|  |  |
| --- | --- |
| **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»

**Лабораторная работа №2**

**по дисциплине «Архитектура ЭВМ»**

|  |  |
| --- | --- |
| **Студент:** Садулаева Т. Р.  **Группа:** ИУ7-54Б  **Преподаватели:** Попов А.Ю., Колотовкин М.И., Куклина Н.И.  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |

### **Цель работы**

### Изучение возможностей работы с JSON, создание HTTP серверов с использованием средств языка JavaScript.

### **Подзадача 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.

**Реализация**

"use strict";

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

const fs = require('fs');

function isEven(n) {

return n % 2 === 0

}

function main() {

const arr = []

let count = readlineSync.question("Input lines count: ");

for (let i = 0; i < count; i++) {

let line = readlineSync.question("Input line: ");

if (isEven(line.length)) {

arr.push(line)

}

}

const jsonArr = JSON.stringify(arr)

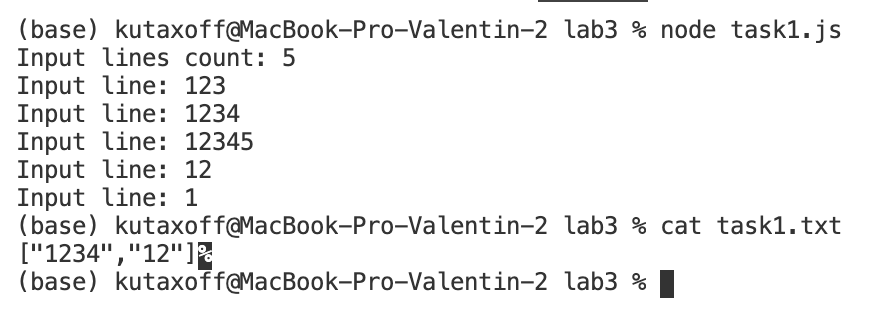
const filename = "task1.txt"

fs.writeFileSync(filename, jsonArr)

}

main()

**Проверка работоспособности**



### **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.

**Реализация**

"use strict";

function print(text) {

console.log(text)

}

function isVowelEng(letter) {

let vowelLetters = ['a', 'e', 'i', 'o', 'u']

return vowelLetters.indexOf(letter.toLowerCase()) !== -1

}

function isVowelRus(letter) {

let vowelLetters = ['а', 'о', 'э', 'е', 'и', 'ы', 'у', 'ё', 'ю', 'я']

return vowelLetters.indexOf(letter.toLowerCase()) !== -1

}

function isVowelEngRus(letter) {

return isVowelEng(letter) || isVowelRus(letter)

}

function isEverySymbolMatches(str, propertyFunc) {

let flag = true

for (let i = 0; i < str.length && flag; i++) {

if (!propertyFunc(str[i])) {

flag = false

}

}

return flag

}

const fs = require('fs');

function main() {

const filename = "task2.txt"

const contentString = fs.readFileSync(filename, "utf8")

const contentArr = JSON.parse(contentString)

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

if (isEverySymbolMatches(contentArr[i], isVowelEngRus)) {

print(contentArr[i])

}

}

}

main()

**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.

**Реализация**

"use strict";

const fs = require('fs');

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

function print(text) {

console.log(text)

}

function main() {

let extension = readlineSync.question("Input file extention (without dot): ");

let dirPath = readlineSync.question("Input directory path: ");

const files = fs.readdirSync(dirPath)

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

if (files[i].indexOf("." + extension) === files[i].length - extension.length - 1){

let content = fs.readFileSync(dirPath + "/" + files[i], "utf8")

print("-")

print(files[i])

print("-\n\n")

print(content)

print("\n\n")

