|  |  |
| --- | --- |
| 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 «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ДИСЦИПЛИНА «Архитектура ЭВМ»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 2**

|  |  |
| --- | --- |
| **Студент** \_\_Кобаренков И.В.\_\_\_\_\_\_\_\_\_\_\_\_\_  **Группа** \_\_ИУ7-51Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Преподаватель** \_\_Попов А. Ю.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |

Москва.

2020 г.

Оглавление

[Задание 3.1 3](#_Toc52146684)

[Задание 3.2 3](#_Toc52146685)

[Задание 3.3 4](#_Toc52146686)

[Задание 3.4 5](#_Toc52146687)

[Задание 3.5 6](#_Toc52146688)

[Задание 3.6 6](#_Toc52146689)

[Задание 3.7 7](#_Toc52146690)

[Задание 4.1 9](#_Toc52146691)

[Задание 4.2 10](#_Toc52146692)

[Задание 4.3 11](#_Toc52146693)

[Задание 4.4 13](#_Toc52146694)

Цель работы: знакомство с форматом JSON, работа с текстовыми файлами, использование фреймворка express.

# Задание 3.1

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 программы

*function* task\_1(*file\_name*) {

*const* N = readlineSync.question("N = ");

*let* string;

*const* arr = [];

    for (*let* i = 0; i < parseInt(N); i++) {

        string = readlineSync.question("Input " + (i + 1) + " string: ");

        if (!(string.length % 2))

            arr.push(string);

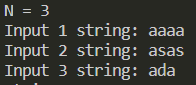
    }

*const* jsonString = JSON.stringify(arr, null, 4);

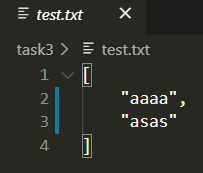
    fs.writeFileSync(*file\_name*, jsonString);

}

Примеры работы



Содержимое файла test.txt:



# Задание 3.2

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

*function* task\_2(*file\_name*) {

    if (fs.existsSync(*file\_name*)) {

*const* contentString = fs.readFileSync(*file\_name*, "utf8");

*const* arr = JSON.parse(contentString);

        console.log("Strings:");

*let* flag;

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

            flag = true;

            for (*let* j = 0; j < arr[i].length; j++) {

                if (['A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u',

                    'А', 'Е', 'Ё', 'И', 'О', 'У', 'Ы', 'Э', 'Ю', 'Я',

                    'а', 'е', 'ё', 'и', 'о', 'у', 'ы', 'э', 'ю', 'я'].indexOf(arr[i].charAt(j)) === -1)

                    flag = false;

            }

            if (flag)

                console.log(arr[i]);

        }

    } else

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

}

Примеры работы



Содержимое входного файла: [ "aaaa", "asas"]

# Задание 3.3

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 программы

*function* task\_3(*extension*, *folder\_address*) {

*const* arr = fs.readdirSync(*folder\_address*);

    console.log("Files:");

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

*const* parse\_arr = arr[i].split(".");

        if (parse\_arr.pop() === *extension*)

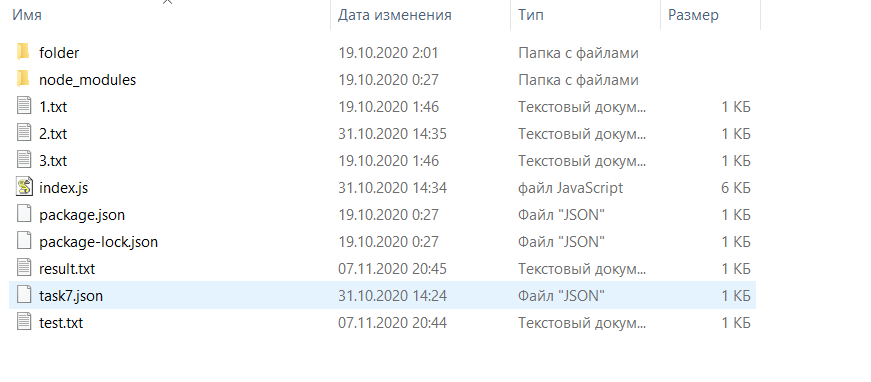
            console.log(arr[i]);

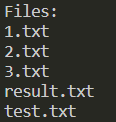
    }

}

Примеры работы

Работа на примере следующей директории





# Задание 3.4

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

Листинг программы

*function* is\_txt\_file(*file\_name*) {

    return (*file\_name*.split(".").pop() === "txt");

}

// перебрать вложенную структуру и вывести имена файлов, у которых содержимое не превышает по длине 10 символов

*function* task\_4(*catalog*) {

*const* arr = fs.readdirSync(*catalog*);

*let* count\_files = 0;

*const* folders = [];

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

        if (is\_txt\_file(arr[i])) {

            count\_files++;

