



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 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.04 Программная инженерия

О Т Ч Е Т

по лабораторной работе № 2

Дисциплина: Архитектура ЭВМ

Студент

ИУ7-52Б

(Группа)

Сучков А.Д.

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

Попов А.Ю.

(Подпись, дата)

(И.О. Фамилия)

Москва, 2020

Цель: изучить формат JSON и возможности фреймворка express.

Часть 1

Задание 1

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

Листинг

```
"use strict";

const readline = require("readline-sync");
const fs = require("fs");

const fileName = "result.txt";
let strArray = [];

let n = parseInt(readline.question("Input N: "));

if (isNaN(n) || n < 0) {
    console.log("Wrong N");
    return;
}

for (let i = 0; i < n; i++) {
    let strTemp = readline.question("Input string: ");

    if ((strTemp.length % 2) == 0) {
        strArray.push(strTemp);
    }
}

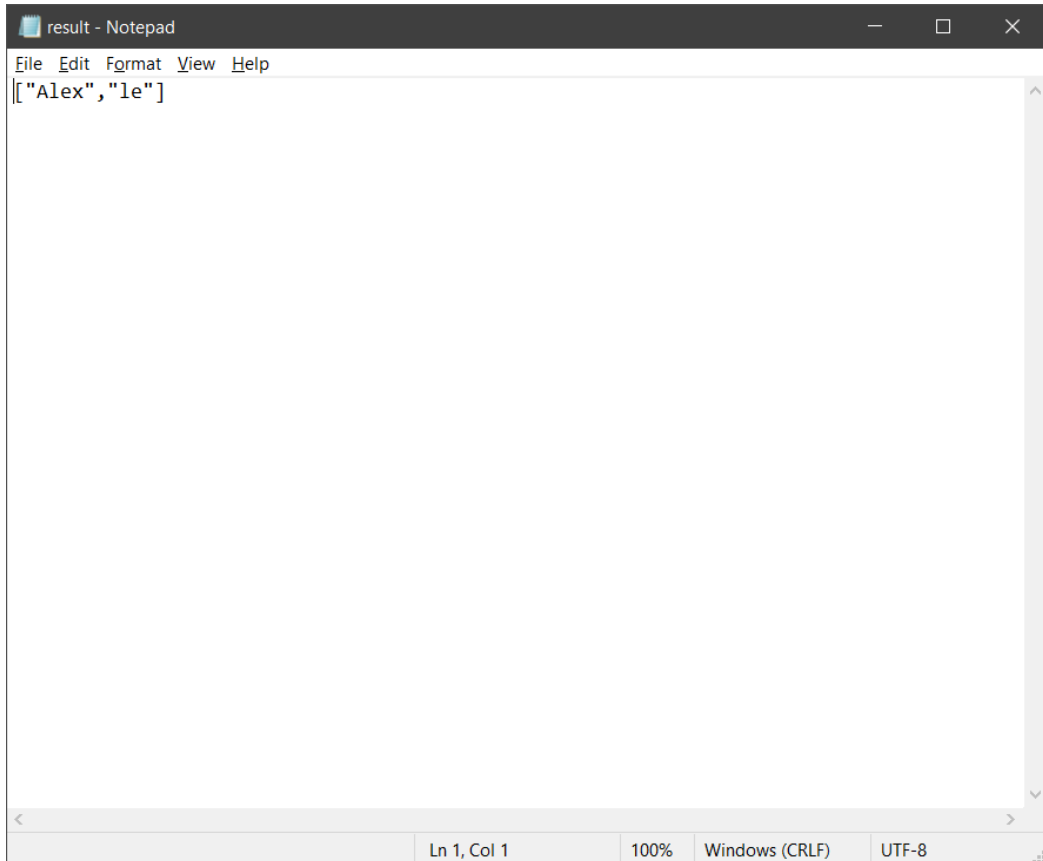
const arrayJson = JSON.stringify(strArray);
fs.writeFileSync(fileName, arrayJson);
```

Тесты

```
PS C:\Repositories\bmstu_archEvm\lab_02\task_11> npm start

> task_01@1.0.0 start C:\Repositories\bmstu_archEvm\lab_02\task_11
> node index.js

Input N: 4
Input string: Alex
Input string: lex
Input string: le
Input string: x
```



