



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 09.03.01 Информатика и вычислительная техника

**О Т Ч Е Т**

по лабораторной работе № 2

Дисциплина: Архитектура ЭВМ

Студент

ИУ7-52Б

(Группа)

\_\_\_\_\_  
(Подпись, дата)

В.А. Иванов

(И.О. Фамилия)

Преподаватель

А.Ю. Попов

\_\_\_\_\_  
(Подпись, дата)

(И.О. Фамилия)

Москва, 2020

## **Цель работы**

Целью лабораторной работы освоение методов работы с html страницами в JavaScript.

## Задание 3.1

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.

### Программная реализация

```
"use strict";

const readlineSync = require('readline-sync');
const fs = require("fs");
const nameString = "test.txt";

const n = parseInt(readlineSync.question("Input N: "));
let data = [];

for (let i=0; i < n; i++) {
  let str = readlineSync.question("Input string: ");
  if (str.length % 2 == 0) {
    data.push(str);
  }
}

console.log(data);
let parse_data = JSON.stringify(data);
console.log(parse_data);

fs.writeFileSync(nameString, parse_data);
```

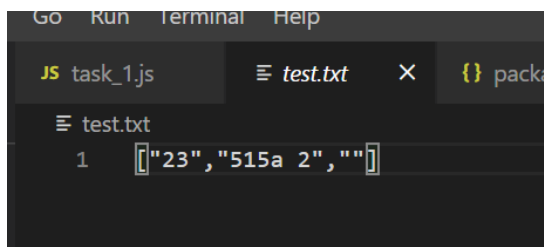
### Тесты

Ввод:

```
> node task_1.js

Input N: 5
Input string: 1
Input string: 23
Input string: 124
Input string: 515a 2
Input string:
[ '23', '515a 2', '' ]
["23","515a 2",""]
```

Вывод:



```
Go Run Terminal Help

JS task_1.js test.txt X {} packa

test.txt
1 ["23","515a 2",""]
```

## 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.

### Программная реализация

```
"use strict";

const vowels = "AEIOU"
const fs = require("fs");
const nameString = "test.txt";

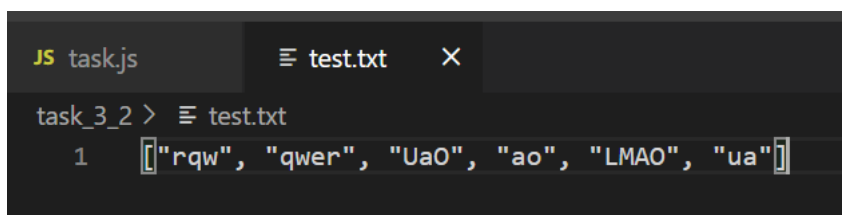
const str = fs.readFileSync(nameString, "utf-8");
console.log("Scanned string: " + str);

console.log("Vowles strings:");
let data = JSON.parse(str);

for (let i in data) {
    let str = data[i].toUpperCase();
    let j=0;
    for (; j < str.length; j++) {
        if (!(vowles.includes(str[j])))
            break;
    }
    if (j == str.length)
        console.log(data[i]);
}
```

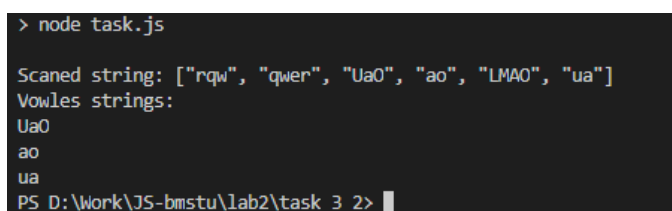
### Тесты

Ввод:



The screenshot shows a code editor with two tabs: 'task.js' and 'test.txt'. The 'test.txt' tab is active, displaying a JSON array: `["rqw", "qwer", "Ua0", "ao", "LMAO", "ua"]`. The cursor is at the end of the array.

Вывод:



The screenshot shows a terminal window with the command `> node task.js` executed. The output is: `Scanned string: ["rqw", "qwer", "Ua0", "ao", "LMAO", "ua"]`, `Vowles strings:`, `Ua0`, `ao`, `ua`. The prompt `PS D:\Work\JS-bmstu\lab2\task_3_2>` is visible at the bottom.