*const* contentString = fs.readFileSync(*catalog* + "/" + arr[i], "utf8");

            if (contentString.length <= 10)

                console.log(arr[i]);

        }

        else

            folders.push(arr[i]);

    }

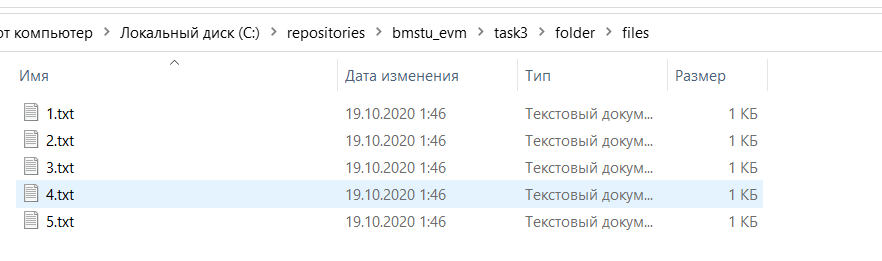
    for (*let* i = 0; i < folders.length; i++)

        task\_4(*catalog* + "/" + folders[i]);

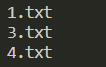
}

Пример работы





Результат:



# Задание 3.5

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 программы

*function* task\_5(*file\_name*) {

*const* N = readlineSync.question("N = ");

*const* arr\_names = [];

*let* string;

    for (*let* i = 0; i < parseInt(N); i++) {

        string = readlineSync.question("Input " + (i + 1) + " name of file: ");

        arr\_names.push(string);

    }

*let* string\_data = "";

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

        if (fs.existsSync(arr\_names[i]))

            string\_data += fs.readFileSync(arr\_names[i], "utf8");

        else

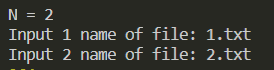
            console.log("File " + arr\_names[i] + " was not found");

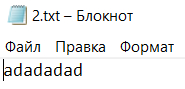
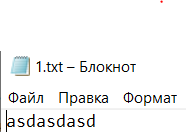
    }

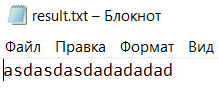
    fs.writeFileSync(*file\_name*, string\_data);

}

Пример работы







# Задание 3.6

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 программы

*function* task\_6() {

*let* noErrorFlag = true

*let* head = {

        num: 0,

        next: null

    }

*let* curNode = head

    while (noErrorFlag) {

        try {

*let* testStr = JSON.stringify(head)

            curNode.next = {

                num: curNode.num + 1,

                next: null

            }

            curNode = curNode.next

        }

        catch (e) {

            noErrorFlag = false

            console.log(curNode.num - 1)

        }

    }

}

Пример работы



# Задание 3.7

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.

*function* task\_7(*filename*){

*function* copyList(*dst*, *src*) {

*let* dstBuf = *dst*

*let* srcBuf = *src*

        while (srcBuf != null) {

            dstBuf.info = srcBuf.info

            dstBuf.next = {

                info: null,

                next: null

            }

            dstBuf = dstBuf.next

            srcBuf = srcBuf.next

        }

    }

*function* find\_max\_deep(*jsonObject*, *i*, *maxDeepInfo*, *buf*, *bufHead*) {

        for (*let* elem in *jsonObject*) {

*buf*.next = {

                info: elem,

                next: null

            }

            if (*i* > *maxDeepInfo*.maxDeep) {

*maxDeepInfo*.maxDeep = *i*

                copyList(*maxDeepInfo*, *bufHead*)

            }

            if (typeof *jsonObject*[elem] === "object") {

                find\_max\_deep(*jsonObject*[elem], *i* + 1, *maxDeepInfo*, *buf*.next, *bufHead*)

            }

        }

    }

*let* jsonString = fs.readFileSync(*filename*, "utf8")

*let* jsonObj = JSON.parse(jsonString)

*let* bufHead = {

        info: jsonObj,

        next: null

    }

*let* res = {

        maxDeep: 0,

        info: null,

        next: null

    }

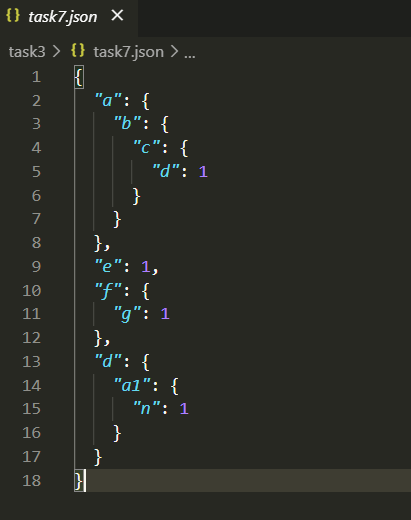
    find\_max\_deep(jsonObj, 1, res, bufHead, bufHead)

    console.log(JSON.stringify(res, null, 4))

