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»

ОТЧЕТ

По лабораторной работе №2

По курсу: «Архитектура ЭВМ»

Студент: Ле Ни Куанг

Группа: ИУ7и-56Б

Преподаватель: Попов А. Ю.

Москва

2020

Цель работы

Познакомить файл с зависимостями, понять, как работать с форматом файла json, запустить сервер, сделать запрос на получение на сервер и познакомиться с html и формами.

Задача 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.

```
import fs from 'fs'
import readlineSync from 'readline-sync'

function isEvenLength(s) {
    return s.length && !(s.length % 2)
}

const n = parseInt(readlineSync.question('Input n: '))
let arr = []

for (let i = 0; i < n; i++) {
    let s = readlineSync.question()
    if (isEvenLength(s))
        arr.push(s)
}

const arrStr = JSON.stringify(arr, null, 2)

fs.writeFileSync('./text/1_out.json', arrStr)
```

Тест 1

```
5/EVM/3 is 📦 v1.0.0 via ⬢ v14.11.0 took 7
s
➜ node 1
Input n: 5
abcdef
abcdefg
12
12345
abcd1234
```

```
5/EVM/3 is 📦 v1.0.0 via ⬢ v14.11.0 took 5
1s
➜ cat text/1_out.json
[
  "abcdef",
  "12",
  "abcd1234"
]
```

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.

```
import { readFileSync} from 'fs'

const contentStr = readFileSync('./text/2_in.json', 'utf8')
let content = JSON.parse(contentStr)

class FilterSymbol {
    constructor(symbols) {
        this.symbols = {}
        for (let i of symbols)
            this.symbols[i] = true
    }

    isPass(str) {
        str = str.toLowerCase()
        for (let i of str)
            if (!this.symbols[i])
                return false
        return true
    }
}

const filterVowel = new FilterSymbol('aeuoi')

content.filter(s => filterVowel.isPass(s)).forEach(s => console.log(s))
```

Тест 2

```
5/EVM/3 is 📦 v1.0.0 via ⬢ v14.11.0
➜ cat text/2 in.json
[
    "abcd",
    "aeiou",
    "AaEeUuOoIi",
    "sDafa"
]%

5/EVM/3 is 📦 v1.0.0 via ⬢ v14.11.0
➜ node 2
aeiou
AaEeUuOoIi
```

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.

```
import fs from 'fs'
import readlineSync from 'readline-sync'

let ext = readlineSync.question('File extension: ')
if (ext[0] !== '.')
    ext = '.' + ext

const dir = readlineSync.question('Foder directory: ')

if (!fs.existsSync(dir)) {
    console.log('File or folder not found!')
    process.exit()
}

if (!fs.statSync(dir).isDirectory()) {
    console.log(dir + ' is not a folder!')
    process.exit()
}


const arr = fs.readdirSync(dir).filter(name => fs.statSync(name).isFile())
const filterFile = arr.filter(f => f.endsWith(ext))

console.log(filterFile)
console.log('All file in ' + dir + ':', arr)
```

Тест 3

```
5/EVM/3 is 📦 v1.0.0 via ⬢ v14.11.0 took 1
0s
➜ node 3
File extension: js
Foder directory: .
[
  '1.js', '2.js',
  '3.js', '4.js',
  '5.js', '6.js',
  '7.js'
]
```

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.

```
import fs from 'fs'
import readlineSync from 'readline-sync'

const dir = readlineSync.question('Foder directory: ')

if (!fs.existsSync(dir)) {
    console.log('File or folder not found!')
    process.exit()
}

function fileExtFilter(f, ext='.txt') {
    return f.endsWith(ext)
}

function fileLenFilter(f, len=10) {
    const contentStr = fs.readFileSync(f)
    return contentStr.length <= len
}

function getFiles(dir) {
    let res = []
    let arr = fs.readdirSync(dir)

    for (let name of arr) {
        name = dir + '/' + name
        if (fs.statSync(name).isDirectory()) {
            res = res.concat(getFiles(name))
        } else {
            res.push(name)
        }
    }
    return res
}

const allFile = getFiles(dir)

const fileFiltered = allFile.filter(f => fileExtFilter(f))
                            .filter(f => fileLenFilter(f))

console.log('All files:', allFile)
console.log('Filtered files:', fileFiltered)
```

