**ЛАБОРАТОРНАЯ РАБОТА №6**

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

**Цель работы**: Изучить основные принципы обработки двумерных массивов, исследовать способы передачи параметров в функции.**1. Вариант задания**

Посчитать сумму элементов таких столбцов матрицы, в которых количество четных элементов больше, чем нечетных. Найти позиции всех седловых точек матрицы. Примечание. Матрица А имеет седловую точку аij, если аij является максимальным элементом в i-й строке и минимальным в j-м столбце.

**2. Алгоритм программы**

Программа состоит из нескольких функций которые вызываются в основной функции main. Для реализации данной программы были сделаны несколько отдельных функций. Схемы всех функций будут приведены ниже.



Рисунок 1 – Основное тело программы



Рисунок 2 – Ввод матрицы с клавиатуры(слева), ввод матрицы рандомом ( по центру) и вывод матрицы на экран (справа)



Рисунок 3 – Алгоритм подсчета суммы в тех стобцах, где нечетных больше четных



Рисунок 4 – Алгоритм работы по поиску седловых точек и их выводу



Рисунок 5 – Подсчет количество столбцов, в которых нет нулевых элементов

**3. Код программы**

// LR6.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#pragma warning(disable : 4996)

#include <Windows.h>

#include<stdio.h>

#include <stdlib.h>

#include <string.h>

#include<math.h>

int\*\* in\_matrix(int\*\* matrix, int a,int b);

int\*\* in\_matrix\_random(int\*\* matrix, int a, int b);

void sum\_stolbs(int\*\* matrix, int a, int b);

void sedlovie(int\*\* matrix, int a, int b);

void vivod\_matrix(int\*\* matrix, int a, int b);

int kol\_noli(int\*\* matrix, int a, int b);

main()

{

int\*\* matrix = NULL;

int a, b; int result;

SetConsoleCP(1251); // Задаем таблицу символов для консоли.

SetConsoleOutputCP(1251);

while (1) {

printf("Выберить способ ввода матрицы, 1 - С клавиатуры, 2 - Рандом ---> ");

scanf("%d", &result);

if ((result == 1) || (result == 2)) break;

}

printf("\nВведите количество строк --> "); scanf("%d", &a);

printf("\nВведите количество столбцов --> "); scanf("%d", &b);

printf("\nЗадается матрица размером %d x %d", a, b);

if (result == 1)

{

matrix = in\_matrix(matrix, a, b);

}

else

{

matrix = in\_matrix\_random(matrix, a, b);

}

vivod\_matrix(matrix, a, b);

sum\_stolbs(matrix, a, b);

sedlovie(matrix, a, b);

printf("\n\nКооличесво столбцов не имеющих 0-вого элемента --->%d", kol\_noli(matrix, a, b));

// for (int i = 0; i < a; i++) // цикл по строкам

// free(matrix[i]); // освобождение памяти под строку

//free(matrix);

}

int\*\* in\_matrix(int\*\* matrix, int a, int b)

{

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

for (int i = 0; i < a; i++)

{

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

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

{

printf("\nВведите элемент matrix[%d,%d] -->", i, j); scanf("%d", &matrix[i][j]);

//printf("\nВведите элемент matrix[%d,%d] -->%d", i, j, matrix[i, j]);

}

}

return matrix;

}

int\*\* in\_matrix\_random(int\*\* matrix, int a, int b)

{

srand(time(0)); //задает постоянное рандомное значение

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

for (int i = 0; i < a; i++)

{

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

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

{

matrix[i][j] = rand() % 200 - 100;

}

}

return matrix;

}

void vivod\_matrix(int\*\* matrix, int a, int b)

{

printf("\n\n\n"); printf("\n Полученная матрица \n");

for (int i = 0; i < a; i++)

{

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

{

printf("%6d\t", matrix[i][j]);

}

printf("\n");

}

}

void sum\_stolbs(int\*\* matrix, int a, int b)

{

int chet, nechet;

for (int i = 0; i < b; i++)

{

chet = 0; nechet = 0;

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

{

int temp = matrix[j][i];

//if ((matrix[j,i]\*(int\*)) % 2) nechet++; else chet++;

if (temp % 2) nechet++; else chet++;

}

if (chet > nechet)

{

int sum = 0;

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

{

int temp = matrix[j][i];

sum += temp;

