



**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

**ФАКУЛЬТЕТ
КАФЕДРА**

**ИУ
ИУ6**

**ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ
КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ**

**Отчет
по домашней работе №1**

Дисциплина: Прикладная теория цифровых автоматов

Название: Реализация конечного автомата

Вариант 18

Студент гр. ИУ6-42

Преподаватель

(Подпись, дата)

(Подпись, дата)

М.А. Мотичев

(И.О. Фамилия)

А.М. Губарь

(И.О. Фамилия)

ЗАДАНИЕ

Вариант 18 - Судоку.

Игра «Судоку» заключается в том, что необходимо заполнить пустые клетки цифрами от 1 до 9, так, чтобы не было совпадений с цифрами стоящими с этой клеткой в одном ряду, в одной строке, в одном блоке по 9 клеточек.

РЕШЕНИЕ

В этой игре будет взаимодействовать пользователь и компьютер, так как я выбрал программную реализацию автомата. То есть сигналы, получаемые автоматом - некоторые действия пользователя:

- A1 - выбор пустой ячейки;
- A2 - выбор заполненной ячейки;
- A3 - в ячейку оставлено число;
- A4 - из ячейки удалено число;
- A5 - нажата кнопка генерации нового судоку.

Эти действия переводят автомат в разные состояния:

- Y1 - выделена пустая ячейка;
- Y2 - выделена заполненная ячейка;
- Y3 - выделена единственная ошибочная ячейка;
- Y4 - выделена пустая ячейка, при наличии хотя бы одной ошибки;
- Y5 - выделена заполненная ячейка, при наличии хотя бы одной ошибки;
- Y6 - решённый судоку;
- Y7 - начальное состояние.

Реализуем автомат в виде графа:

/Место для графа/

А теперь руководствуясь графом реализуем программу. Работа программы:

				5				
4	1	7		3		5	2	
			1					
			6			2	8	
		6	8	2	5	4	1	
				1		3		6
8	5		4	7		9		3
	4	1	3		9		5	2
	6	3		8		1		

Сгенерировать новый

- Выберите пустую ячейку.
- Серые клетки не трогай - они тебе даны изначально.
- Удаление на 0.
- Красные клетки - то что ты не хочешь увидеть - это ошибки.

Рис.1. Состояние У7

				5				
4	1	7		3		5	2	
			1					
			6			2	8	
		6	8	2	5	4	1	
				1		3		6
8	5		4	7		9		3
	4	1	3		9		5	2
	6	3		8		1		

Сгенерировать новый

- Выберите пустую ячейку.
- Серые клетки не трогай - они тебе даны изначально.
- Удаление на 0.
- Красные клетки - то что ты не хочешь увидеть - это ошибки.

Рис. 2. Состояние У1

				5				
4	1	7		3		5	2	
			1					9
			6			2	8	5
		6	8	2	5	4	1	7
				1		3	9	6
8	5	2	4	7	1	9	6	3
7	4	1	3	6	9	8	5	2
9	6	3	5	8	2	1	7	4

Сгенерировать новый

- Выберите пустую ячейку.
- Серые клетки не трогай - они тебе даны изначально.
- Удаление на 0.
- Красные клетки - то что ты не хочешь увидеть - это ошибки.

Рис. 3. Состояние У2

6	3	9	2	5	8	7	4	1
4	1	7		3		5	2	8
			1			7	3	9
			6			2	8	5
		6	8	2	5	4	1	7
				1		3	9	6
8	5	2	4	7	1	9	6	3
7	4	1	3	6	9	8	5	2
9	6	3	5	8	2	1	7	4

Сгенерировать новый

- Выберите пустую ячейку.
- Серые клетки не трогай - они тебе даны изначально.
- Удаление на 0.
- Красные клетки - то что ты не хочешь увидеть - это ошибки.

