|  |  |
| --- | --- |
| 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 **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 2 |

**Дисциплина:** Архитектура ЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | Сучков А.Д. |
|  | (Группа) |  | (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**

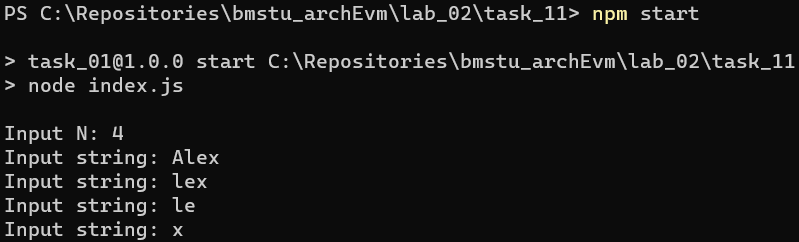
**Задание 1**

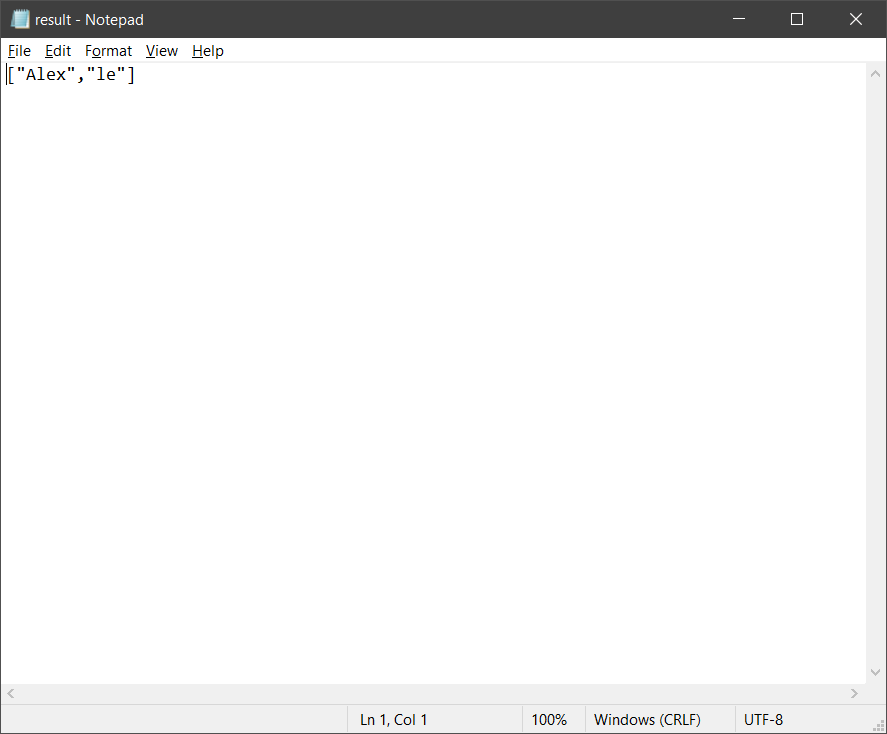
С клавиатуры считывается число N. Далее считывается N строк. Необходимо создать массив и сохранять в него строки только с четной длинной. Получившийся массив необходимо преобразовать в строку JSON и сохранить в файл.

