Министерство образования и науки Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №1

«Операции над множествами»

Выполнил студент

группы ИВТАСбд-11

Абросимов К. С.

Ульяновск, 2022

**Цель работы**

Необходимо написать программу, которая будет проводить операции над множествами, а именно нахождение: их объединения, пересечения, дополнение первого до второго и наоборот и симметрической разницы. Пользователь должен иметь возможность вводить данные. Также должна присутствовать проверка на корректность введённых данных. Первый и второй символы элемента – буква, третий – чётная цифра, четвертый – буква.

**Описание метода выполнения**

Множество – совокупность определенных объектов. Операции над ними – манипуляция над этими объектами. Операции над множествами: объединение, пересечение, симметрическая разница, дополнение первого до второго и наоборот. Объединение – множество, содержащее в себе все элементы других множеств без повторения. Пересечение – множество, содержащее в себе только элементы первого и второго множеств. Дополнением первого до второго называется множество, содержащее все элементы первого множества, которые не принадлежат второму множеству и наоборот. Симметрическая разность – множества, содержащее в себе элементы первого и второго множеств, которые не совпадают между собою.

Код проводимых операций будет написан на PHP. PHP - это язык программирования, специально разработанный для написания web-приложений (сценариев), исполняющихся на Web-сервере. PHP взаимодействует с html.

Сначала сделаем простое меню для выбора лабораторных работ на будущее. Он ничего особенного не имеет, кроме ссылок к другим файлам с их названиями.

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Лабораторные работы</title> - основной заголовок  
 <script type="text/javascript" src="/scripts/scripts.js"></script>  
</head>  
<body>  
 <h1> Мои лабораторные работы </h1>   
 <a href ="mldmLab1.php">Первая лабораторная работа по МЛиДМ</a><br>  
 <a href ="mldmLab2.php">Вторая лабораторная работа по МЛиДМ</a> - ссылки к другим лабораторным работам  
</body>  
</html>

Для того, чтобы пользователь мог вводить данные необходимо написать небольшую html-структуру, состоящую из двух полей для ввода (input), полей для вывода и кнопки, по которой произойдут операции над множествами. Также для полного вывода операций нужно создать значение для вывода.

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Лабораторная работа 1</title>  
 <script type="text/javascript" src="/scripts/scripts.js"></script>  
</head>  
<body>  
 <h1> Лабораторная работа №1 </h1>  
<form>  
 <table>  
 <tr>  
 <td> Первый массив </td>  
 <td> <input type="text" id="mass1" value="" size="50"/></td> - Здесь проводится ввод первого массива  
 </tr>  
 <tr>  
 <td> Второй массив </td>  
 <td> <input type="text" id="mass2" value="" size="50"/></td> - второго массива  
 </tr>  
 <tr>  
 <td colspan="2"> <input type="button" value="Сделать расчёт" onclick="rasschet();"/></td> - кнопка выполняет функцию “rasschet”, которая и приводит к файлу “scripts.js”  
 </tr>  
 </table>  
</form>  
<div id ="outResult"></div> - полный вывод всех операций  
</body>  
</html>

Сначала функция “rasschet” создаёт переменные для считывания информации, введённое пользователем. Но они не являются массивом, а значением. Далее необходимо выполнить проверку введенных элементов множеств. Для этого в функцию передаётся массив, где происходит деление каждого слова на массив символов и проверка каждого символа согласно заданию:

function rasschet()  
{  
 var a = ***document***.getElementById('mass1');  
 var b = ***document***.getElementById('mass2');  
 if(***mass\_1*** = validate(a.value) == false)  
 {  
 alert(***error\_text***);  
 }  
 if (***mass\_2*** = validate(b.value) == false)  
 {  
 alert(***error\_text***);  
 }

После этого функция создаёт переменные для создания массивов и сразу же выполняет функцию для преобразования значений в полноценный массив.

var mass1 = [];  
var mass2 = [];  
mass1 = needmass( a.value );  
mass2 = needmass( b.value );

Наконец, созданию последней переменной, которая и выведет полный ответ. В процессе будут проводиться много функций – операции над множествами, если ошибки не произошло.

let full\_result = "";  
 if (***mass\_1*** == false && ***mass\_2*** == false)  
 {  
 full\_result = "Объединения массивов: " + unification ( mass1, mass2 ) + "\n";  
 full\_result += "Пересечение массивов: " + intersection ( mass1, mass2 ) + "\n";  
 full\_result += "Дополнение A/B: " + additionA ( mass1, mass2 ) + "\n";  
 full\_result += "Дополнение B/A: " + additionB ( mass1, mass2 ) + "\n";  
 full\_result += "Симметрическая разность: " + symmdiff ( mass1, mass2 ) + "\n";

***document***.getElementById('outResult').innerText = "Результат выполнения операций:\n" + full\_result; - переменная с предыдущего файла приобретает полный ответ и выводит его  
 }

