**4 ЛАБОРАТОРНАЯ РАБОТА №4**

**«****ОБРАБОТКА ДВУМЕРНЫХ МАССИВОВ С ПОМОЩЬЮ ФУНКЦИЙ»**

**4.1 Цель работы**

Изучить основные принципы работы с массивами в языках С/С++. Исследовать способы передачи параметров в функции.

**4.2 Вариант задания – 20**

Требуется оформить программу в виде функций, выполняющих:

- выделение памяти и заполнения массива в ручную;

- определить количество и сумму нечетных отрицательных элементов;

- соседями элемента aij в матрице A назовем элементы аkl, для которых верно следующее: (i-1)≤k≤(i+1), (j-1)≤l≤(j+1). Элемент матрицы называется локальным максимумом, если он строго больше всех имеющихся у него соседей. Подсчитать количество локальных максимумов;

- вывод массива на экран;

- выход из программы с очисткой памяти;

Вызов функций осуществлять из интерактивного меню. Все необходимые данные для функций должны передаваться им в качестве параметров.

**4.3 Порядок выполнения работы**

4.3.1 Разработать алгоритм решения задачи.

4.3.2 Разработать структурную схему алгоритма решения задачи.

4.3.3 Разработать программу на языке Си.

4.3.4 Разработать тестовые примеры. Выполнить тестирование используя разработанные тестовые примеры. Выполнить отладку

4.3.6 Сделать выводы по проделанной работе.

**4.4 Ход работы**

4.4.1 Для задания был разработан алгоритм решения задачи:

Для программs была подключена библиотека <stdio.h> для ввода и вывода информации и библиотека <locale.h> для поддержки русского языка. Далее перечисляются прототипы функций. После идёт функция main в которой объявляется что делает программа и её меню. В меню есть 5 действий: создание массива, подсчёт количества и суммы нечётных отрицательных элементов, подсчёт локальных максимумов, вывод матрицы на экран, выход из программы с сопутсвующей очисткой памяти. Доступ к пунктам меню организован с помощью бесконечного цикла «switch». При вводе матрицы запрашивается количество строк и столбцов матрицы, выделяется память с помощью «malloc», и осуществляется непосредственный ввод элементов массива. При поиске кол-ва и суммы нечет. отриц. элементов производится обход массива с помощью двух циклов for и каждый элемент проверяется на четность и отрицательность. При поиске локальных максимумов создаётся дополнительный массив для того чтобы на граничных точках массива программа не выходила за пределы массива, если там присутствует локальный максимум.

4.4.2 Разработана структурная схема алгоритма решения задачи и представлена на рисунках 4.1 и 4.2.

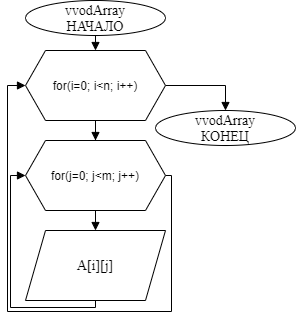
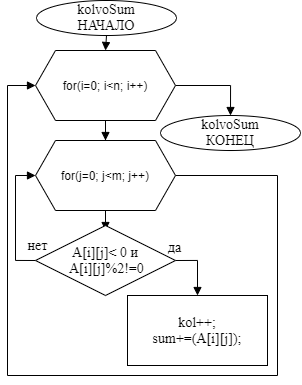
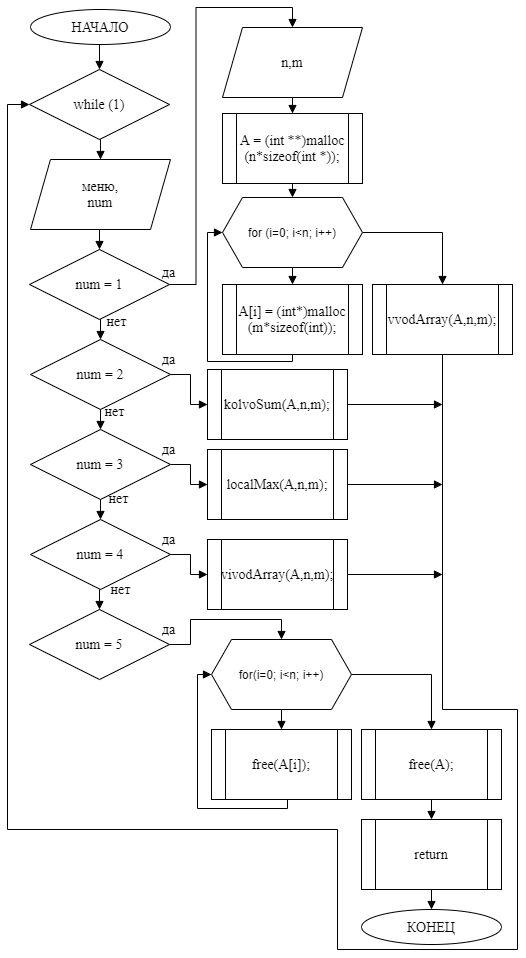
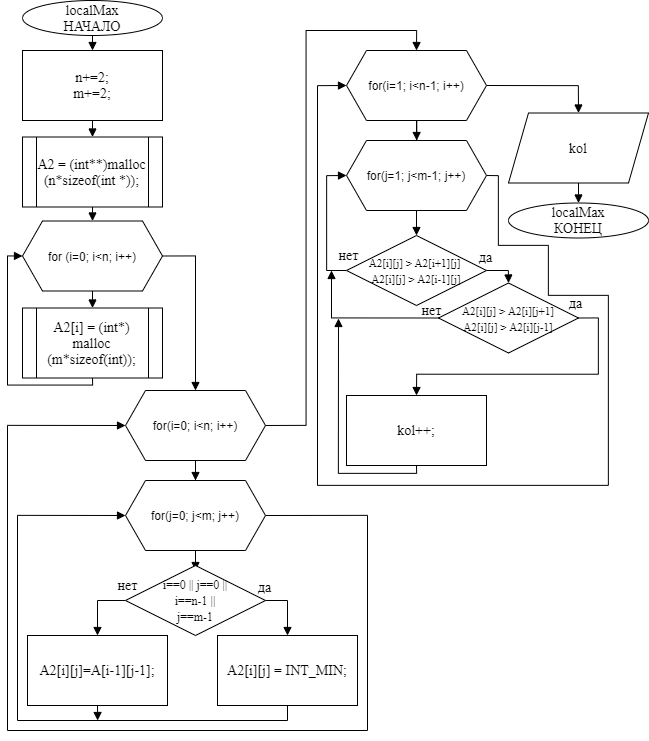
 