Тест 4

```
5/EVM/3 is 📦 v1.0.0 via ⬢ v14.11.0 took 3s
➜ node 4
Foder directory: .
All files: [
  './1.js',              './2.js',
  './3.js',              './4.js',
  './5.js',              './6.js',
  './7.js',              './package-lock.json',
  './package.json',      './text/1_out.json',
  './text/2_in.json',    './text/4/1.txt',
  './text/4/2.txt',      './text/4/4_1/1.txt',
  './text/4/4_2/1.txt', './text/4/4_2/2.txt',
  './text/4/4_2/3.txt', './text/5_out.txt',
  './text/7_in.json',    './text/7_out.json'
]

Filtered files: [
  './text/4/2.txt',
  './text/4/4_1/1.txt',
  './text/4/4_2/1.txt',
  './text/4/4_2/3.txt'
]
```

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.

```
import fs from 'fs'
import readlineSync from 'readline-sync'

const n = parseInt(readlineSync.question('Input n: '))
let arr = []

for (let i = 0; i < n; i++) {
    let s = readlineSync.question()
    if (fs.existsSync(s) && fs.statSync(s).isFile()) {
        arr.push(s)
    } else {
        console.log('File not found!')
    }
}

let bigContent

for (let i in arr) {
    bigContent += fs.readFileSync(arr[i])
}

fs.writeFileSync('text/5_out.txt', bigContent)
```

Тест 5

```
5/EVM/3 is 📦 v1.0.0 via ⬢ v14.11.0 took 2s

→ node 5
Input n: 2
package.json
6.js
```

```
undefined{
  "name": "3",
  "version": "1.0.0",
  "description": "",
  "main": "3.js",
  "scripts": {
    "start": "node 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "fs": "0.0.1-security",
    "readline-sync": "^1.4.10"
  },
  "type": "module",
  "devDependencies": {}
}
let i = 1
let obj = {0:{}}
let ref = obj[0]
let objStr

try {
    while (true) {
        ref = ref[0] = {0:{}}
        i++
        objStr = JSON.stringify(obj);
    }
} catch (err) {
    // console.log(objStr)
    console.log('Max nested level:', i)
    console.error(err.message);
}
```

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.

```
let i = 1
let obj = {0:{}}
let ref = obj[0]
let objStr

try {
    while (true) {
        ref = ref[0] = {0:{}}
        i++
        objStr = JSON.stringify(obj);
    }
} catch (err) {
    // console.log(objStr)
    console.log('Max nested level:', i)
    console.error(err.message);
}
```

Тест 6

```
5/EVM/3 is 📦 v1.0.0 via ⬢ v14.11.0 took 5s
➜ node 6
Max nested level: 2683
Maximum call stack size exceeded
```

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.

```
import fs from 'fs'
import path from 'path'

function dirTree(name) {
    let info = {
        path: name,
        name: path.basename(name)
    }

    if (fs.lstatSync(name).isDirectory()) {
        info.type = 'folder'
        info.childs = fs.readdirSync(name).map(child =>
            dirTree(name + '/' + child)
        )
    } else {
        info.type = 'file'
    }
    return info
}

// fs.writeFileSync('text/7_in.json',
   JSON.stringify(dirTree('/home/ql/snap'), null, 2))


function getDepht(obj) {
    let maxdepth = {
        depth: 1,
        branch: obj
    }

    const isArray = Array.isArray(obj)

    let key
    for (key in obj) {
        if (typeof(obj[key]) === 'object') {
            let mdepth = getDepht(obj[key])

            if (mdepth.depth >= maxdepth.depth) {
                maxdepth.depth = mdepth.depth + 1
```

```
                if (isArray) {
                    maxdepth.branch = [mdepth.branch]
                } else {
                    maxdepth.branch = {}
                    maxdepth.branch[key] = mdepth.branch
                }
            }
        }
    }

    return maxdepth
}
```

