

Содержание

1	Текст задания.....	3
2	Исходные тексты программы.....	4
3	Скриншоты выполнения программы	25
4	Заключение	26
5	Используемая литература.....	27

1 Текст задания

Задача 10

Дана разреженная структурно симметричная матрица. Найти её определитель.

Ввод входных параметров осуществляется с помощью клавиатуры и мыши.

2 Исходные тексты программы

index.html

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <title>Лабораторная работа №2 - Определитель для разреженной
структурно-симметричной матрицы</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" type="text/css" href="css/style.css">
    <script type="text/javascript" src="lib/jquery-
2.2.4.min.js"></script>
    <script type="text/javascript" src="js/det.js"></script>
  </head>
  <body>
    <div class="wrapper">
      <div id="condition">
        <h2>Условие задачи</h2>
        <p>Дана разреженная структурно симметричная матрица.
Найти её определитель.</p>
        <p>Разреженная матрица это матрица, у которой большая
часть элементов равна нулю.</p>
        <p>Структурно-симметричная матрица это матрица в которой
если элементы  $A[i, j] \neq 0$ , то и  $A[j, i] \neq 0$  и  $A[i, j] = 0$ , то и  $A[j, i] = 0$ ,
т.е. элементы матрицы симметричны относительно главной диагонали ( $A[i, j] = A[j, i]$ )</p>
      </div>
      <div id="input">
        <h2>Входные параметры</h2>
        <p><label>Число          строк:          <input          id="row"
type="number"></label></p>
        <p><label>Число          столбцов:        <input          id="col"
type="number"></label></p>
        <div id="det">
          <p>Введите число строк и столбцов в соответствующие
```

текстовые поля.</p>

```
        </div>
    </div>
    <div id="output">
        <h2>Выходные параметры</h2>
        <p id="answer">Входные параметры не введены...</p>
    </div>
</div>
</body>
</html>
```

style.css

```
@font-face
{
    font-family:Roboto;
    src:url(fonts/roboto/Roboto.woff2)                format(woff2),
url(fonts/roboto/Roboto.woff)    format(woff),      url(fonts/roboto/Roboto.ttf)
format(truetype);
    font-weight:400;
    font-style:normal;
}

*
{
    margin:0;
    padding:0;
}

html
{
    font-size:62.5%;
}

body
{
    font-family:Roboto;
```

```
.wrapper
{
  max-width:800px;
  width:calc(100%-20px);
  border-left:1px solid #39c;
  border-right:1px solid #39c;
  margin:0 auto;
}
```

```
h2
{
  font-size:2em;
  color:#fff;
  text-align:center;
  padding:5px 8px;
  background:#39c;
}
```

```
p
{
  font-size:1.6em;
  margin:5px;
}
```

```
input
{
  font-size:1em;
}
```

```
label
{
  display:block;
  width:100%;
  text-align:center;
}
```

```
label > input
```

```

{
  width:100%;
  box-sizing:border-box;
  text-align:center;
  outline:0;
  padding:4px 2px;
}

#condition p
{
  text-align:justify;
}

#det
{
  display:flex;
  flex-wrap:wrap;
  text-align:center;
  margin:16px 6px;
}

.cell
{
  display:inline-block;
  font-size:1.6em;
  text-align:center;
  border:1px solid #39c;
  box-sizing:border-box;
}

#output
{
  border-bottom:1px solid #39c;
}

```

det.js

```
/**
 * Матрица посещений
 * Состояния ячеек сетки в виде двумерного массива
 */
var statesCells = [];

/**
 * Строк в разреженной матрице
 */
var m = 0;

/**
 * Столбцов в разреженной матрице
 */
var n = 0;

/**
 * Ненулевые элементы
 */
var a = [];

/**
 * Столбцы (j-индексы) ненулевых элементов
 */
var lj = [];

/**
 * Позиции i-индексов (строк) первых ненулевых элементов в массиве 'a'
 */
var li = [];

/**
 * Получить количество строк
 */
function getRow()
{
```

```

        return jQuery('#row').val();
    }

    /**
     * Получить количество столбцов
     */
    function getCol()
    {
        return jQuery('#col').val();
    }

    /**
     * Проверка значения на int
     * @param {int} val
     *     Значение для проверки
     */
    function isInt(val)
    {
        var result = false;
        if (Math.floor(val).toString() === val && jQuery.isNumeric(val))
        {
            result = true;
        }
        return result;
    }