Рис. 4. Состояние У3

6	3	9	2	5	8	7	4	1
4	1	7		3	6	5	2	8
			1	4	7	7	3	9
			6			2	8	5
		6	8	2	5	4	1	7
				1		3	9	6
8	5	2	4	7	1	9	6	3
7	4	1	3	6	9	8	5	2
9	6	3	5	8	2	1	7	4

Сгенерировать новый

- Выберите пустую ячейку.
- Серые клетки не трогай - они тебе даны изначально.
- Удаление на 0.
- Красные клетки - то что ты не хочешь увидеть - это ошибки.

Рис. 5. Состояние У4

6	3	9	2	5	8	7	4	1
4	1	7		3	6	5	2	8
			1	4	7	7	3	9
			6			2	8	5
		6	8	2	5	4	1	7
				1		3	9	6
8	5	2	4	7	1	9	6	3
7	4	1	3	6	9	8	5	2
9	6	3	5	8	2	1	7	4

Сгенерировать новый

- Выберите пустую ячейку.
- Серые клетки не трогай - они тебе даны изначально.
- Удаление на 0.
- Красные клетки - то что ты не хочешь увидеть - это ошибки.

Рис. 6. Состояние У5

6	3	9	2	5	8	7	4	1
4	1	7	9	3	6	5	2	8
2	8	5	1	4	7	6	3	9
1	7	4	6	9	3	2	8	5
3	9	6	8	2	5	4	1	7
5	2	8	7	1	4	3	9	6
8	5	2	4	7	1	9	6	3
7	4	1	3	6	9	8	5	2
9	6	3	5	8	2	1	7	4

Сгенерировать новый

- Выберите пустую ячейку.
- Серые клетки не трогай - они тебе даны изначально.
- Удаление на 0.
- Красные клетки - то что ты не хочешь увидеть - это ошибки.

Рис. 7. Состояние У6

ПРИЛОЖЕНИЕ

Код файла index.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Домашняя работа</title>
    <script src="solve.js"></script>
    <script src="p5.js"></script>
    <script src="p5.dom.js"></script>
  </head>
  <body style="font-family: Helvetica, sans-serif">
    <div id="canv" align="center">
    </div>
    <br>
    <table align="center">
      <tr>
        <td>
          <div id="genbut" align="center">
          </div>
        </td>
        <td>
          <div id="txt" align="left">
          </div>
        </td>
      </tr>
    </table>
  </body>
</html>
```

Код файла solve.js:

```
//=====Задание переменных=====

let grid = []; // sudoku на экране массив 9X9 из Cell
let cellw = 0; // размер ячейки
let genButton; // кнопка сброса
let texting;

let currentCell; // текущая выбранная ячейка == Cell
let lessCells = 0; //количество оставшихся заполнить ячеек/сложность
let errors = 0; //количество ошибок

let full = []; //
let done = []; //

const DIFFICULTY = 45; //сложность sudoku - чем больше число, тем сложнее

//=====Реализация=====
//-----Интерфейс-----

function setup() {
  cellw = floor(((windowWidth < windowHeight) ? windowWidth : windowHeight) -
20) / 12); //определяем размер ячейки
  createCanvas(9 * cellw + 1, 9 * cellw + 1).
    parent('canv'); //выделяем поле под sudoku
  textAlign(CENTER); //цифра будет по центру
  textSize(cellw / 2); //цифра в половину высоты и это все для красоты
  for(let i = 0; i < 9; i++) {
    grid[i] = [] // задаём строчку для ячеек
    for(let j = 0; j < 9; j++) {
      grid[i][j] = new Cell(i, j); //заполняем её ячейками
      grid[i][j].getRect(); // координаты квадратов 3x3
    }
  }

  genButton = createButton("Сгенерировать новый"). // объект кнопка
    mousePressed(Generate).
    style('height', '70px'). // повыше
    style('border-radius', '4px'). // закруглим
    style('font-family', 'Helvetica').
    style('font-size', '16px').
    parent('genbut'); // id в html файле

  texting = createElement('pravila', //объект текстовое поле
    '<span style="color: blue">• Выберите пустую ячейку.</span><br>' +
    '<span style="color: grey">• Серые клетки не трогай - они тебе даны
изначально.</span><br>' +
    '<span style="color: black">• Удаление на 0.</span><br>' +
    '<span style="color: red">• Красные клетки - то что ты не хочешь увидеть -
это ошибки.</span>').
    style('font-size', '14px').
    parent('txt'); // id в html файле
}

function draw() {
  background(255);
  strokeWeight(1);
  for(let i = 0; i < 9; i++) {
    for(let j = 0; j < 9; j++) {
      grid[i][j].show();
    }
  }
  drawLines(); // нарисуем линии потолка для квадратов 3x3
}
```