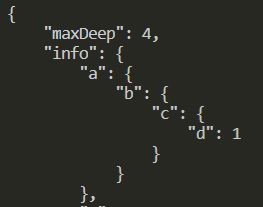
}

Пример работы

Исходный файл:



Результат:



# Задание 4.1

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 программы

"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("/calculate/compare", *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);

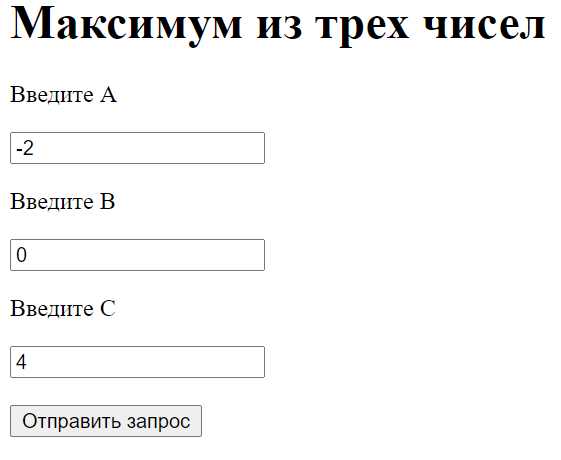
*const* sInt = Math.max(a, b, c);

*const* result = JSON.stringify({result: sInt});

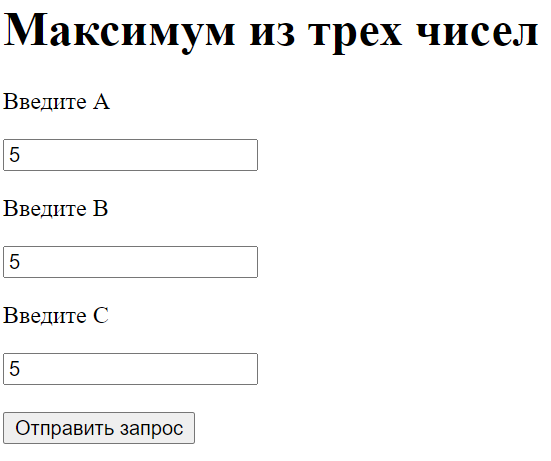
*response*.end(result);

});

Примеры работы









# Задание 4.2

Запустить сервер. На стороне сервера должен храниться файл, внутри которого находится JSON строка. В этой JSON строке хранится информация о массиве объектов. Реализовать на сервере функцию, которая принимает индекс и выдает содержимое ячейки массива по данному индексу. Реализовать страницу с формой ввода для отправки запроса на сервер.

Листинг программы

"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* string = fs.readFileSync(nameString, "utf8");

*response*.end(string);

    } else {

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

*response*.end(string);

    }

});

app.get("/data/get\_el", *function*(*request*, *response*) {

*const* el\_index = *request*.query.el\_index;

*const* index = parseInt(el\_index);

    if (fs.existsSync('data.json')) {

*const* data = fs.readFileSync('data.json', "utf-8");

*const* dataJSON = JSON.parse(data);

*const* result = JSON.stringify({result: dataJSON['data'][index]});

*response*.end(result);

    } else {

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

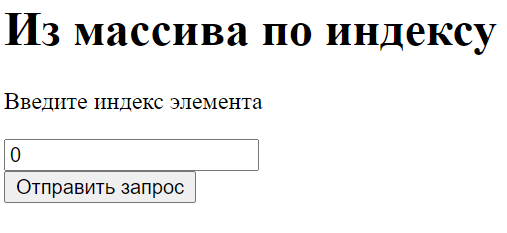
*response*.end(string);

    }

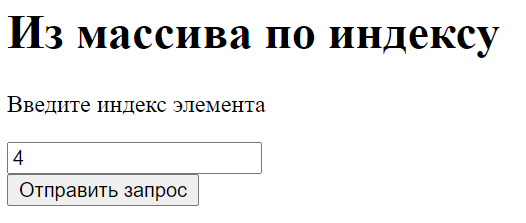
});

Примеры работы

Массив для работы: [1, 2, 3, 4 5]









# Задание 4.3

Написать программу, которая на вход получает массив названий полей и адрес запроса (куда отправлять). Программа должна генерировать HTML разметку страницы, в которую встроена форма для отправки запроса.

