Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра: «Вычислительная техника»

Дисциплина: «Математическая логика и дискретная математика»

Лабораторная работа №1

«Операции над множествами»

Выполнил студент

группы ИВТАСбд-12

Давыдов Д. Д.

Проверил

Игонин А. Г.

Ульяновск, 2022

**Цель работы**

Требуется написать программу, которая выполняет операции над множествами, а именно пересечение, объединение, симметрическая разница, дополнение. Кроме того, необходимо сделать валидацию данных по заданному формату: первый символ – цифра, второй – чётная цифра, третий – нечётная цифра, четвёртый - цифра. Пользователь имеет возможность ввода данных.

**Описание метода выполнения.**

Множество является совокупность объектов. В программе нужно реализовать несколько операций над множествами: объединение, пересечение, симметрическая разница, дополнение первого до второго и наоборот.

Объединение – операция над множествами, возвращающая множество, содержащее все элементы двух других множеств без повторений.

Пересечение – операция над множествами, возвращающая множество, содержащее элементы, которые принадлежат и к первому, и ко второму множеству без повторений.

Симметрическая разница – операция над множествами, возвращающая множество, содержащее элементы и первого и второго множества, не совпадающие между собой.

Дополнение первого до второго – операция над множествами, возвращающая множество, содержащее элементы, которые принадлежат к первому множеству, но не принадлежат ко второму.

Ввод множеств осуществляется с помощью встроенных функций JavaScript путём отображения модального окна для ввода текста. Итоговое множество выводится модальным окном. Код проводимых операций будет написан на JavaScript. JavaScript – это язык, с помощью которого можно писать скрипты для сайтов.

Первым делом рассмотрим функцию валидации элементов множества. Данная функция принимает один элемент множества. Если элемент прошёл проверку, функция возвращает 3, иначе – true.

function checkm (a) {

if (a.length != 4) return true;

if (!(a[0]>='0' && a[0] <= '9' && a[1]>='0' && a[1] <= '9' && a[2]>='0' && a[2] <= '9' && a[3]>='0' && a[3] <= '9')) return true;

if (a[1] % 2 != 0) return true;

if (a[2] % 2 == 0) return true;

return 3;

}

}

Результат данной функции проверяется на равенство «3». И если условие верно, то специальный флаг, следящий за ошибками, остаётся исходного значения, иначе он становится true.

Так осуществляется проверка всех элементов множества:

abc.map((el)=>{

flag = (checkm(el) == 3 ? flag : true)

})}

После этого оба множества проходят через функцию, удаляющую элементы дупликаты.

function checkd (a) {

for (let i = 0; i < a.length; i++){

for (let j = i + 1; j < a.length; j++){

if (a[i] == a[j]) {

a.splice(j,1);

j--;

}

}

}

return a;

}}

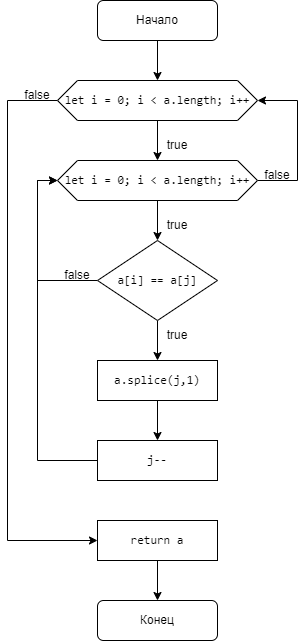


Рис. 1. Блок-схема алгоритма удаления элементов множества.

Переходим к работе операций над множествами.

Первая операция – объединение. Работа функции заключается в объединении элементов двух множеств в одно без повторений.

function ob(a, b){

let res = [...a];

b.map((e) => {

if (!a.includes(e)) res.push(e);

})

return res;

}

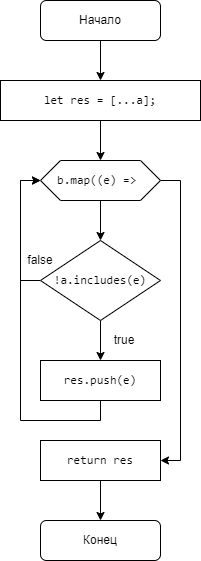


Рис. 2. Блок-схема функции объединения.

Вторая операция – пересечение. Работа функции заключается в поиске одинаковых элементов в двух множествах и их возвращении.

function per(a, b){

let res = [];

a.map((e) => {

if (b.includes(e)) res.push(e);

})

return res;

}

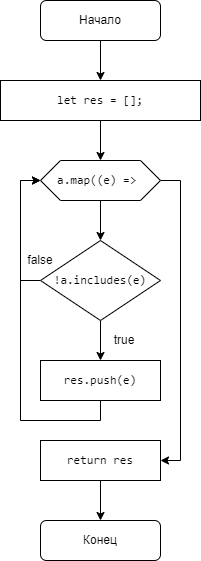


Рис. 3. Блок-схема функции пересечения.

Третья операция – дополнение. Суть функции заключается в поиске элементов в первом множестве, которых нет во втором, и их возврат.

function dop(a, b){

let res = [];

a.map((e) => {

if (!b.includes(e)) res.push(e);

})

return res;

}

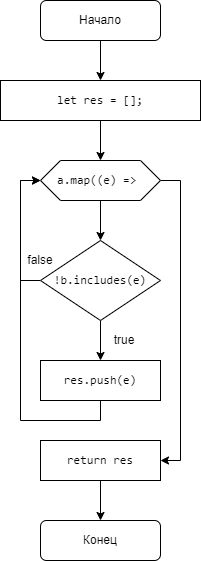


Рис. 4. Блок-схема функции дополнения.

Четвёртая операция – симметрическая разность. Суть функции заключается в запуске функции дополнения 2 раза, меняя аргументы местами.

function obe(a, b){

let res = [];

dop(a,b).map((e)=>{

res.push(e);

})

dop(b,a).map((e)=>{

res.push(e);

})

return res;

}

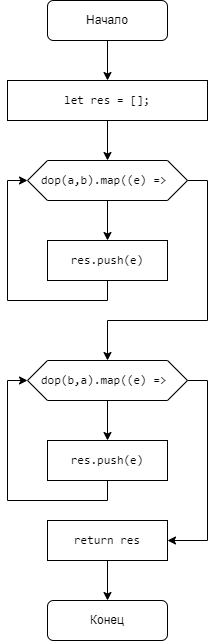


Рис. 5. Блок-схема симметрической разности.

Вывод результатов всех 4 функций и исходных множеств происходит с помощью этой строки.

alert("A: " + abc + "\nB: " + def + "\nОбъединение: " + ob(abc,def) + "\nПересечение: " + per(abc,def) + "\nДополнение A/B: " + dop(abc,def) + "\nДополнение B/A: " + dop(def,abc) + "\nСимметрическая разность: " + obe(abc,def));

**Вывод**

Выполнив данную лабораторную работу, я создал сайт, который может выполнять некоторые операции над двумя множествами и делать проверку вводимых данных по определённому формату.