Рисунок 4.1 – Структурная схема функции ввода массива и функции подсчёта количества и суммы нечетных отрицательных элементов



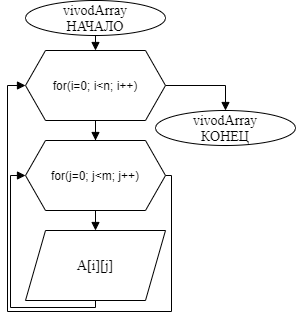


Рисунок 4.2 – Структурная схема функции вывода массива, функции поиска локальных максимумов и основная программа

3.4.4 Написана программы на Си согласно вышеописанного алгоритма.

#include <stdio.h>

#include <locale.h>

//прототипы функций

void vvodArray(int \*\*A, int n, int m);

void vivodArray(int \*\*A, int n, int m);

void kolvoSum(int \*\*A, int n, int m);

void localMax(int \*\*A, int n, int m);

//основная программа

main()

{

setlocale(LC\_ALL, "Rus");

//объявление переменных

int num;

int \*\*A,n,m;

//меню

printf(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \n");

printf("/ \n");

printf(" Программа работает с массивом двумерных чисел \n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \n\n");

while (1)

{

printf("Выберете дальнейшее действие: \n");

printf("1 - создать массив \n");

printf("2 - подсчитать количество и сумму нечёт. отриц. элементов \n");

printf("3 - подсчитать количество локальных максимумов \n");

printf("4 - вывести массив на экран \n");

printf("5 - выход из программы \n");

printf(" -> "); scanf("%d",&num); printf("\n");

switch (num)

{

case 1:

{

printf("Введите количество строк и столбцов матрицы \n");

printf("строки: "); scanf("%d",&n);

printf("столбцы: "); scanf("%d",&m);

//выделение памяти под массив

A = (int \*\*)malloc(n\*sizeof(int \*));

int i;

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

A[i] = (int \*)malloc(m\*sizeof(int));

vvodArray(A,n,m);

break;

}

case 2:

{

kolvoSum(A,n,m);

break;

}

case 3:

{

localMax(A,n,m);

break;

}

case 4:

{

vivodArray(A,n,m);

break;

}

case 5:

{

int i;

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

free(A[i]);

free(A);

return;

}

}

}

}

//------------------------------------------------------------------------------

void vvodArray(int \*\*A, int n, int m)

{

int i,j;

printf("Вводите элементы массива:\n");

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

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

{

printf("a[%d;%d] = ",(i+1),(j+1));

scanf("%d",&A[i][j]);

}

printf("\n");

}

//------------------------------------------------------------------------------

void vivodArray(int \*\*A, int n, int m)

{

int i,j;

printf("Массив: \n");

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

{

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

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

printf("\n");

}

printf("\n");

}

//------------------------------------------------------------------------------

void kolvoSum(int \*\*A, int n, int m)

{

int i,j;

int kol=0,sum=0;

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

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

if (((A[i][j]) < 0) && (((A[i][j]) % 2) != 0))

{

kol++;

sum+=(A[i][j]);

