Министерство образования Республики Беларусь Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования Кафедра инженерной психологии и эргономики

Дисциплина: Современные языки программирования

Лабораторная работа № 9 Вариант № 10

Выполнил:

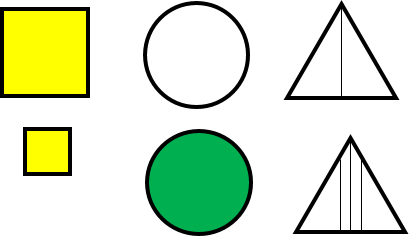
Атаев И.М гр. 910101 Проверила:

Василькова А.Н.

Минск 2022

Задание:

1. Представьте фигуры на картинке в виде объектов. Обратите внимание, фигуры имеют одинаковые параметры.



Выведите: свойства, которые отличают фигуру «зеленый круг»; свойства, которые описывают фигуру «треугольник с тремя линиями»; есть ли у фигуры «маленький квадрат» собственное свойство, которое определяет цвет фигуры.

1. Разработайте алгоритм игры «Судоку». У пользователя есть на выбор 2 варианта игры: судоку с проверками в строках и столбцах; и классическое судоку с проверками в строках, столбцах и малых квадратах (показаны на рисунке). Размер игрового поля 9х9 нельзя изменить. Допустим, пользователю доступны кнопки «Новая игра» и «Проверить». При нажатии на кнопку «Новая игра» генерируется частично заполненное новое игровое поле (в строках и столбцах цифры не повторяются). Для простоты решения задачи предлагается следующий алгоритм генерации чисел: диапазон генерации случайного числа [1, 18], если сгенерированное число больше 9, то текущее значение принимается равным 0,

т.е. клетка пустая.

При нажатии на кнопку «Проверить» осуществляется проверка игрового поля: в первом варианте игры – в столбцах и строках, в классическом судоку – в столбцах, в строках и в малых квадратах.

Для демонстрации работы алгоритма проверки (кнопка «Проверить») разрешается заполнить игровое поле готовым решением.

Листинг кода:

# #index.js

const shape = { borderWidth: 2

};

const square = {

sideLength: 10, color: "yellow",

proto : shape

};

const smallSquare = { sideLength: 5,

proto : square

};

const circle = { radiusSize: 5,

proto : shape

};

const greenCircle = { color: "green",

proto : circle

};

const equilateralTriangle = { sideLength: 10,

numberLines: 1,

proto : shape

};

const equilateralTriangleThreeLines = { numberLines: 3,

proto : equilateralTriangle

};

console.log(`Свойства, которые отличают фигуру «зеленый круг»:

${Object.keys(greenCircle)}`);

const equilateralTriangleThreeLinesProps = [];

for (const prop in equilateralTriangleThreeLines) { equilateralTriangleThreeLinesProps.push(prop);

}

console.log(`Свойства, которые описывают фигуру «треугольник с тремя линиями»: ${equilateralTriangleThreeLinesProps}`);

const isThereColor = smallSquare.hasOwnProperty("color") ? "Да" : "Нет";

console.log(`Есть ли у фигуры «маленький квадрат» собственное свойство, которое определяет цвет фигуры?\n${isThereColor}`);

# #index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

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

<link rel="stylesheet" href="style.css">

<title>Sudoku</title>

</head>

<body>

<div class="wrapper">

<h1>Sudoku</h1><hr>

<div class="result">

<button class="new-game">Новая игра</button>

<button class="check">Проверить</button>

<h2 class="errors">Ошибок: 0</h2>

</div>

<div class="board"></div><br>

<div class="digits"></div>

</div>

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

</body>

</html>

# #script.js

const n = 9

const baseBoard = [

[1, 2, 3, 4, 5, 6, 7, 8, 9],

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | [4, | 5, | 6, | 7, | 8, | 9, | 1, | 2, | 3], |
| [7, | 8, | 9, | 1, | 2, | 3, | 4, | 5, | 6], |
| [2, | 3, | 4, | 5, | 6, | 7, | 8, | 9, | 1], |
| [5, | 6, | 7, | 8, | 9, | 1, | 2, | 3, | 4], |
| [8, | 9, | 1, | 2, | 3, | 4, | 5, | 6, | 7], |
| [3, | 4, | 5, | 6, | 7, | 8, | 9, | 1, | 2], |
| [6, | 7, | 8, | 9, | 1, | 2, | 3, | 4, | 5], |
| [9, | 1, | 2, | 3, | 4, | 5, | 6, | 7, | 8], |
| ]; |  |  |  |  |  |  |  |  |  |

let solutionBoard = [];

let countErrors = 0;

let selectedNumber = 0;

window.onload = function() {

for (let row = 0; row < n; row++) {

for (let col = 0; col < n; col++) {

const tile = document.createElement("div"); tile.classList.add("tile");

if (row == 2 || row == 5) { tile.classList.add("horizontal-line");

}

if (col == 2 || col == 5) { tile.classList.add("vertical-line");

}

const board = document.querySelector(".board"); board.appendChild(tile);

tile.addEventListener("click", selectTile);