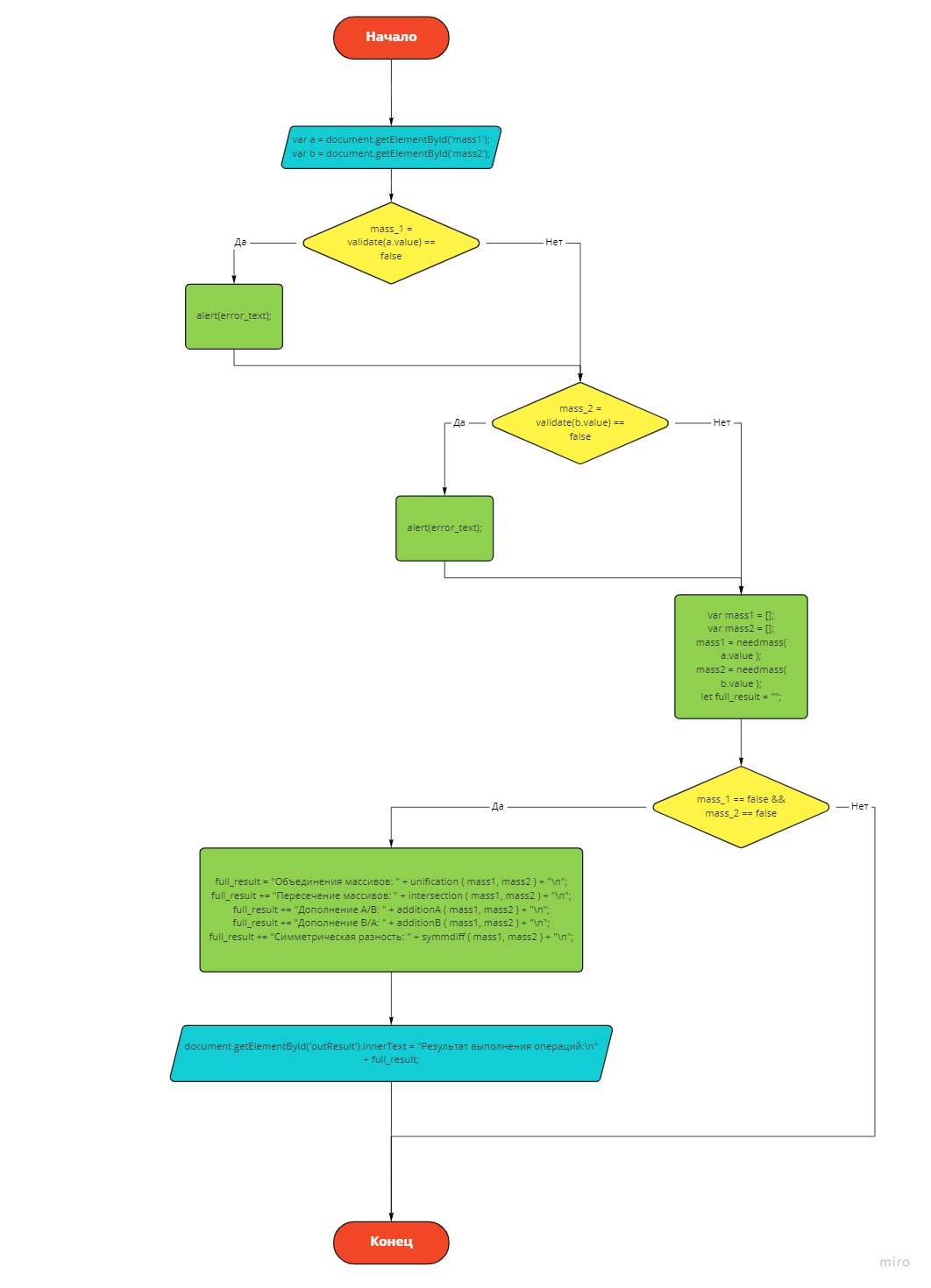
}  


Рис. 1 - функция rasschet

Функция validate и выполняет большую проверку текста. Сначала идёт проверка на наличие в массиве хоть какого-либо текста данным условием:

function validate(str)  
{  
 let mass = false;  
 if(str.length>0)

Следующий шаг – создание из значения массива. Причём деление происходит по знаку “ “ (пробел). Если пользователь вводит больше пробелов между элементами, то данный цикл их удаляет:  
 {  
 mass = str.split(" ");  
 for (let i = 0; i < mass.length; i++)  
 {  
 if (mass[i] == "")  
 {  
 mass.splice(i, 1);  
 }  
 }

Следующие большой цикл уже включает саму проверку. Ошибка (вывод окошка с текстом) будет появляться при:

* Пустом массиве
* Недостаточном количестве символов в элементе
* Неправильном формате элемента  
   for (let i = 0; i < mass.length; i++)  
   {  
   if (mass[i].length != 4)  
   {  
   ***error\_text*** = 'Ошибка при вводе массива формата bbib: ' + str + ' в элементе ' + mass[i] + '. Недостаточно символов или стоит лишний пробел';  
   mass = false;  
   break;  
   }  
   else if (mass[i][0] < 'a' || mass[i][0] > 'z') {  
   ***error\_text*** = 'Ошибка при вводе массива формата bbib: ' + str + ' в элементе ' + mass[i] + '. Нужна буква';  
   mass = false;  
   break;  
   }  
   else if (mass[i][1] < 'a' || mass[i][1] > 'z') {  
   ***error\_text*** = 'Ошибка при вводе массива формата bbib: ' + str + ' в элементе ' + mass[i] + '. Нужна буква';  
   mass = false;  
   break;  
   }  
   else if (mass[i][2] % 2 != 0) {  
   ***error\_text*** = 'Ошибка при вводе массива формата bbib: ' + str + ' в элементе ' + mass[i] + '. Нужна чётная цифра';  
   mass = false;  
   break;  
   }  
   else if (mass[i][3] < 'a' || mass[i][3] > 'z')  
   {  
   ***error\_text*** = 'Ошибка при вводе массива формата bbib: ' + str + ' в элементе ' + mass[i] + '. Нужна буква';  
   mass = false;  
   break;  
   }  
   }  
   }  
   else  
   ***error\_text*** = "Массив не может быть пустым";