**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**





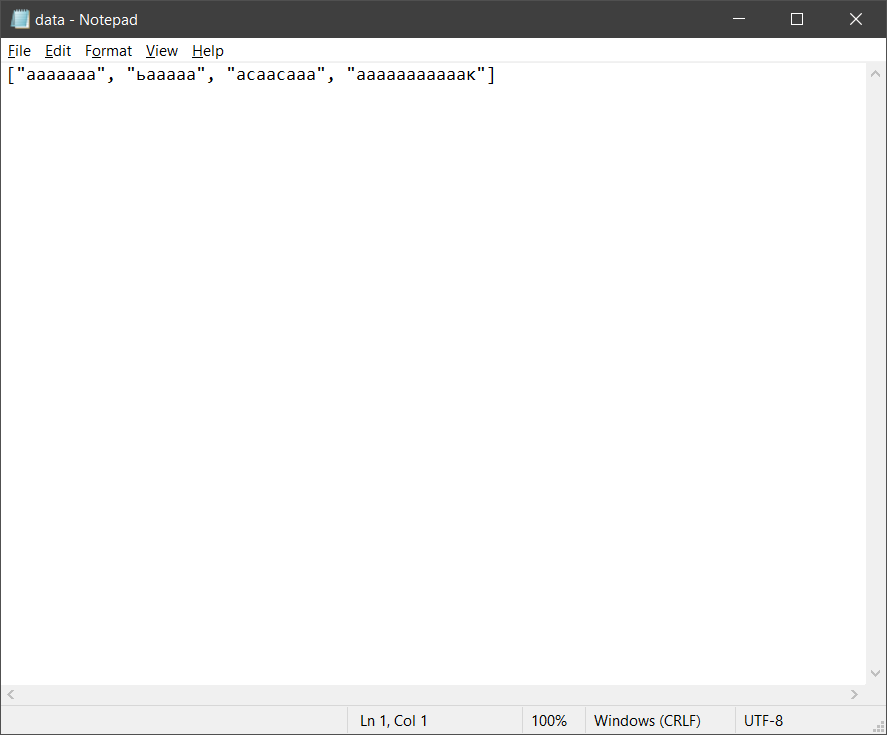
**Задание 2**

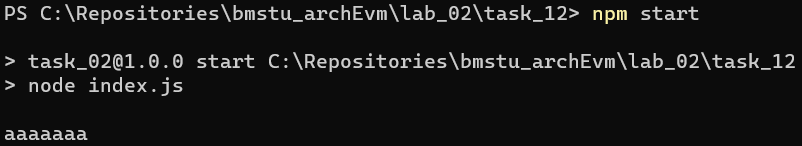
Необходимо считать содержимое файла, в котором хранится массив строк в формате JSON. Нужно вывести только те строки на экран, в которых содержатся только гласные буквы.

**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**





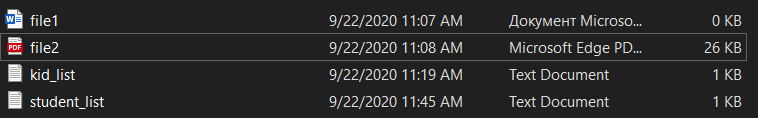
**Задание 3**

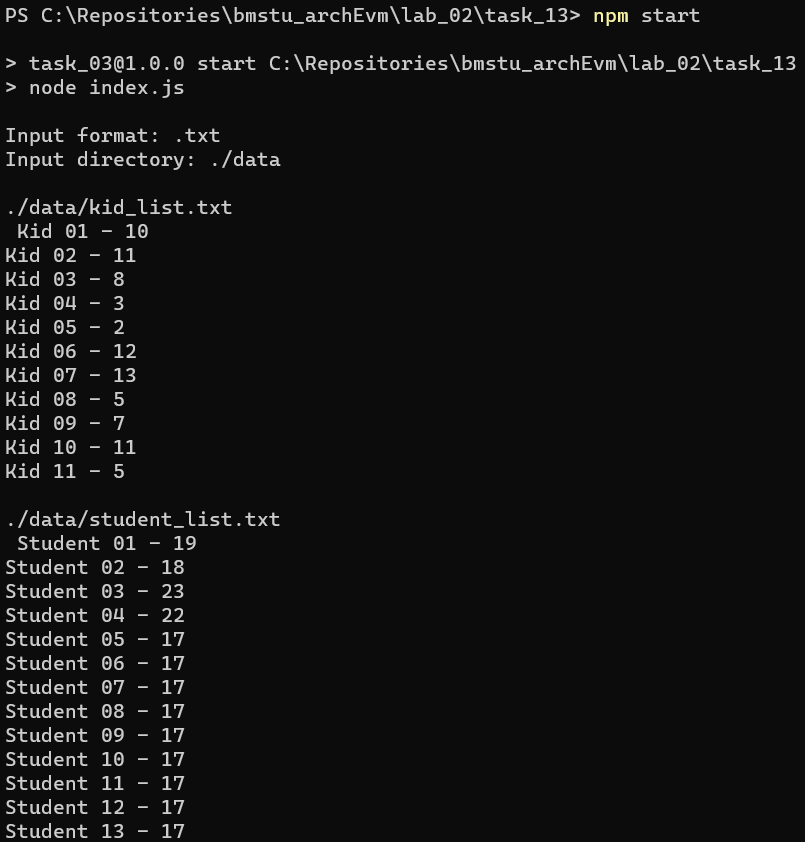
С клавиатуры считывается строка - название расширения файлов. Далее считывается строка - адрес папки. Необходимо перебрать все файлы в папке и вывести содержимое файлов, у которых расширение совпадает с введенным расширением.

