МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ:

по дисциплине «Информационная безопасность»

Исполнитель

студент (ка) 3 курса группы 6 Розель Станислав Александрович

(Ф.И.О.)

Минск 2024

Исходный текст

Зашифрованный текст шифром Цезаря

Зашифрованный текст таблицей Трисемуса

Шифрование текста таблицей Трисемуса: 37.184ms  
Расшифровка текста таблицей Трисемуса: 30.662ms

Шифрование текста шифрем Цезаря: 32.438ms  
Расшифровка текста шифрем Цезаря: 29.635ms

Шифр Цезаря код

|  |
| --- |
| const readline = require('readline');  const fs = require('fs');  const belarusianAlphabet = [  'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'І',  'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т',  'У', 'Ў', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Ы', 'Ь', 'Э',  'Ю', 'Я'  ];  const alphabets = {};  const rl = readline.createInterface({  input: process.stdin,  output: process.stdout  });  function countCharacters(text) {  const charCount = {};  for (const char of text) {  charCount[char] = (charCount[char] || 0) + 1;  }  const charArray = Object.entries(charCount).sort((a, b) => b[1] - a[1]);  console.log('Символ | Количество');  console.log('-----------------');  charArray.forEach(([char, count]) => {  const displayChar = char === ' ' ? '[пробел]' : char;  console.log(`'${displayChar}' | ${count}`);  });  console.log(`\nВсего уникальных символов: ${charArray.length}`);  console.log(`Общая длина текста: ${text.length} символов`);  }  function removeDuplicateLetters(word) {  return Array.from(new Set(word)).join('').toUpperCase();  }  function removeLettersArray(array1, array2) {  const lettersToRemove = new Set(array1);  return array2.filter(letter => !lettersToRemove.has(letter));  }  function createNewAlphabet(keyword, number) {  const belAlphabet = [...belarusianAlphabet];  const arrKeyword = removeDuplicateLetters(keyword).split('');  const chunk1 = belAlphabet.splice(number, belAlphabet.length);  const chunk2 = belAlphabet.splice(0, number);  return [  ...removeLettersArray(arrKeyword, chunk1),  ...arrKeyword,  ...removeLettersArray(arrKeyword, chunk2)  ]  }  function createMapAlphabet(newAlphabet) {  for (let i = 0; i < newAlphabet.length; i++) {  alphabets[belarusianAlphabet[i]] = newAlphabet[i];  }  }  function \_encode(text, replacements) {  let result = '';  for (const char of text.toUpperCase()) {  result += replacements[char] || char;  }  return result;  }  function \_decode(text, replacements) {  const reversedReplacements = Object.fromEntries(  Object.entries(replacements).map(([key, value]) => [value, key])  );  let result = '';  for (const char of text) {  result += reversedReplacements[char] || char;  }  return result;  }  rl.question('Введите число: ', (number) => {  rl.question('Введите ключевое слово: ', (keyword) => {  const text = fs.readFileSync('./texts/originalTexts/newLand.txt', 'utf8');  countCharacters(text);  const newAlphabet = createNewAlphabet(keyword, number);  console.log(newAlphabet);  createMapAlphabet(newAlphabet);  console.time("Execute time");  const encodeText = \_encode(text, alphabets);  console.timeEnd("Execute time");  countCharacters(encodeText);  fs.writeFileSync('./texts/encode/encodeNewLand.txt', encodeText);  const textEncode = fs.readFileSync('./texts/encode/encodeNewLand.txt', 'utf8');  console.time("Execute time");  const textDecode = \_decode(textEncode, alphabets);  console.timeEnd("Execute time");  fs.writeFileSync('./texts/decode/decodeNewLand.txt', textDecode);  rl.close();  });  }); |

Таблица Трисемуса код

|  |
| --- |
| const fs = require('fs');  const readline = require('readline');  const ALPHABET = [  'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'І',  'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т',  'У', 'Ў', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Ы', 'Ь', 'Э',  'Ю', 'Я'  ];  const rl = readline.createInterface({  input: process.stdin,  output: process.stdout  });  function countCharacters(text) {  const charCount = {};  for (const char of text) {  charCount[char] = (charCount[char] || 0) + 1;  }  const charArray = Object.entries(charCount).sort((a, b) => b[1] - a[1]);  console.log('Символ | Количество');  console.log('-----------------');  charArray.forEach(([char, count]) => {  const displayChar = char === ' ' ? '[пробел]' : char;  console.log(`'${displayChar}' | ${count}`);  });  console.log(`\nВсего уникальных символов: ${charArray.length}`);  console.log(`Общая длина текста: ${text.length} символов`);  }  function createTableWithKeyword(keyword) {  const uniqueChars = Array.from(new Set(keyword.toUpperCase().split('')));  const validKeyChars = uniqueChars.filter(char => ALPHABET.includes(char));  const remainingChars = ALPHABET.filter(char => !validKeyChars.includes(char));  const newAlphabet = [...validKeyChars, ...remainingChars];  const rows = 8;  const cols = 4;  const table = [];  let charIndex = 0;  for (let i = 0; i < rows; i++) {  const row = [];  for (let j = 0; j < cols; j++) {  if (charIndex < newAlphabet.length) {  row.push(newAlphabet[charIndex]);  charIndex++;  } else {  row.push('');  }  }  table.push(row);  }  return table;  }  function encrypt(text, keyword) {  const table = createTableWithKeyword(keyword);  return text.toUpperCase().split('').map(char => {  if (!ALPHABET.includes(char)) {  return char;  }  for (let i = 0; i < table.length; i++) {  for (let j = 0; j < table[i].length; j++) {  if (table[i][j] === char) {  return table[i][0];  }  }  }  return char;  }).join('');  }  function decrypt(encryptedText, keyword) {  const table = createTableWithKeyword(keyword);  return encryptedText.toLowerCase().split('').map(char => {  if (!ALPHABET.includes(char)) {  return char;  }  for (let i = 0; i < table.length; i++) {  if (table[i][0] === char) {  return char;  }  }  return char;  }).join('');  }  function printTable(table) {  console.log('Таблица шифрования:');  for (let i = 0; i < table.length; i++) {  console.log(table[i].join('\t'));  }  }  rl.question('Введите ключевое слово: ', (keyword) => {  const originalText = fs.readFileSync('./texts/originalTexts/newLand.txt', 'utf8');  countCharacters(originalText);  const table = createTableWithKeyword(keyword);  printTable(table);  console.time("Execute time");  const encryptedText = encrypt(originalText, keyword);  console.timeEnd("Execute time");  countCharacters(encryptedText);  fs.writeFileSync('./texts/encode/encodeNewLand.txt', encryptedText);  const encodeText = fs.readFileSync('./texts/encode/encodeNewLand.txt', 'utf8');  console.time("Execute time");  const decodeText = decrypt(encodeText, keyword);  console.timeEnd("Execute time");  fs.writeFileSync('./texts/decode/decodeNewLand.txt', decodeText);  rl.close();  }); |