```

function Generate() {
  errors = 0;
  firstSudoku();
  myShuffle();
  deleter();
  for(let i = 0; i < 9; i++) {
    for(let j = 0; j < 9; j++) {
      grid[i][j].clear();
      grid[i][j].value = done[i][j];
      if(grid[i][j].value != 0){
        grid[i][j].block = true;
      }
    }
  }
  console.log("les - " + lessCells);
  console.log('er - ' + errors);
}

//-----Решение sudoku пользователем-----

function keyTyped() { // отработка нажатия цифры 0 ... 9
  if ((key >= 0 && key <= 9) ) {
    typeNumber(key); // набрать число
  }
  console.log("les - " + lessCells);
  console.log('er - ' + errors);
}

function typeNumber(v) {
  if(!currentCell || currentCell.block == true) return; // если ячейка задана
  изначально - выходим
  let i = currentCell.i; // берём номер строки ячейки
  let j = currentCell.j; // берём номер столбца ячейки

  if(((checkRows(i, j, v) && checkCols(i, j, v) && checkRect(i, j, v)) || v ==
0) ){
    if (currentCell.value == 0 && v != 0){
      lessCells = lessCells - 1;
    }
    if (currentCell.value != 0 && v == 0){
      lessCells ++;
    }
    if (currentCell.error == true){
      errors = errors - 1;
    }
    currentCell.value = v;
    currentCell.error = false;
    if (errors == 0 && lessCells == 0){
      for(let i = 0; i < 9; i++) {
        for(let j = 0; j < 9; j++) {
          grid[i][j].block = true;
        }
      }
      alert("Вы всё-таки смогли это решить!");
    }
  } else {
    if (currentCell.error == false){
      errors ++;
      if (currentCell.value == 0){
        lessCells = lessCells -1;
      }
    }
    currentCell.value = v;
    currentCell.error = true;
  }
}

```



```

function checkRect(i, j, n) { // проверяем в квадрате 3x3
  let ni = grid[i][j].rectx;
  let nj = grid[i][j].recty;
  for(let nis = ni; nis < (ni + 3); nis++) {
    for(let njs = nj; njs < (nj + 3); njs++) {
      if(i == nis && j == njs) continue; // обходим нашу ячейку

      if(grid[nis][njs].value == n) // если есть совпадение, то
        return false; // выходим с ошибкой
    }
  }
  return true;
}

function checkCols(i, j, n) { // проверка по столбцу
  for(let nj = 0; nj < 9; nj++) {
    if(nj == j) continue; // обходим нашу ячейку

    if(grid[i][nj].value == n) // если есть совпадение, то
      return false; // выходим с ошибкой
  }
  return true; // раз дошли - то все верно
}

function checkRows(i, j, n) { // проверка по строке
  for(let ni = 0; ni < 9; ni++) {
    if(ni == i) continue; // обходим нашу ячейку

    if(grid[ni][j].value == n) // если есть совпадение, то
      return false; // выходим с ошибкой
  }
  return true; // раз дошли - то все верно
}

function mousePressed() { // нажатие мышкой
  if(mouseX < 0 || mouseX > width || mouseY < 0 || mouseY > height) return; //
  Вернуться, если промазал по всему

  for(let i = 0; i < 9; i++) { // в цикле перебираем
    for(let j = 0; j < 9; j++) { // куда мы попали
      let x = grid[i][j].x; // начало каждой ячейки по x
      let y = grid[i][j].y; // и по y

      if(mouseX >= x && mouseX < x + cellw && mouseY >= y && mouseY < y + cellw)
      { // если попали в проверяемую ячейку
        currentCell = grid[i][j]; // задаем выбранную ячейку текущей
        currentCell.chosen = true; // текущая ячейка выбрана (изменён цвет)
      } else
        grid[i][j].chosen = false; // если промазали, то оставляем цвет белым
      (заодно очищается на белый цвет предыдущая ячейка)
    }
  }
}

function drawLines() { // рисуем сетку 3x3 более толстыми линиями
  strokeWeight(3); //толщина 3

  for(let l = 0; l <= 3; l++) {
    line(0, cellw * (l * 3), width, cellw * (l * 3));
    line(cellw * (l * 3), 0, cellw * (l * 3), height);
  }
}

```

