МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

|  |
| --- |
|  |

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра информационной безопасности**

**Интеллектуальные компьютерные системы**

**ОТЧЕТ**по лабораторной работе №3

на тему «**Кластеризация с неизвестным L**»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Выполнил студент гр. Б8116-09.03.01 | | |  | Гусев М.Д. | |  | | | Проверил | | |  | Ю.С. Москаленко | | дата | | |  | | | зачтено/не зачтено | | |

г. Владивосток

2019

**Постановка задачи и входные данные**

В качестве входных данных выступает матрица заданной размерности, заполненная единицами и нулями.

Задача состоит в том, чтобы в заданной матрице найти строки, которые будут являться масками, определить кодировки всех строк, на основе кодировок составить кластеры.

**Реализация**

Изначально пользователю предоставляются поля, в которых можно задать количество строк и столбцов матрицы. Это было реализовано с помощью метода parseInt(promt(,)), который выводит пользователю диалоговое окно с заданным текстом, а также поле для ввода данных типа integer.

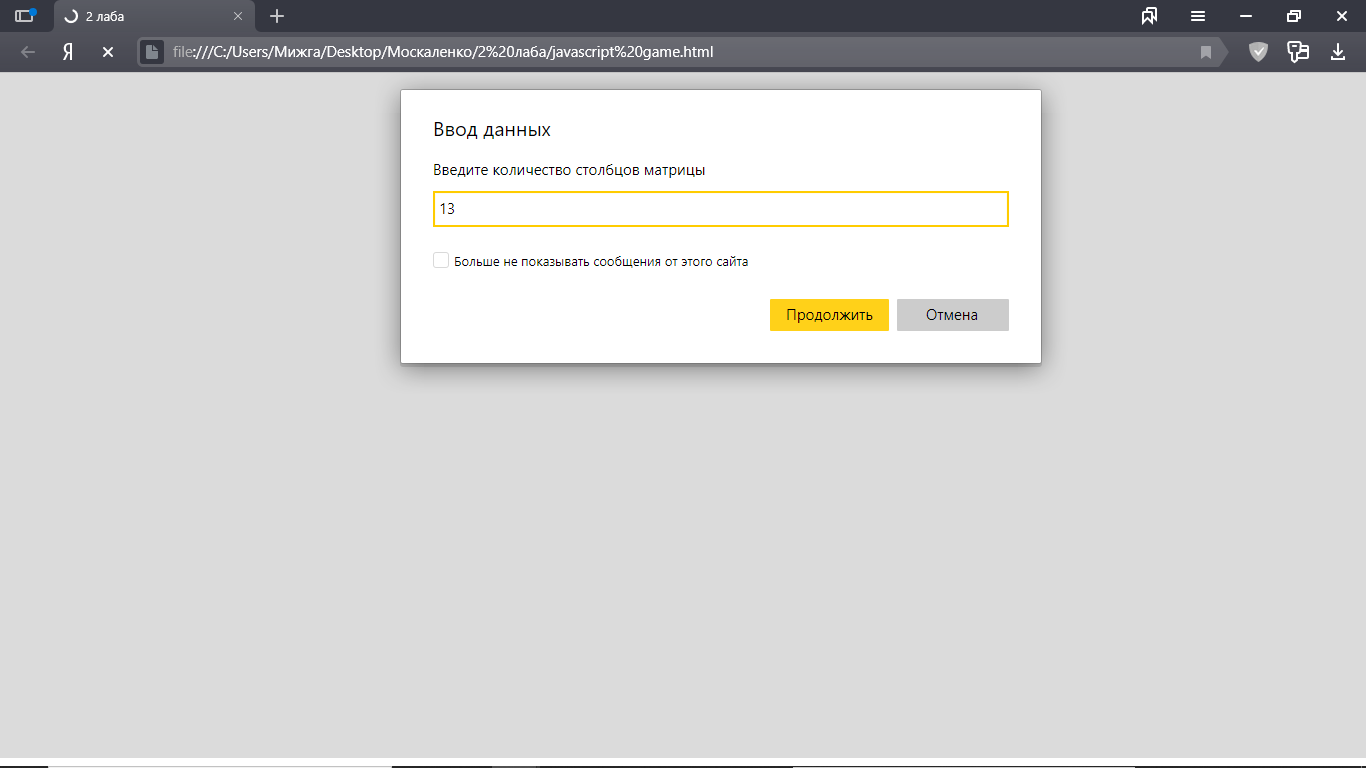


рис. 1 – диалоговое окно для ввода данных.