**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**





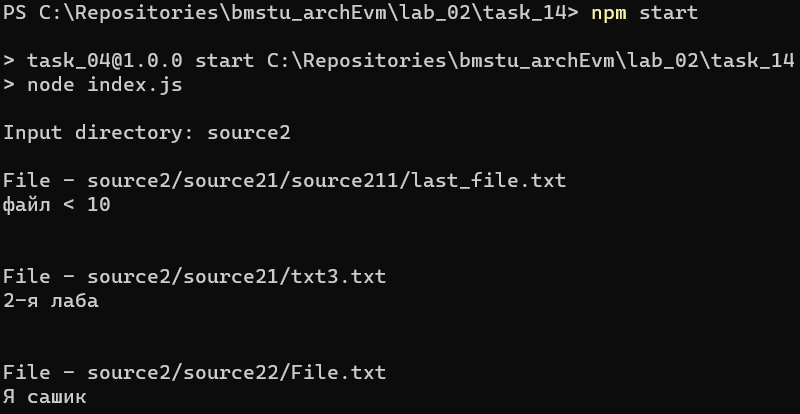
**Задание 4**

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**

|  |
| --- |
| "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**

****

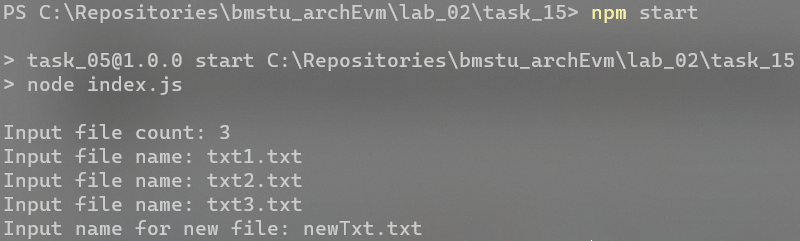
**Задание 5**

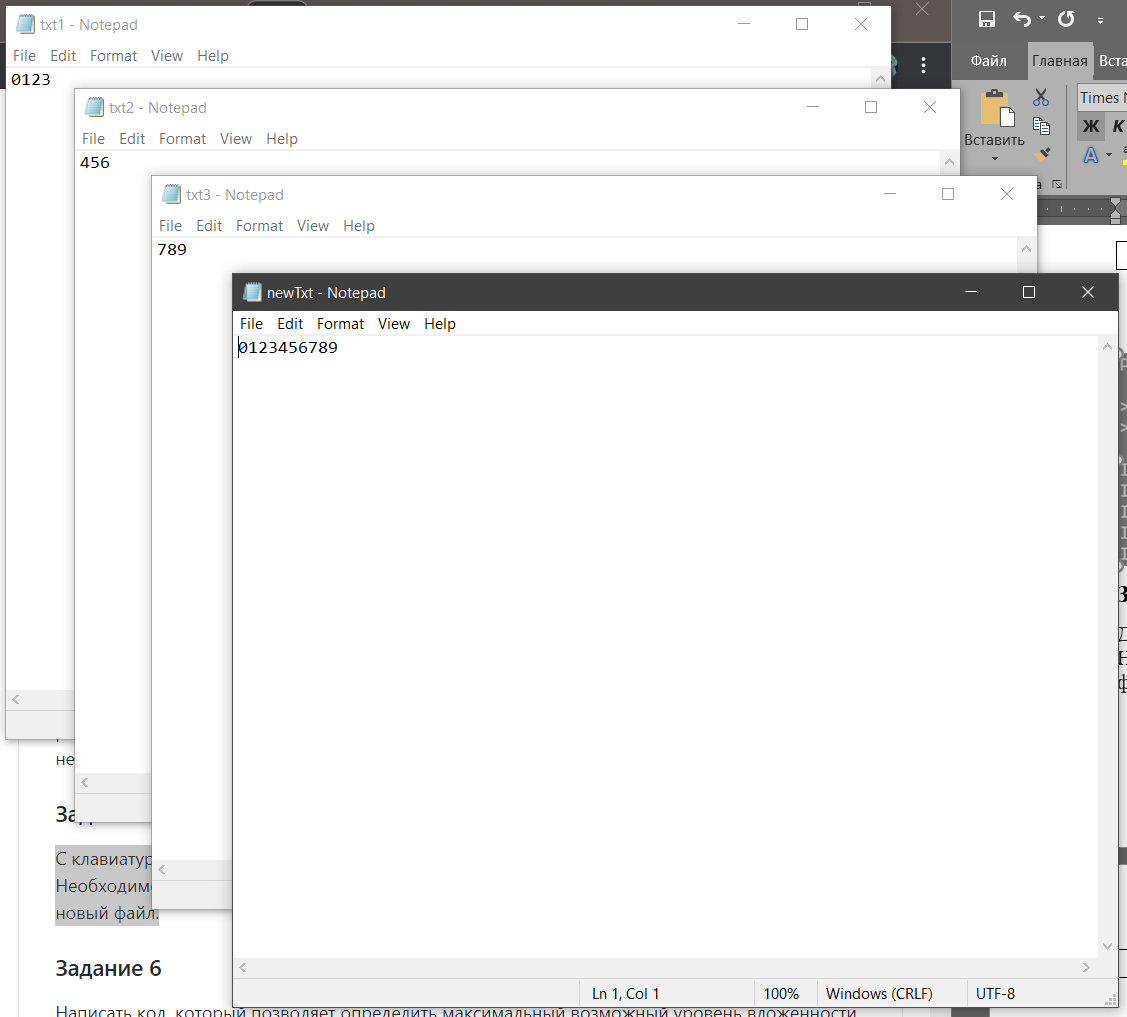
С клавиатуры считывается число N. Далее считывается N строк - имена текстовых файлов. Необходимо склеить всё содержимое введенных файлов в одну большую строку и сохранить в новый файл.

**Листинг**

|  |
| --- |
| "use strict";  const fs = require("fs");  const readLine = require("readline-sync");  function inputFilesName(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**