В конце возвращаем значение mass (значение, а не массив!).  
 return mass;  
}

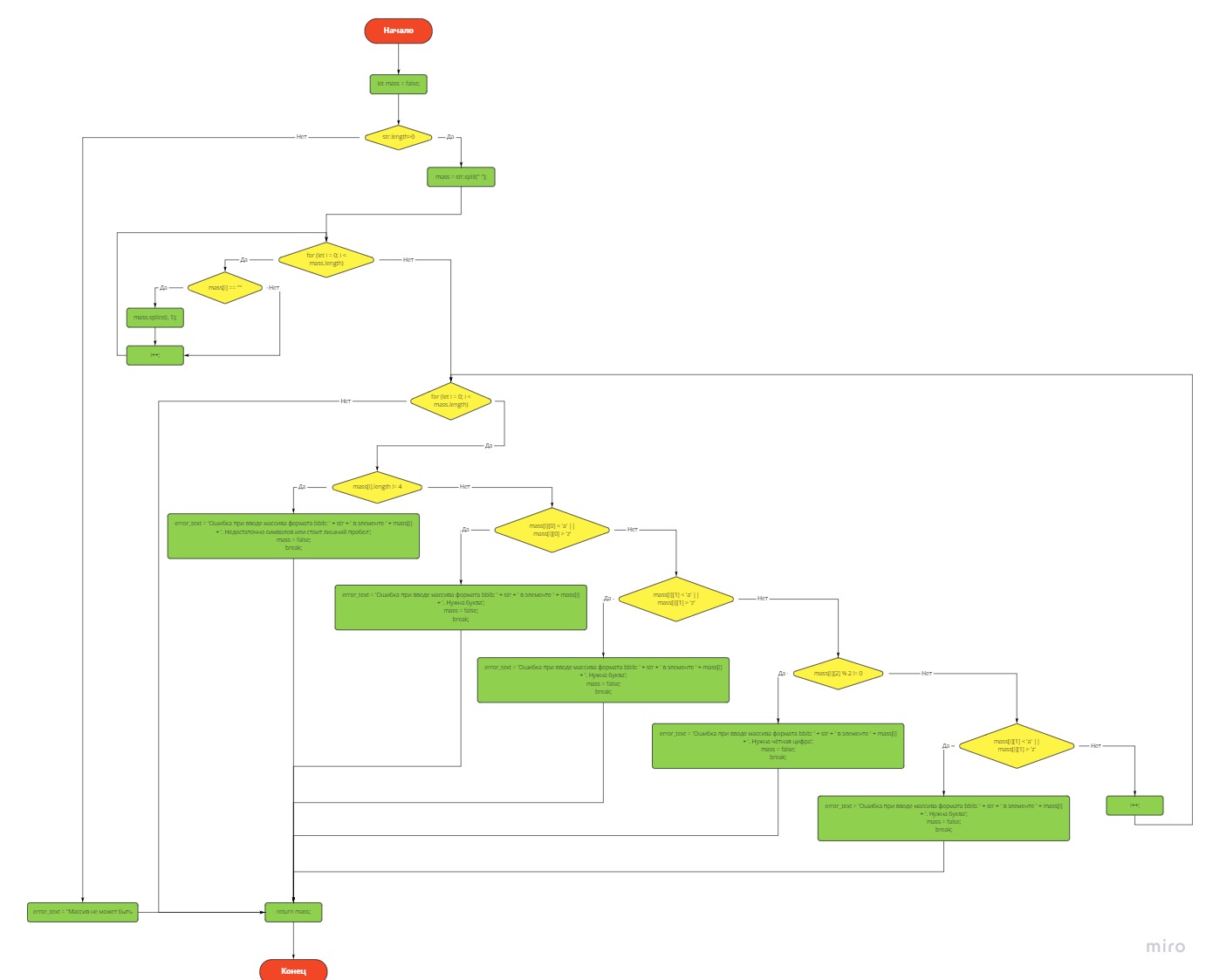


Рис. 2 - функция validate

Именно следующая функция needmass создаёт массивы для дальнейшего пользования, после чего возвращает это пустым массивам, созданным заранее в rasschet.

function needmass( n )  
{  
 var massiv = [];  
 massiv = [].join.call(n,'');  
 massiv = n.split(" ");  
 for (let i = 0; i < massiv.length; i++)  
 {  
 if(countElement(massiv,massiv[i])>1)  
 {  
 massiv.splice(i,1);  
 }  
 }  
 return massiv;  
}

