МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 НА ТЕМУ:**

**Элементы теории информации. Параметры и характеристики дискретных информационных систем**

Выполнил студент 3 курса 4 группы

Смолик Валерий Александрович

Минск 2024

**Задание А**.

Рассчитать энтропию указанных преподавателем алфавитов: один – на латинице, другой – на кириллице.

Энтропию алфавита по К. Шеннону рассчитывают по следующей формуле:

Код, позволяющий посчитать энтропию английского и русского языков представлен в листинге 1, написан на языке Java.

|  |
| --- |
| package com.ib.lab1;  import java.io.File;  import java.io.FileNotFoundException;  import java.util.HashMap;  import java.util.Map;  import java.util.Scanner;  public class Lab1 {  public static void main(String[] args) {  String filePath1 = "D:\\3k2s\\IB\\labs\\lab1\\CIR.txt";  String filePath2 = "D:\\3k2s\\IB\\labs\\lab1\\LAT.txt";  final String LATIN\_ALPHABET = "abcdefghijklmnopqrstuvwxyz";  final String CYRILLIC\_ALPHABET = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";  try {  String textCyrillic = loadTextFromFile(filePath1);  String textLatin = loadTextFromFile(filePath2);  double latinEntropy = calculateEntropy(textLatin, LATIN\_ALPHABET);  System.out.println("Энтропия латинского алфавита: " + latinEntropy);  double cyrillicEntropy = calculateEntropy(textCyrillic, CYRILLIC\_ALPHABET);  System.out.println("Энтропия кириллического алфавита: " + cyrillicEntropy);  } catch (FileNotFoundException e) {  System.out.println("Файл не найден: " + e.getMessage());  }  }  private static String loadTextFromFile(String filePath) throws FileNotFoundException {  StringBuilder sb = new StringBuilder();  File file = new File(filePath);  try (Scanner scanner = new Scanner(file)) {  while (scanner.hasNextLine()) {  sb.append(scanner.nextLine());  }  }  return sb.toString();  }  // Расчет энтропии  private static double calculateEntropy(String text, String alphabet) {  Map<Character, Integer> charFrequency = new HashMap<>();  int totalCount = 0;  // Подсчет частот символов  for (char c : text.toCharArray()) {  if (alphabet.indexOf(c) != -1) {  charFrequency.put(c, charFrequency.getOrDefault(c, 0) + 1);  totalCount++;  }  }  double entropy = 0.0;  for (char c : charFrequency.keySet()) {  double probability = (double) charFrequency.get(c) / totalCount;  entropy -= probability \* (Math.log(probability) / Math.log(2));  }  return entropy;  }  } |

Листинг 1 – Код подсчета энтропии

В результате выполнения данного кода, были получены значения, представленные на рисунке 1.

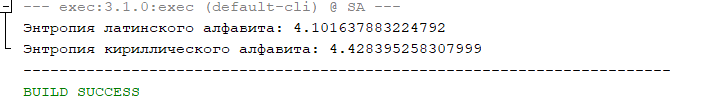
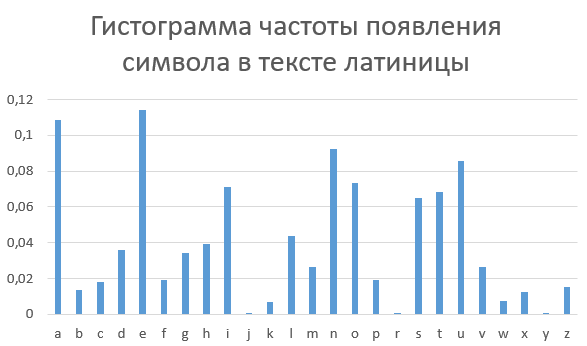


Рисунок 1 – показатели энтропии для русского и английского алфавитов.

Гистограммы частоты появления символов в тексте кириллицы:



Гистограмма появления символов в тексте латиницы:



**Задание Б**.

Для входных документов, представленных в бинарных кодах, определить энтропию бинарного алфавита.