}

printf("\n Строка %3d, колличество четных %3d , колличество нечетных %3d , сумма = %7d", i,chet,nechet, sum);

}

}

}

void sedlovie(int\*\* matrix, int a, int b)

{

int sedl\_i, sedl\_j,tempj=0;

for (int i = 0; i < a; i++)

{

int max\_i = 0, min\_j = 0;

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

{

if (max\_i < matrix[i][j]) { max\_i = matrix[i][j]; tempj = j; }

}

min\_j = max\_i;

for (int L = 0; L < a; L++)

{

if (min\_j > matrix[L][tempj])

{

min\_j = matrix[L][tempj];

break;

}

}

if (min\_j == max\_i)

{

printf("\nСедловая точка sedl[%d,%d] = %d", i, tempj, matrix[i][tempj]);

}

}

}

int kol\_noli(int\*\* matrix, int a, int b)

{

int col = 0;

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

{

int schet = 0;

for (int i = 0; i < a; i++)

{

if (matrix[i][j] == 0) schet++;

}

if (schet == 0) col++;

}

return col;

}

**4.Результаты тестирования и отладки программы**

Для тестирования программы были разработаны несколько тестовых вариантов.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 9 | 20 | -9 | 0 |
| -9 | -9 | 40 | 3 | 4 |
| 0 | 2 | 30 | 4 | 6 |
| 23 | 3 | 45 | 3 | 5 |
| 2 | 1 | 20 | 1 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -9 | 10 | 45 | 22 | -5 |
| 7 | 23 | 45 | 12 | 0 |
| 67 | 0 | 76 | 32 | 24 |
| 45 | 6 | 55 | 0 | -7 |
| 98 | 76 | 67 | 54 | 3 |

Таблица 1 – Тестовые примеры для выполнения проверки работоспособности программы

Из данных тестовых примеров следует, что, в первой табличке 2 седловые точки , это число 20. Так же колличество столбцов не имеющих 0 – 3. Так же присутствуют две строки в которых колличество четных элементов больше нечетных , это строки 3 и 5( в программе отсчет не от 1 а от 0, соответственно это столбцы 2 и 4).

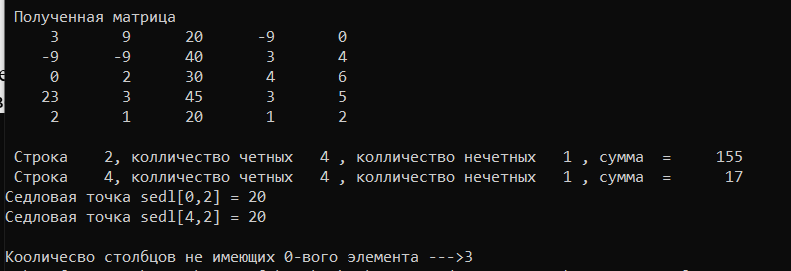


Рисунок 6 – Результат выполнения программы с первым тестовым примером

Из второго тестового примера следует, что , седловых точек в нем две , это число 45. Количество строк где четных больше чем нечетных – 2-я строка и 4-я строка (в программе это 1-я и 3-я). Количество столбцов не имеющих нулеевые элементы – 2.

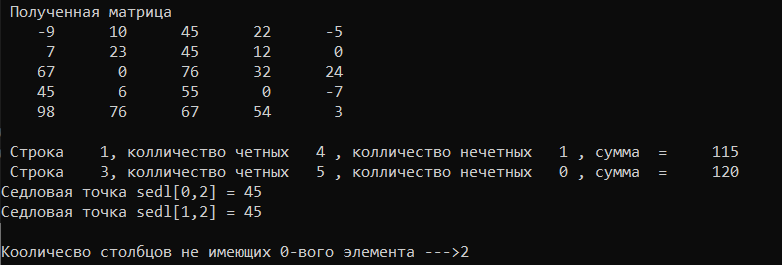


Рисунок 7 – Результат выполнения программы с вторым тестовым примером

Как видим, программа работает корректно.

**Вывод**

В ходе выполнения данной лабораторной работы были изучены особенности представления и обработки двумерных массивов в языках С/С++ с учетом связи указателей и массивов указателей. Получены практические навыки реализации алгоритмов обработки двумерных динамических массивов средствами языков С/С++. Были про исследованы особенности обработки двумерных динамических массивов.ё