Министерство образования и науки Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №3

«Функции»

Выполнил студент

группы ИВТАСбд-11

Абросимов К. С.

Ульяновск, 2022

Содержание

[1. Цель работы 3](#_Toc103871043)

[2. Описание метода выполнения 3](#_Toc103871044)

[2.1 Теория 3](#_Toc103871045)

[2.2 Файл «index» 3](#_Toc103871046)

[2.3 Файл «mldmLab3.php» 4](#_Toc103871047)

[2.4 Файл «Scriptslab3.js» 5](#_Toc103871048)

[3. Выводы 13](#_Toc103871049)

[4. Список литературы 13](#_Toc103871050)

1. **Цель работы**

Необходимо написать программу, которая будет определять – является ли введённое отношение функцией. Отношения вводятся по парам элементам. Также вводятся все элементы двух множеств. При компиляции программа должна проверить корректность ввода данных пользователя.

1. **Описание метода выполнения**

## **2.1 Теория**

Функцией, определенной на множестве А и со значениями на множестве В, называется бинарное отношение, при котором каждый элемент из множества А связан с единственным элементом множества В. Другими словами, для каждого а из А существует ровно одна пара из отношения вида (а, Ь).

Для того, чтобы определить, является ли отношение функцией, сперва его нужно преобразовать в матрицу. К примеру, элементы одного множества – строки, а другого - столбцы. Нужно, чтобы каждая строка имела только одну связь с одним из столбцов, так же может действовать и наоборот.

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

## **2.2 Файл «index»**

Первый файл «index», является html страницей, которая имеет ссылки (переходы) на другие страницы – лабораторные работы.

<!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> - ссылка на папку со скриптами  
</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><br> - ссылки к другим лабораторным работам

<a href ="mldmLab3.php">Третья лабораторная работа по МЛиДМ</a>   
</body>  
</html>

## **2.3 Файл «mldmLab3.php»**

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

Как только пользователь введёт элементы множеств A, B и пар элементов и нажмёт кнопку «Сделать расчёт», то программа выполнит функцию «rasschetfunction», ссылаясь на файл «Scriptslab3.js», а затем выведет либо полученный результат в виде матрицы отношения элементов (Matrix) и определения функции в паре элементов (является ли функцией или нет -outResult).

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Лабораторная работа 3</title> - заголовок страницы  
 <script type="text/javascript" src="/scripts/Scriptslab3.js"></script> - ссылка на скрипт с обработкой  
</head>  
<body>  
<h1> Лабораторная работа №3 </h1>  
<form>  
 <table>  
 <tr>  
 <td> Множество A </td>  
 <td> <input type="text" id="mass1" value="" size="50"/></td>  
 </tr>  
 <tr>  
 <td> Множество B </td>  
 <td> <input type="text" id="mass2" value="" size="50"/></td>  
 </tr>  
 <tr>  
 <td> Пары элементов </td>  
 <td> <input type="text" id="mass3" value="" size="50"/></td> - всё это ввод значений для пользователя  
 </tr>  
 <tr>  
 <td colspan="2"> <input type="button" value="Сделать расчёт" onclick="rasschetfunction();"/></td> - кнопка, выполняющая функцию с другого файла  
 </tr>  
 </table>  
  
</form>  
<div id ="Matrix"></div>  
<div id ="outResult"></div> - вывод результатов  
  
</body>  
</html>

## **2.4 Файл «Scriptslab3.js»**

Как и было сказано, выполняется функция “rasschetfunction. Сначала эта функция создаёт переменные для считывания информации, введённое пользователем – A, B, и couples. Но они не являются массивом, а значением. После преобразования значений в массив с помощью функции «normalarr» (об этом позже), а также создания сетов A и B (те же самые массивы, только могут использовать некоторые особенные функции), проверяем массивы на правильность ввода (по условиям задачи и с помощью функции validate). После проверки код либо выведет окно с текстом ошибки, либо же продолжит компилироваться (начнёт основную обработку) и в конце выведет текст, является ли введённая пара элементов функцией или нет.