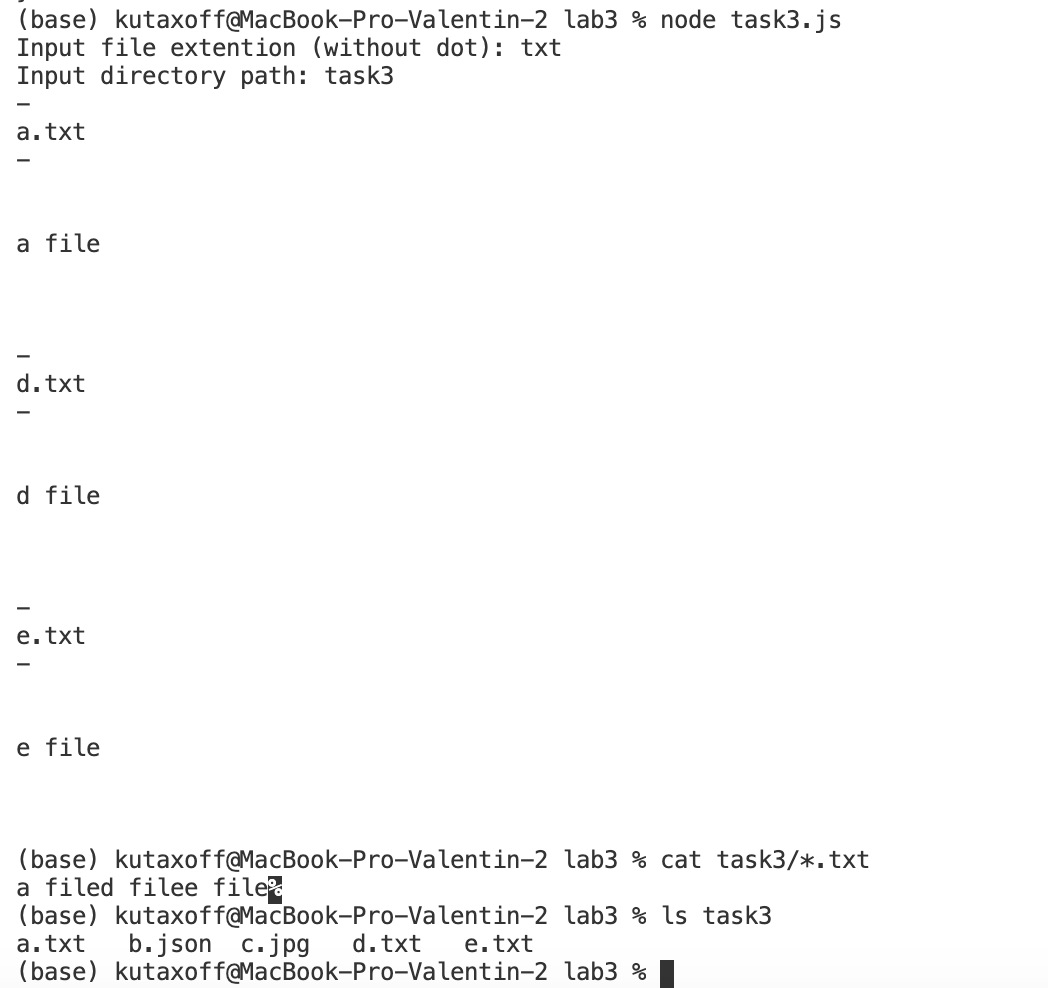
}

}

}

main()

**Проверка работоспособности**

****

**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.

**Реализация**

"use strict";

const fs = require('fs');

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

const path = require('path')

function print(text) {

console.log(text)

}

function printAlltxt(dir) {

const rootContent = fs.readdirSync(dir)

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

if (path.extname(rootContent[i]) === ".txt") {

const fileContent = fs.readFileSync(dir + "/" + rootContent[i])

if (fileContent.length <= 10) {

print(rootContent[i])

}

}

else {

let elemPath = path.join(dir, rootContent[i])

let stat = fs.statSync(elemPath)

if (stat.isDirectory()) {

printAlltxt(elemPath)