В сравнении с прошлой лабораторной работой были доработаны поля для ввода данных. Теперь они представляют собой строки, которые можно полностью заполнять аналогичными строками, которые будут в матрице. Такой способ более быстрый и удобный, ведь если допущена ошибка, то ее можно тут же исправить.

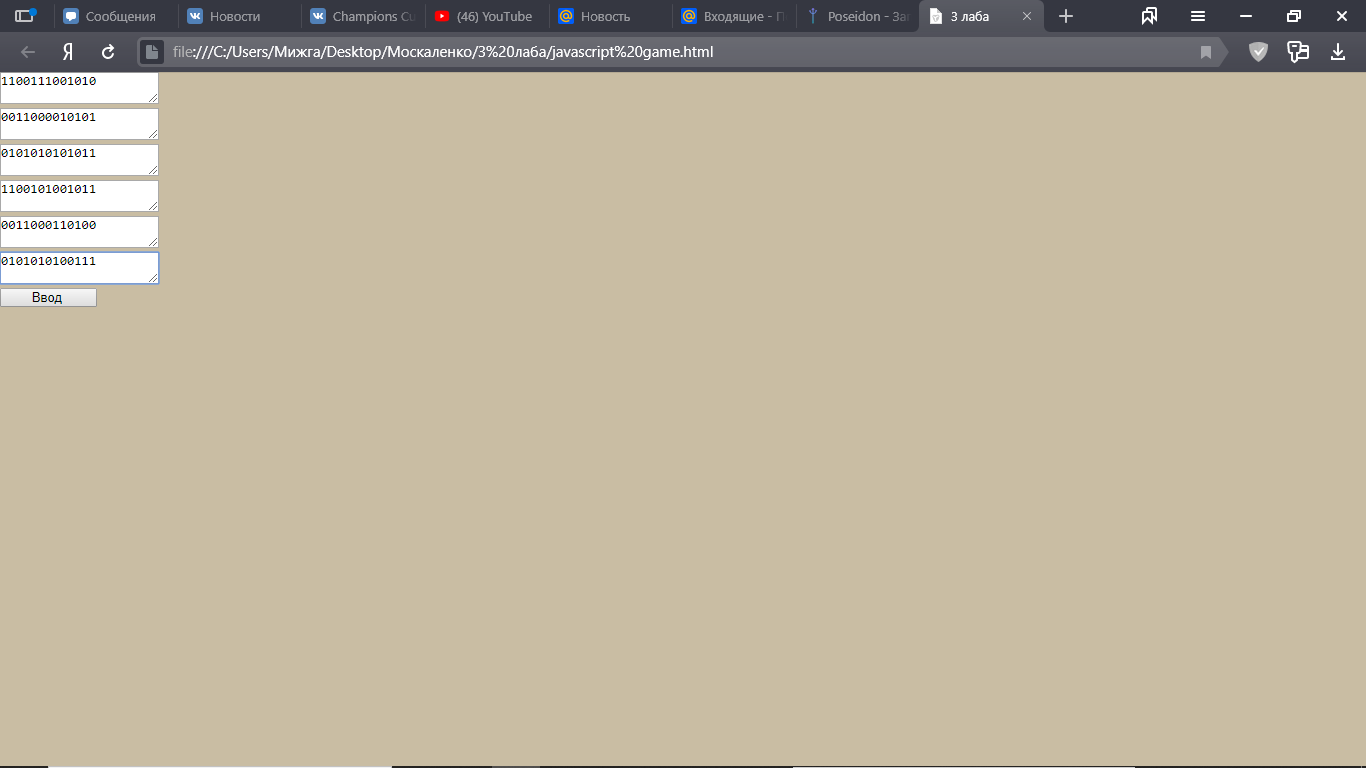


Рис.2 – построчный ввод данных матрицы

После ввода данных пользователю отображается заданная матрица. Для этого введенные данные были сохранены в двумерном массиве.   
Также создается еще один двумерный массив, каждый элемент которого будет являться блоком div, которому задается нужный визуальный стиль. Именно внутрь блоков div помещены данные из первого массива.  
Если кратко, в первом массиве хранятся введенные цифры, во втором – блоки div, которые являются визуальной оболочкой для цифр.