Задание 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

```
"use strict";

const fs = require("fs");
const fileName = "data.txt";

function checkVowel(string) {
    let consonantArray = "йцкнгшщзхъфвпрлджчсмтьб";

    for (let letter of string) {
        for (let consonant of consonantArray) {
            if (letter === consonant) {
                return false;
            }
        }
    }

    return true;
}

function main() {
    let error = null;

    if (fs.existsSync(fileName)) {
        let strArrayJSON = fs.readFileSync(fileName, "utf8");

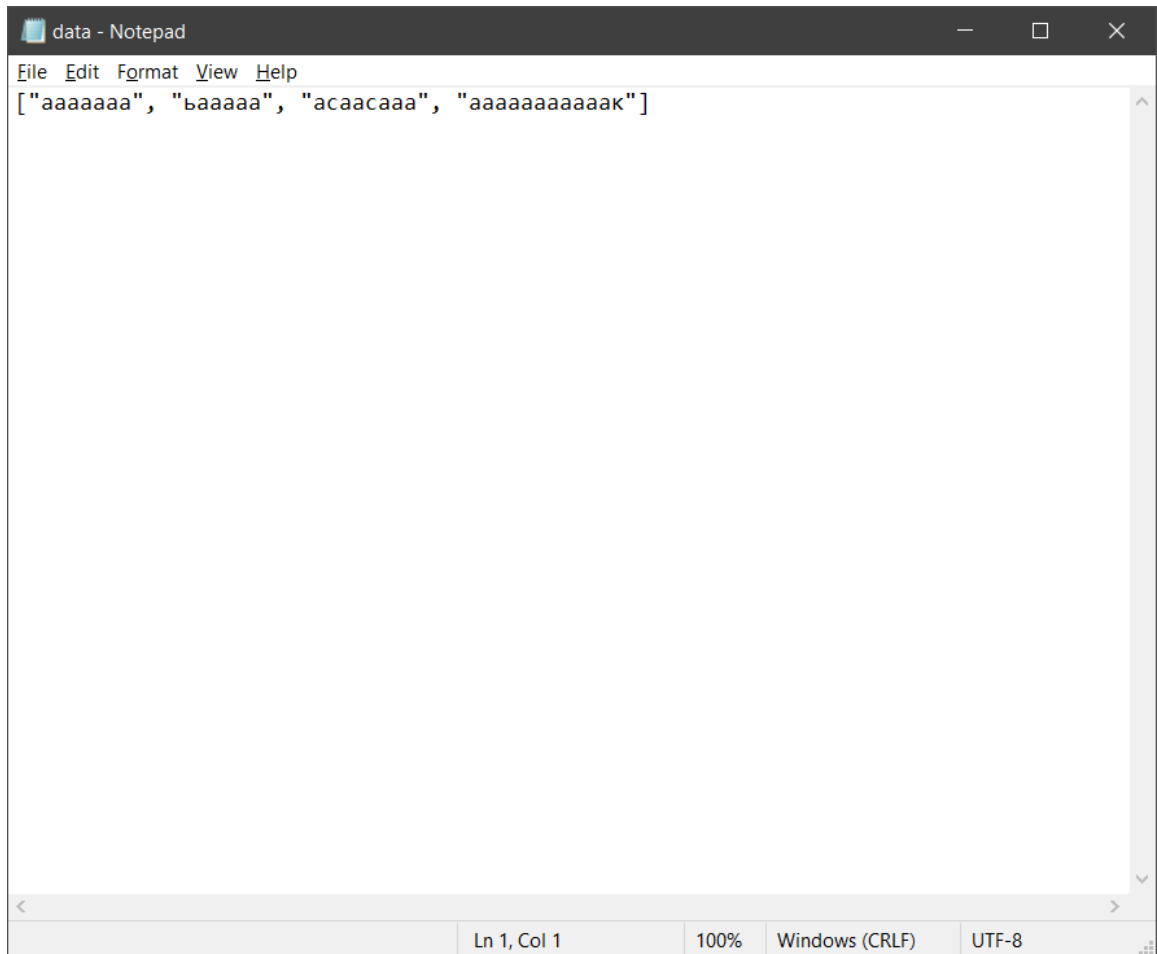
        const strArray = JSON.parse(strArrayJSON);

        if (strArray) {
            for (let str of strArray) {
                if (checkVowel(str)) {
                    console.log(str);
                }
            }
        } else {
            error = "Wrong data";
        }
    } else {
        error = "File does not exist";
    }

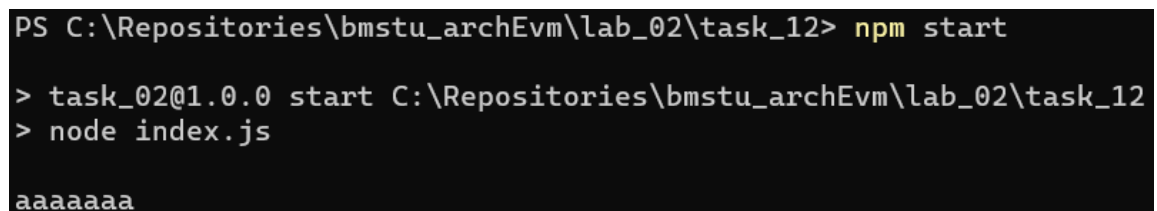
    if (error) {
        console.log(error);
    }
}

main()
```

Тесты



```
data - Notepad
File Edit Format View Help
[\"aaaaaaa\", \"ьaaaaa\", \"асаасааа\", \"aaaaaaaaаак\"]
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```



```
PS C:\Repositories\bmstu_archEvm\lab_02\task_12> npm start
> task_02@1.0.0 start C:\Repositories\bmstu_archEvm\lab_02\task_12
> node index.js
aaaaaaa
```

Задание 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