}

}

}

}

function main() {

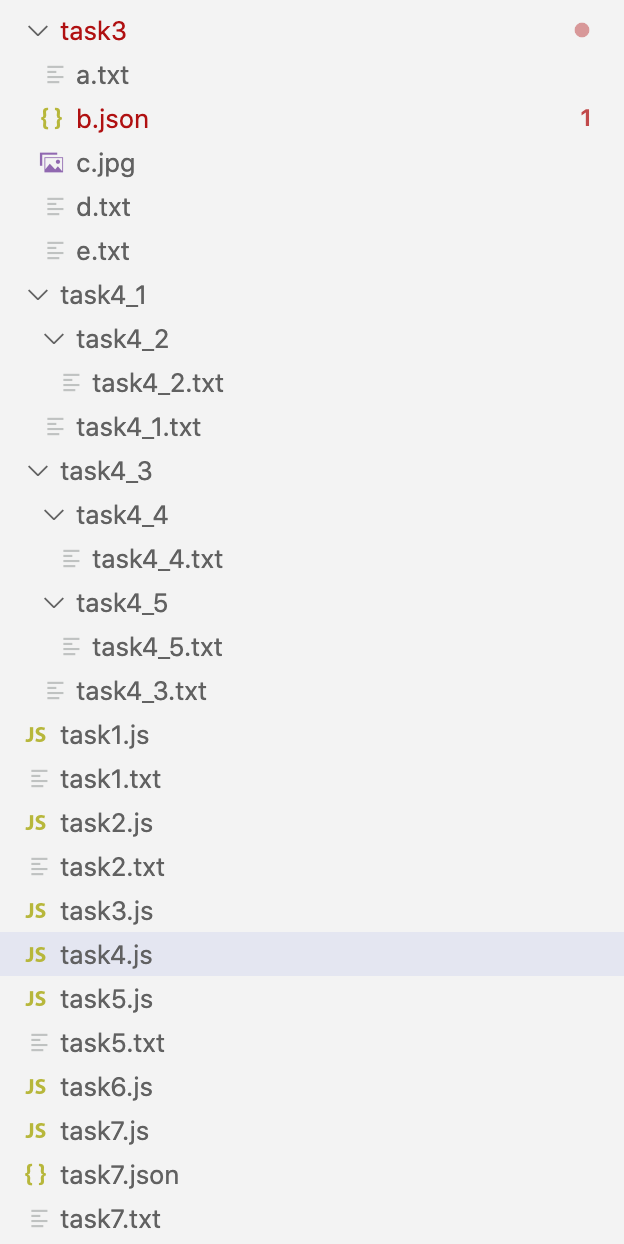
const rootDir = readlineSync.question("Input root directory: ")

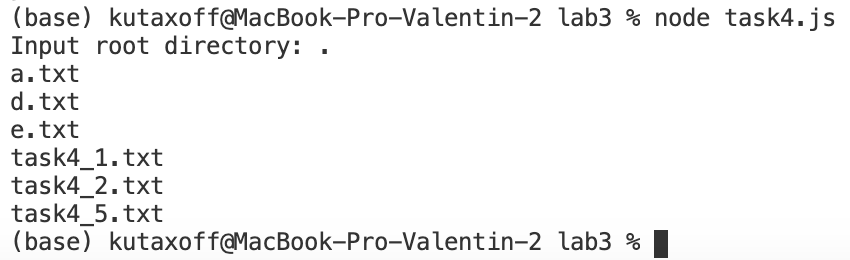
printAlltxt(rootDir)

}

main()

**Проверка работоспособности**

****

****

**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.

**Реализация**

"use strict";

const fs = require('fs');

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

function print(text) {

console.log(text)

}

function main() {

let contentsString = ""

const count = readlineSync.question("Input files count: ")

for (let i = 0; i < count; i++) {

let filename = readlineSync.question("Input filename: ")

contentsString += fs.readFileSync(filename, "utf8") + '\n'

}

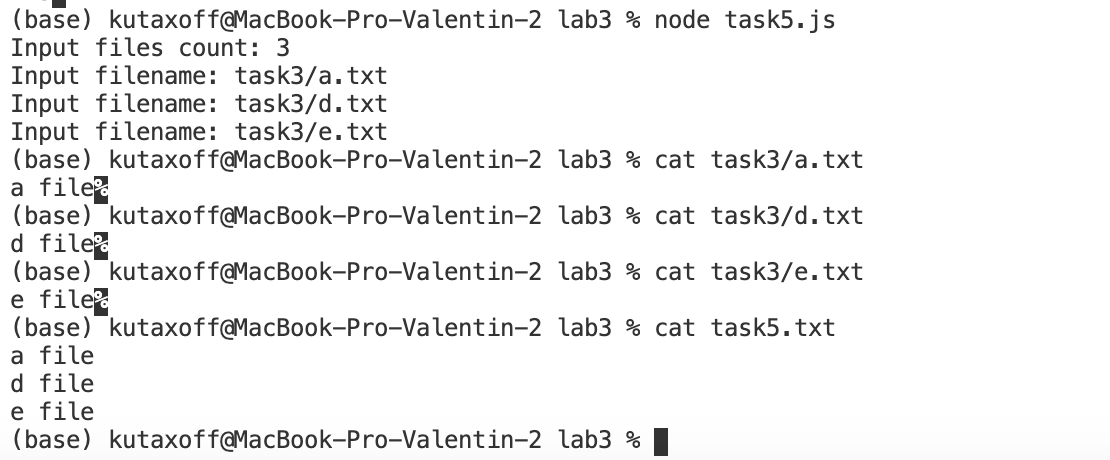
const outputFileName = "task5.txt"

fs.writeFileSync(outputFileName, contentsString)