### 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.

#### Программная реализация

```
"use strict";

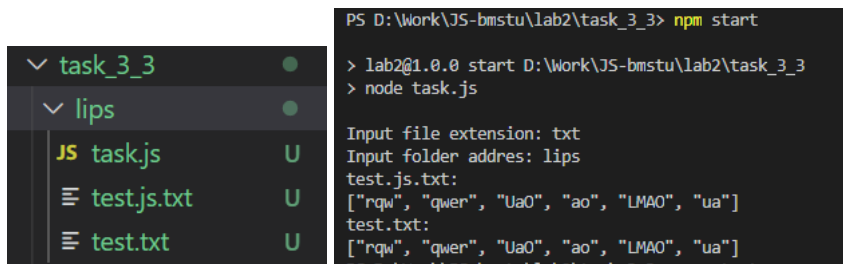
const readlineSync = require('readline-sync');
const fs = require("fs");
const ext = readlineSync.question("Input file extension: ");
const addr = readlineSync.question("Input folder address: ");

if (fs.existsSync(addr)) {
    const file_arr = fs.readdirSync(addr);

    file_arr.forEach(f_name => {
        let parts = f_name.split('.');
        if (parts[parts.length-1] == ext) {
            console.log(f_name + ":");

            let str = fs.readFileSync(addr+"\\\\"+f_name, "utf-8");
            if (str == "") {
                console.log("Empty file");
            } else {
                console.log(str);
            }
        }
    });
} else {
    console.log("Folder not exists");
}
```

#### Тесты



The image shows a file explorer on the left and a terminal window on the right. The file explorer displays a directory structure with 'task\_3\_3' expanded, showing 'lips' and 'task.js'. Below 'lips' are 'test.js.txt' and 'test.txt'. The terminal window shows the command 'npm start' and the output of the program, which prompts for 'Input file extension: txt' and 'Input folder address: lips'. It then lists the contents of 'test.js.txt' and 'test.txt', both showing an array of strings: ["rqw", "qwer", "Ua0", "ao", "LMAO", "ua"].

```
PS D:\Work\JS-bmstu\lab2\task_3_3> npm start
> lab2@1.0.0 start D:\Work\JS-bmstu\lab2\task_3_3
> node task.js

Input file extension: txt
Input folder address: lips
test.js.txt:
["rqw", "qwer", "Ua0", "ao", "LMAO", "ua"]
test.txt:
["rqw", "qwer", "Ua0", "ao", "LMAO", "ua"]
```

### Задание 3.4

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.

#### Программная реализация

```
"use strict";

const fs = require("fs");

function lookup_folder(addr){
  if (!fs.existsSync(addr)) { return; }

  const file_arr = fs.readdirSync(addr);
  file_arr.forEach(f_name => {
    let parts = f_name.split('.');

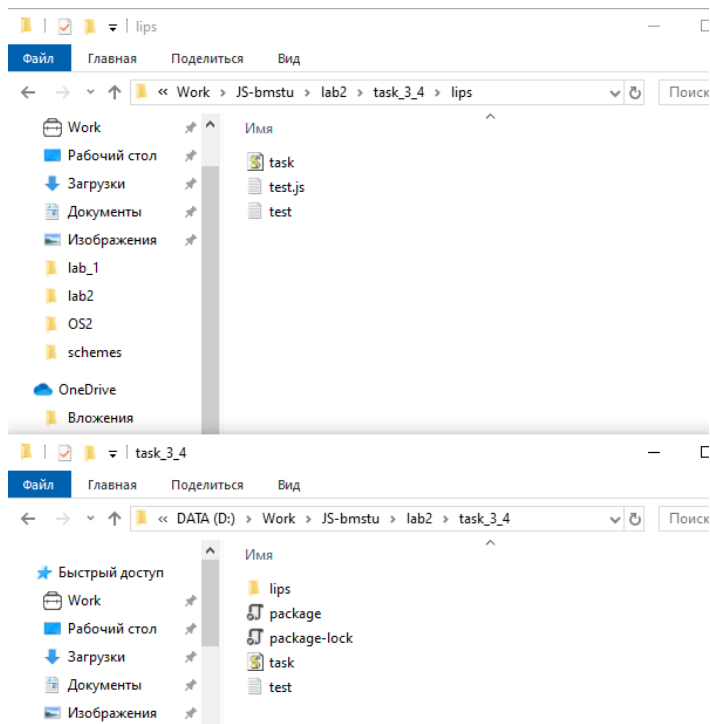
    if (parts.length == 1) {
      lookup_folder(addr+"\\ "+f_name);
    } else {
      let str = fs.readFileSync(addr+"\\ "+f_name, "utf-8");

      if (str.length <= 10) {
        console.log(addr+"\\ "+f_name);
        console.log(str);
      }
    }
  });
}

lookup_folder(".");
```