    /**
     * Является ли число положительным
     * @param {int} val
     *     Значение для проверки
     */
    function numPositive(val)
    {
        var result = false;
        if (val >= 0)
        {
            result = true;
        }
    }

```



```

        return result;
    }

    /**
     * Округление результата
     * @param {double} value
     *     Значение
     * @param {double} decimals
     *     Число знаков после запятой
     */
    function round(value, decimals)
    {
        return Number(Math.round(value + 'e' + decimals) + 'e-' + decimals);
    }

    /**
     * Конвертировать одномерный массив в двумерный
     * @param {array} vector
     *     Одномерный массив
     * @param {int} col
     *     Количество ячеек в строке
     */
    function convMatrix(vector, col)
    {
        var matrix = [];
        while (vector.length)
        {
            matrix.push(vector.splice(0, col));
        }
        return matrix;
    }

    /**
     * Построение сетки
     * @param {int} row
     *     Число строк
     * @param {int} col
     *     Число столбцов

```

```

*/
function buildGrid(row, col)
{
    // Общее количество ячеек
    var totalCells = row * col;
    // Разметка ячеек
    var htmlCells = '';
    // Каскадные стили
    var styles = '.cell {width: calc(100% / ' + col + '); padding:
calc(50% / ' + col + ' - 1.6em) 0;}}';
    for (var i = 0; i < totalCells; i++)
    {
        htmlCells += '<input class="cell" type="number" value="0">';
        if ((i + 1) % col === 0)
        {
            htmlCells += '<br>';
        }
    }

    jQuery('head style').remove();
    jQuery('<style>' + styles + '</style>').appendTo('head');
    jQuery('#det').html(htmlCells);
}

/**
 * Установка размерности разреженной матрицы
 *
 * Построение сетки
 * @param {int} row
 *     Число строк
 * @param {int} col
 *     Число столбцов
 */
function setDimSparseMatrix(row, col)
{
    a = [];
    li = [];
    for (var i = 0; i < row + 1; i++)

```

```

        {
            li[i] = 1;
        }
        lj = [];
        n = row;
        m = col;
    }

/**
 * Проверка индексов
 *
 * @param {int} row
 *     Номер строки
 *
 * @param {int} col
 *     Номер столбца
 */
function validateCoordinates(row, col)
{
    try
    {
        if (row < 1 || col < 1 || row > m || col > n)
        {
            throw new Error('Ошибка в индексах');
        }
    }
    catch (e)
    {
        console.log(e.message);
    }
}

/**
 * Формирование элементов разреженной матрицы
 * в виде трех массивов
 *
 * @param {int} row

```

```

*   Номер строки
*
* @param {int} col
*   Номер столбца

* @param {double} val
*   Добавляемое значение
*/
function insert(row, col, val)
{
    if (a === [])
    {
        a = array();
        lj = array();
    }
    else
    {
        a.push(val);
        lj.push(col);
    }

    for (var i = row; i <= m; i++)
    {
        li[i] = li[i] + 1;
    }
}

/**
*   Если с в строке все элементы нулевые
*
* @param {int} row
*   Строка матрицы
*/
function remove(row)
{
    for (var i = row; i <= m; i++)
    {
        li[i] = li[i] + 1;
    }
}

```

```

    }
}

/**
 * Добавление значения разреженной матрицы
 *
 * @param {double} val
 *     Добавляемое значение
 *
 * @param {int} row
 *     Номер строки
 *
 * @param {int} col
 *     Номер столбца
 */
function set(val, row, col)
{
    validateCoordinates(row, col);

    var currCol = 0;
    for (var pos = li[row - 1] - 1; pos < li[row] - 1; pos++)
    {
        currCol = lj[pos];
        if (currCol >= col)
        {
            break;
        }
    }
    if (currCol !== col)
    {
        if (!(val === 0))
        {
            insert(row, col, val);
        }
    }
    else if (val === 0)
    {
        remove(row);
    }
}

```