Тест 7

```
let obj = JSON.parse(fs.readFileSync('text/7_in.json'))

let maxdepth = getDepht(obj)

console.log(maxdepth.depth)

const objStr = JSON.stringify(getDepht(obj), null, 1)

fs.writeFileSync('text/7_out.json', objStr)
```

```
5/EVM/3 is 📦 v1.0.0 via ⬢ v14.11.0 took 3s
➜ node 7
43
```

Задача 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.

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.

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.

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.

Листинг 1: File index.js

```
import fs from 'fs'
import express from 'express'
import { sum, f4 } from './func.js'
import { buildHtml } from './3.js'

const app = express()
const port = 5015
app.listen(port)

console.log('Server running on: http://localhost:' + port)

let cache = {}

app.get('/', (request, response) => {
    let page = request.query.p

    if (!page || !fs.existsSync('html/' + page)) {
        page = 'index.html'
    }

    if (!cache[page])
        cache[page] = fs.readFileSync('html/' + page, 'utf8')
    response.end(cache[page])
})

app.get('/sum3', (request, response) => {
    const a = parseFloat(request.query.a)
    const b = parseFloat(request.query.b)
    const c = parseFloat(request.query.c)
    const s = sum(a, b, c)
    response.end(JSON.stringify({sum: s}))
})


app.get('/find', (request, response) => {
    const i = parseInt(request.query.i)

    if (cache['data'] === undefined) {
        cache['data'] = JSON.parse(fs.readFileSync('data/mock_data.json',
            'utf8'))
    }

    if (i > 0 && i <= cache['data'].length) {
        const answerJSON = JSON.stringify(cache['data'][i-1])
        response.end(answerJSON)
    }
    response.end()
```

```
})


app.get('/htmlbuilder', (request, response) => {
    console.log(request.query)
    response.end(buildHtml(request.query))
})


app.get('/f4', (request, response) => {
    const a = parseInt(request.query.a)
    const b = parseInt(request.query.b)
    const c = parseInt(request.query.c)
    const arr = f4(a, b, c)
```

Листинг 2: File 3.js

```
import fs from 'fs'

const template = fs.readFileSync('html/template.html', 'utf-8')

function buildInput(name) {
    return `<p>Input %name%:</p>
    <input name="%name%" type="text">"
`.replace("%name%", name).replace("%name%", name)
}
```

```
export function buildHtml(query, title="Title") {
    let res
    res = template.replace('%title%', title)
    res = res.replace('%action%', query.action)
            .replace('%action%', query.action)

    let input = ''

    let field = query.field
    if (!Array.isArray(field))
        field = [field]

    for (let i in field) {
        if (field[i].length)
            input += buildInput(field[i])
    }

    res = res.replace('%input%', input)

    console.log(res)
    return res
}
```

Листинг 3: File func.js

```
export function sum() {
    let s = 0
    for (let i of arguments) {
        s += i
    }
    return s
}

export function f4(a, b, c) {
    let arr = []
    for (let i = a; i <= b; i++) {
        if (i % c === 0)
            arr.push(i)
    }
    return arr
}
```

Листинг 4: File html/index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Lab 2-2</title>
</head>
<body>
    <h1>Lab 2 - 2</h1>
    <a href="?p=1.html">Task 1</a><br>
    <a href="?p=2.html">Task 2</a><br>
    <a href="?p=3.html">Task 3</a><br>
    <a href="?p=4.html">Task 4</a><br>
</body>
</html>
```

Листинг 5: File html/1.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Lab 2-2 - Task 1</title>
</head>
<body>
    <form method="GET" action="/sum3">
        <p>Введите A</p>
        <input name="a" type="number">
        <p>Введите B</p>
        <input name="b" type="number">
```

```
        <p>Введите C</p>
        <input name="c" type="number">
        <br><br>
        <input type="submit" value="Отправить␣запрос">
    </form>
</body>
</html>
```