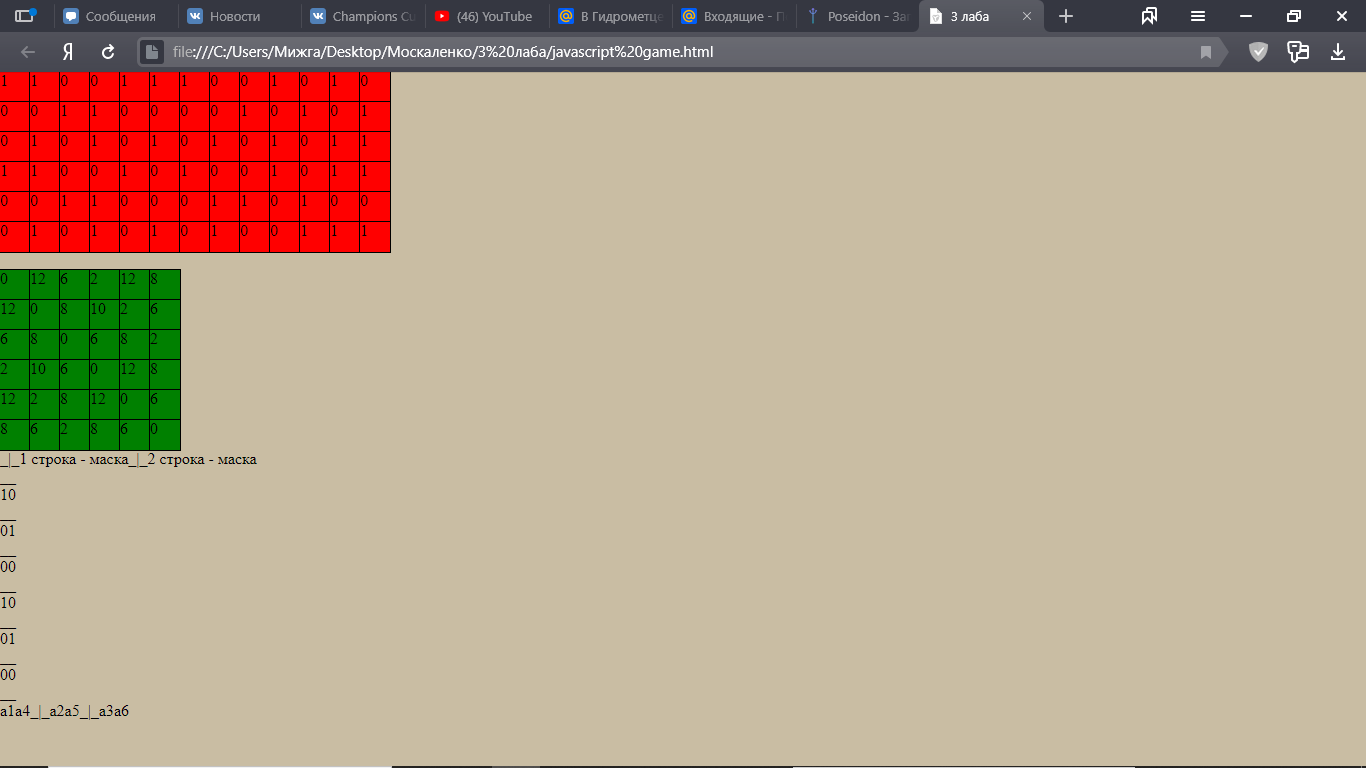


рис. 3 – матрица введенных элементов

Также из прошлой лабораторной работы «перекочевала» матрица несоответствий. Здесь она необязательна, но такая матрица позволяет ускорить вычисление количества несоответствий, а также делает результат этих вычислений более удобным и наглядным.  
 Для этого используется тройной цикл, внутри которого стоит условие, проверяющее совпадает ли элемент с аналогичным элементом в другой строке. Также вместе с условием запущен счетчик, считающий количество несовпадений. Результат записывается в очередной двумерный массив.  
Для визуального отображения создается еще один двумерный массив, который будет хранить блоки div, с заданным визуальным стилем, как уже это было ранее.