}

printf("Количество нечётных элементов массива = %d\n",kol);

printf("Сума нечётных отрицательных элементов массива = %d\n\n",sum);

}

//------------------------------------------------------------------------------

void localMax(int \*\*A, int n, int m)

{

int i,j, kol=0;

//создаём вспомогательный массив для того чтобы на граничных

//точках массива прога не вылетала, если там локальный максимум

n+=2;

m+=2;

//выделение памяти

int \*\*A2 = (int \*\*)malloc(n\*sizeof(int \*));

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

A2[i] = (int \*)malloc(m\*sizeof(int));

//заполнение вспомогательного массива

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

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

{

//если граничная точка

if ((i==0) || (j==0) || (i==(n-1)) || (j==(m-1)))

A2[i][j] = -9999999999;

else

A2[i][j]=A[i-1][j-1];

}

//поиск локальных максимумов

for(i=1; i<(n-1); i++)

for(j=1; j<(m-1); j++)

{

if ((A2[i][j] > A2[i+1][j]) && (A2[i][j] > A2[i-1][j]) && (A2[i][j] > A2[i][j+1]) && (A2[i][j] > A2[i][j-1]))

{

printf("Элемент %d является локальным максимумом;\n", (A2[i][j]));

kol++;

}

}

printf("В матрице %d локальных максимумов\n\n", kol);

}

2.4.5 Выполнена отладка программы.

Результаты тестирования отображены на рисунках 4.3 и 4.4.

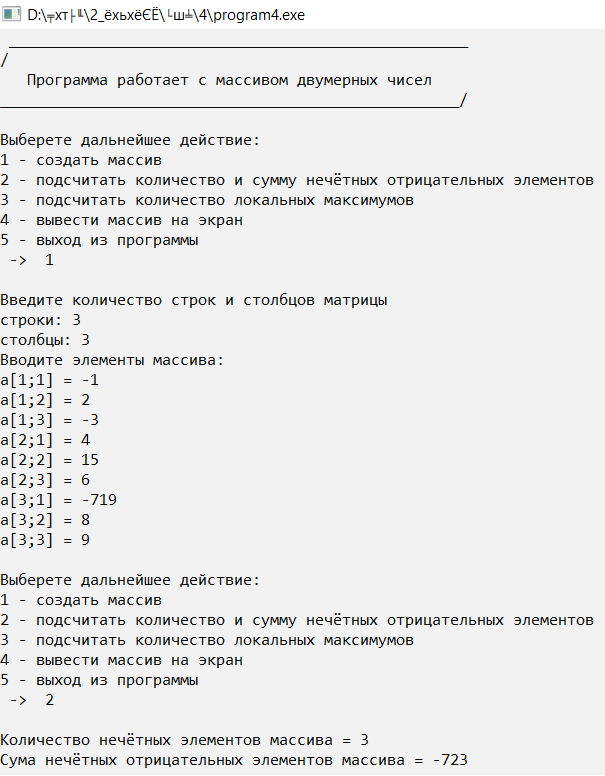


Рисунок 3.2 – Тест 1 часть

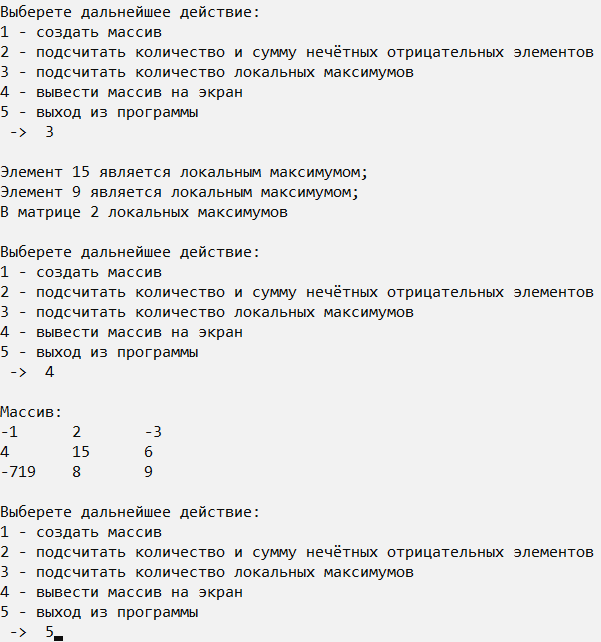


Рисунок 3.3 – Тест 2 часть

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

В ходе выполнения данной лабораторной работы были получены навыки обработки двумерных массивов с помощью функций. Были изучены способы инициализации двумерных массивов статических и динамических, а также способы осуществления доступа к элементам многомерных массивов. Получен навык выделения и очистки памяти для двумерного динамического массива. Были изучены различные алгоритмы при работе с массивами и способы их применения. Полученные во время разработки навыки помогут разрабатывать более сложные программы с массивами, более эффективные по времени выполнения алгоритмы.