МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Лабораторная работа №4

«Рекурсия и головоломки»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к отчету по лабораторной работе

по дисциплине

Технологии программирования

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_ Капранов С.Н.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_ Казанцев А.В.

(подпись) (фамилия, и.,о.)

18-ИСТ-3

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

# Задание

**Задача 10.** **Разработать программу, решающую головоломку "Филломино"**

Филломино ("Fillomino") представляет собой прямоугольную сетку произвольного

размера; в некоторых клетках находятся числа. Необходимо разбить игровое поле на

блоки; блок должен содержать столько клеток, сколько обозначено числом в клетках

блока. Блоки, имеющие одинаковый размер, не должны соприкасаться по горизонтали или

по вертикали. Клетки, которые изначально не содержали чисел, также могут быть

объединены в блоки, необходимые для решения головоломки.

**Для разработки данной программы я использовал фреймворк Vue.js**

**для языка Javascript**

# Код

<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*="icon" *href*="<%= BASE\_URL %>favicon.ico">  
 <link *rel*="stylesheet" *href*="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" *integrity*="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" *crossorigin*="anonymous">  
  
 <title><%= htmlWebpackPlugin.options.title %></title>  
 </head>  
 <body>  
 <noscript>  
 <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't work properly without JavaScript enabled. Please enable it to continue.</strong>  
 </noscript>  
 <div *id*="app"></div>  
 *<!-- built files will be auto injected -->* </body>  
</html>

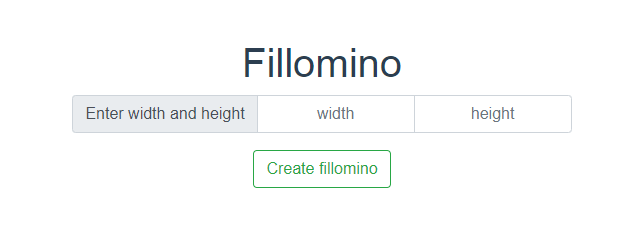
<App.vue> - Основной компонент

<template>  
 <div *id*="app">  
 <h1>Fillomino</h1>  
 <div *class*="input-group w-26 mx-auto mb-3" *v-if*="!render">  
 <div *class*="input-group-prepend">  
 <span *class*="input-group-text">Enter width and height</span>  
 </div>  
 <input *type*="text" *v-model*="width" *placeholder*="width" *aria-label*="width" *class*="form-control text-center" >  
 <input *type*="text" *v-model*="height" *placeholder*="height" *aria-label*="height" *class*="form-control text-center">  
 </div>  
 <h3 *v-bind:class*="{green: render, red: !render}">{{ message }}</h3>  
 <h3 *v-if*="render">{{size}}</h3>  
 <Fillomino  
 *v-bind:grid*="grid"  
 *v-if*="render"  
 *:height*="height"  
 *:width*="width"  
 *@change-message*="changeMessage"  
 *@remove-grid*="removeGrid"  
 />  
 <button *class*="btn btn-outline-success" *@click*="renderGrid" *v-if*="!render">Create fillomino</button>  
 </div>  
</template>  
  
<script>  
  
  
*import* Fillomino *from* "./components/fillomino";  
*export default* {  
 name: 'App',  
 components: {  
 Fillomino  
 },  
 data() {  
 *return* {  
 height: *null*,  
 width: *null*,  
 grid: *null*,  
 render: *false*,  
 message: ''  
 }  
 },  
 methods: {  
 renderGrid() {  
 *if* (Number(*this*.height) && Number(*this*.width)) {  
 *this*.width = parseInt(*this*.width);  
 *this*.height = parseInt(*this*.height);  
 *this*.grid = *new* Array(*this*.height+2);  
 *for* (*let* i = 0; i < *this*.grid.length; i++) {  
 *this*.grid[i] = *new* Array(*this*.width+2);  
 }  
 *this*.render = *true*;  
 *this*.message = `Fillomino grid has been created`;  
 *this*.size = `Fillomino size: ${*this*.width}x${*this*.height}`;  
  
 } *else* {  
 *this*.message = `Fillomino grid was not created, enter integers`;  
 }  
 },  
 removeGrid() {  
 *this*.width = '';  
 *this*.height = '';  
 *this*.render = *false*;  
 *this*.message = `Fillomino grid has been removed`;  
 },  
 changeMessage(mes) {  
 *this*.message = mes;  
 }  
 }  
}  
</script>  
  
<style *lang*="scss">  
#app {  
 font-family: Avenir, Helvetica, Arial, sans-serif;  
 text-align: center;  
 color: #2c3e50;  
 margin-top: 60px;  
 font-size: 1.5rem;  
}  
 .*green* {  
 color: #4c800d;  
 }  
 .*red* {  
 color: #ff2023;  
 }  
 .*w-26* {  
 width: 26%;  
 }  
</style>

