Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №2»**

“Исследование криптографических шифров на основе подстановки (замены) символов”

**Выполнил:** студент 4 курса

1 группы специальности ИСИт

Палазник Арсений Викторович

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2024

**Разработать авторское приложение в соответствии с целью лабораторной работы. Приложение должно реализовывать следующие операции:**

* **выполнять зашифрование/расшифрование текстовых документов (объемом не менее 5 тысяч знаков), созданных на основе алфавита языка в соответствии с нижеследующей таблицей вариантов задания; при этом следует использовать шифры подстановки из третьего столбца данной таблицы (варианты задания представлены в табл. 2.6);**

Реализация приложения:

|  |
| --- |
| const fs = require("fs");  const path = require("path");  // Немецкий алфавит  const alphabet = "abcdefghijklmnopqrstuvwxyzäöüß";  // Функция для шифрования с использованием соотношений  function encryptUsingRelation(text, k, alphabet) {  return text  .split("")  .map((char) => {  const index = alphabet.indexOf(char);  if (index === -1) return char; // Возвращаем символ, если он не в алфавите  const newIndex = (index + k) % alphabet.length;  return alphabet[newIndex];  })  .join("");  }  // Функция для расшифрования с использованием соотношений  function decryptUsingRelation(text, k, alphabet) {  return text  .split("")  .map((char) => {  const index = alphabet.indexOf(char);  if (index === -1) return char; // Возвращаем символ, если он не в алфавите  const newIndex = (index - k + alphabet.length) % alphabet.length;  return alphabet[newIndex];  })  .join("");  }  // Функция для создания таблицы Трисемуса  function createTrisemusTable(keyword, alphabet) {  const uniqueChars = [...new Set(keyword + alphabet)];  const table = [];  while (uniqueChars.length) {  table.push(uniqueChars.splice(0, 6)); // создаем строки таблицы  }  return table;  }  // Функция для шифрования с таблицей Трисемуса  function trisemusCipher(text, keyword, alphabet, encrypt = true) {  const table = createTrisemusTable(keyword, alphabet);  return text  .split("")  .map((char) => {  const rowIndex = table.findIndex((row) => row.includes(char));  if (rowIndex === -1) return char;  const colIndex = table[rowIndex].indexOf(char);  if (encrypt) {  return rowIndex === table.length - 1  ? table[0][colIndex]  : table[rowIndex + 1][colIndex];  } else {  return rowIndex === 0  ? table[table.length - 1][colIndex]  : table[rowIndex - 1][colIndex];  }  })  .join("");  }  // Функция для построения гистограммы  function createHistogram(text) {  const frequency = {};  text.split("").forEach((char) => {  frequency[char] = (frequency[char] || 0) + 1;  });  return frequency;  }  // Функция для записи гистограммы в CSV  function writeHistogramToCSV(histogram, filename) {  const csvContent =  "Character,Frequency\n" +  Object.entries(histogram)  .map(([char, freq]) => `${char};${freq}`)  .join("\n");  fs.writeFileSync(filename, csvContent);  }  // Главная программа  const keyword = "enigma"; // ключевое слово  const k = 7; // значение k  // Чтение исходного текста из файла  const originalText = fs.readFileSync("input.txt", "utf-8");  // Шифрование с использованием соотношений  const startRelation = Date.now();  const encryptedRelation = encryptUsingRelation(originalText, k, alphabet);  const endRelation = Date.now();  const relationTime = endRelation - startRelation;  // Расшифрование с использованием соотношений  const startRelationDecrypt = Date.now();  const decryptedRelation = decryptUsingRelation(encryptedRelation, k, alphabet);  const endRelationDecrypt = Date.now();  const relationDecryptTime = endRelationDecrypt - startRelationDecrypt;  // Шифрование с таблицей Трисемуса  const startTrisemus = Date.now();  const encryptedTrisemus = trisemusCipher(originalText, keyword, alphabet, true);  const endTrisemus = Date.now();  const trisemusTime = endTrisemus - startTrisemus;  // Расшифрование с таблицей Трисемуса  const startTrisemusDecrypt = Date.now();  const decryptedTrisemus = trisemusCipher(  encryptedTrisemus,  keyword,  alphabet,  false  );  const endTrisemusDecrypt = Date.now();  const trisemusDecryptTime = endTrisemusDecrypt - startTrisemusDecrypt;  // Создание гистограмм  const originalHistogram = createHistogram(originalText);  const relationHistogram = createHistogram(encryptedRelation);  const trisemusHistogram = createHistogram(encryptedTrisemus);  // Запись зашифрованных текстов и гистограмм в файлы  fs.writeFileSync("encrypted\_relation.txt", encryptedRelation);  fs.writeFileSync("decrypted\_relation.txt", decryptedRelation);  fs.writeFileSync("encrypted\_trisemus.txt", encryptedTrisemus);  fs.writeFileSync("decrypted\_trisemus.txt", decryptedTrisemus);  writeHistogramToCSV(originalHistogram, "original\_histogram.csv");  writeHistogramToCSV(relationHistogram, "relation\_histogram.csv");  writeHistogramToCSV(trisemusHistogram, "trisemus\_histogram.csv");  // Вывод результатов  console.log(  `Зашифрованный текст с использованием соотношений: ${encryptedRelation}`  );  console.log(  `Время выполнения шифрования с использованием соотношений: ${relationTime}ms`  );  console.log(  `Расшифрованный текст с использованием соотношений: ${decryptedRelation}`  );  console.log(  `Время выполнения расшифрования с использованием соотношений: ${relationDecryptTime}ms`  );  console.log(  `Зашифрованный текст с использованием таблицы Трисемуса: ${encryptedTrisemus}`  );  console.log(  `Время выполнения шифрования с использованием таблицы Трисемуса: ${trisemusTime}ms`  );  console.log(  `Расшифрованный текст с использованием таблицы Трисемуса: ${decryptedTrisemus}`  );  console.log(  `Время выполнения расшифрования с использованием таблицы Трисемуса: ${trisemusDecryptTime}ms`  );  console.log("Все выполнено"); |

Результат выполнения:

