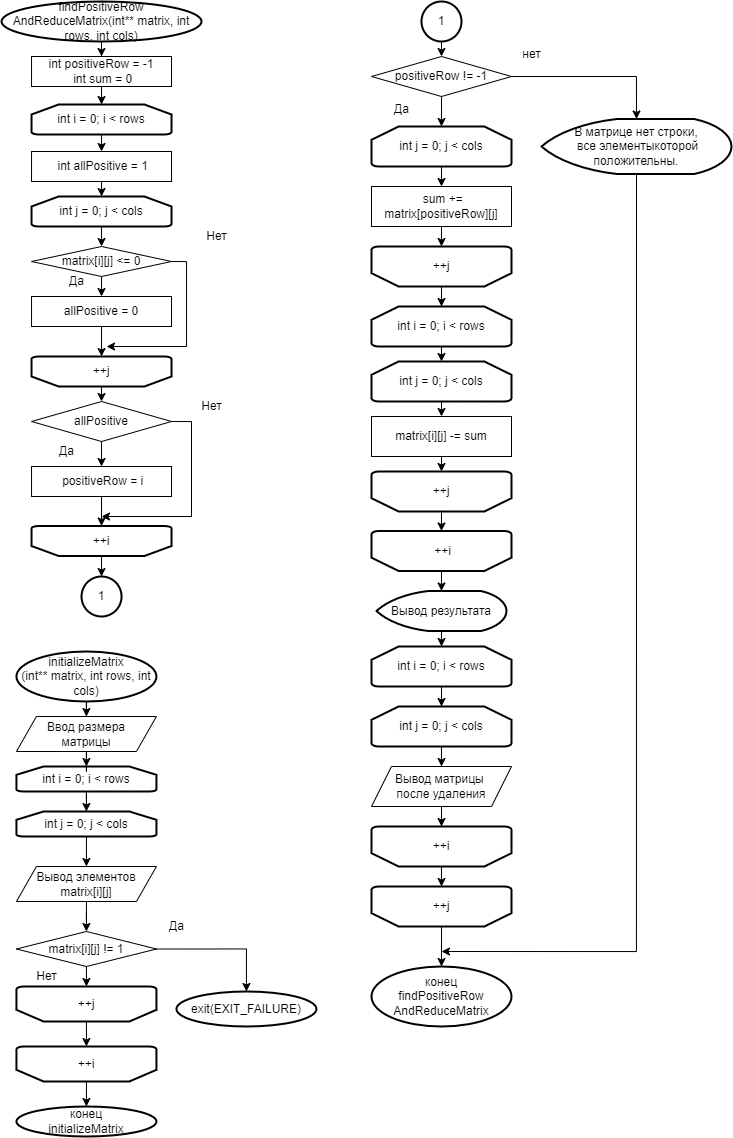


Рисунок 1.1 – Блок-схема №1

**Приложение Б**

Блок-схема алгоритма 2 

(*продолжение на стр. 9*)

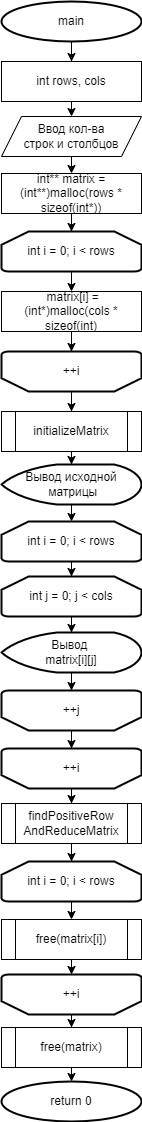


Рисунок 1.2 – Блок-схема №2

**Приложение В**

Блок-схема алгоритма 3

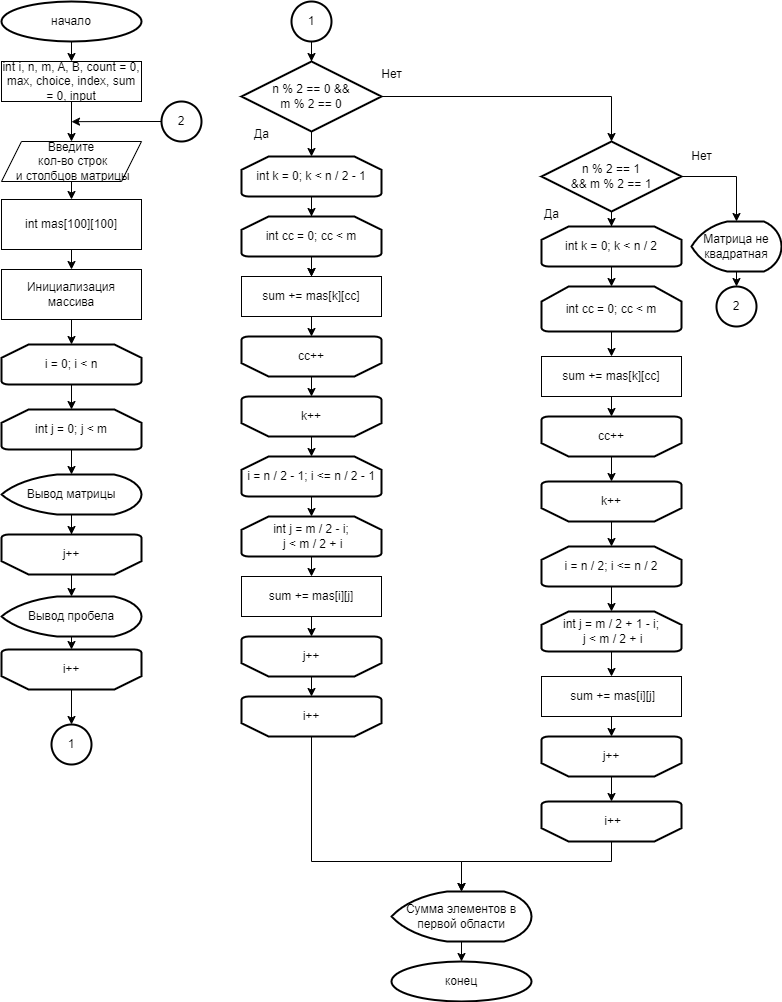


Рисунок 1.3 – Блок-схема №3