}

}

for (let i = 1; i <= n; i++) {

let number = document.createElement("div"); number.innerText = i;

number.classList.add("number");

const digits = document.querySelector(".digits"); digits.appendChild(number);

number.addEventListener("click", selectNumber);

}

}

const newGameBtn = document.querySelector(".new-game"); newGameBtn.addEventListener("click", newGame);

const checkBtn = document.querySelector(".check"); checkBtn.addEventListener("click", check);

const errorsTitle = document.querySelector(".errors");

function selectNumber(event) {

const digits = document.querySelector(".digits");

const numbers = digits.querySelectorAll(".number"); numbers.forEach(number => {

number.classList.remove("selected");

});

event.target.classList.add("selected"); selectedNumber = event.target.textContent;

}

function selectTile(event) {

if (event.target.textContent === "" && selectedNumber !== 0) { event.target.textContent = selectedNumber;

}

}

function newGame() {

const endGameTitle = document.querySelector(".end-game"); if (endGameTitle) {

endGameTitle.remove();

}

const numbers = document.querySelectorAll(".number"); numbers.forEach(number => {

number.classList.remove("selected");

});

selectNumber = 0;

countErrors = 0;

errorsTitle.textContent = `Ошибок: ${countErrors}`;

const tiles = document.querySelectorAll(".tile"); tiles.forEach(tile => {

tile.innerText = ""; tile.classList.remove("tile-error");

});

solutionBoard = shuffleBoard(baseBoard);

const shuffledBoard = solutionBoard.slice(); for (let i = 0; i < solutionBoard.length; i++) {

shuffledBoard[i] = solutionBoard[i].slice();

}

const removedTilesBoard = deleteBoardsTiles(shuffledBoard);

const flatTilesBoard = removedTilesBoard.flat();

for (let i = 0; i < tiles.length; i++) { if (flatTilesBoard[i] !== 0) {

tiles[i].innerText = flatTilesBoard[i];

}

}

}

function check() {

const endGameTitle = document.querySelector(".end-game"); if (endGameTitle) {

endGameTitle.remove();

}

let isWin = true;

const tiles = document.querySelectorAll(".tile");

const flatSolutionBoard = solutionBoard.flat();

for (let i = 0; i < tiles.length; i++) { if (tiles[i].textContent !== "") {

if (+tiles[i].textContent !== flatSolutionBoard[i]) { tiles[i].classList.add("tile-error");

errorsTitle.textContent = `Ошибок: ${++countErrors}`;

setTimeout(() => { tiles[i].textContent = "";

tiles[i].classList.remove("tile-error");

}, 1000);

}

} else {

isWin = false;

}

}

if (isWin) {

endGame();

}

}

function endGame() {

const numbers = document.querySelectorAll(".number"); numbers.forEach(number => {

number.classList.remove("selected");

});

selectNumber = 0;

const wrapper = document.querySelector(".wrapper");

const endGameTitle = document.createElement("h2"); endGameTitle.classList.add("end-game");

endGameTitle.innerText = "Вы выиграли!!!";

wrapper.appendChild(endGameTitle);

}

//transponing board

function transponing(board) {

return board[0].map((col, i) => board.map(row => row[i]));

}

//swap rows

function swapRows(board) {

for (let i = 0; i < 3; i++) {

const area = Math.floor(Math.random() \* 3);

const lineOne = Math.floor(Math.random() \* 3); const indexLineOne = area \* 3 + lineOne;

let lineTwo = Math.floor(Math.random() \* 3); while (lineOne === lineTwo) {

lineTwo = Math.floor(Math.random() \* 3);

}

const indexLineTwo = area \* 3 + lineTwo;

[board[indexLineOne], board[indexLineTwo]] = [board[indexLineTwo], board[indexLineOne]];

}

return board;

}

//swap columns

function swapColumns(board) {

const transposeBoard = transponing(board); const swapRowsBoard = swapRows(transposeBoard); return transponing(swapRowsBoard);

}

//swap rows areas

function swapRowsAreas(board) { for (let i = 0; i < 3; i++) {

const areaOne = Math.floor(Math.random() \* 3);

let areaTwo = Math.floor(Math.random() \* 3); while (areaOne === areaTwo) {

areaTwo = Math.floor(Math.random() \* 3);

}

for (let j = 0; j < 3; j++) {

const indexLineOne = areaOne \* 3 + j; const indexLineTwo = areaTwo \* 3 + j;

[board[indexLineOne], board[indexLineTwo]] = [board[indexLineTwo], board[indexLineOne]];

}

}

return board;

}

//swap columns areas

function swapColumnsAreas(board) {

const transposeBoard = transponing(board);

const swapRowsAreasBoard = swapRowsAreas(transposeBoard); return transponing(swapRowsAreasBoard);

}

//shuffle board

function shuffleBoard(board) {

const swapRowsBoard = swapRows(board);

const swapColumnsBoard = swapColumns(swapRowsBoard);

const swapRowsAreasBoard = swapRowsAreas(swapColumnsBoard);

return swapColumnsAreas(swapRowsAreasBoard);