```

    }
    else
    {
        a[pos] = val;
    }
}

/**
 * Получение значения элемента матрицы по индексам
 *
 * @param {int} row
 *   Номер строки
 *
 * @param {int} col
 *   Номер столбца
 *
 * @return {double}
 */
function get(row, col)
{
    validateCoordinates(row, col);

    var currCol;
    for (var pos = li[row - 1] - 1; pos < li[row] - 1; pos++)
    {
        currCol = lj[pos];
        if (currCol === col)
        {
            return a[pos];
        }
        else if (currCol > col)
        {
            break;
        }
    }

    return 0;
}

```

```

/**
 * Конвертация матрицы в разреженную матрицу
 *
 * @param {array} matrix
 *   Входная матрица
 */
function readMatrix(matrix)
{
    var rows = matrix.length;
    for (var i = 0; i < rows; i++)
    {
        var cols = matrix[i].length;
        for (var j = 0; j < cols; j++)
        {
            set(matrix[i][j], i + 1, j + 1);
        }
    }
}

/**
 * Вывод разреженной матрицы
 */
function printMatrix()
{
    var show = '';
    for (var i = 1; i <= m; i++)
    {
        for (var j = 1; j <= n; j++)
        {
            show += get(i, j) + ' ';
        }
        show += '\n';
    }
    return show;
}

/**

```

```

* Получение все разреженной матрицы
*/
function getMatrix()
{
    var matrix = [];
    for (var i = 1; i <= m; i++)
    {
        matrix.push([]);
        for (var j = 1; j <= n; j++)
        {
            matrix[i - 1][j - 1] = get(i, j);
        }
    }
    return matrix;
}

```

```

/**
* Вывод массива A
*/
function getA()
{
    var show = '';
    a.forEach(function(val)
    {
        show += val + ' ';
    });
    return show;
}

```

```

/**
* Вывод массива LI
*/
function getLI()
{
    var show = '';
    li.forEach(function(val)
    {

```



```

        show += val + ' ';
    });
    return show;
}

/**
 * Вывод массива LJ
 */
function getLJ()
{
    var show = '';
    lj.forEach(function(val)
    {
        show += val + ' ';
    });
    return show;
}

/**
 * Проверка матрицы на признак структурно-симметричности
 */
function checkStructSym()
{
    var answer = true;
    var col = m; // Количество строк в матрице
    for (var i = 0; i < col; i++)
    { // Проход по матрице
        var row = n; // Элементы строки
        if (col === row)
        { // Симметричная матрица всегда квадратная
            for (var j = 0; j < row; j++)
            {
                if (i !== j)
                { // Главную диагональ не учитываем
                    answer = answer && get(i + 1, j + 1) === get(j + 1,
i + 1);

                    if (answer === false)
                    {

```

```

        break;
    }
}
}
}
else
{
    answer = false;
    break;
}
}
return answer;
}

/**
 * Поиск определителя методом Гаусса
 * Приводит матрицу к верхнему-треугольному виду
 * и перемножает элементы на главной диагонали
 */
function detGauss()
{
    var dimension = m; // Размерность
    var tempMatrix = getMatrix(); // Временная матрица
    var det = 1.0; // Определитель
    if (dimension >= 1)
    {
        for (var i = 0; i < dimension; i++)
        {
            if (tempMatrix[i][i] === 0)
            {
                // Поиск максимального элемента в колонке
                var maxEl = tempMatrix[i][i];
                var maxRow = i;
                for (var k = i + 1; k < dimension; k++)
                {
                    if (Math.abs(tempMatrix[k][i]) > maxEl)
                    {
                        // Запоминаем максимальный элемент и его строку

```

```

        maxEl = Math.abs(tempMatrix[k][i]);
        maxRow = k;
    }
    if (maxEl === 0)
    {
        return 0;
    }
}
// Складываем строки (по столбцам)
for (var k = i; k < dimension; k++)
{
    tempMatrix[i][k] += tempMatrix[maxRow][k];
}
}
for (var k = i + 1; k < dimension; k++)
{
    var c = -tempMatrix[k][i] / tempMatrix[i][i];
    for (var j = i; j < dimension; j++)
    {
        if (i !== j)
        {
            tempMatrix[k][j] += c * tempMatrix[i][j];
        }
        else
        {
            tempMatrix[k][j] = 0;
        }
    }
}
}
}
else
{
    det = 0;
}
// Перемножаем элементы на главной диагонали
for (var i = 0; i < dimension; i++)
{

```