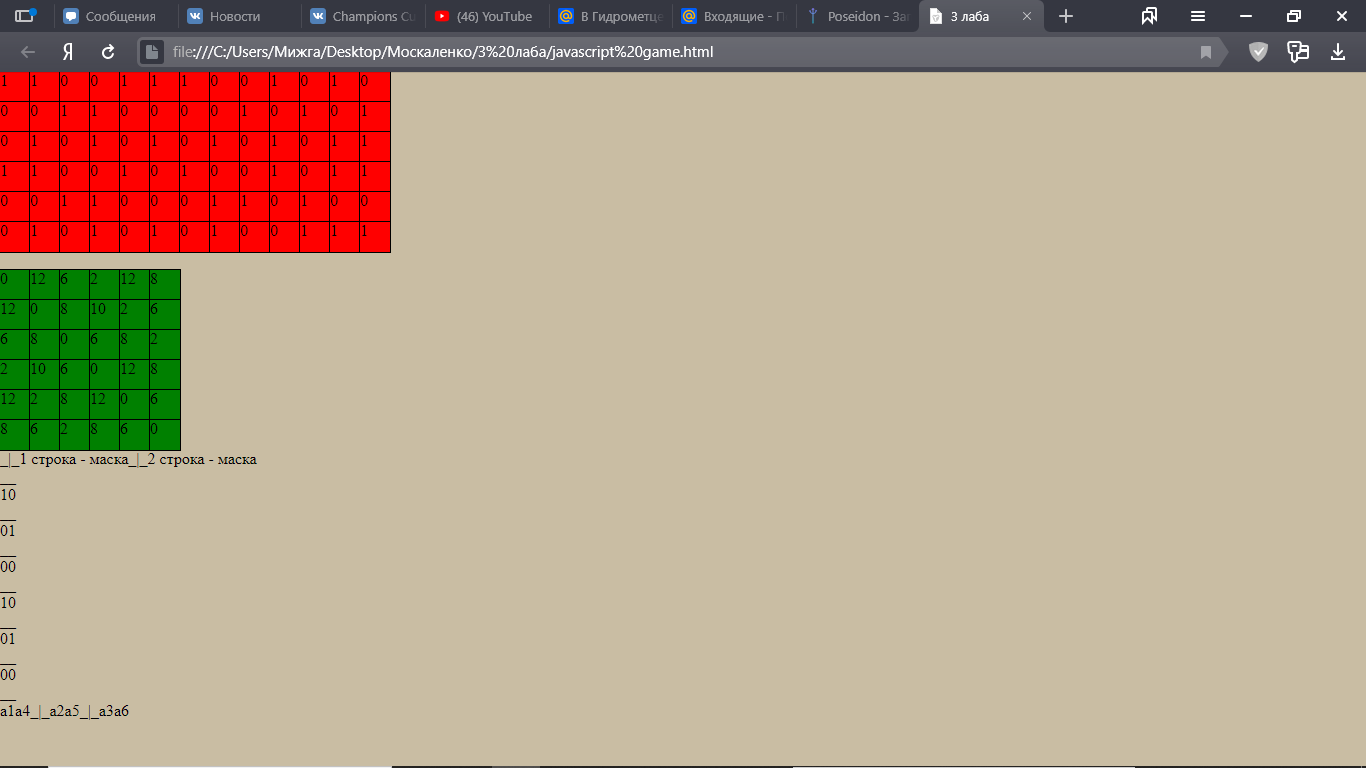


рис.4 – матрица введенных элементов и матрица несоответствий

Поиск масок начинается с первой строки матрицы несоответствий. Она всегда является первой маской. После чего в этой строке ищется элемент, больший чем R1=n/2+n/4.  
При нахождении такого элемента программа запоминает его индекс и переходит на строку, которую этот элемент связывает с прошлой строкой. Уже на этой строке идет проверка, является ли прошлая строка маской. После проверки снова ищется элемент, больший чем R1 и несоответствующий элементам, которые связывают данную маску с предыдущими. Все это продолжается до тех пор, пока в очередной строке не будет найдено ни одного элемента большего, чем R1.