}

//delete board's tiles

function deleteBoardsTiles(board) { for (let i = 0; i < 9; i++) {

let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9];

const countDeleteNumber = Math.floor(Math.random() \* (8 - 4) + 4);

for (let j = 0; j < countDeleteNumber; j++) {

const deleteIndex = Math.floor(Math.random() \* numbers.length); numbers.splice(deleteIndex, 1);

board[i][deleteIndex] = 0;

}

}

return board;

}

# #style.css

body {

font-family: Arial, Helvetica, sans-serif; text-align: center;

}

.wrapper {

display: flex;

justify-content: center; flex-direction: column; align-items: center;

}

h1 {

}

hr {

}

margin: 10px 0px;

width: 450px;

.board {

width: 450px; height: 450px; display: flex; flex-wrap: wrap;

}

.tile {

width: 48px; height: 48px;

border: 1px solid lightgray; font-size: 20px;

font-weight: bold; display: flex;

justify-content: center; align-items: center; transition: .5s;

}

.tile:hover {

cursor: pointer;

}

.digits {

width: 450px; height: 50px; display: flex; flex-wrap: wrap;

}

.number {

width: 44px; height: 44px;

border: 1px solid black; margin: 2px;

font-size: 20px; font-weight: bold; display: flex;

justify-content: center; align-items: center;

}

.number:hover { cursor: pointer;

background-color: gray;

}

.selected {

background-color: gray;

}

.tile-start {

background-color: whitesmoke;

}

.tile-error {

background-color: #fa856e;

}

.horizontal-line {

border-bottom: 1px solid black;

}

.vertical-line {

border-right: 1px solid black;

}

.result {

width: 450px; margin: 10px; display: flex;

justify-content: space-between; align-items: center;

}

.new-game {

width: 120px; height: 35px;

background-color: #57f542; border: 1px solid gray;

}

.new-game:hover {

background-color: #36d651; cursor: pointer;

}

.check {

width: 120px; height: 35px;

}

.check:hover { cursor: pointer;

}

.errors {

font-size: 21px;

}

.end-game {

font-size: 23px;

}

Результат выполнения

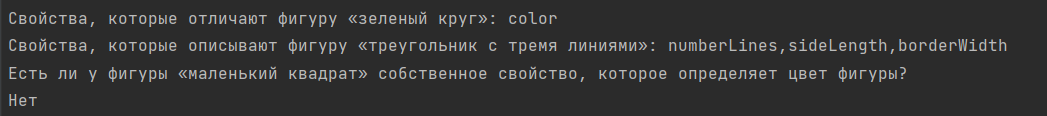


Рисунок 1 – Вывод результата в консоли (задача 1)

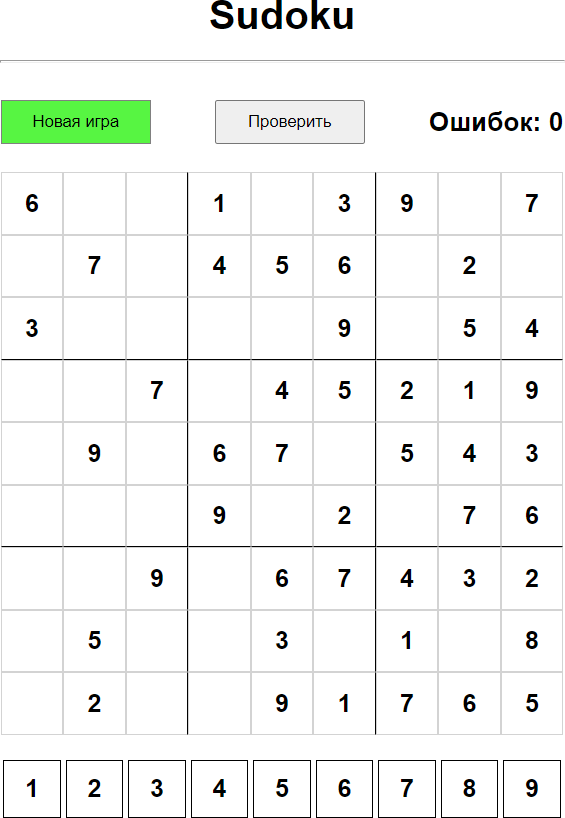
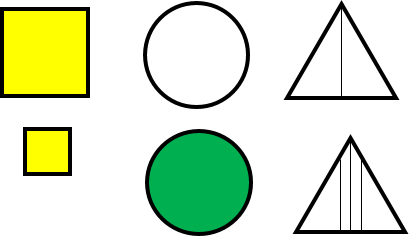


Рисунок 2 – Игровой процесс (задача 2)

Вывод: решил задачи:

1. Представил фигуры на картинке в виде объектов



Вывод: свойства, которые отличают фигуру «зеленый круг»; свойства, которые описывают фигуру «треугольник с тремя линиями»; есть ли у фигуры «маленький квадрат» собственное свойство, которое определяет цвет фигуры.