}

main()

**Проверка работоспособности**



**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.

**Реализация**

"use strict";

function print(text) {

console.log(text)

}

function main() {

const obj = {}

let A = obj

let count = 0

let flag = true

while (flag) {

try {

const jsonString = JSON.stringify(obj);

} catch (err) {

print("Максимальная вложенность: " + count);

flag = false

}

if (count % 1000 === 0) {

print("Текущая вложенность: " + count)

}

A.field = {}

A = A.field

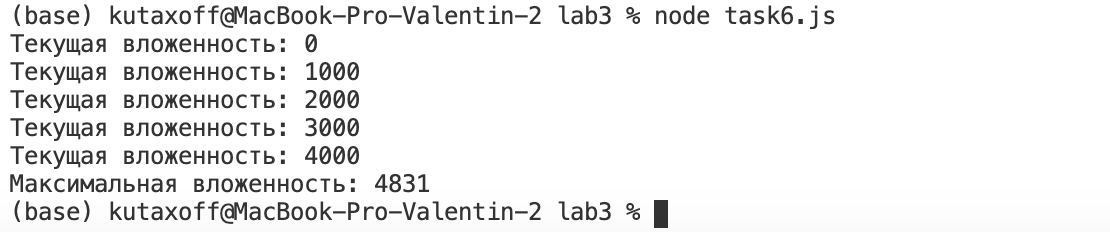
count++

}

}

main()

**Проверка работоспособности**



### **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.

**Реализация**

"use strict";

const fs = require('fs');

function print(text) {

console.log(text)

}

function getDeepestBranch(obj) {

let deepestBranch = {}

let deepestKey = null

let maxLevel = 0

if (obj === null || typeof obj != 'object') {

return { "branch": obj, "level": 0 }

}

for (let key in obj) {

const { branch, level } = getDeepestBranch(obj[key])

if (level + 1 > maxLevel) {

deepestKey = key

deepestBranch = branch

maxLevel = level + 1

}

}

return { "branch": {[deepestKey] : deepestBranch}, "level": maxLevel }

}

function main() {

const stringJSON = fs.readFileSync("task7.json", "utf-8")

const obj = JSON.parse(stringJSON)

let res = getDeepestBranch(obj)

