МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Казанский национальный исследовательский технический университет

им. А.Н. Туполева – КАИ»

Институт компьютерных технологий и защиты информации

Отделение СПО в ИКТЗИ (Колледж информационных технологий)

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине

Прикладное программирование

# Тема: «JS + html + CSS»

Работу выполнил

Студент гр.4238

Бусов В.Р.

Принял преподаватель

Калинина А.В.

Казань 2024

**Цели**

1. Изучить JavaScript
2. Научиться применять JS.
3. Создать сайты в соответствии с вариантом.

**Задание на лабораторную работу**

**Задание 1**: вычислить квадратный корень числа x с точностью до 6 знака после запятой. Не использовать Math.

**Задание 2**: создать класс Matrix2x2 - двумерная матрица из вещественных чисел. Аргументы - содержимое матрицы (лучше, разумеется, хранить двумерным массивом, а то замучаетесь). Методы:  
• Matrix2x2() - конструктор для нулевой матрицы;  
• Matrix2x2(double) - конструктор для матрицы, у которой каждый элемент равен поданному числу;  
• Matrix2x2(double [][]) - конструктор для матрицы, содержимое подается на вход в виде массива;  
• Matrix2x2 add(Matrix2x2) - сложение матрицы с другой;  
• Matrix2x2 sub(Matrix2x2) - вычитание из матрицы другой матрицы;  
• Matrix2x2 multNumber(double) - умножение матрицы на вещественное число;  
• Matrix2x2 mult(Matrix2x2) - умножение матрицы на другую матрицу;  
• double det() - определитель матрицы;  
• void transpon() - транспонировать матрицу;  
• Matrix2x2 inverseMatrix() - возвратить обратную матрицу для заданной. Если это невозможно, вывести сообщение об ошибке и вернуть нулевую матрицу (кто вдруг знает исключения, может их использовать).  
• Vector2D multVector(Vector2D) - умножить матрицу на двумерный вектор  (считая его столбцом) и возвратить получившийся столбец в виде вектора.