```

function Cell(i, j) { // объект ячейка
  this.i = i; // номер строки
  this.j = j; // номер столбца
  this.x = j * cellw; // координата начала ячейки по x
  this.y = i * cellw; // координата ячейки по y
  this.rectx = 0; // координата x квадрата ячейки
  this.recty = 0; // координата y квадрата ячейки
  this.value = 0; // число в ячейке
  this.chosen = false; // выбрана ли ячейка
  this.block = false; // была ли ячейка заранее задана
  this.error = false; // поставлена ли ошибка
  this.backgroundColor = 255; // цвет экрана
  this.getRect = function() { // задание координат квадрата каждой ячейки
    for(let ni = 0; ni < 9; ni += 3) {
      for(let nj = 0; nj < 9; nj += 3) {
        if(this.i >= ni && this.i < ni + 3 && this.j >= nj && this.j < nj + 3) {
          this.rectx = ni;
          this.recty = nj;
          return;
        }
      }
    }
  }
}

this.clear = function(){ // очищение(вызывается при генерации нового)
  noFill();
  this.value = 0;
  this.chosen = false;
  this.block = false;
  this.error = false;
}

this.show = function() { // показ ячейки (её цвет, )
  noFill();

  if(this.block)
    fill(225); // серый блок, заполненный изначально
  if(this.error)
    fill(255, 155, 155); // красненький, неправильный блок
  if(this.chosen)
    fill(0, 180, 255); // голубой блок, выбранный изначально
  if(this.block && this.chosen) // тёмно серый, выбран заполненный изначально
    fill(175);
  if(this.chosen && this.error) // тёмно красненький, выбран неправильный блок
    fill(255, 125, 125);

  rect(this.x, this.y, cellw, cellw); // рисуем границы ячейки !

  if(this.value > 0) { // пишем в неё поставленное число
    fill(0);
    text(this.value, this.x + cellw / 2, this.y + cellw / 1.5);
  }
}

}

//-----Генерация-----

function firstSudocu(){ // задаём простейший судоку
  for(let i = 0; i < 9; i++) {
    full[i] = [];
    for(let j = 0; j < 9; j++) {
      full[i][j] = (((i*3 + j) + Math.floor(i / 3)) % 9) + 1; // вот и формула
    }
  }
}

```

```

function myShuffle(){
  let num = Math.floor(Math.random()* DIFFICULTY + DIFFICULTY);
  for(let i = 0; i < num; i++){
    let func_num = Math.floor(Math.random()*5);
    switch (func_num){
      case 0 :
        change_a();
        break;
      case 1 :
        change_b();
        break;
      case 2 :
        change_c();
        break;
      case 3 :
        change_d();
        break;
      case 4 :
        change_e();
        break;
      default : break;
    }
  }
}

function change_b(){ //смена двух строк
  let buf = [];
  for(let i = 0; i < 9; i++) {
    buf[i] = [];
    for(let j = 0; j < 9; j++) {
      buf [i][j] = full[i][j];
    }
  }
  let one = Math.floor(Math.random()*3);
  let two = Math.floor(Math.random()*3);
  if (one == 0){
    for(let j = 0; j < 9; j++) {
      full[two*3][j] = buf [two*3 + 1][j];
      full[two*3 + 1][j] = buf [two*3][j];
    }
  }
  if (one == 1){
    for(let j = 0; j < 9; j++) {
      full[two*3 + 1][j] = buf [two*3 + 2][j];
      full[two*3 + 2][j] = buf [two*3 + 1][j];
    }
  }
  if (one == 2){
    for(let j = 0; j < 9; j++) {
      full[two*3][j] = buf [two*3 + 2][j];
      full[two*3 + 2][j] = buf [two*3][j];
    }
  }
  return;
}

```