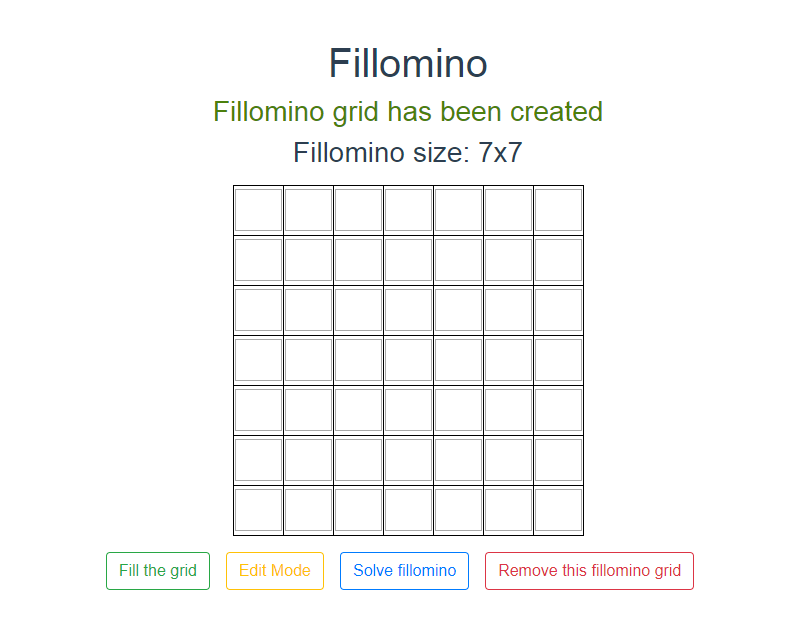
<Fillomino.vue> - дочерний компонент отвечающий за сетку филломино, содержащий всю логику необходимую для решения задачи

<template>  
 <div>  
 <table *class*="my-3">  
 <tr *v-for*="(row, rowIndex) *of* grid" *:key*="rowIndex-1">  
 <td *v-for*="(ceil, index) *of* row" *:key*="index" *v-bind:id*="index+(rowIndex-1)\*row.length" *:style*="{backgroundColor: ceil.bgColor}" *v-bind:class*="{'d-none': grid[rowIndex][index].value == -1}">  
 <input *type*="text" *v-if*="editMode" *v-model*="grid[rowIndex][index].value">  
 <span *v-if*="!editMode">{{ceil.value}}</span>  
 </td>  
 </tr>  
 </table>  
  
 <button *class*="btn btn-outline-success" *@click*="gridFilling">Fill the grid</button>  
 <button *class*="btn btn-outline-warning" *@click*="editGrid">Edit Mode</button>  
 <button *class*="btn btn-outline-primary" *@click*="solve">Solve fillomino</button>  
 <button *class*="btn btn-outline-danger" *@click*="removeGrid">Remove this fillomino grid</button>  
 </div>  
</template>  
  
<script>  
 *export default* {  
 name: "fillomino",  
 data() {  
 *return* {  
 editMode: *true*,  
 itaration: 0,  
 bottomDistance: 0,  
 topDistance: 0,  
 rightDistance: 0,  
 leftDistance: 0,  
 colors: [  
 'rgba(255,59,0,0.58)',  
 'rgba(255,235,100,0.58)',  
 'rgba(255,144,54,0.58)',  
 'rgba(255,58,58,0.58)',  
 'rgba(79,255,230,0.58)',  
 'rgba(46,41,255,0.58)',  
 'rgba(255,105,252,0.58)',  
 'rgba(100,255,64,0.58)',  
 'rgba(128,0,255,0.58)'  
 ]  
 }  
 },  
 props: ['grid', 'width', 'height'],  
 methods: {  
 gridFilling() {  
 *let* flag = *true*;  
 *for* (*let* i = 0; i < *this*.grid.length; i++) {  
 *for* (*let* j = 0; j < *this*.grid[i].length; j++) {  
 *if*(i === 0 || i === *this*.grid.length-1 || j === 0 || j === *this*.grid[i].length-1) {  
 *continue*;  
 }  
 *if*(Number(*this*.grid[i][j].value) && Number(*this*.grid[i][j].value) === parseInt(*this*.grid[i][j].value) && Number(*this*.grid[i][j].value) > 0 && Number(*this*.grid[i][j].value) < 10 || *this*.grid[i][j].value === '') {  
 *continue*;  
 } *else* {  
 flag = *false*;  
 }  
 }  
 }  
 *if* (flag) {  
 *this*.editMode = *false*;  
 } *else* {  
 alert('You can enter only integer numbers from 1 to 9');  
 }  
 },  
 removeGrid() {  
 *this*.$emit('remove-grid');  
 },  
 editGrid() {  
 *this*.editMode = *true*;  
 },  
 solve() {  
 *let* sum = 0;  
 *for* (*let* i = 1; i < *this*.grid.length-1; i++) {  
 *for* (*let* j = 1; j < *this*.grid[i].length-1; j++) {  
 *if*(Number(*this*.grid[i][j].value))  
 sum += Number(*this*.grid[i][j].value);  
 }  
 }  
 *if*(sum === (Number(*this*.width) \* Number(*this*.height))) {  
 *for* (*let* i = 1; i < *this*.grid.length-1; i++) {  
 *for* (*let* j = 1; j < *this*.grid[i].length-1; j++) {  
 *for* (*let* k = 1; k < 10; k++) {  
 *if*(Number(*this*.grid[i][j].value) === k) {  
 *this*.itaration = 0;  
 *this*.dfs(i, j, k);  
 *break*;  
 }  
 }  
 }  
 }  
 *this*.editMode = *true*; *//  
 this*.editMode = *false*; *//render  
 this*.changeMessage('Fillomino was successfully resolved');  
 } *else* {  
 *let* conf = confirm('Error! The sum of all entered numbers must correspond to the product of the width and height of the grid. Want to see what happens?');  
 *if*(conf) {  
 *for* (*let* i = 1; i < *this*.grid.length-1; i++) {  
 *for* (*let* j = 1; j < *this*.grid[i].length-1; j++) {  
 *for* (*let* k = 1; k < 10; k++) {  
 *if*(Number(*this*.grid[i][j].value) === k) {  
 *this*.itaration = 0;  
 *this*.dfs(i, j, k);  
 *break*;  
 }  
 }  
 }  
 }  
 *this*.editMode = *true*; *//  
 this*.editMode = *false*; *//render* }  
 }  
  