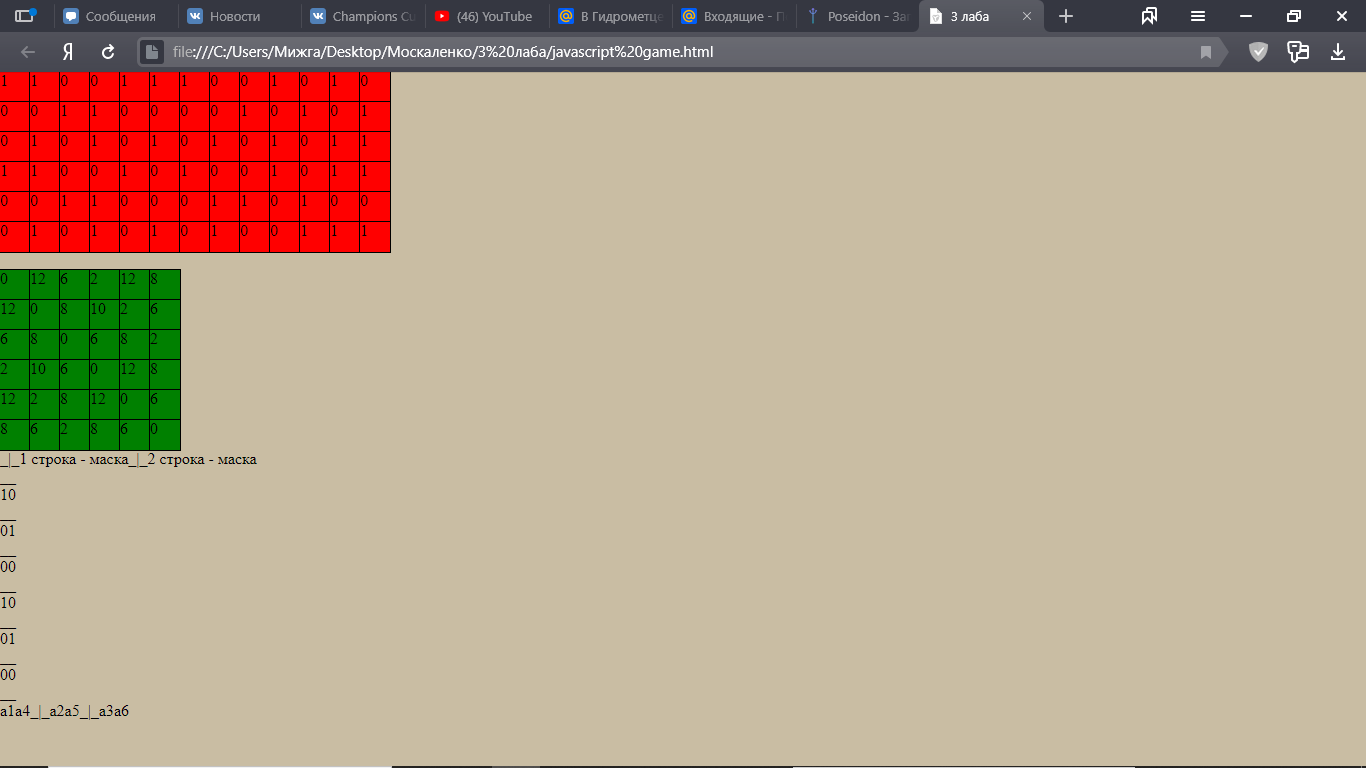


рис. 5 - матрица введенных элементов и матрица несоответствий, маски

Для определения кодировок программа «пробегает» в таблице несоответствий по строкам, и ищет элементы, меньшие чем R2=n-R1, у таких элементов будет код 1 относительно одной из масок, в противном случае код будет 0. Таким образом, например если маски будет две, то код строки будет состоять из 2 цифр. Коды сначала последовательно записываются в двумерный массив, далее он объединяется в новом одномерном массиве и фильтруется от пустых элементов (которые могут возникать при объединении). Строки с одинаковой кодировкой объединяются в кластеры и далее программа последовательно выводит кодировки строк, а также сами кластеры.