```

function change_c(){ //смена двух столбцов
    let buf = [];
    for(let i = 0; i < 9; i++) {
        buf[i] = [];
        for(let j = 0; j < 9; j++) {
            buf [i][j] = full[i][j];
        }
    }
    let one = Math.floor(Math.random()*3);
    let two = Math.floor(Math.random()*3);
    if (one == 0){
        for(let i = 0; i < 9; i++) {
            full[i][two*3] = buf [i][two*3 + 1];
            full[i][two*3 + 1] = buf [i][two*3];
        }
    }
    if (one == 1){
        for(let i = 0; i < 9; i++) {
            full[i][two*3 + 1] = buf [i][two*3 + 2];
            full[i][two*3 + 2] = buf [i][two*3 + 1];
        }
    }
    if (one == 2){
        for(let i = 0; i < 9; i++) {
            full[i][two*3] = buf [i][two*3 + 2];
            full[i][two*3 + 2] = buf [i][two*3];
        }
    }
    return;
}

```

```

function change_d(){ // смена строк по три
    let buf = [];
    for(let i = 0; i < 9; i++) {
        buf[i] = [];
        for(let j = 0; j < 9; j++) {
            buf [i][j] = full[i][j];
        }
    }
    let one = Math.floor(Math.random()*3);
    if (one == 0){
        for(let i = 0; i < 3; i++) {
            for(let j = 0; j < 9; j++){
                full[i][j] = buf [i + 3][j];
                full[i + 3][j] = buf [i][j];
            }
        }
    }
    if (one == 1){
        for(let i = 0; i < 3; i++) {
            for(let j = 0; j < 9; j++){
                full[i][j] = buf [i + 6][j];
                full[i + 6][j] = buf [i][j];
            }
        }
    }
    if (one == 2){
        for(let i = 3; i < 6; i++) {
            for(let j = 0; j < 9; j++){
                full[i][j] = buf [i + 3][j];
                full[i + 3][j] = buf [i][j];
            }
        }
    }
    return;
}

```

```

function change_e(){ // смена столбцов по три
    let buf = [];
    for(let i = 0; i < 9; i++) {
        buf[i] = [];
        for(let j = 0; j < 9; j++) {
            buf [i][j] = full[i][j];
        }
    }
    let one = Math.floor(Math.random()*3);
    if (one == 0){
        for(let i = 0; i < 9; i++) {
            for(let j = 0; j<3; j++){
                full[i][j] = buf [i][j+3];
                full[i][j+3] = buf [i][j];
            }
        }
    }
    if (one == 1){
        for(let i = 0; i < 9; i++) {
            for(let j = 0; j<3; j++){
                full[i][j] = buf [i][j+6];
                full[i][j+6] = buf [i][j];
            }
        }
    }
    if (one == 2){
        for(let i = 0; i < 9; i++) {
            for(let j = 3; j < 6; j++){
                full[i][j] = buf [i][j+3];
                full[i][j+3] = buf [i][j];
            }
        }
    }
    return;
}

function change_a(){ // транспонирование
    let buf = []
    for(let i = 0; i < 9; i++) {
        buf[i] = [];
        for(let j = 0; j < 9; j++) {
            buf [i][j] = full[i][j];
        }
    }
    for(let i = 0; i < 9; i++) {
        for(let j = 0; j < 9; j++) {
            full[i][j] = buf [j][i];
        }
    }
    return;
}

function deleter(){ //удаление эл-тов sudoku
    lessCells = 0;
    for(let i = 0; i < 9; i++) {
        done[i] = [];
        for(let j = 0; j < 9; j++) {
            done[i][j] = full[i][j];
        }
    }
    for(let i = 0; i < 81; i++){
        if( Math.floor(Math.random() * 81) < DIFFICULTY){
            done[Math.floor(i / 9)][i % 9] = 0;
            lessCells++;
        }
    }
    return;
}

```