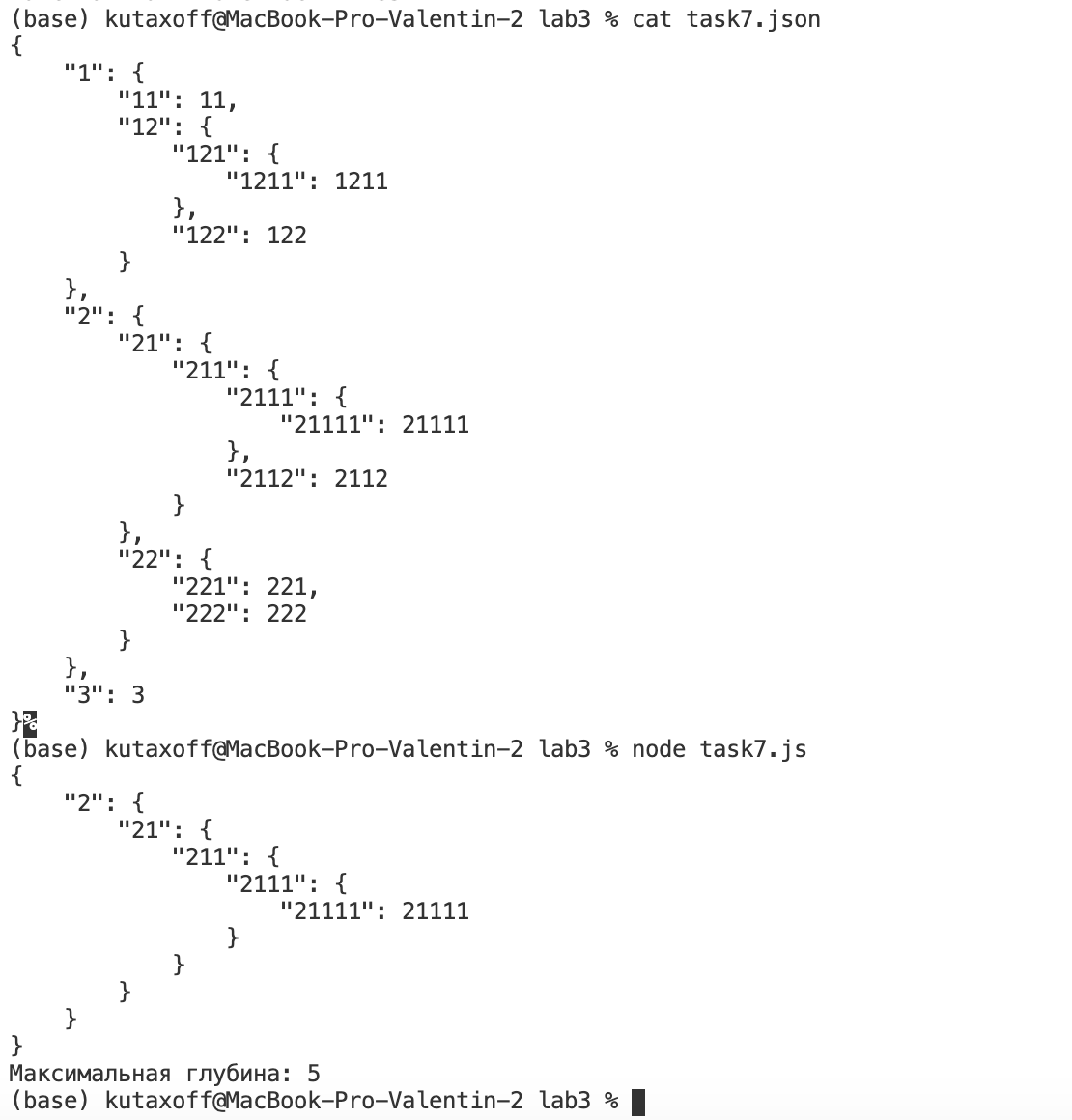
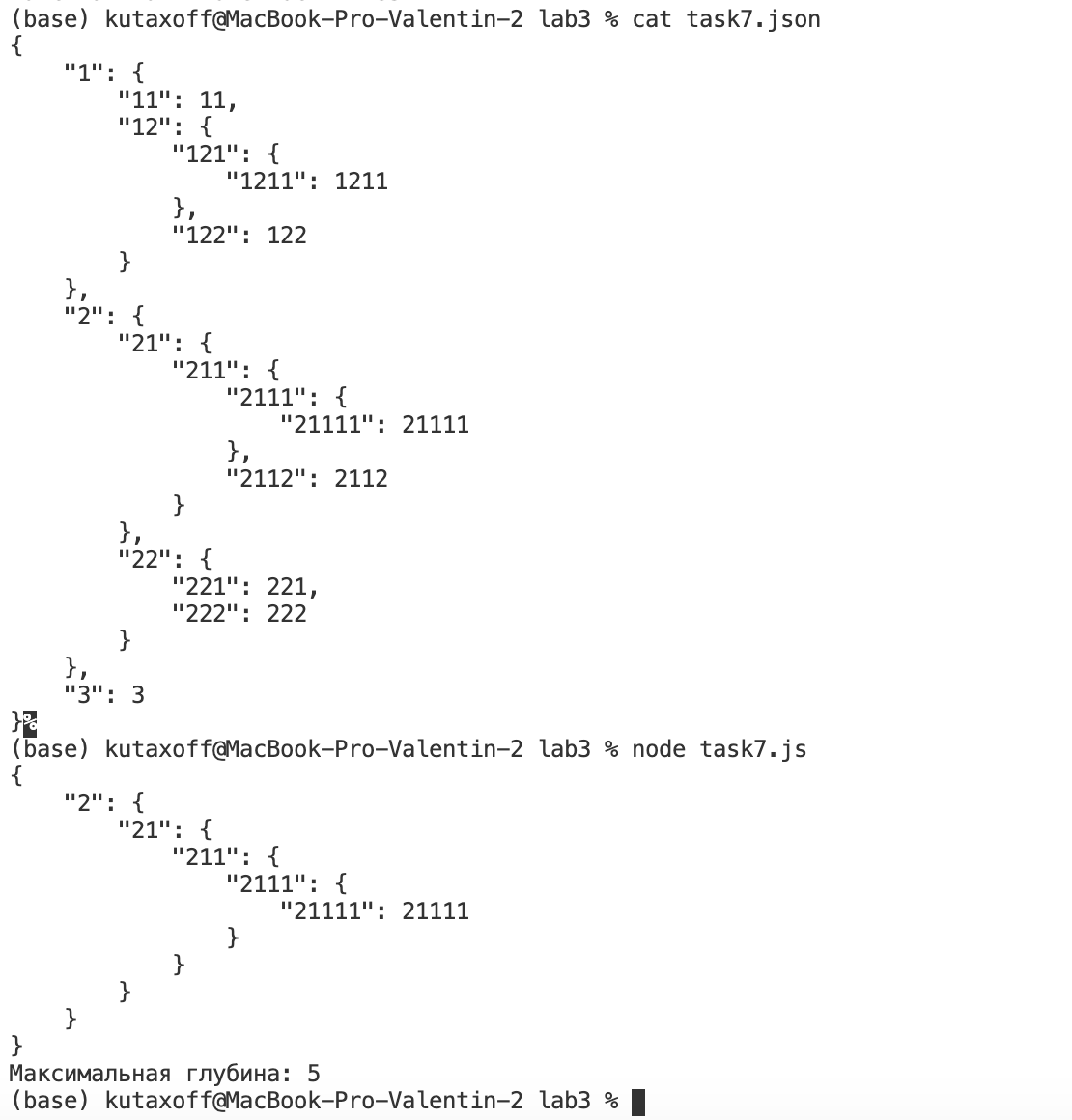
print(JSON.stringify(res["branch"], null, 4))