 },  
 dfs(x, y, n) {  
 *if*(Number(*this*.grid[x][y].value) === n && *this*.itaration < n && Number(*this*.grid[x][y].value) != -1 && *this*.grid[x][y].visited === *false*){  
 *this*.itaration += 1;  
 *this*.grid[x][y].visited = *true*;  
 *this*.grid[x][y].bgColor = *this*.colors[n-1];  
 *this*.dfs(x+1, y, n);  
 *this*.dfs(x-1, y, n);  
 *this*.dfs(x, y+1, n);  
 *this*.dfs(x, y-1, n);  
 }  
 *if* (*this*.grid[x][y].value === '' && *this*.itaration < n) {  
 *this*.grid[x][y].value = n;  
 *this*.grid[x][y].visited = *true*;  
 *this*.grid[x][y].bgColor = *this*.colors[n-1];  
 *this*.itaration += 1;  
 *this*.dfs(x+1, y, n);  
 *this*.dfs(x-1, y, n);  
 *this*.dfs(x, y+1, n);  
 *this*.dfs(x, y-1, n);  
  
 }  
 },  
 changeMessage(mes) {  
 *this*.$emit('change-message', mes);  
 }  
 },  
 created() {  
 *for* (*let* i = 0; i < *this*.grid.length; i++)  
 *for* (*let* j = 0; j < *this*.grid[i].length; j++) {  
 *if*(i === 0 || i === *this*.grid.length-1 || j === 0 || j === *this*.grid[i].length-1) {  
 *this*.grid[i][j] = {value: -1, visited: *true*, bgColor: '#fff'};  
 } *else  
 this*.grid[i][j] = {value: '', visited: *false*, bgColor: '#fff'};  
 }  
  
 }  
 }  
</script>  
  
<style *lang*="scss" *scoped*>  
 table {  
 margin: 0 auto;  
 tr {  
 td {  
 width: 50px;  
 height: 50px;  
 border: 1px solid black;  
 input {  
 width: 100%;  
 text-align: center;  
 }  
 }  
 }  
 }  
 .*btn-outline-warning* {  
 &:*hover* {  
 color: white;  
 }  
 }  
 .*btn* {  
 margin-right: 1rem;  
 }  
</style>

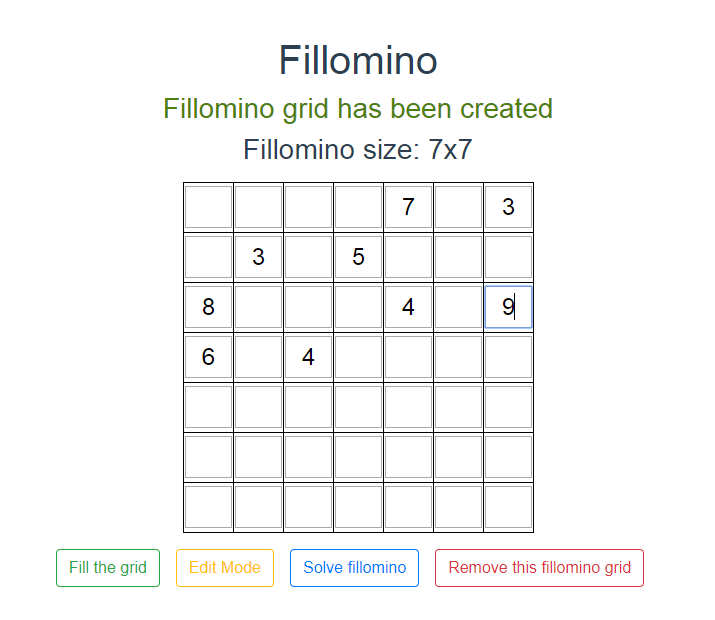
**Результат работы программы**

****

****

****

Заполним сетку



И нажмем кнопочку Solve fillomino



Удаляем сетку и можем попробовать заново