function rasschetfunction()  
{  
 A = ***document***.getElementById('mass1');  
 B = ***document***.getElementById('mass2'); - берём значения из предыдущего файла  
 var couples = ***document***.getElementById('mass3'); - берём значения, введённые пользователем  
 A = normalarr(A.value);   
 B = normalarr(B.value); - функция преобразования в массив  
 const setA = new ***Set***(A); - создаём «Сеты» только для функции проверки ввода  
 const setB = new ***Set***(B);  
 couples = normalarr(couples.value); - преобразовываем значения в массивы  
 let full\_result = ""; - уже объявляем переменную типа String для будущего вывода текста  
 if (validate(setA, setB, couples) == false) { - функция проверки ввода  
 alert(***error\_text***); - вывод окна с текстом ошибки  
 }

Легкий способ определения функции у пар элементов – через бинарную матрицу. Сделаем так, чтобы элементы множества A были строками, а элементы множества B – столбцы. Если у элементов есть связь, то на пересечении у них стоит знак «1», в ином случае – «0». Для начала, создадим переменные типа Boolean, которые в будущем будут определять, является ли отношение функцией. Но также стоит учитывать, что отношение может быть функцией как А к В, так и В к А. Также подготовим матрицу, при котором код циклично будет заполнять нулями элементы (с пустой матрицей, без значений, никакие действия не работали), причём длина самой матрицы будет составлять длины множеств A и B.   
 else {  
 var matrix = []; - будущее бинарное отношение  
 var flagA = true;  
 var flagB = true; - эти «флаги» будут отвечать за вывод информации о том, является ли отношение функцией  
 for (let i = 0; i < A.length; i++) {  
 matrix[i] = [];  
 for (let j = 0; j < B.length; j++) {  
 matrix[i][j] = 0; - массив пустой, его нужно заполнить нулями, чтобы с ним работать  
 }  
 }

Далее нужно понять, как именно расставить связи в матрице. Массив пар элементов скопируем с помощью couples.slice, чтобы в дальнейшем менять оригинальный массив, не влияя на значение длин при цикле. Определяя местоположение элементов в массивах A и B, мы присваиваем сперва 0 к паре элементов (сбрасываем элемент), а затем добавляем их индекс, чтобы в итоге получился массив со связями, с которыми можно спокойно заполнить матрицу. К примеру, элемент a1 станет 00 (если в множестве A «a» - первый по счёту массив, также и с множеством B). После того, как мы преобразовали массив пар элементов, по его индексам расставляем связи в матрице (присваиваем «1»).  
 var coarr = couples.slice(); - копируем массив пар элементов  
 for (let i = 0; i < couples.length; i++) {  
 for (let j = 0; j < couples.length; j++) {  
 if (couples[i][0] == A[j]) {  
 couples[i] = 0; - присваиваем первому символу элемента (кроме пробела), цифру 0 (будущее положение в массиве), к которой потом прибавим индекс (местоположение элемента в множестве A)  
 couples[i] += j;  
 break;  
 }  
 }  
 for (let j = 0; j < couples.length; j++) {  
 if (coarr[i][1] == B[j]) {  
 couples[i] += '' + j; - Такое же действие с множеством B  
 break;  
 }  
 }  
 }  
 for (let i = 0; i < couples.length; i++) {  
 matrix[couples[i][0]][couples[i][1]] = 1; - присваиваем подготовленной матрице «положение» пар элементов в виде единиц  
 }

Данный цикл нужен для того, чтобы без запятых и в типе String вывести полученную матрицу на страницу.  
 let outmatrix = ""; - копируем матрицу, чтобы просто вывести её на сайте  
 for (let i = 0; i < A.length; i++) {  
 for (let j = 0; j < B.length; j++) {  
 outmatrix += matrix[i][j] + ' ';  
 }  
 outmatrix += '\n';  
 }  
 ***document***.getElementById('Matrix').innerText = "Матрица отношения:\n" + outmatrix +  
 "A - строки, B - столбцы"; вывод полученной матрицы