print("Максимальная глубина: " + res["level"])

}

main()

**Проверка работоспособности**

****

**Подзадача 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.

**Реализация**

"use strict";

const express = require("express");

const fs = require("fs");

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/maxOfThree", function(request, response) {

const a = request.query.a;

const b = request.query.b;

const c = request.query.c;

const aFloat = parseFloat(a);

const bFloat = parseFloat(b);

const cFloat = parseFloat(c);

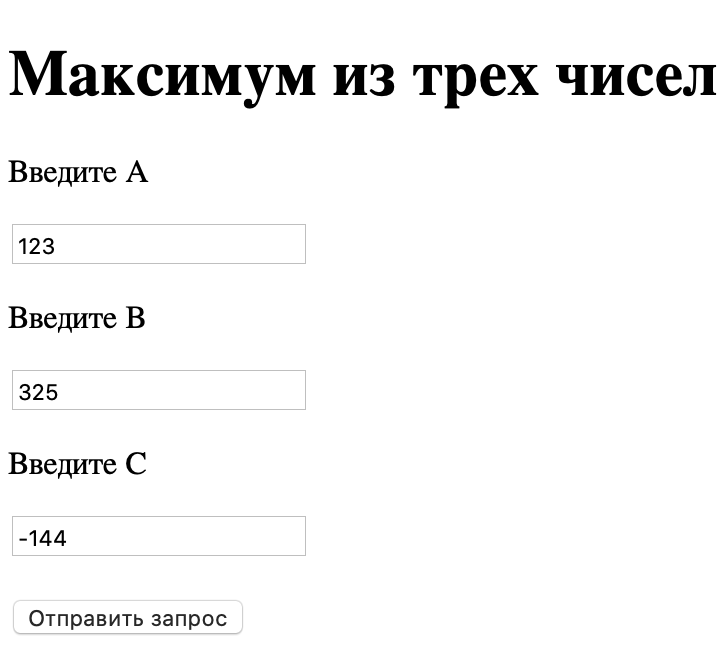
const maxFloat = Math.max(aFloat, bFloat, cFloat)

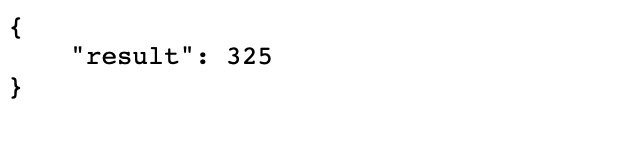
const answerJSON = JSON.stringify({result: maxFloat}, null, 4);

response.end(answerJSON);

});

**Проверка работоспособности**





**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.

**Реализация**

"use strict";

const express = require("express");

const fs = require("fs");

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("/array/get", function(request, response) {

const index = request.query.index;

const arrayJSON = fs.readFileSync("array.json", "utf-8");

const array = JSON.parse(arrayJSON);

let res = null;

if (array.length > index && index >= 0) {

res = JSON.stringify({result: array[index]}, null, 4);

}

const answerJSON = res;