```
"use strict";

const fs = require("fs");
const readline = require("readline-sync");

function main() {
    let error = null;

    let fileFormat = readline.question("Input format: ");
    let directory = readline.question("Input directory: ");

    if (fs.existsSync(directory)) {
        const fileArray = fs.readdirSync(directory);





        if (fileArray.length > 0) {
            for (let file of fileArray) {
                const filePath = directory + '/' + file;

                if (file.endsWith(fileFormat) && fs.existsSync(filePath)) {
                    const fileContent = fs.readFileSync(filePath, "utf-8");
                    console.log('\n' + filePath + '\n', fileContent);
                }
            }
        }
        else {
            error = "Folder is empty";
        }
    }
    else {
        error = "Directory does not exist";
    }

    if (error) {
        console.log(error);
    }
}

main()
```

Тесты

 file1	9/22/2020 11:07 AM	Документ Microso...	0 KB
 file2	9/22/2020 11:08 AM	Microsoft Edge PD...	26 KB
 kid_list	9/22/2020 11:19 AM	Text Document	1 KB
 student_list	9/22/2020 11:45 AM	Text Document	1 KB

```
PS C:\Repositories\bmstu_archEvm\lab_02\task_13> npm start

> task_03@1.0.0 start C:\Repositories\bmstu_archEvm\lab_02\task_13
> node index.js

Input format: .txt
Input directory: ./data

./data/kid_list.txt
Kid 01 - 10
Kid 02 - 11
Kid 03 - 8
Kid 04 - 3
Kid 05 - 2
Kid 06 - 12
Kid 07 - 13
Kid 08 - 5
Kid 09 - 7
Kid 10 - 11
Kid 11 - 5

./data/student_list.txt
Student 01 - 19
Student 02 - 18
Student 03 - 23
Student 04 - 22
Student 05 - 17
Student 06 - 17
Student 07 - 17
Student 08 - 17
Student 09 - 17
Student 10 - 17
Student 11 - 17
Student 12 - 17
Student 13 - 17
```

Задание 4

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

Листинг

```
"use strict";

const fs = require("fs");
const readline = require("readline-sync");

const contentLength = 10;
const fileFormat = ".txt";

function directoryProcessing(path) {
    let fileArray = []

    if (fs.existsSync(path)) {
        fileArray = fs.readdirSync(path);

        for (let object of fileArray) {
            let fullFilePath = path + '/' + object;

            if (fs.statSync(fullFilePath).isDirectory()) {
                directoryProcessing(fullFilePath);
            }
            else if (fullFilePath.endsWith(fileFormat)) {
                if (fs.existsSync(fullFilePath)) {
                    let fileContent = fs.readFileSync(fullFilePath, "utf-8");

                    if (fileContent.length <= contentLength) {
                        console.log("\nFile - " + fullFilePath, "\n" + fileContent, "\n");
                    }
                }
                else {
                    console.log("File ", fullFilePath, " is not available");
                }
            }
        }
    }
    else {
        console.log("Directory does not exist");
    }
}

function main() {
    let directoryStart = readline.question("Input directory: ");
    directoryProcessing(directoryStart);
}

main()
```


Тесты

```
PS C:\Repositories\bmstu_archEvm\lab_02\task_14> npm start

> task_04@1.0.0 start C:\Repositories\bmstu_archEvm\lab_02\task_14
> node index.js

Input directory: source2

File - source2/source21/source211/last_file.txt
файл < 10

File - source2/source21/txt3.txt
2-я лаба

File - source2/source22/File.txt
Я сашик
```

Задание 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

```
"use strict";

const fs = require("fs");
const readline = require("readline-sync");

function inputFileName(count) {
    let fileArray = [];

    for (let i = 0; i < count; i++) {
        let fileName = readline.question("Input file name: ");
        fileArray.push(fileName);
    }

    return fileArray;
}

function main() {
    let n = parseInt(readline.question("Input file count: "));

    if (isNaN(n) || n < 0) {
        console.log("File number is wrong");
        return;
    }
}
```

```

    }

    let fileArray = inputFilesName(n);

    let newFileName = readline.question("Input name for new file: ");
    let newFileContent = "";

    for (let file of fileArray) {
        if (fs.existsSync(file)) {
            let content = fs.readFileSync(file, "utf-8");
            newFileContent += content;
        }
        else {
            console.log("Some files are not available");
            break;
        }
    }