**Результат**

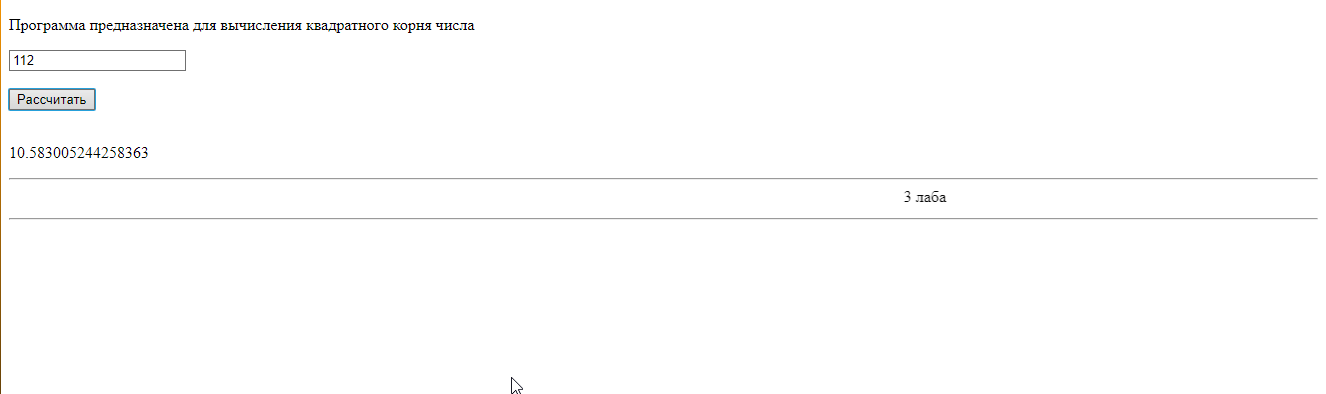


Рисунок 1 – результат выполнения задания №1

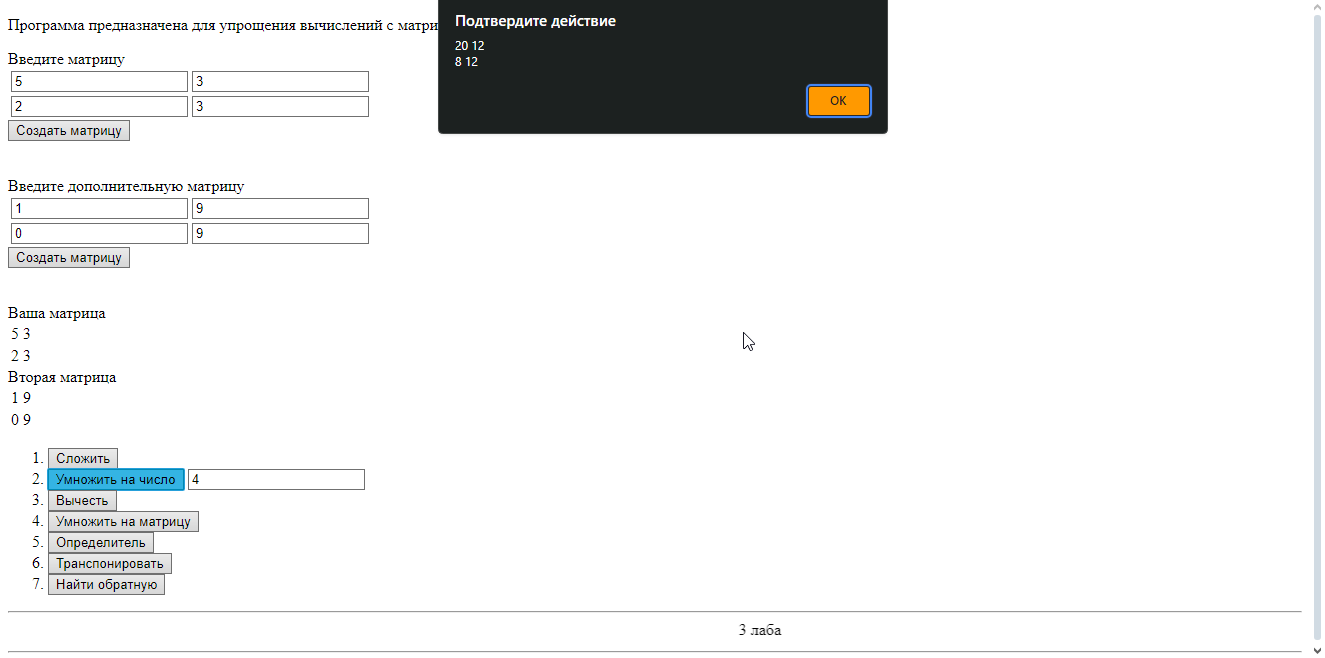


Рисунок 2 – результат выполнения задания №2

**Листинг**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<p>Программа предназначена для вычисления квадратного корня числа</p>

<input type="number" id="request-input" placeholder="Введите число">

<br><br>

<button onclick="sqrt();" id="submit-btn">Рассчитать</button>

<br><br>

<p id="result">Здесь будет показан результат....</p>

<hr>

<marquee>3 лаба</marquee>

<hr>

</body>

</html>

function sqrt() {

var n = parseFloat(document.querySelector("#request-input").value);

if (n >= 0) {

var x = n;

var y = (x + 1) / 2;

console.log(x, y);

while (y < x) {

x = y;

y = (x + n / x) / 2;

}

document.querySelector("#result").innerHTML = x;

} else if (n < 0) {

alert("Число не может быть отрицательным!");

}

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="script2.js"></script>

<title>Document</title>

</head>

<body>

<p>Программа предназначена для упрощения вычислений с матрицами</p>

<div class="create-first-matrix">

<table>

<thead>Введите матрицу</thead>

<tbody>

<tr>

<td><input class="el-input" type="number"></td>

<td><input class="el-input" type="number"></td>

</tr>

<tr>

<td><input class="el-input" type="number"></td>

<td><input class="el-input" type="number"></td>

</tr>

</tbody>

</table>

<button id="create-matrix" onclick="create\_matrix(document.querySelector('#create-matrix'));">Создать матрицу</button>

</div>

<br><br>

<div class="create-second-matrix">

<table>

<thead>Введите дополнительную матрицу</thead>

<tbody>

<tr>

<td><input class="second-el-input" type="number"></td>

<td><input class="second-el-input" type="number"></td>

</tr>

<tr>

<td><input class="second-el-input" type="number"></td>

<td><input class="second-el-input" type="number"></td>

</tr>

</tbody>

</table>

<button id="create-second-matrix" onclick="create\_matrix(document.querySelector('#create-second-matrix'));">Создать матрицу</button>

</div>

<br><br>

<div class="user-first-matrix">

<table>

<thead>Ваша матрица</thead>

<tbody>

<tr>

<td class="user-first-matrix-el">-</td>

<td class="user-first-matrix-el">-</td>

</tr>

<tr>

<td class="user-first-matrix-el">-</td>

<td class="user-first-matrix-el">-</td>

</tr>

</tbody>

</table>

</div>

<div class="user-second-matrix">

<table>

<thead>Вторая матрица</thead>

<tbody>

<tr>

<td class="user-second-matrix-el">-</td>

<td class="user-second-matrix-el">-</td>

</tr>

<tr>

<td class="user-second-matrix-el">-</td>

<td class="user-second-matrix-el">-</td>

</tr>

</tbody>

</table>

</div>

<div class="interface">

<ol>

<li>

<button onclick="add\_matrix();">Сложить</button>

</li>

<li>

<button onclick="multiNumber\_matrix();">Умножить на число</button>

<input type="number" id="multiNumber">

</li>

<li>

<button onclick="sub\_matrix();">Вычесть</button>

</li>

<li>

<button onclick="multi\_matrix();">Умножить на матрицу</button>

</li>

<li>

<button onclick="det\_matrix();">Определитель</button>

</li>

<li>

<button onclick="transpon\_matrix();">Транспонировать</button>

</li>

<li>

<button onclick="inverse\_matrix();">Найти обратную</button>

</li>

</ol>

</div>

<hr>

<marquee>3 лаба</marquee>

<hr>

</body>

</html>

class Matrix2x2 {

constructor(n) {

this.matrix = [[], []];

if (typeof(n) == "number") {

for (var i = 0; i < 2; i++) {

for (var j = 0; j < 2; j++) {

this.matrix[i][j] = n;