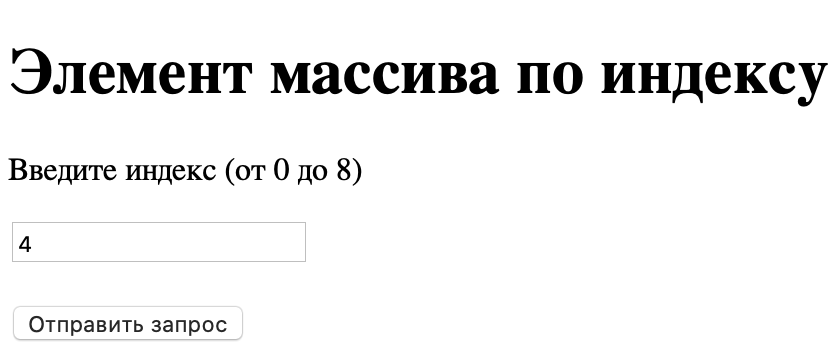
response.end(answerJSON);

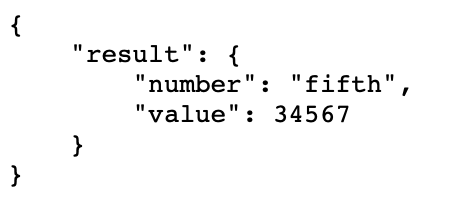
});

**Содержимое файла**



**Проверка работоспособности**

****

****

**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.

**Реализация**

"use strict"

const fs = require("fs");

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

const startFormFileName = "part1.html";

const fieldFormFileName = "field.html"

const endFormFileName = "part2.html";

let startForm = fs.readFileSync(startFormFileName, "utf-8");

let fieldForm = fs.readFileSync(fieldFormFileName, "utf-8")

let endForm = fs.readFileSync(endFormFileName, "utf-8");

const formTemplate = {"start": startForm, "field": fieldForm, "end": endForm};

function generateHTML(fields, reqAdress, outputFile, title="Title", header="Header", buttonText="Submit", form=formTemplate) {

let searchRegExp = /@title@/g;

let start = form["start"].replace(searchRegExp, title);

searchRegExp = /@header@/g;

start = start.replace(searchRegExp, header);

searchRegExp = /@destination@/g;

start = start.replace(searchRegExp, reqAdress);

start += "\n"

searchRegExp = /@buttonText@/g;

let end = form["end"].replace(searchRegExp, buttonText);

let fieldsStr = "";

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

let field = form["field"].split("@field@");

fieldsStr += field.join(fields[i]) + "\n";

}

let result = start + fieldsStr + end;

fs.writeFileSync(outputFile, result)

}

function main() {

let outputFileName = "c.html";

let title = null;

let header = null;

let buttonText = null;

let extras = false;

const choice = readlineSync.question("Type Y for extra settings: ");

if (choice === "Y" || choice === "y") {

outputFileName = readlineSync.question("Output File: ");

title = readlineSync.question("Title: ");

header = readlineSync.question("Header: ");

buttonText = readlineSync.question("Text on Button: ");

extras = true;

}

console.log("Input names of fields (empty input for end)");

let fields = [];

let field = readlineSync.question(": ");

while (field !== "") {

fields.push(field);

field = readlineSync.question(": ");

}

let reqAdress = readlineSync.question("Request Adress: ");

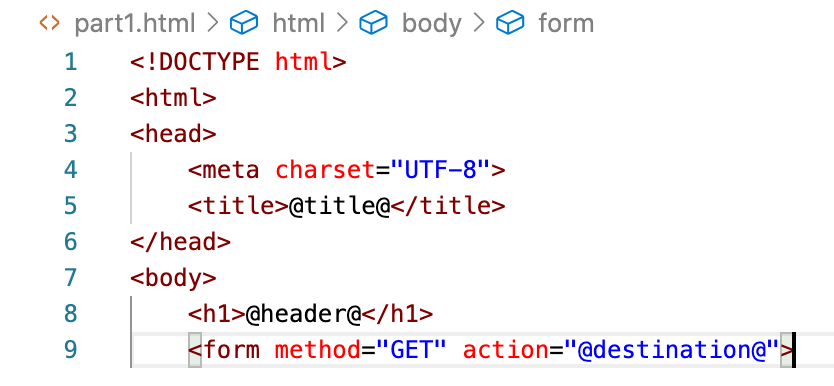
if (extras) generateHTML(fields, reqAdress, outputFileName, title, header, buttonText);

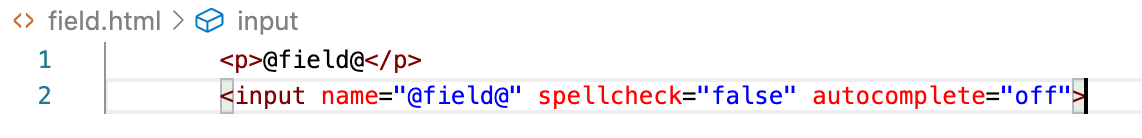
else generateHTML(fields, reqAdress, outputFileName);