```

        det *= tempMatrix[i][i];
    }
    return round(det, 2);
}

/**
 * Считывание состояния ячеек из сетки
 * @param {int} col
 *   число колонок
 */
function readGrid(col)
{
    col = parseInt(col);
    var states = []; // Состояния ячеек сетки
    var det = 'Невозможно вычислить'; // Определитель матрицы в виде
строки
    var vals = ''; // Ненулевые элементы в виде строки
    var rows = ''; // Позиции i-индексов (строк) первых ненулевых
элементов в массиве 'a' в виде строки
    var cols = ''; // Столбцы (j-индексы) ненулевых элементов в виде
строки
    var allData = ''; // Сбор всех выходных данных в виде строки
    var structSym = false; // Признак структурно-симметричности в виде
строки
    var quantity = jQuery('#det .cell').length;
    if (quantity > 0)
    {
        // Перебор ячеек сетки
        jQuery('#det .cell').each(function()
        {
            states.push(parseFloat(jQuery(this).val()));
        });
    }

    // Одномерные массивы конвертировать в двумерные
    statesCells = convMatrix(states, col);

```

```

// Размерность разреженной матрицы
setDimSparseMatrix(col, col);

// Добавление значений в разреженную матрицу
readMatrix(statesCells);

// Получаем данные трех массивов 'a', 'li', 'lj'
vals = getA();
rows = getLI();
cols = getLJ();

structSym = checkStructSym();
if (structSym === true)
{
    det = detGauss();
}
else
{
    det = 'Матрица не является структурно-симметричной';
}

vals = 'A : <b>' + vals + '</b><br>';
rows = 'LI: <b>' + rows + '</b><br>';
cols = 'LJ: <b>' + cols + '</b><br>';
det = 'Определитель матрицы: <b>' + det + '</b>';
allData = vals + rows + cols + det;
// Вывести ответ на экран
jQuery('#answer').html(allData);
}

/**
 * Выполняем действия когда DOM полностью загружен
 */
jQuery(document).ready(function()
{
    // Отслеживаем изменение числа строк и числа столбцов
    jQuery('#row, #col').on('keyup', function()

```

```

{
    var row = getRow(); // Количество строк на сетке
    var col = getCol(); // Количество столбцов на сетке
    var message = '';    // Сообщение пользователю

    if (row === '' || col === '')
    { // Значения не пустые
        message = 'Введите число строк и столбцов в соответствующие
текстовые поля.';
    }
    else if (!isInt(row) || !isInt(col))
    { // Являются Int
        message = 'Входные параметры не являются целочисленными.';
    }
    else if (!numPositive(row) || !numPositive(col))
    { // Положительные числа
        message = '<b>Ошибка! </b>Входные параметры должны иметь
положительные значения!';
    }
    else if (row > 15 || col > 15)
    {
        message = '<b>Ошибка! </b>Введите значения текстовых полей
<= 15.';
    }

    if (message === '')
    {
        buildGrid(row, col); // Построение сетки
        readGrid(col);       // Считывание состояния ячеек из сетки
    }
    else
    {
        jQuery('#det').html('<p>' + message + '</p>');
    }
});
// Отслеживаем клик по текстовому полю
jQuery('#det').on('click', '.cell', function()
{

```

```
        jQuery(this).select();
    });
    // Отслеживаем изменение текстового поля
    jQuery('#det').on('keyup', '.cell', function()
    {
        var col = getCol(); // Количество столбцов на сетке
        readGrid(col);      // Считывание состояния ячеек из сетки
    });
});
```

3 Скриншоты выполнения программы


Входные параметры					
Число строк:					
6					
Число столбцов:					
6					
0	2	1	5	0	0
2	0	0	0	0	0
1	0	5	0	0	0
5	0	0	7	1 	0
0	0	0	1	0	0
0	0	0	0	0	3

Рисунок 1 – Ввод входных параметров с клавиатуры

Выходные параметры
A: 2 1 5 2 1 5 5 7 1 1 3 Ll: 1 4 5 7 1 0 1 1 1 2 LJ: 2 3 4 1 1 3 1 4 5 4 6 Определитель матрицы: 60

Рисунок 2 – Результат выполнения программы

4 Заключение

В ходе выполнения лабораторной работы изучены принципы хранения и обработки разреженных матриц.

5 Используемая литература

- 1 Методические указания по выполнению лабораторных работ по дисциплине «Технологии программирования» – Нижний Новгород.: НГТУ, 2015. – 18 с.
- 2 Стандарт предприятия СТП 1-У-НГТУ-2004. Общие требования к оформлению пояснительных записок дипломных и курсовых проектов. – Нижний Новгород.: НГТУ, 2004. – 22 с.