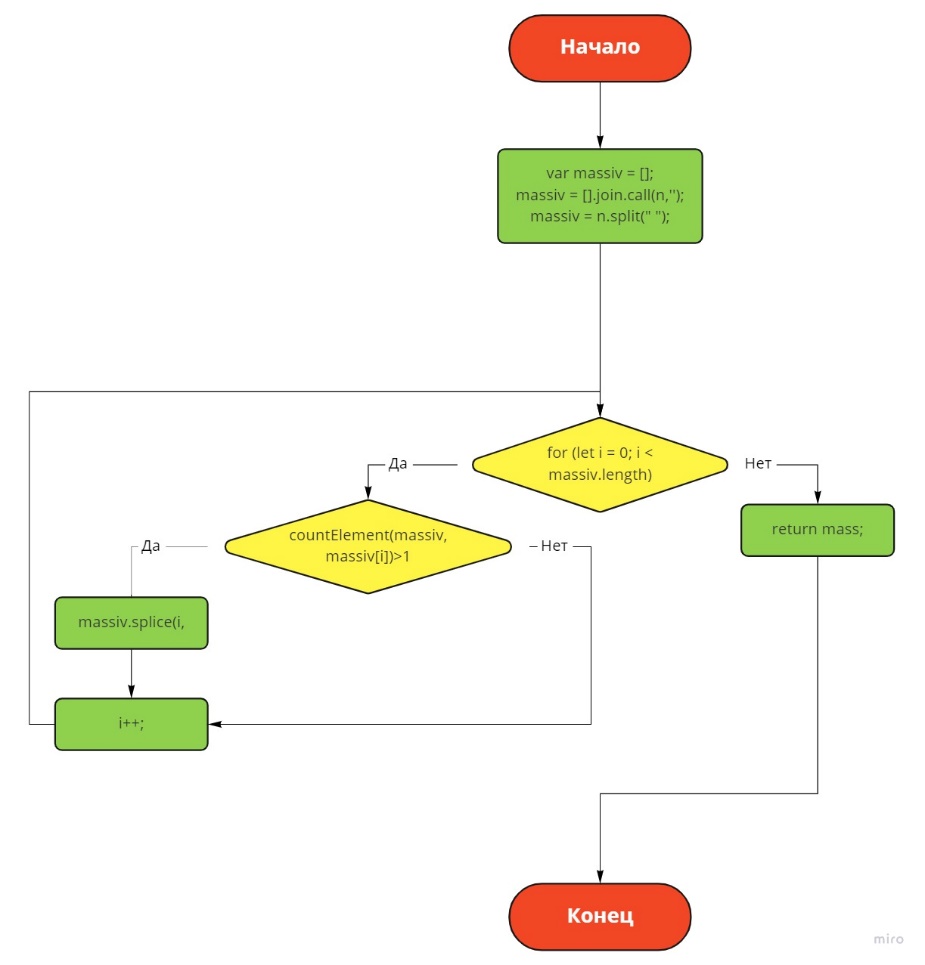


Рис. 3 - функция needmass

Внутри needmass есть функция countElement, которая удаляет дубликаты элементов в одном массиве.

function countElement(mass, element)  
{  
 let count = 0;  
 for (let i = 0; i < mass.length; i++)  
 {  
 if (mass[i] == element)  
 {  
 count++;  
 }  
 }  
 return count;  
}

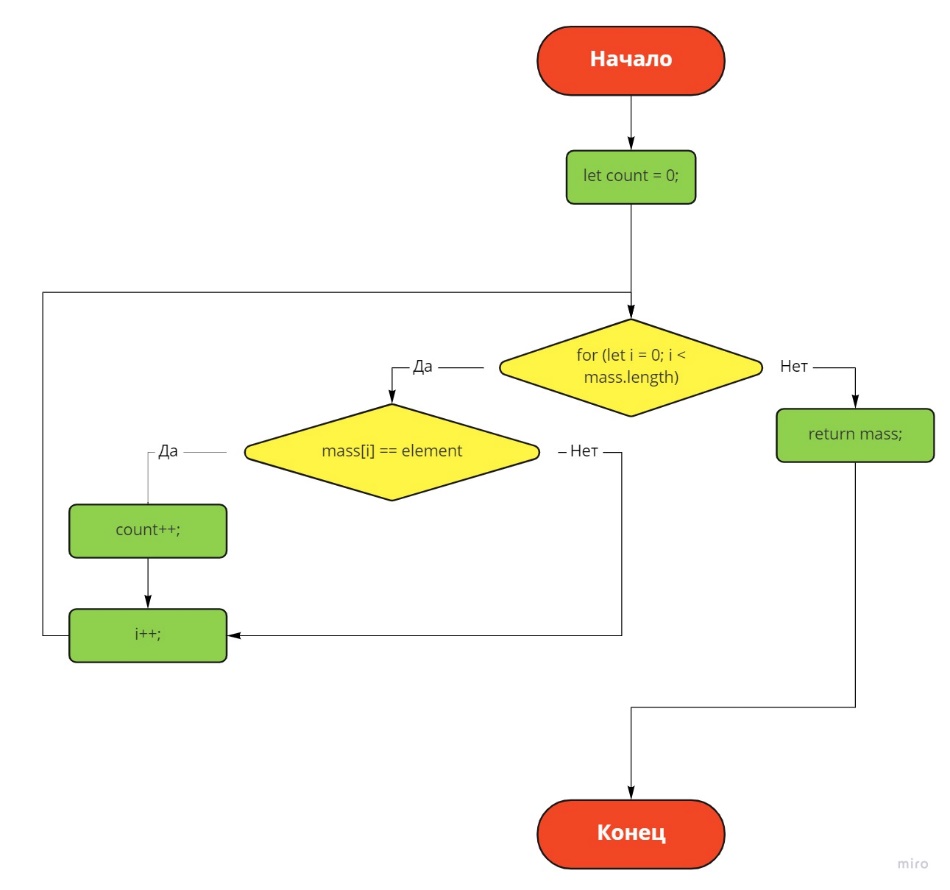


Рис. 4 - функция countElement

Далее идут сами функции операций над множествами (массивами). Сперва идёт объединение: в пустой массив сперва записывает все элементы первого множества одним циклом, а затем, проверяя наличие элементов второго множества у первого, записывает элементы второго множества другим циклом:

function unification ( n1, n2 ) {  
 var massivUnif = [];  
 for (let i = 0; i < n1.length; i++)  
 {  
 massivUnif += n1[i] + " ";  
 }  
 for (let i = 0; i < n2.length; i++) {  
 if (n1.indexOf(n2[i]) == -1)  
 {  
 massivUnif += n2[i] + " ";  
 }  
 }  
 return massivUnif;  
}

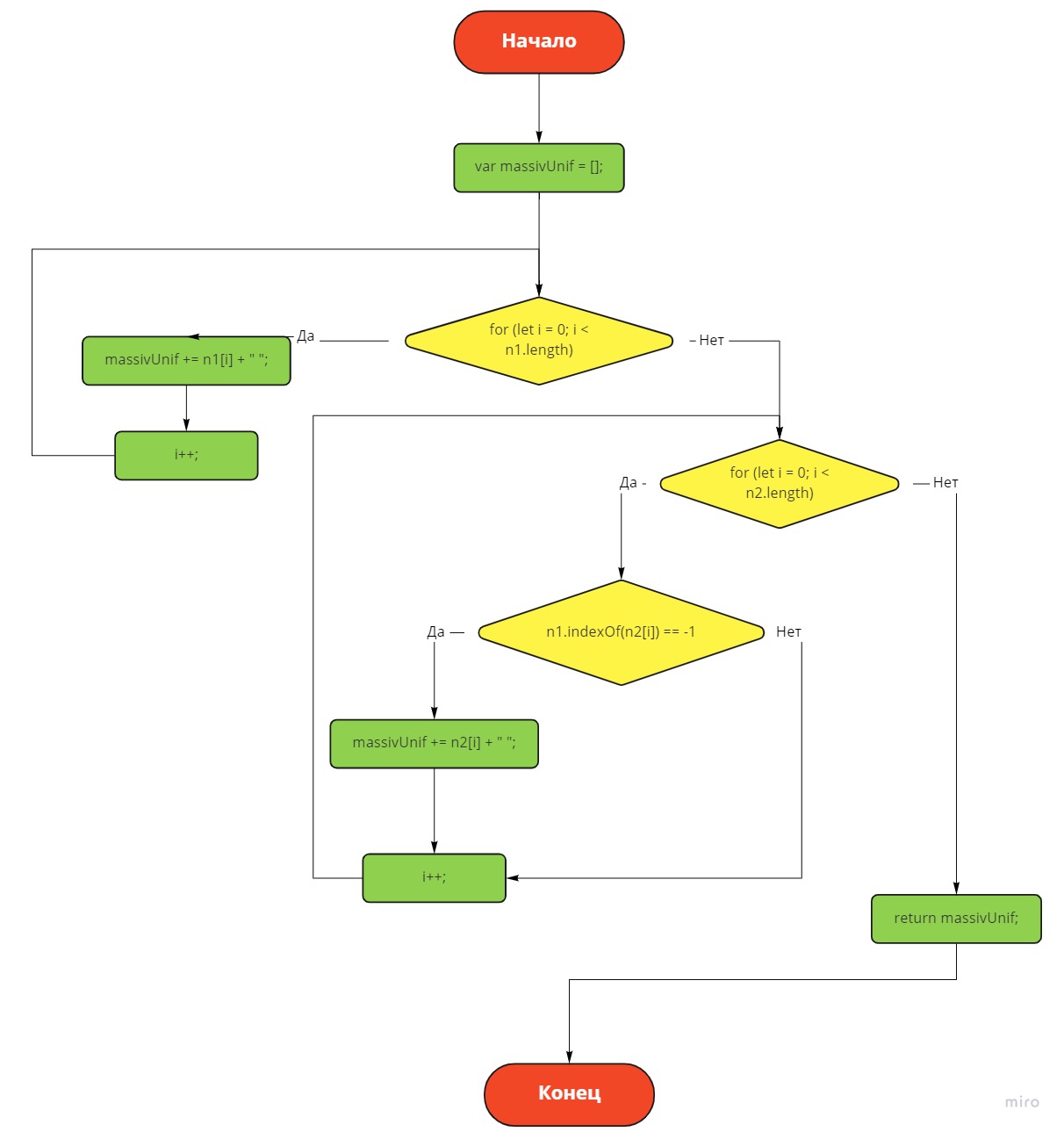


Рис. 5 - функция unification

Далее пересечение – записывает в пустой массив только те элементы, которые есть у обоих множеств.

function intersection ( n1, n2 )  
{  
 var massivInter = [];  
 for (let i = 0; i < n2.length; i++) {  
 if (n1.indexOf(n2[i]) != -1)  
 {  
 massivInter += n2[i] + " ";  
 }  
 }  
 return massivInter;  
}