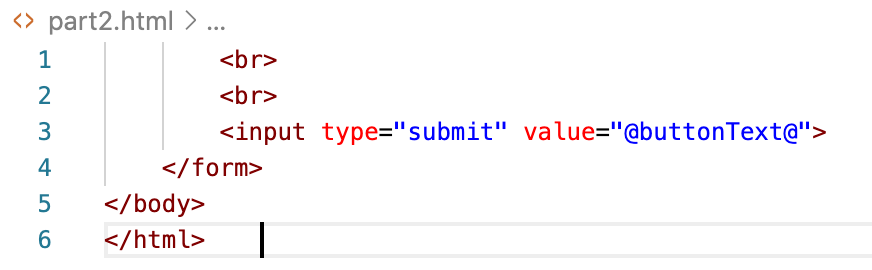
}

main()

**Содержимое вспомогательных файлов**







### **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.

**Реализация**

"use strict";

const express = require("express");

const fs = require("fs");

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/multiplesOf", 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 answer = []

for (let n = aInt; n <= bInt; n++) {

if (n % cInt === 0) {

answer.push(n)

}

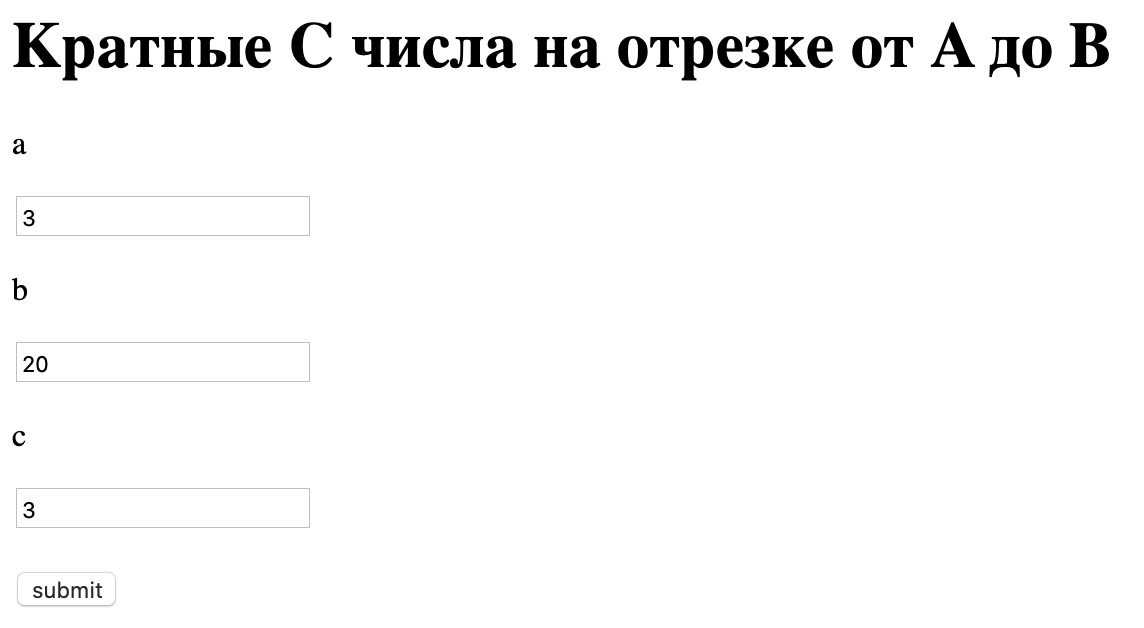
}

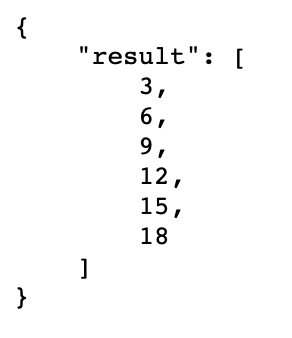
const answerJSON = JSON.stringify({result: answer}, null, 4);

response.end(answerJSON);

});

**Проверка работоспособности**

****

****

**Вывод**

Язык программирования JavaScript позволяет создавать веб-сервера при помощи библиотеки request. Ввод информации, её преобразование и выдача пользователю могут быть осуществлены с использованием таких базовых инструментов языка JavaScript, как структуры, массивы и классы. Язык JavaScript также предоставляет эффективные встроенные методы решения математических задач.