Для выполнения задания Б, дополним код прошлого задания тремя функциями, которые будут находить энтропию бинарного алфавита. Код, функций представлен в Листинге 2.

|  |
| --- |
| private static double calculateBinaryEntropy(String filePath) throws IOException {  byte[] binaryData = loadBinaryDataFromFile(filePath);  return calculateEntropy(binaryData);}  private static byte[] loadBinaryDataFromFile(String filePath) throws IOException {  File file = new File(filePath);  byte[] data = new byte[(int) file.length()];  try (FileInputStream fis = new FileInputStream(file)) {  fis.read(data);  }  return data;}  private static double calculateEntropy(byte[] data) {  Map<Byte, Integer> byteFrequency = new HashMap<>();  int totalCount = 0;  for (byte b : data) {  byteFrequency.put(b, byteFrequency.getOrDefault(b, 0) + 1);  totalCount++;  }  double entropy = 0.0;  for (byte b : byteFrequency.keySet()) {  double probability = (double) byteFrequency.get(b) / totalCount;  entropy -= probability \* (Math.log(probability) / Math.log(2));  }  return entropy;  }  } |

Листинг 2 – Код функций, вычисляющих бинарную энтропию

Результат выполнения кода, представлен на рисунке 2.



Рисунок 2 – Бинарные энтропии русского и английского языков соответственно.

**Задание В.**

Используя значения энтропии алфавитов, полученные в заданиях а и б, подсчитать количество информации в сообщении, состоящем из собственных имени, фамилии и отчества.  
 Для выполнения задания дополним код еще одним методом, который вычислит количество информации. Листинг 3.

|  |
| --- |
| private static double QuantityOfInformation(double entropy, String text)  {  return entropy \* text.length();} |

Листинг 3 – метод класса, находящий количество информации в сообщении.

Результат выполнения метода представлен на рисунке 3.

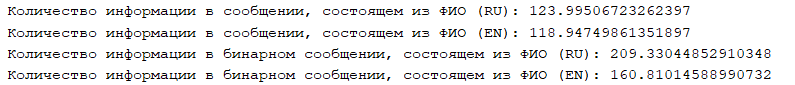


Рисунок 3 – Количество информации в сообщении, содержащим ФИО на русском языке и английский

**Задание Г.**

Выполнить задание в) при условии, что вероятность ошибочной передачи единичного бита сообщения составляет: 0.1, 0.5, 1.  
 Для того чтобы вычислить количество информации в сообщении воспользуется переопределением метода QuantityOfInformation. Код представлен в Листинге 4.

|  |
| --- |
| public static double QuantityOfInformation(double entropy, String text, double errorProbability) {  return entropy \* text.length() \* (1 - errorProbability);  } |

Листинг 4 – Переопределение метода QuantityOfInformation

В результате выполнения кода получим вывод, представленный на рисунке 4.

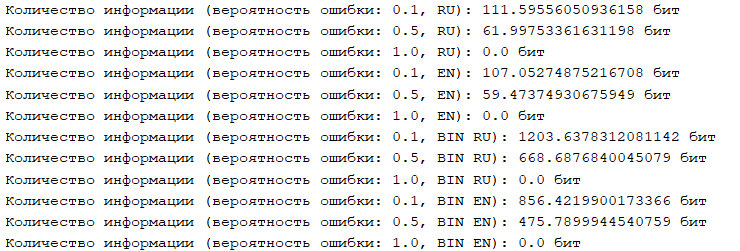


Рисунок 4 – Вывод после выполнения.

Полный код представлен в Листинге 5.