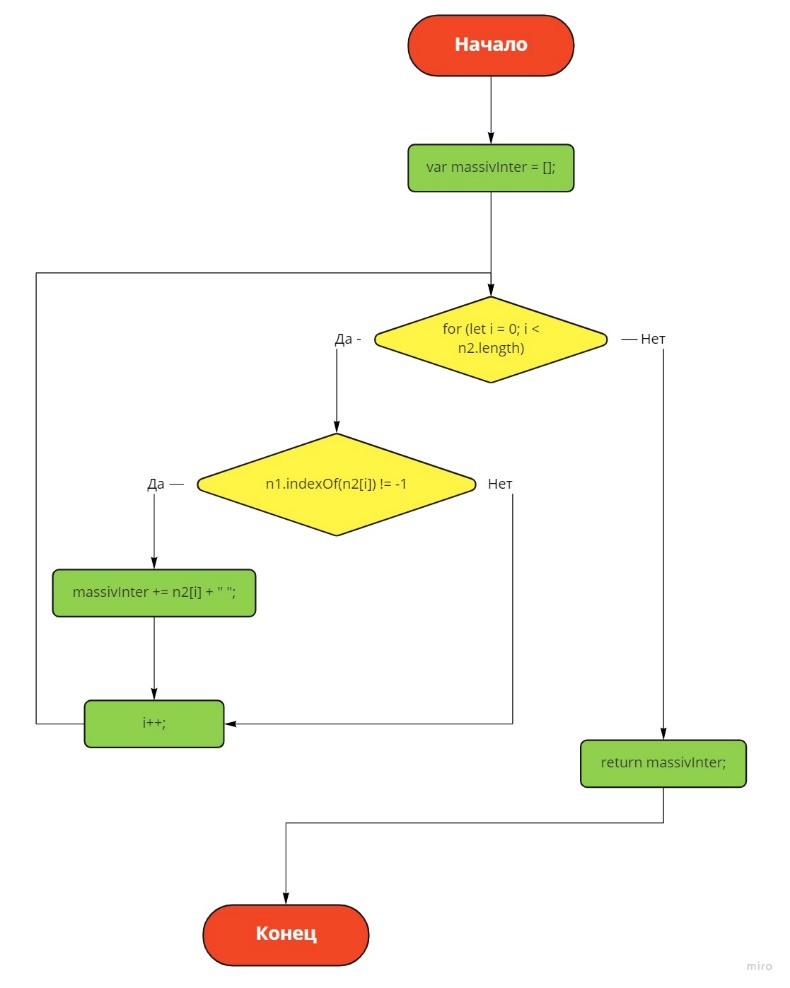


Рис. 6 - функция intersection

Дополнение A/B – записываем в пустой массив только те элементы, которые находятся в первом множестве и не пересекаются со вторым:

function additionA ( n1, n2 )  
{  
 var massivAdda = [];  
 for (let i = 0; i < n1.length; i++) {  
 if (n2.indexOf(n1[i]) == -1)  
 {  
 massivAdda += n1[i] + " ";  
 }  
 }  
 return massivAdda;  
}

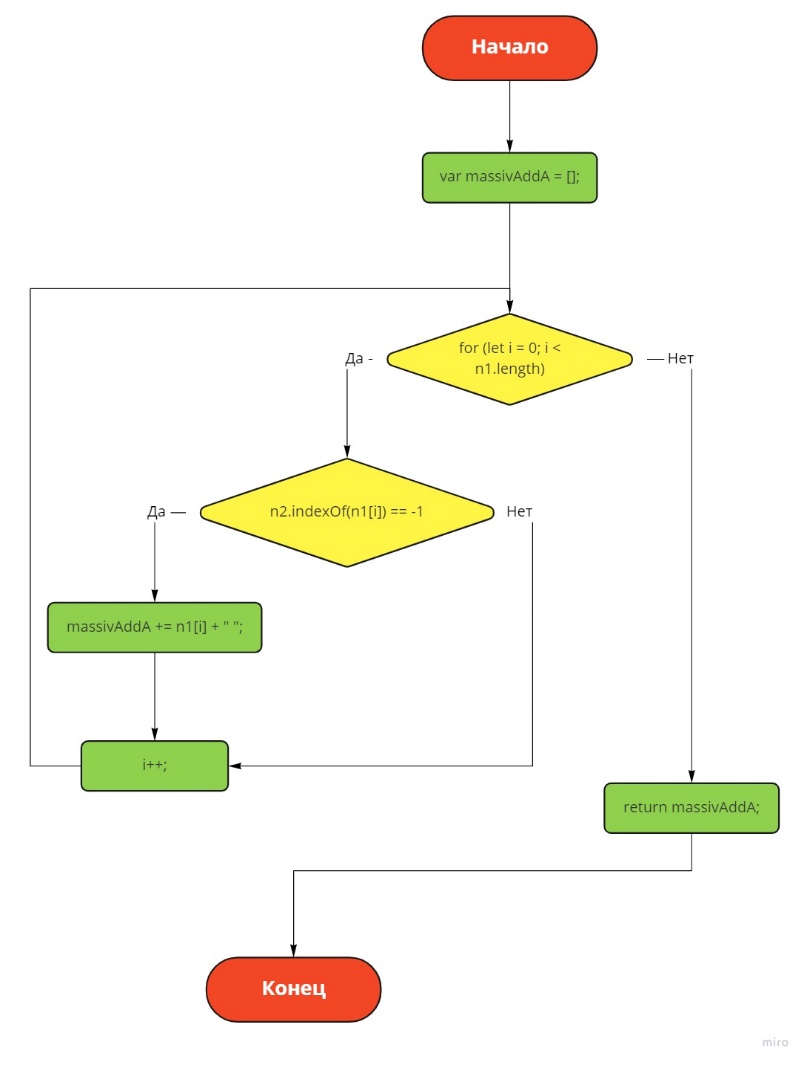


Рис. 7 - функция additionA

Дополнение B/A – теперь наоборот: в пустой массив записываем только те элементы, который находятся во втором множестве и не пересекаются с первым.