Листинг 6: File html/2.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Lab 2-2 - Task 2</title>
</head>
<body>
    <form method="GET" action="/find">
        <p>Input index</p>
        <input name="i" type="number">
        <br>
        <input type="submit" value="Отправить␣запрос">
    </form>
</body>
</html>
```

Листинг 7: File html/3.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Lab 2-2 - Task 3</title>
</head>

<body>
    <form method="GET" action="/htmlbuilder">
        <input name="action" type="text" placeholder="Action">
        <br><br>
        <table id="TB">
            <tr>
            <td><input name="field" type="text" placeholder="field"></td>
            </tr>
        </table>
        <br>
        <input type="button" value="Add␣Row" onclick="addRow()">
        <input type="submit" value="Build␣Html">
    </form>
```

```
22
23      <script language="javascript">
24          function addRow() {
25              let table = document.getElementById("TB")
26
27              let rowCount = table.rows.length
28              let row = table.insertRow(rowCount)
29              let cell = row.insertCell(0)
30              let elem = document.createElement("input")
31              elem.type = "text"
32              elem.name = "field"
33              elem.placeholder = "field"
34              cell.appendChild(elem)
35          }
36      </script>
37  </body>
38
39  </html>
```

Листинг 8: File html/4.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Lab 2-2 - Task 4</title>
6  </head>
7  <body>
8      <form method="GET" action="/f4">
9          <p>Введите A</p>
10         <input name="a" type="number">
11         <p>Введите B</p>
12         <input name="b" type="number">
13         <p>Введите C</p>
14         <input name="c" type="number">
15         <br><br>
16         <input type="submit" value="Отправить␣запрос">
17     </form>
18 </body>
19 </html>
```

Листинг 9: File html/template.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>%title%</title>
6  </head>
```

```
<body>
    <form method="GET" action="%action%">
        <h3>Get request to %action%</h3>
        %input%
        <br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Тест 1

Введите A

4

Введите B

7

Введите C

8

Отправить запрос

