|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **ПРОГРАМНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.03 ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 2 |

**Дисциплина:** Архитектура ЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | Н.А. Гарасев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.Ю. Попов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

Оглавление

[Цель работы: 3](#_Toc52194168)

[Task 1 4](#_Toc52194169)

[Задание 1 4](#_Toc52194170)

[Задание 2 10](#_Toc52194172)

[Задание 3 15](#_Toc52194174)

[Task 2 21](#_Toc52194176)

[Задание 1 21](#_Toc52194177)

[Задание 2 23](#_Toc52194178)

[Задание 3 26](#_Toc52194179)

[Вывод: 28](#_Toc52194180)

# Цель работы:

Приобретение базовых знаний JavaScript, написание программ, демонстрирующих знание циклов, строк, массивов, объектов и функций, получение знаний в области ООП языка JavaScript.

## Task 3

### Задание 1

**Условие задачи:**

С клавиатуры считывается число N. Далее считывается N строк. Необходимо создать массив и сохранять в него строки только с четной длинной. Получившийся массив необходимо преобразовать в строку JSON и сохранить в файл.

### Листинг кода

"use strict";

let tmp = 0;

let arr = [];

let str;

const readlineSync = require('readline-sync');

const n = parseInt(readlineSync.question("Input count of string: "));

while(tmp !== n) {

    str = readlineSync.question("Input string: ");

    if (str.length % 2 ==  0) {

         arr.push(str);

    }

    tmp++;

}

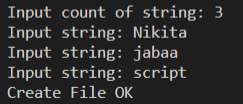
const fs = require("fs");

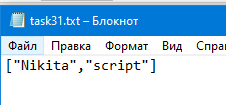
const nameString = "task31.txt";

const contentString = JSON.stringify(arr);

fs.writeFileSync(nameString, contentString);

console.log("Create File OK");





### Задание 2

**Условие задачи:**

### Необходимо считать содержимое файла, в котором хранится массив строк в формате JSON. Нужно вывести только те строки на экран, в которых содержатся только гласные буквы.

### Листинг программы:

"use strict";

function checkOnlyVowels(str) {

    let vowel = ['A', 'E', 'U', 'Y', 'I', 'O'];

    let tmp\_str = str.toUpperCase();

    for (let i = 0; i < tmp\_str.length; i++) {

        if (vowel.indexOf(tmp\_str[i]) == -1) {

            return false;

        }

    }

    return true;

}

const fs = require("fs");

const nameString = "task32.txt";

if (fs.existsSync(nameString)) {

    const contentString = fs.readFileSync(nameString, "utf8");

    const obj = JSON.parse(contentString);

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

        if (checkOnlyVowels(obj[i])) {

            console.log(obj[i]);

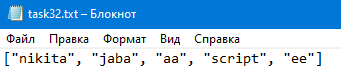
        }

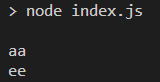
    }

} else {

    console.log("File was not found");

}





### Задание 3

**Условие задачи:**

### С клавиатуры считывается строка - название расширения файлов. Далее считывается строка - адрес папки. Необходимо перебрать все файлы в папке и вывести содержимое файлов, у которых расширение совпадает с введенным расширением.

### Листинг кода:

"use strict";

function checkExtension(extension, needCheckExtension) {

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

        if (extension[extension.length - 1 - i] != needCheckExtension[needCheckExtension.length - 1 - i]) {

            return false;

        }

    }

    return true;

}

const fs = require("fs");

const readlineSync = require('readline-sync');

const extension = readlineSync.question("Input file extension with . : ")

const folder = readlineSync.question("Input adress folder: ")

const arr = fs.readdirSync(folder);

let readData = '';

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

    const fileName = folder + arr[i];

    if (checkExtension(extension, arr[i])) {

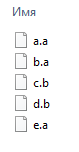
        const contentString = fs.readFileSync(fileName, "utf8");

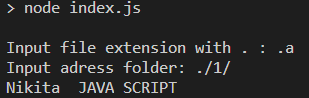
        readData += contentString;

    }

}

console.log(readData);





### Задание 4

**Условие задачи:**

Дана вложенная структура файлов и папок. Все файлы имеют раcширение "txt". Необходимо рекурсивно перебрать вложенную структуру и вывести имена файлов, у которых содержимое не превышает по длине 10 символов.

### Листинг кода:

"use strict";

function recursionReadFile(dir) {

    const arr = fs.readdirSync(dir);

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

        try {

            const fileText = fs.readFileSync(dir + arr[i], "utf8");

            if (fileText.length <= 10) {

                console.log(fileText);

            }

        } catch(err) {

            recursionReadFile(dir + arr[i] + '/');

        }

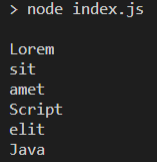
    }

}

const fs = require("fs");

const dir = "./1/"

recursionReadFile(dir);



### Задание 5

**Условие задачи:**

С клавиатуры считывается число N. Далее считывается N строк - имена текстовых файлов. Необходимо склеить всё содержимое введенных файлов в одну большую строку и сохранить в новый файл.

