|  |  |
| --- | --- |
|  | **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**  Чалый А. А.  **Группа** ИУ7 – 52 Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Попов А.Ю. |  |

Москва.

2020 г.

# Оглавление

[Оглавление 2](#_Toc1)

[Цель 3](#_Toc2)

[Task 3 4](#_Toc3)

[Задание 1 5](#_Toc4)

[Задание 2 6](#_Toc5)

[Задание 3 7](#_Toc6)

[Задание 4 8](#_Toc7)

[Задание 5 9](#_Toc8)

[Задание 6 10](#_Toc9)

[Задание 7 11](#_Toc10)

[Task 4 12](#_Toc11)

[Задание 1 13](#_Toc12)

[Задание 2 14](#_Toc13)

[Задание 3 15](#_Toc14)

[Задание 4 16](#_Toc15)

[Вывод 17](#_Toc16)

# Цель

По окончании данной работы будут приобретены такие умения как:

1. создание файла с зависимостями;
2. опыт работы с форматом JSON;
3. работа с файлами в JavaScript;
4. считывание с клавиатуры в JavaScript;
5. запуск сервера в node.js;
6. обработка get запросов.

# Task 3

## Задание 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 1. задание 1

**"use strict"**;

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

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

**const** n = readlineSync.question(**"Input N: "**);

**let** arr = [];

**for**(**let** i = 0; i < n; i++)

{

**let** str = readlineSync.question(**"Input str: "**);

**if**(str.**length** % 2 === 0) arr.push(str);

}

**const** filename = **"task3\_1.txt"**;

**let** res = ***JSON***.stringify(arr);

***fs***.*writeFileSync*(filename, res);

Листинг 2. тесты к заданию 1

**Input:**

Input N: 3

Input str: klop

Input str: op

Input str: pok

**Output:**

["klop","op"]

## 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 3. задание 2

**"use strict"**;

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

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

**function** vowel\_word(word)

{

**let** vowels = [**'A'**, **'E'**, **'O'**, **'U'**, **'Y'**, **'I'**, **'a'**, **'e'**, **'o'**, **'u'**, **'y'**, **'i'**];

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

**if** (vowels.includes(word[i]) === **false**)

**return false**

**return true**

}

**const** filename = **"task3\_1.txt"**;

**if** (***fs***.*existsSync*(filename))

{

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

**let** arr = ***JSON***.parse(filecontent);

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

**if**(vowel\_word(arr[i]) === **true**) ***console***.log(arr[i]);

}

**else**

{

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

**return** -1;

}

Листинг 4. тесты к заданию 2

**Input:**

["aoao","poka","OeeO"]

**Output:**

aoao

OeeO

## 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 5. задание 3

**"use strict"**;

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

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

**const** ext = readlineSync.question(**"Input file extension: "**);

**const** diraddr = readlineSync.question(**"Input folder address: "**);

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

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

{

**let** file = arr[i].split(**'.'**);

**let** curext = file[file.**length** - 1];

**if** (curext === ext)

{

***console***.log(**"Content of file "** + arr[i].toString());

***console***.log(***fs***.*readFileSync*(arr[i], **"utf8"**));

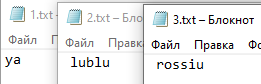
}

}

**let** res = ***JSON***.stringify(arr);

Листинг 6. тесты к заданию 3

**Input:**



**Output:**

Content of file 1.txt

ya

Content of file 2.txt

lublu

Content of file 3.txt

rossiu

## Задание 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 7. задание 4

**"use strict"**;

**function** rec\_dirs(diraddr)

{

**let** dir = ***fs***.*readdirSync*(diraddr);

***console***.log(**'Current directory '** + diraddr)

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

{

**if** (dir[i].split(**'.'**).length === 1)

{

**let** curdir = diraddr + **'/'** + dir[i];

rec\_dirs(curdir);

}

**else**

{

**let** filetext = fs.readFileSync(diraddr + **'/'** + dir[i], **"utf8"**);

**if** (filetext.length <= 10)

console.log(filetext);

}

}

}

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

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

**const** diraddr = **'./tests'**;

rec\_dirs(diraddr);