Следующие несколько циклов будут определять, является ли функцией отношением. Он будет функцией только тогда, когда **каждая** строка (элемент множества A) будет иметь одну **единственную** связь «1» (с элементов множества B). Для этого будем складывать сумму строк и следить за тем, чтобы не сумма не была нулем или больше 1. Если все условия выполнены, значение типа Boolean не меняется (Является функцией), в ином случае – принимает значение false (Не является функцией) – цикл прерывается. Такие действия будут проводиться относительно A к B. Если же с случаем B к A, будет определяться сумма столбцов.

let countrow = 0; - счёт строки  
 let countheight = 0; - счёт столбца  
 for (let i = 0; i < A.length; i++) {  
 for (let j = 0; j < B.length; j++) {  
 countrow += matrix[i][j];  
 }  
 if (countrow >= 2) { - элемент А должен иметь связь только с одним элементов В  
 flagA = false;   
 break;  
 } else if (countrow == 0) { - каждый элемент А должен иметь связь  
 flagA = false;  
 break;  
 } else {  
 countrow = 0;  
 }  
 }

Такой же принцип и с В  
 for (let i = 0; i < A.length; i++) {  
 for (let j = 0; j < B.length; j++) {  
 countheight += matrix[j][i];  
 }  
 if (countheight >= 2) {  
 flagB = false;  
 break;  
 } else if (countheight == 0) {  
 flagB = false;  
 break;  
 }  
 else {  
 countheight = 0;  
 }  
 }   
 if (flagA) { - вывод полученных результатов  
 full\_result += "Отношение является функцией А в B\n";  
 }else {  
 full\_result += "Отношение не является функцией А в B\n";  
 }  
 if (flagB) {  
 full\_result += "Отношение является функцией B в A";  
 }else {  
 full\_result += "Отношение не является функцией B в A";  
 }  
 ***document***.getElementById('outResult').innerText = "\n" + full\_result; - вывод полученных выражений на страницу  
 }  
}

Функция normalarr создаёт переменную (будущий массив, который будем возвращать), присваиваем значение str, взятую с основной функции и сразу же раздёленный по пробелам. Если были лишние пробелы, то они превращаются в пустой элемент, поэтому с помощью цикла и функции splice удаляем эти ненужные элементы.

function normalarr ( str )  
{  
 var mass = str.split(" ");  
 for (let i = 0; i < mass.length; i++) {  
 if (mass[i] == "") {  
 mass.splice(i, 1);  
 }  
 }  
 return mass;  
}