}

}

} else if (typeof(n) == "object") {

for (var i = 0; i < 2; i++) {

for (var j = 0; j < 2; j++) {

this.matrix[i][j] = n[i][j];

}

}

}

}

add(mat) {

var temp = [[], []]

for (var i = 0; i < 2; i++) {

for (var j = 0; j < 2; j++) {

temp[i][j] = this.matrix[i][j] + mat.matrix[i][j];

}

}

return temp;

}

sub(mat) {

var temp = [[], []]

for (var i = 0; i < 2; i++) {

for (var j = 0; j < 2; j++) {

temp[i][j] = this.matrix[i][j] - mat.matrix[i][j];

}

}

return temp;

}

multiNumber(n) {

var temp = [[], []]

for (var i = 0; i < 2; i++) {

for (var j = 0; j < 2; j++) {

temp[i][j] = n \* this.matrix[i][j];

}

}

return temp;

}

multi(mat, a, b) {

if (mat) var matrix = mat.matrix;

else var matrix = [[a], [b]]

var result = [[], []];

for (var i = 0; i < 2; i++) {

for (var j = 0; j < matrix[0].length; j++) {

var sum = 0;

for (var k = 0; k < 2; k++) {

sum += this.matrix[i][k] \* matrix[k][j];

}

result[i][j] = sum;

}

}

return result;

}

det() {

var det = this.matrix[0][0] \* this.matrix[1][1] - this.matrix[1][0] \* this.matrix[0][1];

return det;

}

transpon() {

var temp = [[], []];

for (var i = 0; i < 2; i++) {

for (var j = 0; j < 2; j++) {

temp[j][Math.abs(1 - i)] = this.matrix[i][j];

}

}

return temp;

}

inverseMatrix() {

var determinant = this.det();

if (determinant == 0) {

// обратной матрицы не существует

}

var inverseMatrix = [

[this.matrix[1][1] / determinant, -this.matrix[0][1] / determinant],

[-this.matrix[1][0] / determinant, this.matrix[0][0] / determinant]];

return inverseMatrix;

}

multiVector(a, b) {

return this.multi(null, a, b)

}

}

function create\_matrix(btn) {

if (btn.id == "create-matrix") {

var inputs = document.querySelector(".create-first-matrix").querySelectorAll(".el-input");

} else {

var inputs = document.querySelectorAll(".second-el-input");

}

var correct = true;

inputs.forEach((e) => {

if (e.value == "") {

if (correct == true) {

alert("Необходимо заполнить все столбцы");

correct = false;

}

}

});

if (correct) {

var matrix = [

[Number(inputs[0].value), Number(inputs[1].value)],

[Number(inputs[2].value), Number(inputs[3].value)]

];

if (btn.id == "create-matrix") {

var user\_matrix\_elements = document.querySelectorAll(".user-first-matrix-el");

} else {

var user\_matrix\_elements = document.querySelectorAll(".user-second-matrix-el");

}

for (var i = 0; i < 4; i++) {

user\_matrix\_elements[i].innerHTML = inputs[i].value;

inputs[i].value = "";

}

if (btn.id == "create-matrix") {

m1 = new Matrix2x2(matrix);

alert("Матрица успешно создана!");

console.log(m1.matrix);

} else {

m2 = new Matrix2x2(matrix);

alert("Дополнительная матрица успешно создана!");

console.log(m2.matrix);

}

}

}

window.onload = (event) => {

var m1 = null;

var m2 = null;

}

function print\_matrix(matrix) {

alert(matrix[0][0] + " " + matrix[0][1] + "\n" + matrix[1][0] + " " + matrix[1][1])

}

function add\_matrix() {

print\_matrix(m1.add(m2));

console.log(m1.add(m2));

}

function multiNumber\_matrix() {

print\_matrix(m1.multiNumber(

document.querySelector(

"#multiNumber"

).value

));

}

function sub\_matrix() {

print\_matrix(m1.sub(m2));

}

function multi\_matrix() {

print\_matrix(m1.multi(m2));

}

function det\_matrix() {

alert(m1.det());

}

function transpon\_matrix() {

print\_matrix(m1.transpon());

}

function inverse\_matrix() {

print\_matrix(m1.inverseMatrix());

}

// var m1 = new Matrix2x2([[1, 2], [3, 4]]);

// var m2 = new Matrix2x2([[8, 2], [1, 4]]);

// var m = m1.add(m2);

// print\_matrix(m);

// var m = m1.sub(m2);

// console.log(m);

// var m = m1.multiNumber(10);

// console.log(m);

// var m = m1.multi(m2);

// console.log(m);

// var det = m1.det();

// console.log(det);

// var m = m1.inverseMatrix();

// console.log(m);

// var tm = m1.multiVector(2, 2);

// console.log(tm);