function additionB ( n1, n2 )  
{  
 var massivAddb = [];  
 for (let i = 0; i < n2.length; i++) {  
 if (n1.indexOf(n2[i]) == -1)  
 {  
 massivAddb += n2[i] + " ";  
 }  
 }  
 return massivAddb;  
}

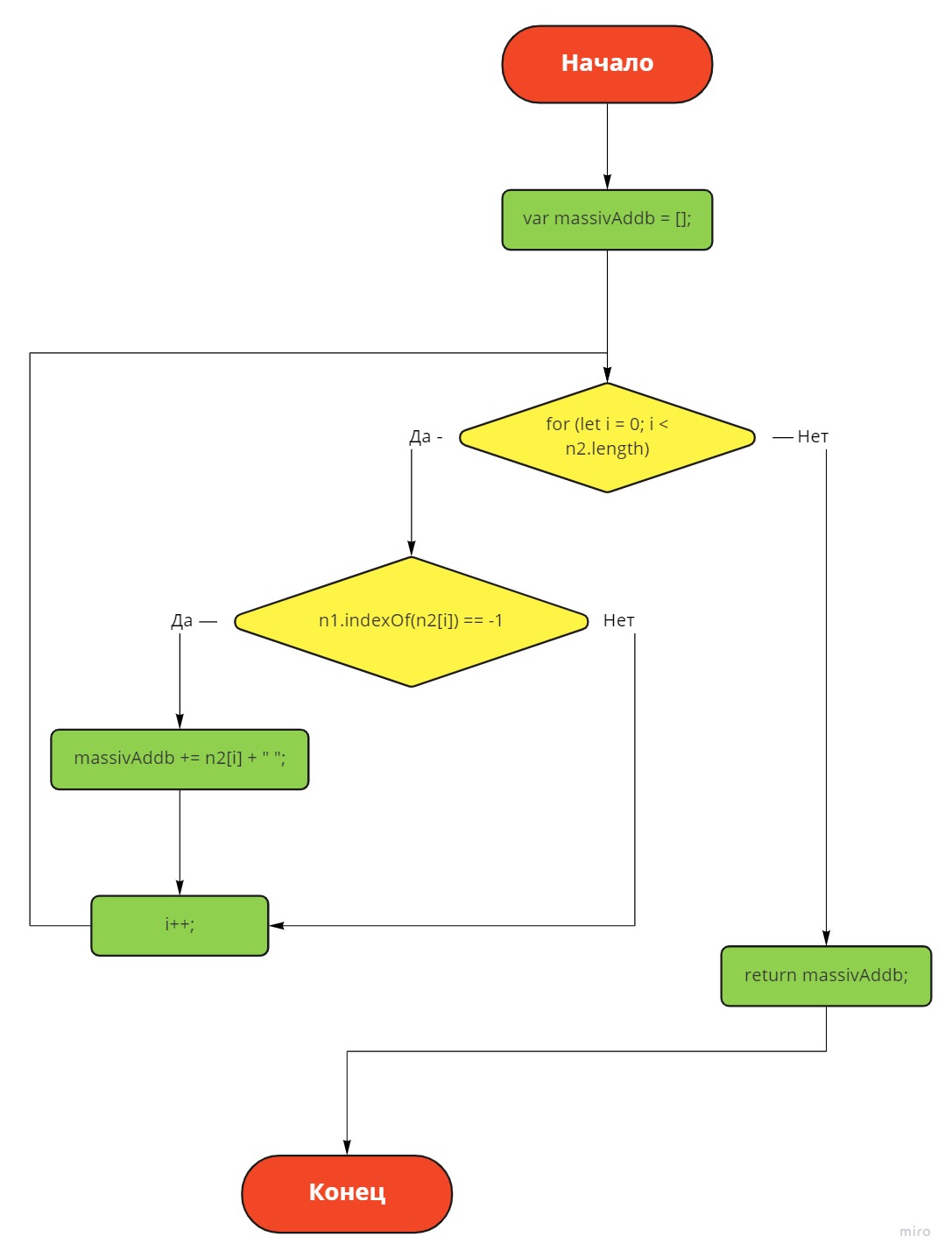


Рис. 8 - функция additionB

Последняя функция – симметрическая разность. Записывает в массив все элементы, кроме тех, которые пересекаются в множествах:

function symmdiff( n1, n2 )  
{  
 var massivsymm = [];  
 for (let i = 0; i < n1.length; i++) {  
 if (n2.indexOf(n1[i]) == -1)  
 {  
 massivsymm += n1[i] + " ";  
 }  
 }  
 for (let i = 0; i < n2.length; i++) {  
 if (n1.indexOf(n2[i]) == -1)  
 {  
 massivsymm += n2[i] + " ";  
 }  
 }  
 return massivsymm;  
}