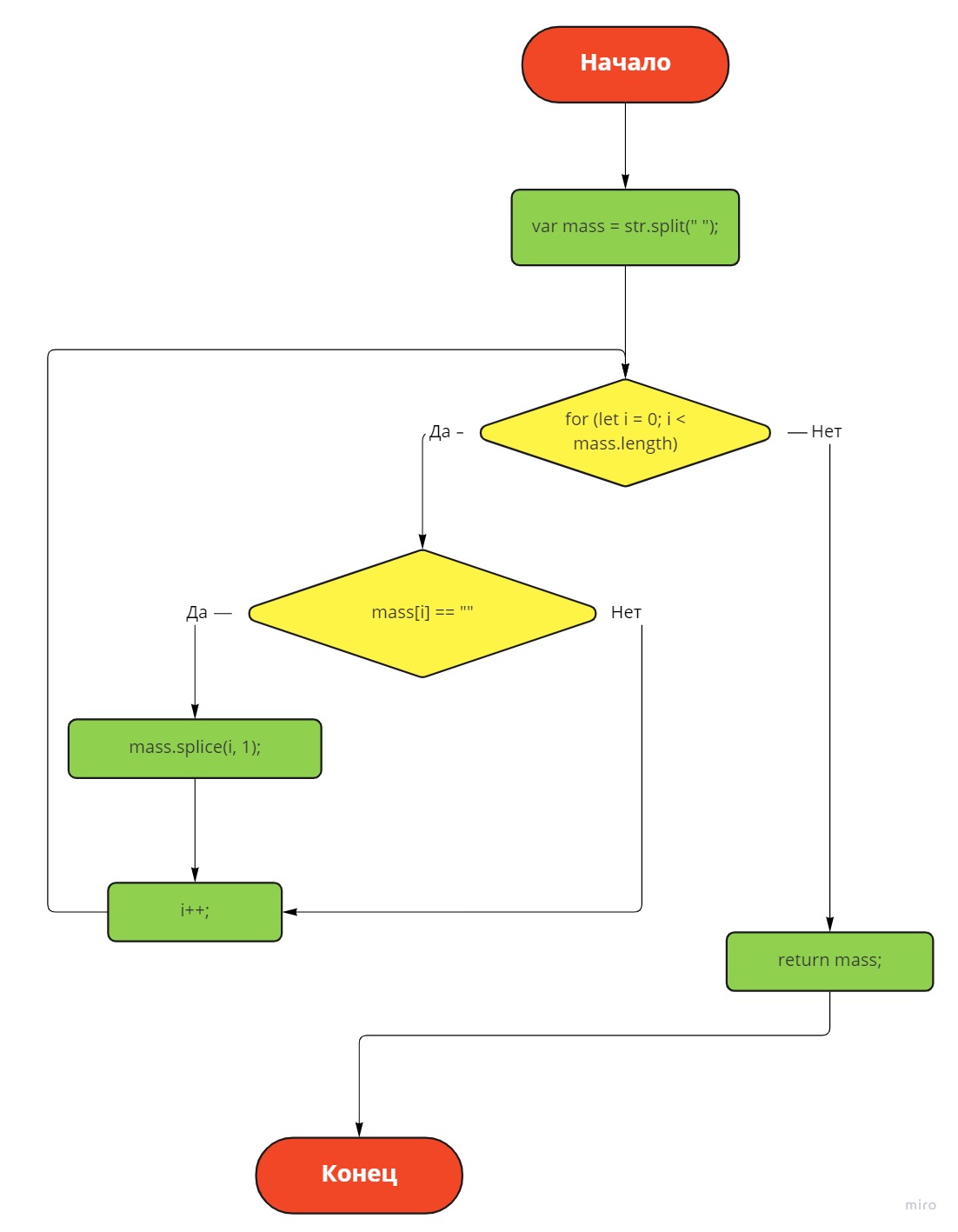


Рис. 1 - функция normalarr

Функция validate проверяет несколько условий:

* Имеются ли у множеств А и В те элементы, которые есть у массива с парами элементов. Вместе с этим проверяется, записаны ли именно пары элементов с двумя символами.
* Имеют ли множества А, В и пар элементов более одного элемента.

Для этого на каждый элемент пар элементов мы будем проверять данные условия, в случае нахождения ошибки записывается текст ошибки. После всего этого идёт последняя проверка – пустое ли значение error\_text. Если да, то возвращается true, и программа будет продолжаться, в ином случае – возвращает false и далее окно с ошибкой. Проверяется наличие элементов с помощью функции has, который взаимодействуют со сетами, а не с простыми значениями, которые выглядят, как массив.

function validate(A, B, couples)

{

for(elem of couples) {

if(!(A.has(elem[0]) && B.has(elem[1])) || elem.length != 2) { - 1 условие

error\_text = "Пары элементов должны быть вида Ai, Bi ... )";

}

}

if (A.size <= 1) {

error\_text = "Не менее 2 элементов в множестве A\n";

}

else if (B.size <= 1) {

error\_text = "Не менее 2 элементов в множестве B\n";

}

else if (couples.length <= 1) {

error\_text = "Не менее двух пар элементов"; - второе условие для каждого множества

}

if(error\_text) {

return false;

}else { - возвращение значения

return true;

}

}

1. **Выводы**

В результате проведения работы была изучена теория об определении функции в бинарных отношениях.

# **Список литературы**

(б.д.).

*studfile*. (б.д.). Получено из 2. Бинарные отношения: https://studfile.net/preview/4229244/