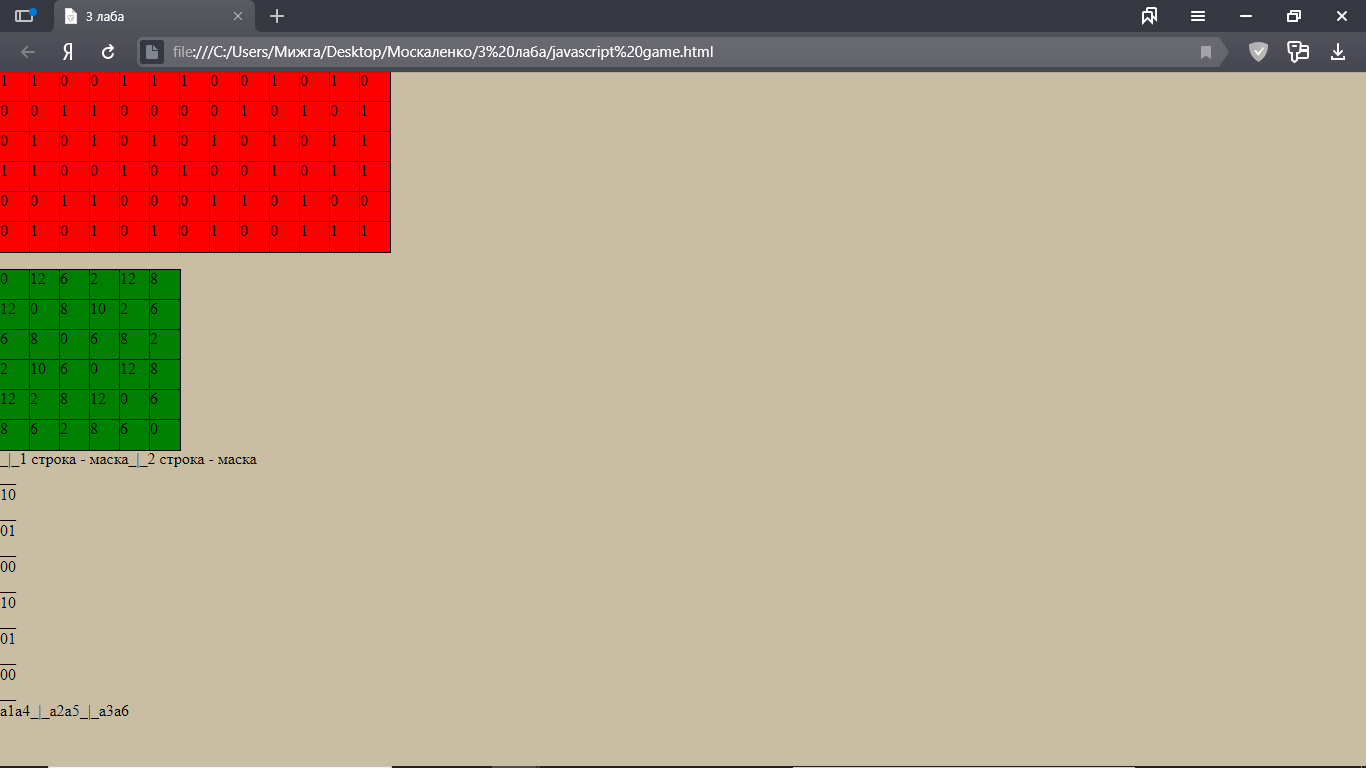


Рис.5 – матрица введенных данных, матрица несоответствий,маски, кодировки строк, кластеры.

**Код программы.**

var count=0; count1 = 0;

var command,command1;

var mas = [];var mas1 = []; mas2 = []; mas3 = []; mask = [];cod = [];cod1 = [];cod2 = [];cod3 = [];

var pmask;

var block,val1,val2,R1,R2,count3;

mask[0] = 0;

window.onload = function()

{

block = document.createElement('div');

document.body.appendChild(block);

val1 = parseInt(prompt("Введите количество строк матрицы", 2));

val2 = parseInt(prompt("Введите количество столбцов матрицы", 2));

R1 = (val2/2) + (val2/4);

R2 = val2 - R1;

for(var i=0;i<val1;i++){

text1=document.createElement('textarea');

document.body.appendChild(text1);

br=document.createElement('br');

br.id = 'i'+i;

document.body.appendChild(br);

text1.id=i;

}

for(var i=0;i<1;i++){

input = document.createElement('input');

document.body.appendChild(input);

input.type = 'button';

input.value=" Ввод ";

}

br1 = document.createElement('br');

document.body.appendChild(br1);

input.onclick=function(){

for(var i=0;i<val1;i++){

mas1[i] = [];

count3=0;

t=document.getElementById(i).value;

for(let int of t){

//alert(int);

mas1[i][count3]=int;

count3++;