    fs.writeFileSync(newFileName, newFileContent);
}

main()

```

Тесты

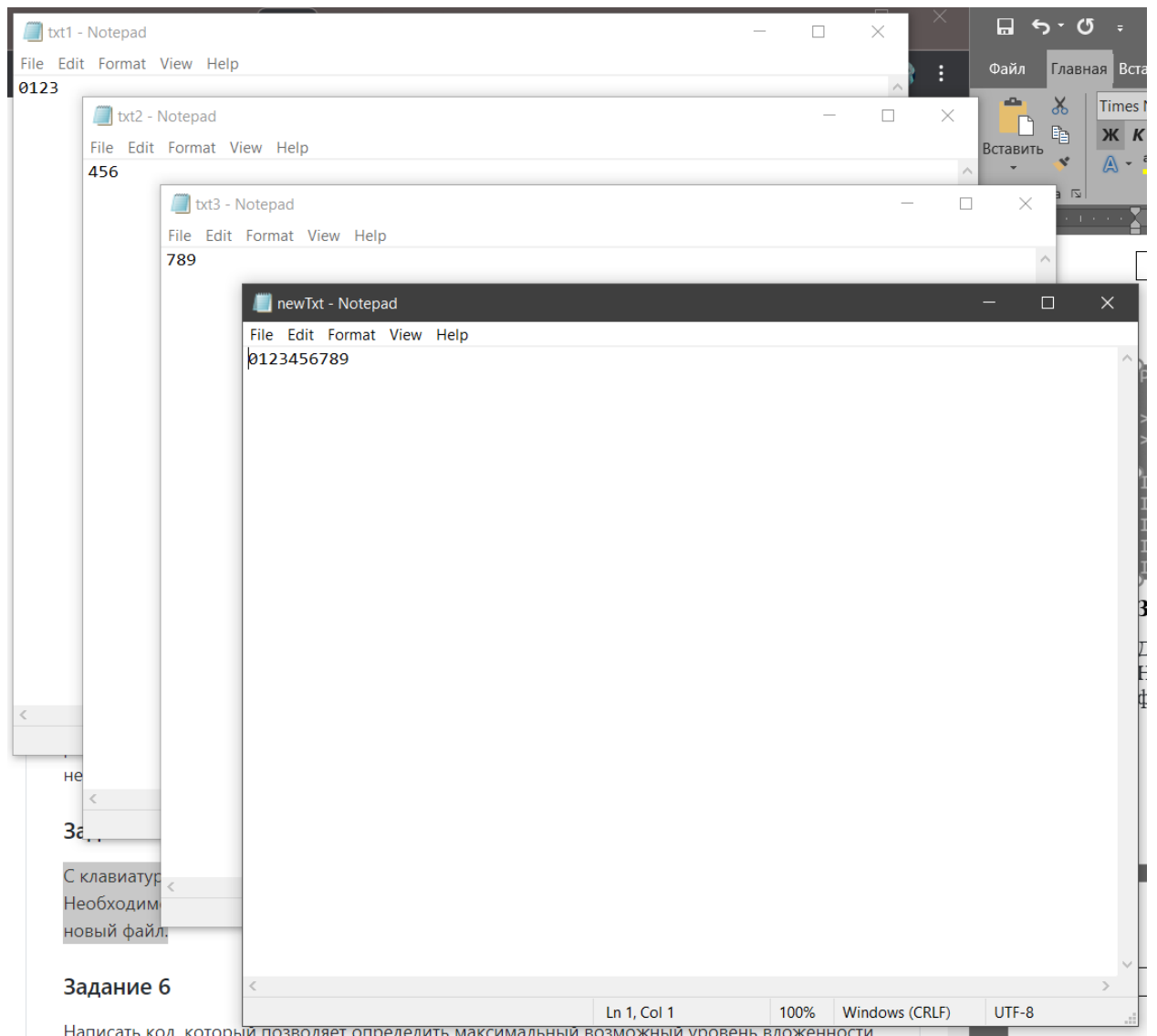
```

PS C:\Repositories\bmstu_archEvm\lab_02\task_15> npm start

> task_05@1.0.0 start C:\Repositories\bmstu_archEvm\lab_02\task_15
> node index.js

Input file count: 3
Input file name: txt1.txt
Input file name: txt2.txt
Input file name: txt3.txt
Input name for new file: newTxt.txt

```



Задание 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

```
"use strict";

function shoveObject(object) {
    let newObject = { data : object };
    return newObject;
}

function main() {
    let result = 0;
    let obj = { data : "Бесконечность не предел" };

    while (true) {
        try {
            let objJson = JSON.stringify(obj);
        }
        catch {
            console.log("Maximum possible nesting level = ", result);
            break;
        }

        result ++;
        obj = shoveObject(obj);
    }
}

main()
```

Тесты

```
PS C:\Repositories\bmstu_archEvm\lab_02\task_16> npm start

> task_06@1.0.0 start C:\Repositories\bmstu_archEvm\lab_02\task_16
> node index.js

Maximum possible nesting level = 965
```

Задание 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.

Листинг

```
"use strict";

const fs = require('fs');

const treeFile = "tree.txt";

function randInt(min, max) {
    let rand = min - 0.5 + Math.random() * (max - min + 1);
    return Math.round(rand);
}

function generateBranch() {
    const symbols = "qwertyuiopasdfghjklzxcvbnm";

    let branch = randInt(0, 2);

    let index = randInt(0, symbols.length - 1);
    let tree = { "value" : symbols[index] };

    if (branch >= 1) {
        tree["left"] = generateBranch();
    }

    if (branch == 2) {
        tree["right"] = generateBranch();
    }

    return tree;
}

function generateTree() {
    const string = JSON.stringify(generateBranch(), null, ' ');
    fs.writeFileSync(treeFile, string);
}

function parseTree() {
```

```
    let out = false;

    let content;
    let tree;

    if (fs.existsSync(treeFile)){
        content = fs.readFileSync(treeFile, "utf8");