#### Тесты

Ввод:



## Вывод:

```
PS D:\Work\JS-bmstu\lab2\task_3_4> npm start

> lab2@1.0.0 start D:\Work\JS-bmstu\lab2\task_3_4
> node task.js

.\lips\task.js
"use";
.\lips\test.txt
["rqw"]
```

### 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.

#### Программная реализация

```
"use strict";

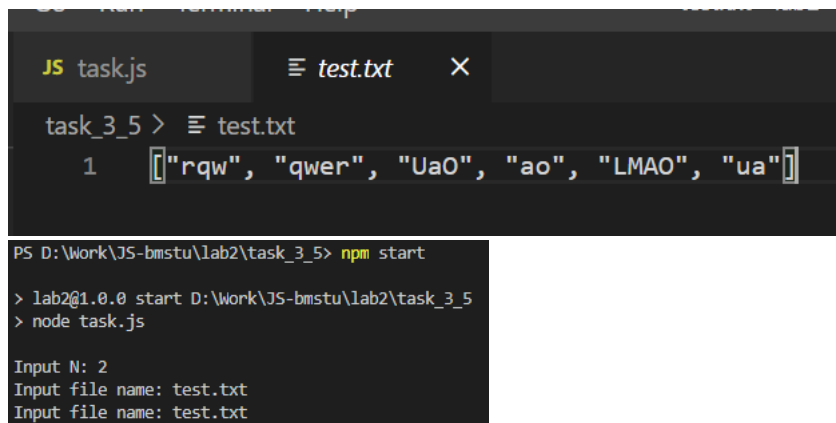
const fs = require("fs");
const readlineSync = require('readline-sync');

let result_str = "";
const n = parseInt(readlineSync.question("Input N: "));

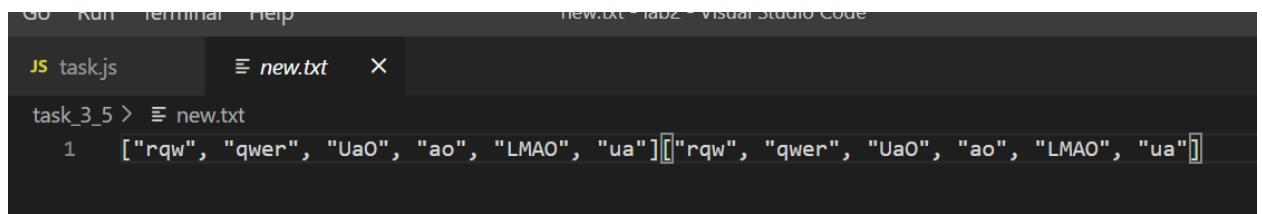
for (let i=0; i<n; i++){
    let f_name = readlineSync.question("Input file name: ");
    if (fs.existsSync(f_name)) {
        let str = fs.readFileSync(f_name, "utf-8");
        result_str += str;
    } else {
        console.log("File not exists");
    }
}
fs.writeFileSync("new.txt", result_str);
```

#### Тесты

Ввод:



Вывод:





### 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.

#### Программная реализация

```
"use strict";

const e = require("express");

class Box {
  constructor (depth) {
    this.d = depth;
    if (depth > 0)
      this.next = new Box(depth-1);
    else
      this.next = NaN;
  }
}

let size = 100;
let step = 128;
while (step > 1) {
  try {
    let b = new Box(size);
    let jsonStr = JSON.stringify(b);
    size += step;
  } catch (RangeError) {
    size -= step;
    step /= 2;
  }
}
console.log(size);
```