### Листинг кода:

"use strict";

const readlineSync = require('readline-sync');

const fs = require("fs");

const n = parseInt(readlineSync.question("Input count of file: "));

console.log(n);

let tmp = 0;

let str = '';

while(tmp !== n) {

    const file\_name = readlineSync.question("Input file: ");

    if (fs.existsSync(file\_name)) {

        str += fs.readFileSync(file\_name, "utf8");

        tmp++;

    } else {

        console.log("File was not found");

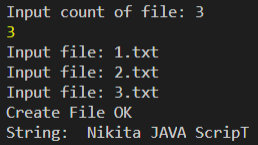
    }

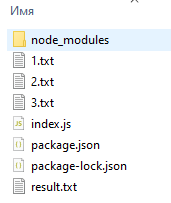
}

fs.writeFileSync("result.txt", str);

console.log("Create File OK");

console.log("String: ", str);





### Задание 6

**Условие задачи:**

Написать код, который позволяет определить максимальный возможный уровень вложенности друг в друга полей в объекте, чтобы данный объект можно было преобразовать в строку формата JSON. Ответом является целое число.

### Листинг кода:

"use strict";

let exObj = {name: "obj0", child:null};

for (let i = 1;;i++) {

    try {

        const obj = {name: "obj" + i, child: exObj};

        JSON.stringify(obj);

        exObj = obj;

    } catch (err) {

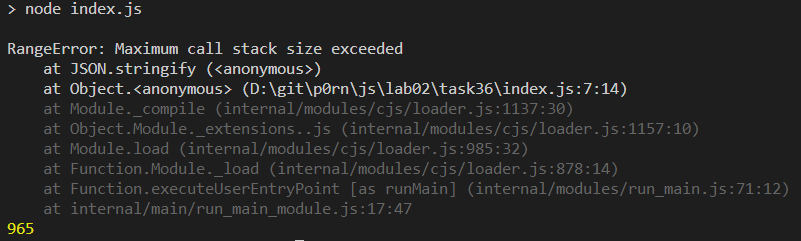
        console.log(err);

        console.log(i);

        break;

    }

}



### Задание 7

**Условие задачи:**

Из файла считывается строка в формате JSON. В этой строке информация об объекте, в котором находится большое количество вложенных друг в друга полей. Объект представляет из себя дерево. Необходимо рекурсивно обработать дерево и найти максимальную вложенность в дереве. Необходимо вывести на экран ветку с максимальной вложенностью.

### Листинг кода:

"use strict";

const fs = require("fs");

const node9 = {name: "9", child1: null, child2: null};

const node8 = {name: "8", child1: node9, child2: null};

const node7 = {name: "7", child1: null, child2: null};

const node6 = {name: "6", child1: null, child2: null};

const node5 = {name: "5", child1: null, child2: null};

const node4 = {name: "4", child1: node7, child2: node8};

const node3 = {name: "3", child1: node5, child2: node6};

const node2 = {name: "2", child1: node3, child2: node4};

const node1 = {name: "1", child1: node2, child2: null};

console.log(JSON.stringify(node1));

fs.writeFileSync("tree.txt", JSON.stringify(node1));

let longestbranch = "";

function search(node, branch) {

    branch += node.name;

    if (branch.length > longestbranch.length)

        longestbranch = branch;

    if (node.child1 !== null)

        search(node.child1, branch)

    if (node.child2 !== null)

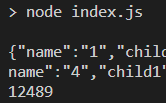
        search(node.child2, branch)

}

const obj = JSON.parse(fs.readFileSync("tree.txt", "utf8"));

search(obj, "");

console.log(longestbranch);



## Task 4

### Задание 1

**Условие задания:**

Запустить сервер. Реализовать на сервере функцию для сравнения трёх чисел и выдачи наибольшего из них. Реализовать страницу с формой ввода для отправки запроса на сервер.

**Листинг кода:**

"use strict";

const fs = require("fs");

const express = require("express");

const app = express();

const port = 5015;

app.listen(port);

console.log("My server on port " + port);

app.get("/me/page", function(request, response) {

    const nameString = request.query.p;

    if (fs.existsSync(nameString)) {

        const contentString = fs.readFileSync(nameString, "utf8");

        response.end(contentString);

    } else {

        const contentString = fs.readFileSync("bad.html", "utf8");

        response.end(contentString);

    }

});

app.get("/find/max", function(request, response) {

    const a = request.query.a;

    const b = request.query.b;

    const c = request.query.c;

    const aInt = parseInt(a);

    const bInt = parseInt(b);

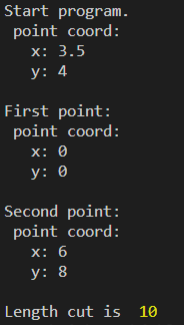
    const cInt = parseInt(c);

    const maxInt = Math.max(Math.max(aInt, bInt), cInt);

    const answerJSON = JSON.stringify({result: maxInt});

    response.end(answerJSON);

});



### Задание 2

**Условие задачи:**

Создать класс *Треугольник*.