        try {
            tree = JSON.parse(content);
            out = tree;
        }
        catch (error) {
            console.error("Exist file");
        }
    }
    else {
        console.error("File not found");
    }

    return out;
}

function getMaxTrace(tree) {

    if (!tree)
        return "";

    if (!tree["left"] && !tree["right"])
        return tree["value"];

    let left = getMaxTrace(tree["left"]);
    let right = getMaxTrace(tree["right"]);

    return tree["value"] + ((left.length > right.length) ? left : right);
}

function main() {
    generateTree();

    let tree = parseTree();

    if (tree) {
        let maxTrace = getMaxTrace(tree)

        console.log("Tree: ");
        console.log(tree);
    }
}
```

```
        console.log("Trace: " + maxTrace);
        console.log("Max depth: " + maxTrace.length);
    }
}

main();
```

Тесты

```
PS C:\Repositories\bmstu_archEvm\lab_02\task_17> npm start

> task_07@1.0.0 start C:\Repositories\bmstu_archEvm\lab_02\task_17
> node index.js

Tree:
{
  value: 'p',
  left: { value: 'l', left: { value: 'i', left: [Object] } },
  right: {
    value: 'n',
    left: { value: 'y', left: [Object], right: [Object] },
    right: { value: 'z', left: [Object], right: [Object] }
  }
}
Trace: pnzajzdgrj
Max depth: 10
```

Часть 2

Задание 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

```
"use strict";

const fs = require("fs");
const express = require("express");

class LocalServer {
  constructor(port) {
    this.port = port;
    this.app = express();
  }

  startServer() {
    try {
      this.app.listen(this.port);
      console.log("Server started on port ", this.port);
    }
    catch {
      console.log("Server startup error");
      throw new Error("Server start error");
    }

    this.app.get("/input_numbers", this.getStartPage);
    this.app.get("/find_maximum", this.getMaxNumber);
  }

  getStartPage(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("html_source/bad_page.html",
"utf8");
      response.end(contentString);
    }
  }
}
```



```

getMaxNumber(request, response) {
  const number1 = parseInt(request.query.a);
  const number2 = parseInt(request.query.b);
  const number3 = parseInt(request.query.c);

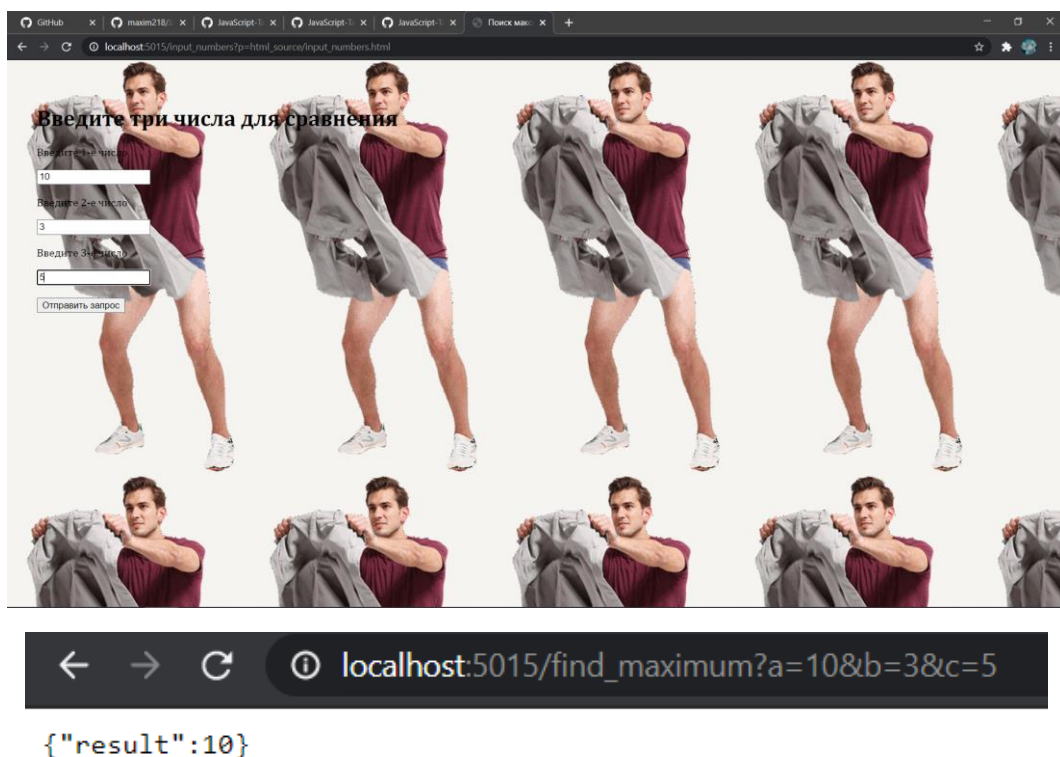
  if (isNaN(number1) || isNaN(number2) || isNaN(number3)) {
    const contentString = fs.readFileSync("html_source/nan_numbers.html
");
    response.end(contentString);
  }
  else {
    const maxNumber = Math.max(number1, number2, number3);

    const answerJSON = JSON.stringify({ result : maxNumber });
    response.end(answerJSON);
  }
}

function main() {
  let server = new LocalServer(5015);
  server.startServer();
}
main()

```

Тесты



Задание 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