#### Тесты

```
PS D:\Work\JS-bmstu\lab2\task_3_6> npm start

> lab2@1.0.0 start D:\Work\JS-bmstu\lab2\task_3_6
> node task.js

964
PS D:\Work\JS-bmstu\lab2\task_3_6> █
```

### 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.

#### Программная реализация

```
"use strict";

const fs = require("fs");
const f_name = "json-data.txt";

function create_file() {
  const probability = 0.53;
  class Box {
    constructor (depth) {
      this.d = depth;
      this.leaf1 = NaN;
      this.leaf2 = NaN;
      if (Math.random() > 1 - probability)
        this.leaf1 = new Box(depth+1);
      if (Math.random() > 1 - probability)
        this.leaf2 = new Box(depth+1);
    }
  }
  let b = new Box(0);
  let jStr = JSON.stringify(b);
  console.log(jStr);
  fs.writeFileSync(f_name, jStr);
}

function path_lenght(path) {
  return (path.match(new RegExp("/", "g")) || []).length
}

function process_tree(tree) {
  let path = "";
  for (let key in tree) {
    if (typeof(tree[key]) == "object" && tree[key] != null) {
      let t_path = key + "/" + process_tree(tree[key]);
      if (path_lenght(t_path) > path_lenght(path))
        path = t_path;
    }
  }
  return path;
}
```

```
// create_file();

let jStr = fs.readFileSync(f_name, "utf-8");
console.log("Scaned JSON-data: " + jStr);

let original_tree = JSON.parse(jStr);
let path = process_tree(original_tree);
let path_l = path_length(path);

console.log("Path: " + path.slice(0, path.length-1));
console.log("Length: " + path_l);
```

## Тесты

## Ввод:

Для проверки работоспособности программы была создана функция, записывающая в файл древовидную структуру в формате JSON. В тесте использована строка:

```
{
  "d":0,"leaf1":null,"leaf2":{"d":1,"leaf1":{"d":2,"leaf1":null,"leaf2":{"d":3,"leaf1":null,"leaf2":{"d":4,"leaf1":{"d":5,"leaf1":{"d":6,"leaf1":{"d":7,"leaf1":null,"leaf2":null},"leaf2":null},"leaf2":{"d":6,"leaf1":{"d":7,"leaf1":{"d":8,"leaf1":{"d":9,"leaf1":null,"leaf2":{"d":10,"leaf1":{"d":11,"leaf1":null,"leaf2":null},"leaf2":null}}},"leaf2":{"d":9,"leaf1":null,"leaf2":null}}},"leaf2":null},"leaf2":null}}},"leaf2":{"d":5,"leaf1":null,"leaf2":null}}}},"leaf2":null}}
```

## Вывод:

```
Scanned JSON-data: {"d":0,"leaf1":null,"leaf2":{"d":1,"leaf1":{"d":2,"leaf1":null,"leaf2":{"d":3,"leaf1":null}, "leaf2":{"d":4,"leaf1":null}}, "leaf2":{"d":6,"leaf1":{"d":7,"leaf1":{"d":8,"leaf1":{"d":9,"leaf1":null,"leaf2":null}}}, "leaf2":null}}, "leaf2":{"d":5,"leaf1":null}}
```

Path: leaf2/leaf1/leaf2/leaf2/leaf1/leaf2/leaf1/leaf1/leaf1/leaf2/leaf1

Length: 11

### Задание 3.3

#### **Программная реализация**

#### **Тесты**

Ввод:

Вывод:

### Задание 3.3

#### **Программная реализация**

#### **Тесты**

Ввод:

Вывод:

### Задание 3.3

#### **Программная реализация**

#### **Тесты**

Ввод:

Вывод:

### Задание 3.3

#### **Программная реализация**

#### **Тесты**

Ввод:

Вывод:

## **Вывод**

В рамках лабораторной работы было выполнено ознакомление и практическое закрепление основ работы с JSON, потоками ввода/вывода, управления серверами в языке JS.