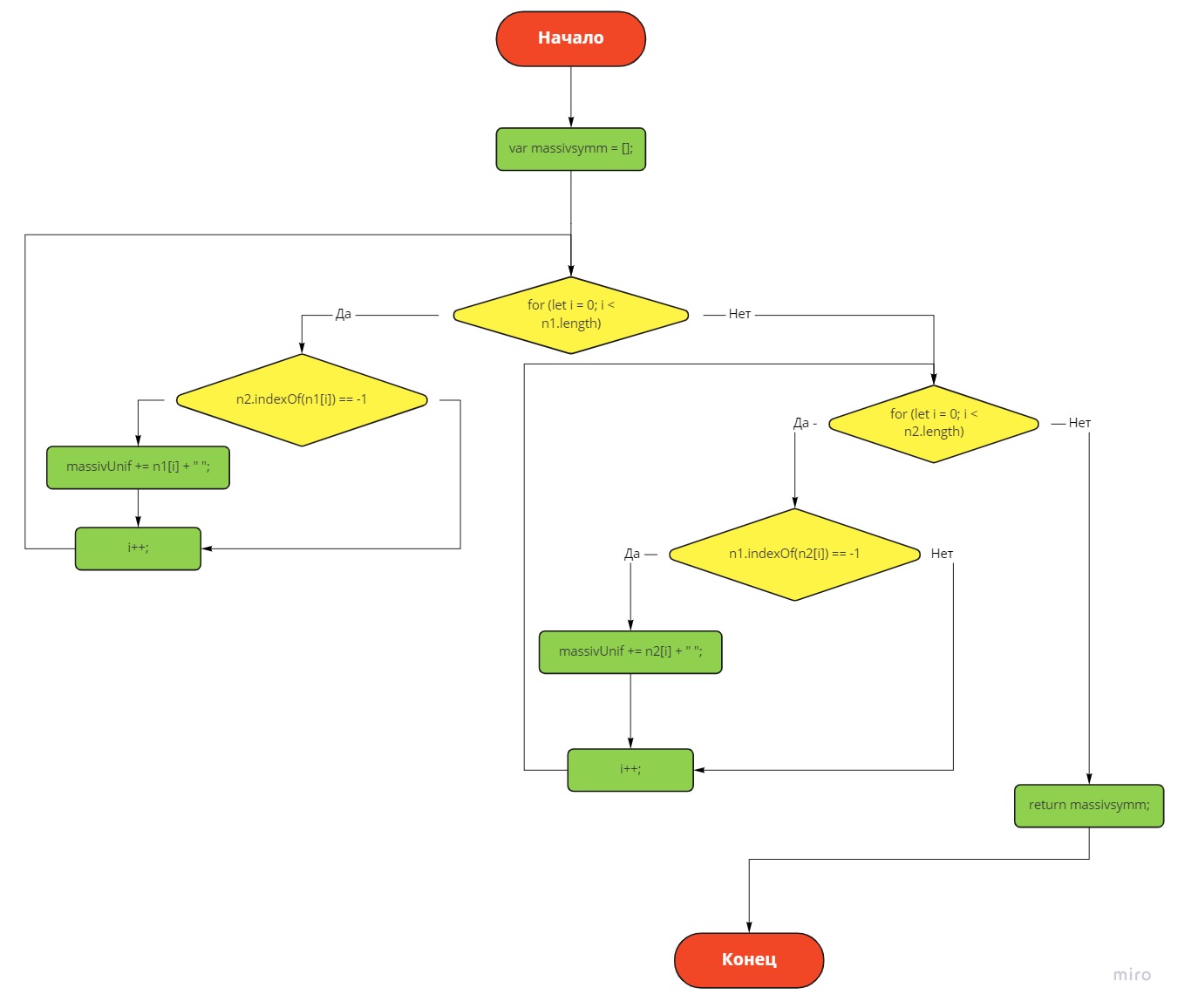


Рис. 9 - функция symmdiff

**Выводы**

Сайт был протестирован, он работает нормально и стабильно. Все требуемые условия задачи выполнены успешно. В этой работе я узнал, как создавать сайты и как работать с ними, больше узнал о языке html (новые теги, функции) и о программе PHPStorm. Также я научился делать разные операции над множествами.

# **Список литературы**

(б.д.).

*JSExpert*. (б.д.). Получено из indexOf(): https://www.jsexpert.net/handbook/javascript/array/indexof/#:~:text=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%20indexOf()%20%D0%B8%D1%89%D0%B5%D1%82%20%D0%B2,1%2C%20%D0%B5%D1%81%D0%BB%D0%B8%20%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%20%D0%BD%D0%B5%20%D0%BD%D0%B0%D

*MDN web docs*. (б.д.). Получено из Array.prototype.join(): https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global\_Objects/Array/join

Дворников, В. (б.д.). *Интернет-технологии*. Получено из Изучаем удаление элемента из массива JS: https://www.internet-technologies.ru/articles/kak-udalit-opredelennyy-element-massiva-javascript.html

*ЯКласс*. (б.д.). Получено из Пересечение и объединение множеств: https://www.yaklass.ru/