Листинг 8. тесты к заданию 4

Current directory ./tests

hdfhdfh

as

Current directory ./tests/dir 2

as

hdfhdfh

Current directory ./tests/dir 2/dir 4

as

hdfhdfh

Current directory ./tests/dir 3

as

hdfhdfh

## 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 9. задание 5

**"use strict"**;

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

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

**const** n = readlineSync.question(**"Input N: "**);

**let** arr = [];

**let** filetext = **''**;

**for**(**let** i = 0; i < n; i++)

{

**let** filename = readlineSync.question(**"Input file name: "**);

**let** text = ***fs***.*readFileSync*(filename + **'.txt'**, **"utf8"**);

filetext = filetext.concat(text);

}

**const** filename = **"task3\_5.txt"**;

***fs***.*writeFileSync*(filename, filetext);

Листинг 10. тесты к заданию 5

Input:

Input N: 3

Input file name: 1

Input file name: 2

Input file name: 3  
Output:

ya lublu rossiu

## 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 11. задание 6

**function** *checkError*(deepness, myObj)

{

**try**{

***console***.log(***JSON***.stringify(myObj));

}**catch**(err){

***console***.log(**"Error"**);

***console***.log(**"Max deepness is"**, deepness);

}

}

**let *o1*** = {**deep**: 0, **child**: **null**}

**for**(**let** i = 0;;i++)

{

**try**{

**const** obj = {**deep**: i, **child**: ***o1***}

***JSON***.stringify(obj)

***o1*** = obj

} **catch** (e) {

***console***.log(**"Max level of deepness"**);

***console***.log(i);

**break**;

}

}

Листинг 12. тесты к заданию 6

Output:

Max level of deepness

964

## 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 13. задание 7

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

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

**const** node6 = {**name**: **"6"**, **child1**: node7, **child2**: **null**};

**const** node5 = {**name**: **"5"**, **child1**: node6, **child2**: **null**};

**const** node4 = {**name**: **"4"**, **child1**: **null**, **child2**: **null**};

**const** node3 = {**name**: **"3"**, **child1**: **null**, **child2**: **null**};

**const** node2 = {**name**: **"2"**, **child1**: node4, **child2**: node5};

**const** node1 = {**name**: **"1"**, **child1**: node2, **child2**: node3};

***fs***.*writeFileSync*(**"task3\_7.txt"**, ***JSON***.stringify(node1));

**let** lobgestbranch = **""**;

**function** search(node, branch)

{

branch += node.**name**;

**if** (branch.**length** > longestWay.**length**)

lobgestbranch = branch;

**if** (node.child1 !== **null**)

{

*//console.log(node.name);*

search(node.child1, branch);

}

**if** (node.**child2** !== **null**)

{

*//console.log(node.name);*

search(node.**child2**, branch);

}

}

**const** obj = ***JSON***.parse(***fs***.*readFileSync*(**"task3\_7.txt"**, **"utf8"**));

search(obj, **""**);

***console***.log(lobgestbranch);

Листинг 14. тесты к заданию 7

Input:

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

**const** node6 = {**name**: **"6"**, **child1**: node7, **child2**: **null**};

**const** node5 = {**name**: **"5"**, **child1**: node6, **child2**: **null**};

**const** node4 = {**name**: **"4"**, **child1**: **null**, **child2**: **null**};

**const** node3 = {**name**: **"3"**, **child1**: **null**, **child2**: **null**};

**const** node2 = {**name**: **"2"**, **child1**: node4, **child2**: node5};

**const** node1 = {**name**: **"1"**, **child1**: node2, **child2**: node3};

Output:

Longest branch:

12567

# Task 4

## 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.

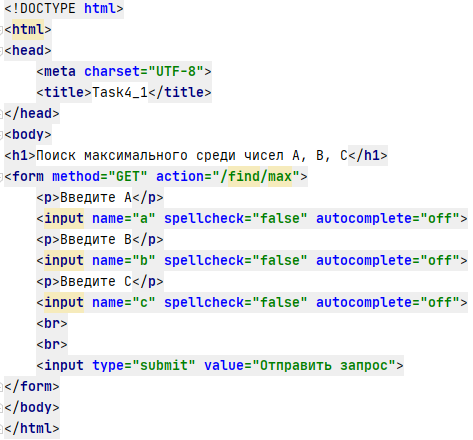


Рисунок 1. task4\_1.html

Листинг 15. задание 1

**"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** res = ***Math***.max(aInt, bInt, cInt);

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

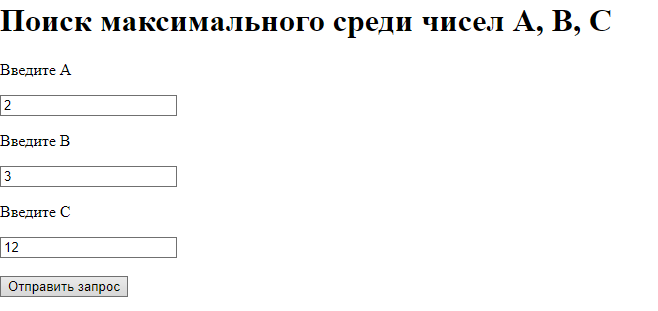
response.end(answerJSON);

});

Листинг 16. тесты к заданию 1

Input:

http://localhost:5015/me/page?p=task4\_1.html



Output:

{"result":12}

## 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.



Рисунок 2. task4\_2.html

Листинг 17. задание 2

**"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(**"/index/mas"**, **function**(request, response) {

**const** index = request.**query**.index;

**const** indInt = *parseInt*(index);

**let** res = ***fs***.*readFileSync*(**"task4\_2.txt"**, **"utf-8"**);

res = ***JSON***.parse(res);

**if** (indInt < 0 || indInt >= res.**length**)

response.end(**'Wrong index'**);

**const** resind = ***JSON***.stringify(res[indInt]);

response.end(resind);

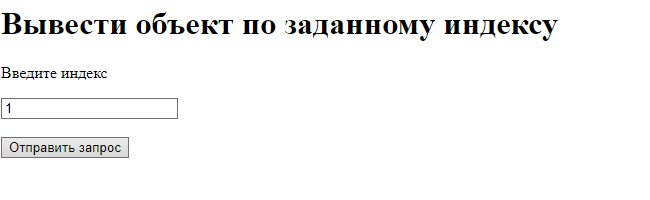
});

Листинг 18. тесты к заданию 2

Input:

http://localhost:5015/me/page?p=task4\_2.html

[{"a":4,"b":"342"},{"c":"mama","d":"papa"},{"e":"34kon","a":424},{"f":334,"g":"chpek"}]



Output:

{"c":"mama","d":"papa"}

## 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.

Рисунок 3. task4\_3.html

Листинг 19. задание 3

Листинг 20. тесты к заданию 3

## Задание 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.



Рисунок 4. task4\_4.html

Листинг 21. задание 4

**"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(**"/get/mas"**, **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);

**let** res = [];

**for** (**let** i = aInt; i <= bInt; i++)

**if** (i % cInt === 0)

res.push(i);

res = ***JSON***.stringify(res);

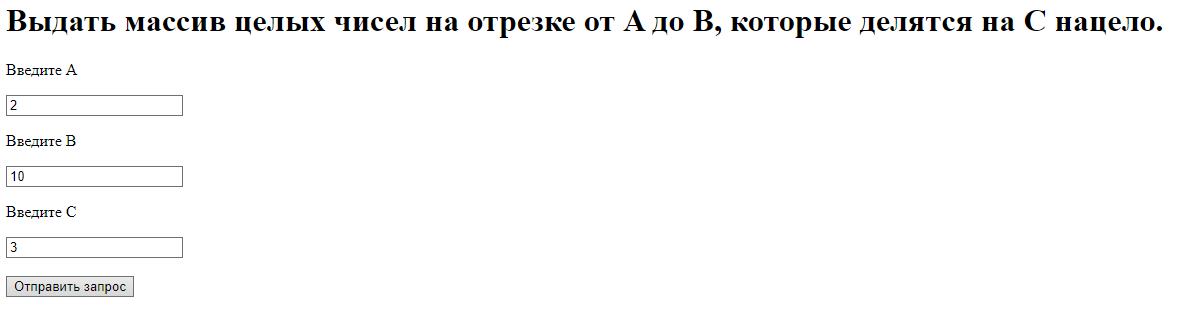
response.end(res);

});

Листинг 22. тесты к заданию 4

Input:

http://localhost:5015/me/page?p=task4\_4.html



Output:

[3,6,9]

# Вывод

Все поставленные задачи были выполнены. Цель лабораторной работы достигнута.