****

****

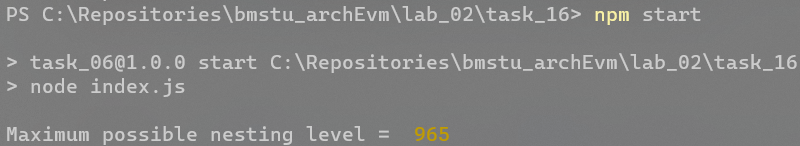
**Задание 6**

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**

****

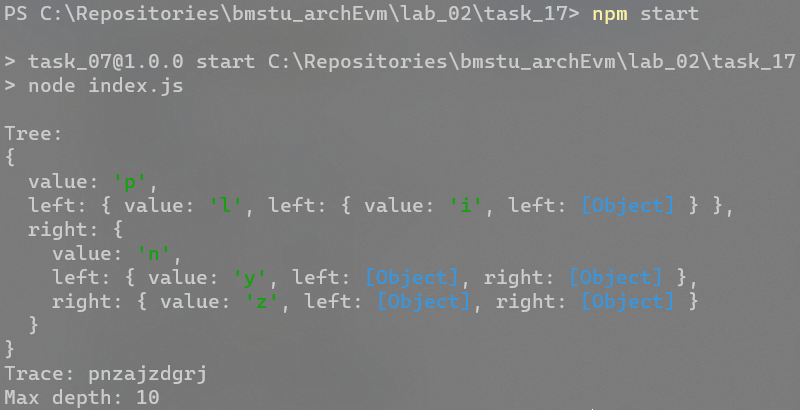
**Задание 7**

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**



**Часть 2**

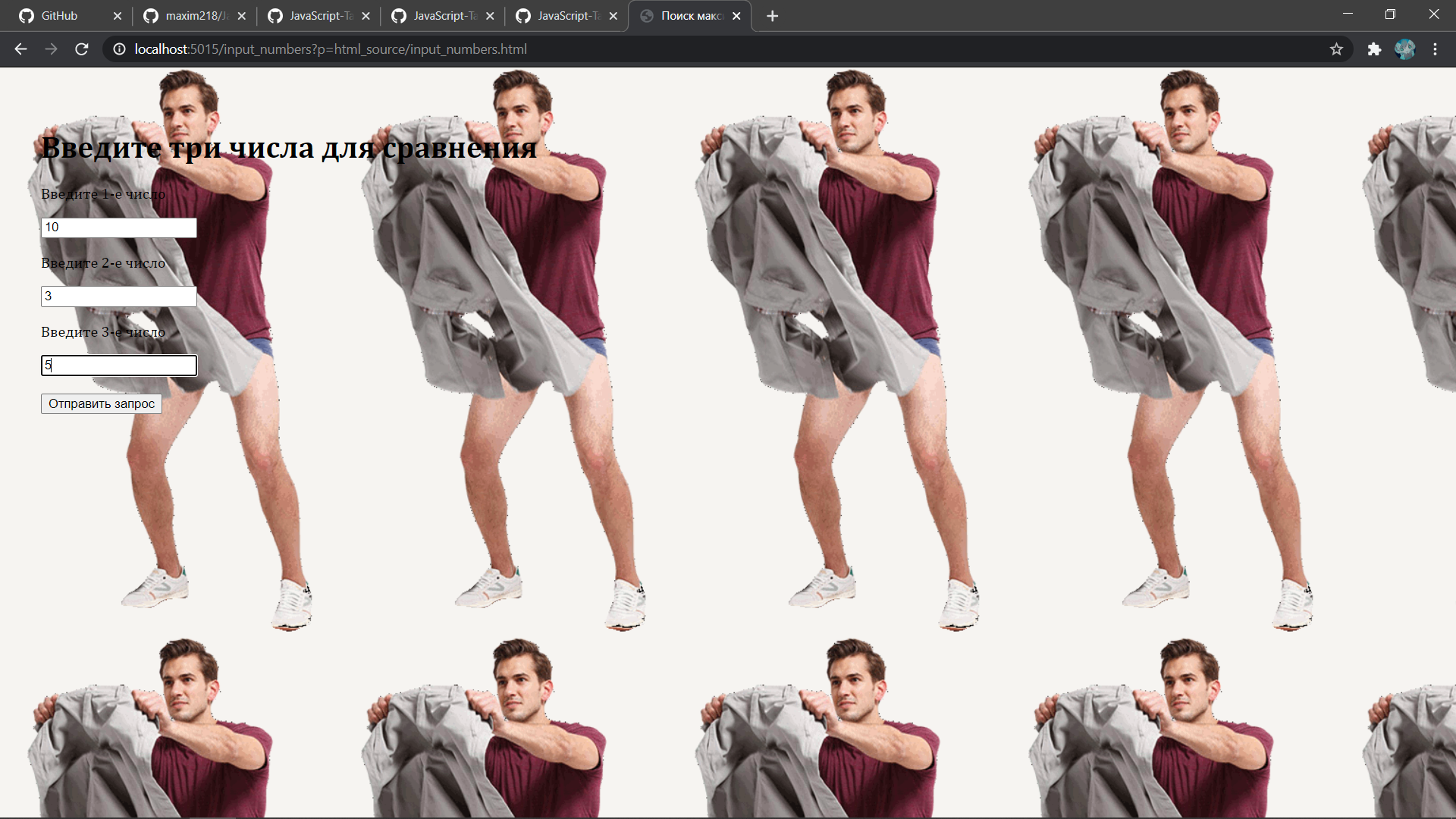
**Задание 1**

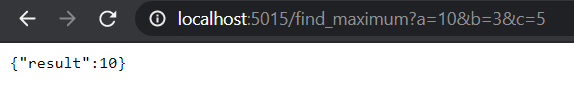
Запустить сервер. Реализовать на сервере функцию для сравнения трёх чисел и выдачи наибольшего из них. Реализовать страницу с формой ввода для отправки запроса на сервер.

**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**



****

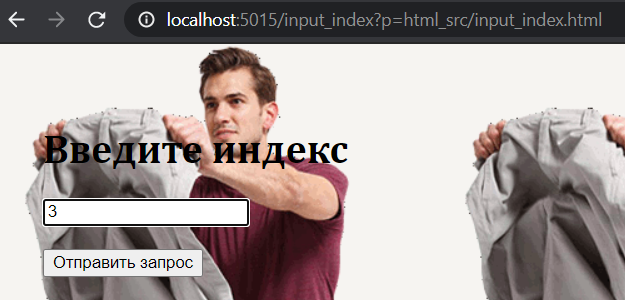
**Задание 2**

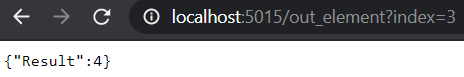
Запустить сервер. На стороне сервера должен храниться файл, внутри которого находится JSON строка. В этой JSON строке хранится информация о массиве объектов. Реализовать на сервере функцию, которая принимает индекс и выдает содержимое ячейки массива по данному индексу. Реализовать страницу с формой ввода для отправки запроса на сервер.