Листинг программы

"use strict";

*const* fs = require("fs");

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

*const* fields = readlineSync.question('Input array of fields: ').split(',');

*const* url = readlineSync.question('Input url: ');

if (fs.existsSync('template\_form.html')) {

*let* html = fs.readFileSync('template\_form.html', "utf-8").split('\n');

*let* fields\_index = html.indexOf('    INPUT\_FORM');

*let* url\_index = fields\_index - 1;

    html[fields\_index] = '';

    fields.forEach(*function* (*field*){

        html[fields\_index] +=

            "    <p>" + *field* + "</p>\n" +

            "    <input name=\"" + *field* + "\" spellcheck=\"false\" autocomplete=\"off\">\n"

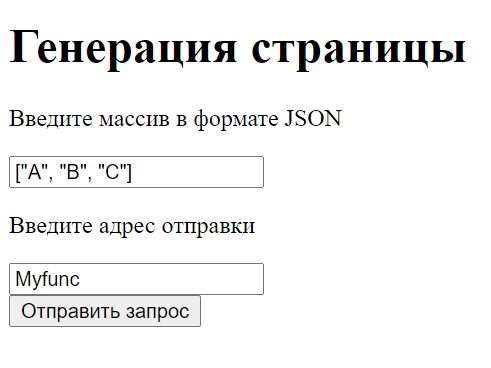
    })

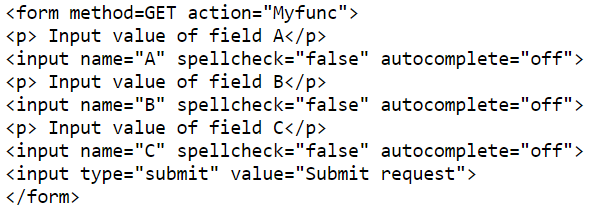
    html[url\_index] = html[url\_index].replace('INPUT\_URL', url);

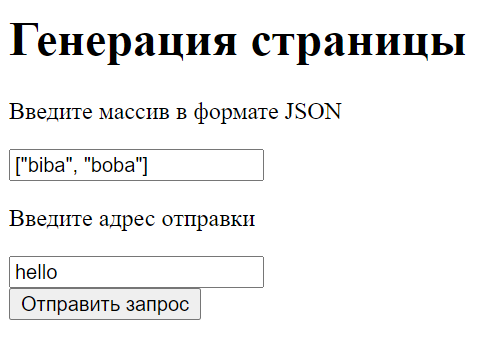
    fs.writeFileSync('form.html', html.join('\n'));

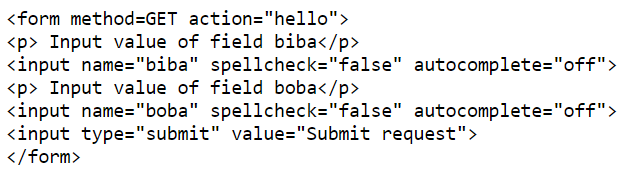
}

Примеры работы









# Задание 4.4

Запустить сервер. Реализовать на сервере функцию, которая принимает на вход числа A, B и C. Функция должна выдавать массив целых чисел на отрезке от A до B, которые делятся на C нацело.

Листинг программы

"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("/calc/arr", *function*(*request*, *response*) {

*const* a = *request*.query.a;

*const* b = *request*.query.b;

*const* c = *request*.query.c;

*const* aInt = parseFloat(a);

*const* bInt = parseFloat(b);

*const* cInt = parseFloat(c);

*let* arr = [];

    for (*let* num = aInt; num <= bInt; num += 1){

        if (!(num % cInt))

            arr.push(num);

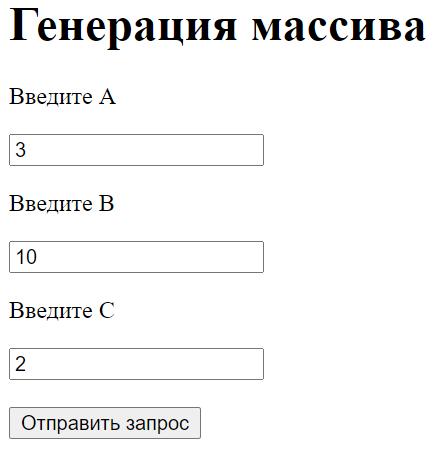
    }

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

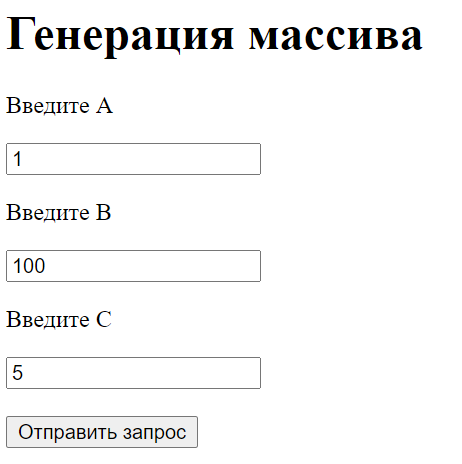
*response*.end(answerJSON);

});

Примеры работы









**Вывод**: в ходе данной лабораторной работы были изучен формат JSON, приобретены навыки работы с фреймворком express и текстовыми файлами.