```
{"sum":19}
```

Тест 2

Input index

6

Отправить запрос

```
{"id":6,"first_name":"Davita","last_name":"Minget","email":"
dminget5@washingtonpost.com","gender":"Female","ip_address":
"37.130.235.41"}
```

Тест 3

http://youtube.com/watch

v

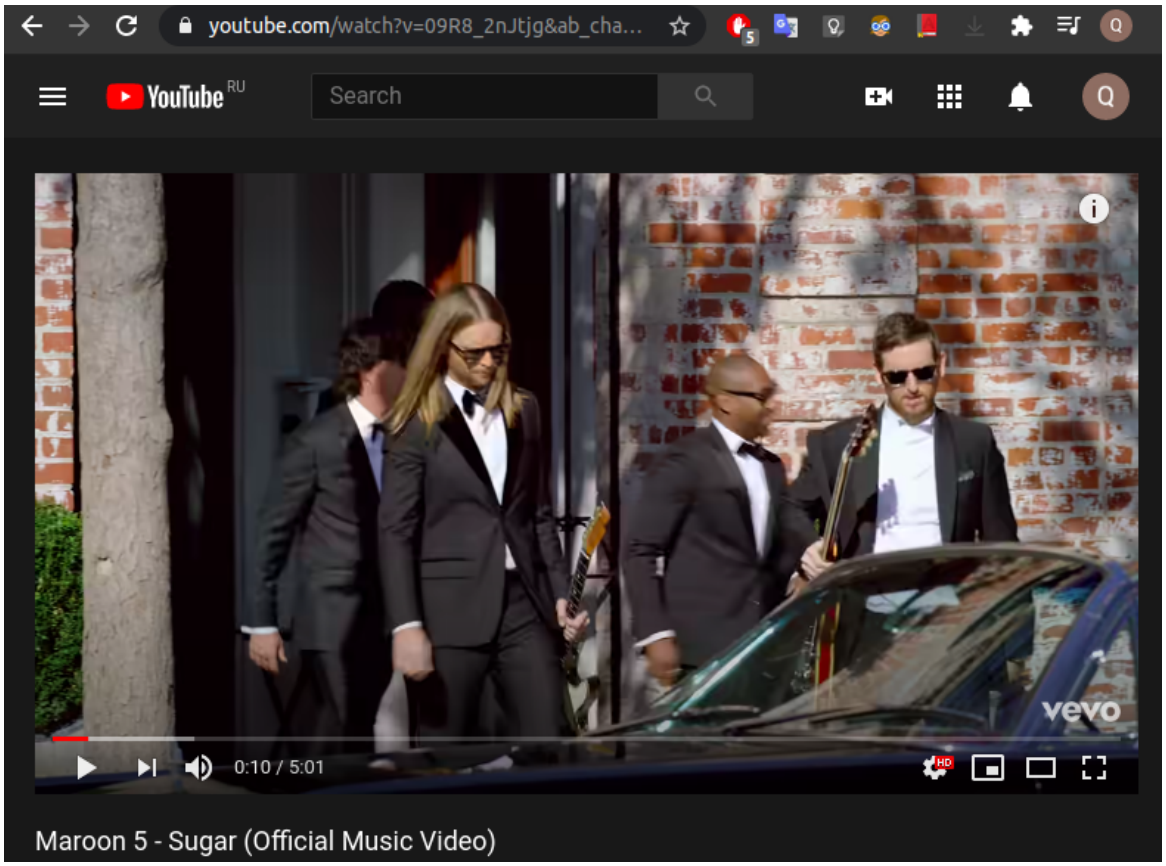
Add Row Build Html

Get request to http://youtube.com/watch

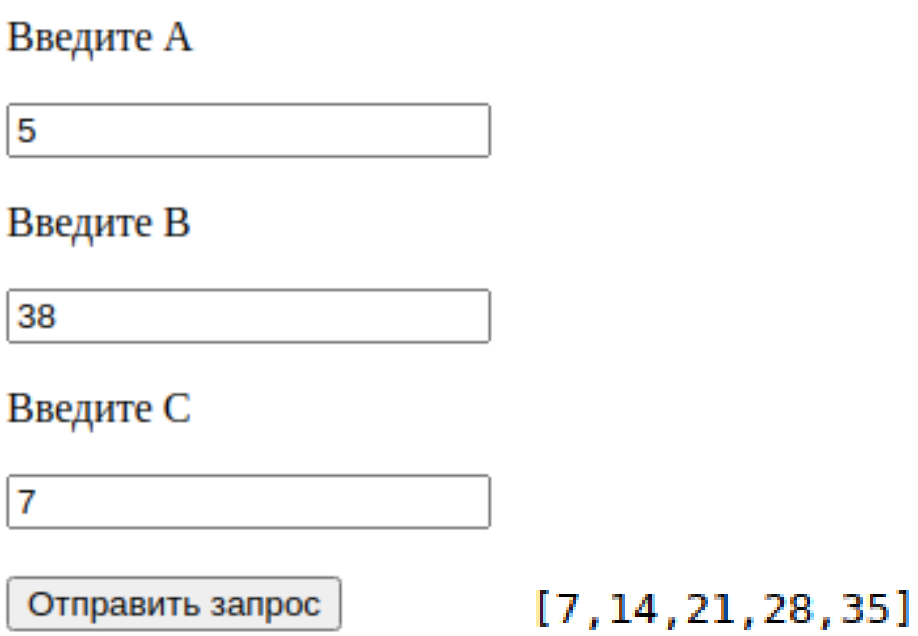
Input v:

09R8_2nJtjg"

Submit



Тест 4



Вывод

При выполнении лабораторной работы ознакомился с файлом зависимостей, новым форматом файлов json, научился работать с файлами и потоком ввода-вывода, научился запускать простой сервер, немного изучил html и форму.