}

document.getElementById(i).style.display = 'none';

input.style.display = 'none';

br1.style.display = 'none';

document.getElementById('i'+i).style.display = 'none';

}

matrix();

}

}

function matrix()

{

for(var i = 0;i<val1;i++)

{

mas[i] = [];

for (var j = 0;j<val2;j++)

{

div = document.createElement('div');

document.body.appendChild(div);

mas[i][j] = div;

mas[i][j].style.width='30px';

mas[i][j].style.height='30px';

mas[i][j].style.background='red';

mas[i][j].style.outline = '1px solid #000';

mas[i][j].style.display = 'inline-block';

mas[i][j].innerHTML = mas1[i][j];

}

br1 = document.createElement('br');

document.body.appendChild(br1);

}

count = 0;

br1 = document.createElement('br');

document.body.appendChild(br1);

for(var i=0; i<val1;i++)

{

mas2[i] = [];

for(var j=0;j<val1;j++)

{

div = document.createElement('div');

document.body.appendChild(div);

mas2[i][j] = div;

mas2[i][j].style.width='30px';

mas2[i][j].style.height='30px';

mas2[i][j].style.background='green';

mas2[i][j].style.outline = '1px solid #000';

mas2[i][j].style.display = 'inline-block';

}

br1 = document.createElement('br');

document.body.appendChild(br1);

}

count = 0;

for(var i = 0;i<val1;i++)

{ mas3[i] = [];

for(var j=0;j<val1;j++)

{

for(var k=0;k<val2;k++)

{

if(mas1[i][k] != mas1[j][k]) count=count+1;

}

mas3[i][j] = count;

mas2[i][j].innerHTML = mas3[i][j];

count=0;

}

}

count = 0;

for(var i=0;i<1;i++){

for(var j=0;j<val1;j++){

if(mas3[i][j] > R1){

pmask = j; break;

}

}

}

var count1 = 0;

for(var i=0;i<val1;i++){

if(mas3[i][pmask] > R1){

for(var j=0;j<mask.length;j++){

if(mask[j]== i){

count++;

}

if(count == mask.length){

count1++;

mask[count1]= pmask;

}

}

if(count != mask.length-1){

pmask = i;

}

}

}

for(var i=0;i<mask.length;i++){

p = document.createElement('p');

document.body.appendChild(p);

p.style.display = 'inline-block';

p.innerHTML ='\_|\_'+ (mask[i]+1)+ ' строка - маска';

}

for(var i=0;i<mask.length;i++){

cod[i] = [];

var n = mask[i];

for(var j=0;j<val1\*1;j++){

var p=0\*1;

if(mas3[j][n] < R2){

p=1;

}

cod[i][j] = p;

}

}

p=0;

count = 0;

for(var j=0;j<val1;j++){

for(var i =0;i<mask.length;i++){

if(j!=count) p+=mask.length;

cod1[i+p] = cod[i][j];

}

count=j;

}

let cod2 = cod1.filter(element => element !=='');

for(var i=0;i<cod2.length;i++){

if((i)%mask.length == 0) {

p = document.createElement('p');

document.body.appendChild(p);

//p.style.display = 'inline-block';

p.innerHTML = "\_\_" ;

}

p = document.createElement('p');

document.body.appendChild(p);

p.style.display = 'inline-block';

p.innerHTML = cod2[i] ;

}

for(var i=0;i<1;i++){

p = document.createElement('p');

document.body.appendChild(p);

//p.style.display = 'inline-block';

p.innerHTML = "\_\_" ;

}

for (var i = 0; i < val1; i++)

{

var m = '';

for (var j = 0; j < mask.length; j++)

{

var k = i \* mask.length + j;

m += cod2[k].toString();

cod3[i] = m;

}

}

count=cod3.length;

for(var j=0;j<cod3.length;j++){

p = document.createElement('p');

document.body.appendChild(p);

p.style.display = 'inline-block';

p.innerHTML = 'a'+ (j\*1+1);

for(var i=j+1;i<cod3.length;i++){

if(cod3[j] == cod3[i]){

p = document.createElement('p');

document.body.appendChild(p);

p.style.display = 'inline-block';

p.innerHTML =' '+'a'+ (i\*1+1);

count--;

}

}

count--;

if(count==0)break;

p = document.createElement('p');

document.body.appendChild(p);

p.style.display = 'inline-block';

p.innerHTML = "\_|\_";

}

}