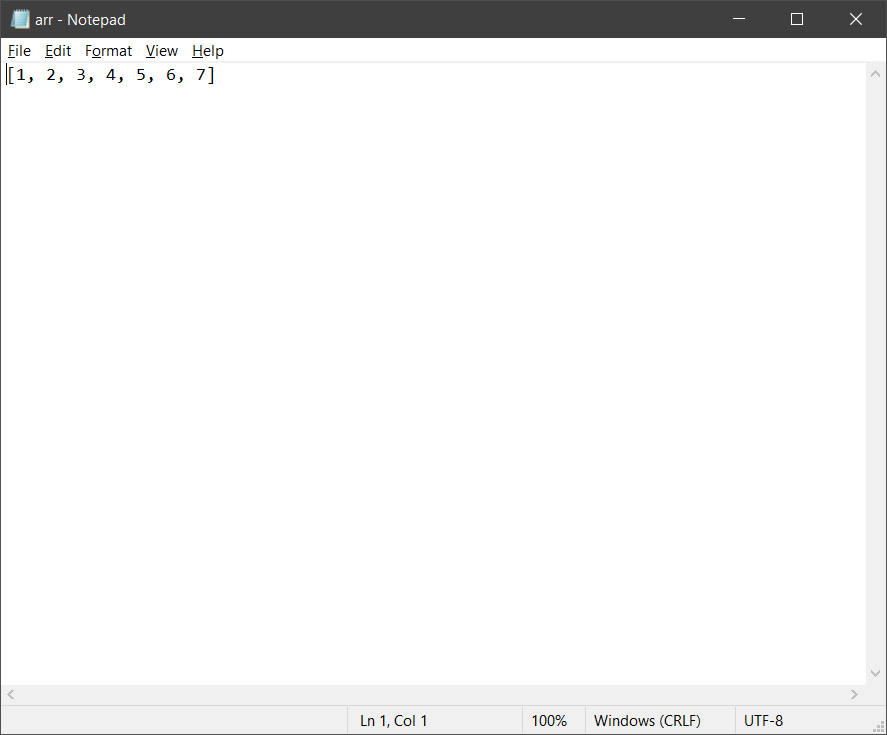
**Листинг**

|  |
| --- |
| "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**

****

****



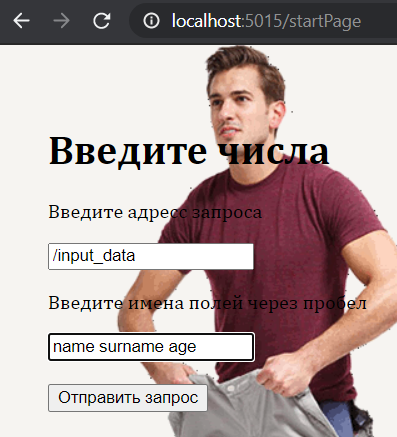
**Задание 3**

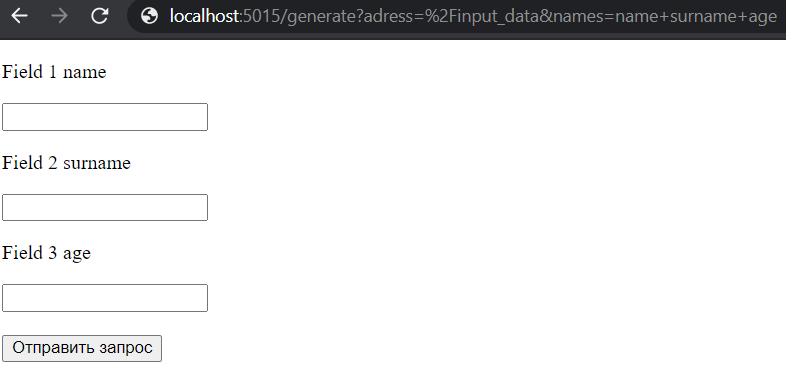
Написать программу, которая на вход получает массив названий полей и адрес запроса (куда отправлять). Программа должна генерировать HTML разметку страницы, в которую встроена форма для отправки запроса.

**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) + " " + fieldName + '</p>';              this.pageBody += '\n\t\t' + '<input name="' + fieldName + '" spellcheck="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.pageBody + '\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**

****

****

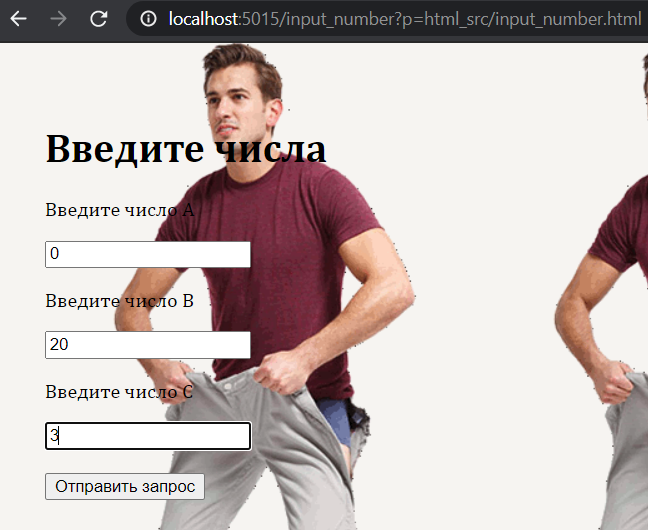
**Задание 4**

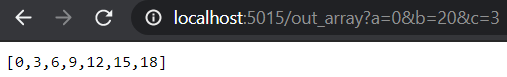
Запустить сервер. Реализовать на сервере функцию, которая принимает на вход числа A, B и C. Функция должна выдавать массив целых чисел на отрезке от A до B, которые делятся на C нацело.

**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**

****

****

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