Класс *Треугольник* должен иметь поля, хранящие длины сторон треугольника.

Реализовать следующие методы:

* Метод инициализации полей
* Метод проверки возможности существования треугольника с такими сторонами
* Метод получения периметра треугольника
* Метод получения площади треугольника
* Метод для проверки факта: является ли треугольник прямоугольным

**Листинг кода:**

"use strict";

class Triangle {

    constructor(a,  b, c) {

        this.a = a;

        this.b = b;

        this.c = c;

    }

    console\_log() {

        let start = " triangle len: \n"

        let a = "   a: " + this.a + "\n";

        let b = "   b: " + this.b + "\n";

        let c = "   b: " + this.c + "\n";

        let message = start + a + b + c;

        console.log(message);

    }

    triangle\_check() {

        if ((this.a + this.b > this.c) && (this.a + this.c > this.b) && (this.c + this.b > this.a)) {

            return true;

        }

        return false;

    }

    hypotenuse(leg1, leg2, hypotense) {

        if ((leg1 \* leg1 + leg2 \* leg2) == hypotense \* hypotense) {

            return true;

        }

        return false;

    }

    rectangular() {

        if (this.triangle\_check()) {

            let tmp = Math.max(this.a, this.b, this.c);

            if (tmp == this.a) {

                return this.hypotenuse(this.b, this.c, tmp);

            } else if (tmp == this.b) {

                return this.hypotenuse(this.a, this.c, tmp);

            } else {

                return this.hypotenuse(this.a, this.b, tmp);

            }

        }

        return NaN;

    }

    perimetr() {

        if (this.triangle\_check()) {

            return this.a + this.b + this.c;

        }

        return NaN;

    }

    square() {

        if (this.triangle\_check()) {

            let p = this.perimetr() / 2;

            return Math.sqrt(p \* (p - this.a) \* (p - this.b) \* (p - this.c));

        }

        return NaN;

    }

}

let triangle = new Triangle(10, 9, 11);

console.log("Data: 10, 9, 11");

console.log("Is it triangle?");

console.log(triangle.triangle\_check());

console.log("Perimetr = ", triangle.perimetr());

console.log("Square = ", triangle.square());

console.log("Is it rectangular triangle?");

console.log(triangle.rectangular());

let triangle1 = new Triangle(15, 15, 15);

console.log("Data: 15, 15, 15");

console.log("Is it triangle?");

console.log(triangle1.triangle\_check());

console.log("Perimetr = ", triangle1.perimetr());

console.log("Square = ", triangle1.square());

console.log("Is it rectangular triangle?");

console.log(triangle1.rectangular());

let triangle2 = new Triangle(3, 4, 5);

console.log("Data: 3, 4, 5");

console.log("Is it triangle?");

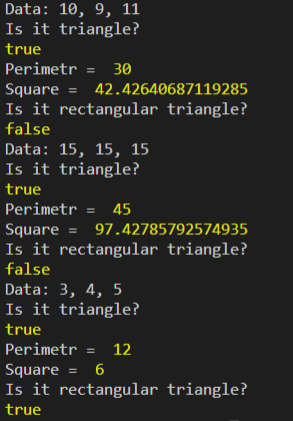
console.log(triangle2.triangle\_check());

console.log("Perimetr = ", triangle2.perimetr());

console.log("Square = ", triangle2.square());

console.log("Is it rectangular triangle?");

console.log(triangle2.rectangular());



### Задание 3

**Условие задачи:**

Реализовать программу, в которой происходят следующие действия:

Происходит вывод целых чисел от 1 до 10 с задержками в 2 секунды.

После этого происходит вывод от 11 до 20 с задержками в 1 секунду.

Потом опять происходит вывод чисел от 1 до 10 с задержками в 2 секунды.

После этого происходит вывод от 11 до 20 с задержками в 1 секунду.

Это должно происходить циклически.

**Листинг кода:**

"use strict";

let count = 0;

let needCount = 2;

function printNumber(delay, stage){

    let num = 0;

    let interval = setInterval(() => {

        if (num >= 0) {

            num++;

            if (stage === 0) {

                console.log(num);

            } else {

                console.log(num + 10);

            }

        }

        if (num === 10) {

            clearInterval(interval);

            num = 0;

            if (stage === 0) {

                printNumber(1000, 1);

            } else {

                count++;

                if (count != needCount) {

                    printNumber(2000, 0);

                }

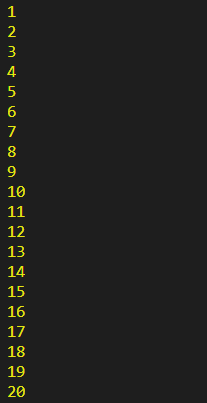
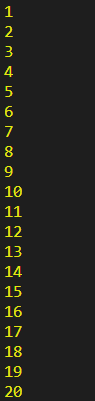
            }

        };

    }, delay);

}

printNumber(2000, 0);

****

## Вывод:

Были приобретены базовые знания JavaScript. Все задачи на знание основ JavaScript, а также ООП в JavaScript выполнены успешно.