```
"use strict";

const fs = require("fs");
const express = require("express");

class LocalServer {
  constructor(port) {
    this.port = port;
    this.app = express();
  }

  startServer() {
    try {
      this.app.listen(this.port);
      console.log("Server started on port ", this.port);
    }
    catch {
      console.log("Server startup error");
      throw new Error("Server start error");
    }

    this.app.get("/input_index", this.getStartPage);
    this.app.get("/out_element", this.getElement);
  }

  getStartPage(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("html_src/bad_page.html", "utf8");
      response.end(contentString);
    }
  }
}
```

```

getElement(request, response) {
    const index = parseInt(request.query.index);

    if (isNaN(index)) {
        const contentString = fs.readFileSync("html_src/nan_index.html", "utf8");
        response.end(contentString);
    }
    else {
        const array = JSON.parse(fs.readFileSync("src/arr.txt", "utf8"));

        if (index < 0 || index >= array.length) {
            const contentString = fs.readFileSync("html_src/out_of_range_index.html", "utf8");
            response.end(contentString);
        }
        else {
            const result = array[index];

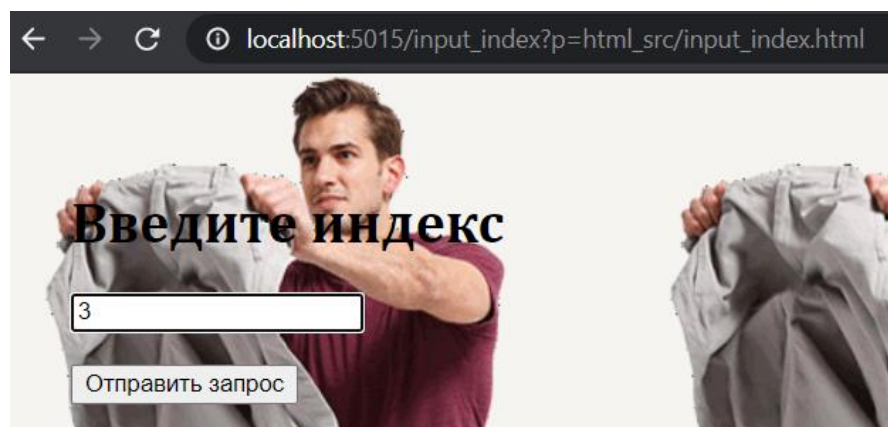
            const answerJSON = JSON.stringify({ Result : result});
            response.end(answerJSON);
        }
    }
}

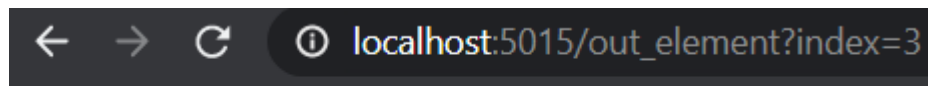
function main() {
    let server = new LocalServer(5015);
    server.startServer();
}

main()

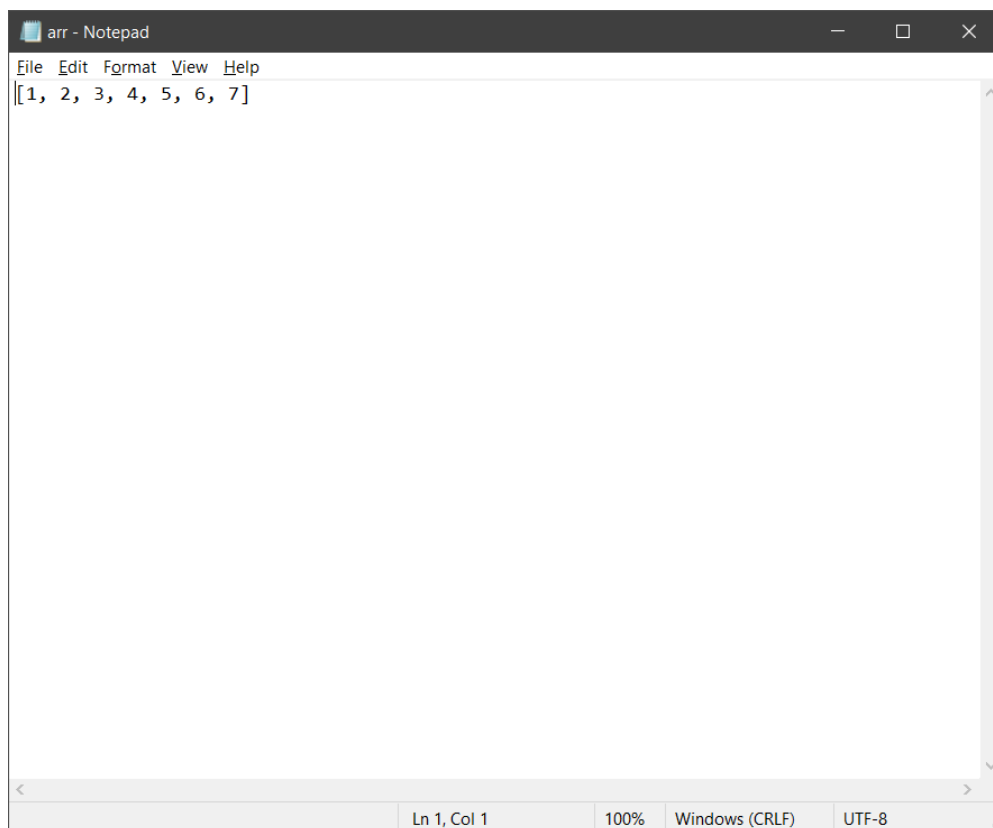
```

Тесты





```
{"Result":4}
```



Задание 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

```
"use strict";

const fs = require("fs");
const express = require("express");

const newPageName = "html_src/gen_page.html";

class PageHTML {
  initPageHead() {
    this.pageHead = '<head>\n\t<meta charset="UTF-8">\n\t<title>Сгенерированная страница</title></head>';
  }
}
```

```

    initPageFields(count, nameArray, queryAdress) {
        if (queryAdress[0] !== '/') {
            queryAdress = '/' + queryAdress;
        }

        this.pageBody = '<body>\n\t<form method="GET" action="' + queryAdress +
        '">';

        for (let i = 0; i < count; i++) {
            const fieldName = nameArray[i];

            this.pageBody += '\n\t\t' + '<p>' + 'Field ' + (i + 1) + " " + field
            dName + '</p>';
            this.pageBody += '\n\t\t' + '<input name="' + fieldName + '" spellc
            heck="false" autocomplete="off">'
        }

        this.pageBody += '\n\t\t<br>\n\t\t<br>\n\t\t<input type="submit" value=
        "Отправить запрос">';
        this.pageBody += '\n\t</form>\n</body>';
    }

    pageSave(fileName) {
        const pageData = '<!DOCTYPE html>\n<html>\n' + this.pageHead + this.pag
        eBody + '\n</html>';

        fs.writeFileSync(fileName, pageData);
        console.log(">>> HTML save successfully!");
    }
}

class Server {
    constructor(port=5015) {
        this.port = port;
        this.app = express();
    }

    startServer() {
        try {
            this.app.listen(this.port);
            console.log("Server started on port " + this.port);
        }
        catch {
            console.log("Server startup error");
            throw new Error("Server start error");
        }

        this.app.get("/startPage", this.getStartPage);
        this.app.get("/generate", this.generatePage);
    }
}

```

```

    }

    getStartPage(request, response) {
        const nameString = "html_src/start_page.html";

        if (fs.existsSync(nameString)) {
            const contentString = fs.readFileSync(nameString, "utf8");
            response.end(contentString);
        }
        else {
            const contentString = fs.readFileSync("html_src/bad_page.html", "utf8");
            response.end(contentString);
        }
    }

    generatePage(request, response) {
        const queryAddress = request.query.adress;
        const nameString = request.query.names;

        if (nameString.length < 1 || queryAddress.length < 1) {
            const contentString = fs.readFileSync("html_src/wrong_data.html", "utf8");
            response.end(contentString);
        }
        else {
            // Get names from string
            function parseNameArray(string) {
                let nameArray = [];
                let name = "";

                for (let ch of string) {
                    if (ch === ' ' && name.length > 0) {
                        nameArray.push(name);
                        name = "";
                    }
                    else {
                        name += ch;
                    }
                }

                return nameArray;
            }

            const fieldsName = parseNameArray(nameString);
            let pageHTML = new PageHTML();

            pageHTML.initPageHead();
            pageHTML.initPageFields(fieldsName.length, fieldsName, queryAddress
        );
    }

```

```

        pageHTML.pageSave(newPageName);

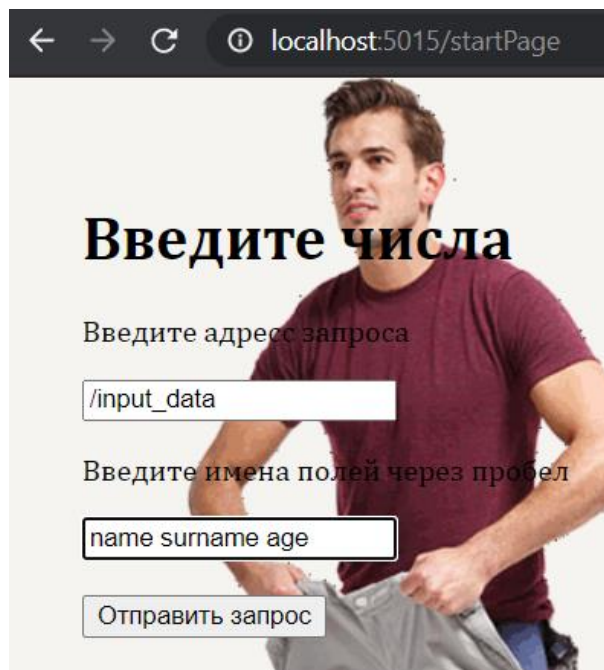
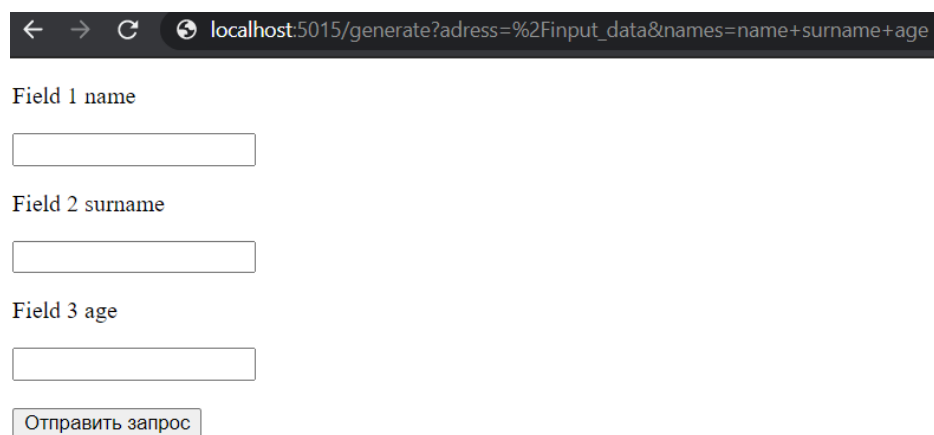
        const contentString = fs.readFileSync(newPageName, "utf8");
        response.end(contentString);
    }
}

function main() {
    let server = new Server(5015);
    server.startServer();
}

main()

```

Тесты

Задание 4

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

Листинг

```
"use strict";

const fs = require("fs");
const express = require("express");

class Server {
  constructor(port = 5015) {
    this.port = port;
    this.app = express();
  }

  startServer() {
    try {
      this.app.listen(this.port);
      console.log("Server started on port " + this.port);
    }
    catch {
      console.log("Server startup error");
      throw new Error("Server start error");
    }
    this.app.get("/input_number", this.getStartPage);
    this.app.get("/out_array", this.outArray);
  }

  getStartPage(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("html_src/bad_page.html", "utf8");
      response.end(contentString);
    }
  }

  static getArrayByRange(A, B, C) {
    const array = [];
    for (let number = A; number <= B; number++) {
```



```

        if (!(number % C)) {
            array.push(number);
        }
    }
    return array;
}

outArray(request, response) {
    const A = parseInt(request.query.a);
    const B = parseInt(request.query.b);
    const C = parseInt(request.query.c);

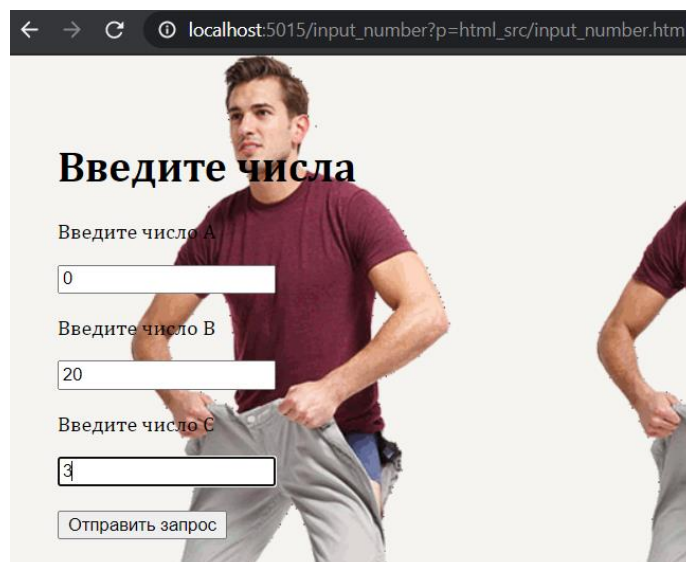
    if (isNaN(A) || isNaN(B) || isNaN(C)) {
        const contentString = fs.readFileSync("html_src/nan_numbers.html",
"utf8");
        response.end(contentString);
    }
    else if (A > B) {
        const contentString = fs.readFileSync("html_src/wrong_range.html",
"utf8");
        response.end(contentString);
    }
    else {
        const numberArray = Server.getArrayByRange(A, B, C);

        const answerJSON = JSON.stringify(numberArray);
        response.end(answerJSON);
    }
}

let server = new Server(5015);
server.startServer();

```

Тесты



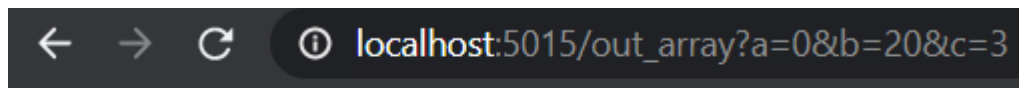
← → ↻ 🌐 localhost:5015/input_number?p=html_src/input_number.html

Введите числа

Введите число A

Введите число B

Введите число C



[0,3,6,9,12,15,18]

Вывод: в ходе выполнения лабораторной работы были изучены формат JSON и его ключевые особенности, а также фреймворк express.