|  |
| --- |
| package com.ib.lab1;  import java.io.File;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.IOException;  import java.util.HashMap;  import java.util.Map;  import java.util.Scanner;  public class Lab1 {  public static void main(String[] args) {  String filePath1 = "D:\\3k2s\\IB\\labs\\lab1\\CIR.txt";  String filePath2 = "D:\\3k2s\\IB\\labs\\lab1\\LAT.txt";  final String LATIN\_ALPHABET = "abcdefghijklmnopqrstuvwxyz";  final String CYRILLIC\_ALPHABET = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";  try {  String textCyrillic = loadTextFromFile(filePath1);  String textLatin = loadTextFromFile(filePath2);  double latinEntropy = calculateEntropy(textLatin, LATIN\_ALPHABET);  System.out.println("Энтропия латинского алфавита: " + latinEntropy);  double cyrillicEntropy = calculateEntropy(textCyrillic, CYRILLIC\_ALPHABET);  System.out.println("Энтропия кириллического алфавита: " + cyrillicEntropy);  String binaryFilePath = "D:\\3k2s\\IB\\labs\\lab1\\BIN.txt";  double binaryEntropy = calculateBinaryEntropy(binaryFilePath);  System.out.println("бинарная энтропия: " + binaryEntropy);    String binaryFilePath2 = "D:\\3k2s\\IB\\labs\\lab1\\BIN2.txt";  double binaryEntropy2 = calculateBinaryEntropy(binaryFilePath2);  System.out.println("бинарная энтропия: " + binaryEntropy2);    String text1 = "Смолик Валерий Александрович";  System.out.println("Количество информации в сообщении, состоящем из ФИО (RU): "+QuantityOfInformation(cyrillicEntropy,text1));  String text2 = "Smolik Valeriy Aleksandrovich";  System.out.println("Количество информации в сообщении, состоящем из ФИО (EN): "+QuantityOfInformation(latinEntropy,text2));  System.out.println("Количество информации в бинарном сообщении, состоящем из ФИО (RU): "+ calculateInformation(textToBinary(text1)));  System.out.println("Количество информации в бинарном сообщении, состоящем из ФИО (EN): "+ calculateInformation(textToBinary(text2)));    double errorProbability1 = 0.1;  double quantity1 = QuantityOfInformation(cyrillicEntropy, text1, errorProbability1);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability1 + ", RU): " + quantity1 + " бит");  double errorProbability2 = 0.5;  double quantity2 = QuantityOfInformation(cyrillicEntropy, text1, errorProbability2);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability2 + ", RU): " + quantity2 + " бит");  double errorProbability3 = 1.0;  double quantity3 = QuantityOfInformation(cyrillicEntropy, text1, errorProbability3);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability3 + ", RU): " + quantity3 + " бит");  //EN  double quantity4 = QuantityOfInformation(latinEntropy, text2, errorProbability1);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability1 + ", EN): " + quantity4 + " бит");  double quantity5 = QuantityOfInformation(latinEntropy, text2, errorProbability2);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability2 + ", EN): " + quantity5 + " бит");  double quantity6 = QuantityOfInformation(latinEntropy, text1, errorProbability3);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability3 + ", EN): " + quantity6 + " бит");    //BIN RU    //RU  double quantity7 = QuantityOfInformation(cyrillicEntropy, textToBinary(text1), errorProbability1);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability1 + ", BIN RU): " + quantity7 + " бит");  double quantity8 = QuantityOfInformation(cyrillicEntropy, textToBinary(text1), errorProbability2);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability2 + ", BIN RU): " + quantity8 + " бит");  double quantity9 = QuantityOfInformation(cyrillicEntropy, textToBinary(text1), errorProbability3);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability3 + ", BIN RU): " + quantity9 + " бит");    //EN  //RU  double quantity10 = QuantityOfInformation(latinEntropy, textToBinary(text2), errorProbability1);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability1 + ", BIN EN): " + quantity10 + " бит");  double quantity11 = QuantityOfInformation(latinEntropy, textToBinary(text2), errorProbability2);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability2 + ", BIN EN): " + quantity11 + " бит");  double quantity12 = QuantityOfInformation(latinEntropy, textToBinary(text2), errorProbability3);  System.out.println("Количество информации (вероятность ошибки: " + errorProbability3 + ", BIN EN): " + quantity12 + " бит");        } catch (FileNotFoundException e) {  System.out.println("Файл не найден: " + e.getMessage());  } catch (IOException e) {  System.out.println("Ошибка при чтении бинарного файла: " + e.getMessage());  }  }  private static String loadTextFromFile(String filePath) throws FileNotFoundException {  StringBuilder sb = new StringBuilder();  File file = new File(filePath);  try (Scanner scanner = new Scanner(file)) {  while (scanner.hasNextLine()) {  sb.append(scanner.nextLine());  }  }  return sb.toString();  }  private static double QuantityOfInformation(double entropy, String text)  {  return entropy \* text.length();  }    public static double QuantityOfInformation(double entropy, String text, double errorProbability) {  return entropy \* text.length() \* (1 - errorProbability);  }  private static double calculateEntropy(String text, String alphabet) {  Map<Character, Integer> charFrequency = new HashMap<>();  int totalCount = 0;  // Подсчет частот символов  for (char c : text.toCharArray()) {  if (alphabet.indexOf(c) != -1) {  charFrequency.put(c, charFrequency.getOrDefault(c, 0) + 1);  totalCount++;  }  }  double entropy = 0.0;  for (char c : charFrequency.keySet()) {  double probability = (double) charFrequency.get(c) / totalCount;  entropy -= probability \* (Math.log(probability) / Math.log(2));  }  return entropy;  }  private static double calculateBinaryEntropy(String filePath) throws IOException {  byte[] binaryData = loadBinaryDataFromFile(filePath);  return calculateEntropy(binaryData);  }  private static byte[] loadBinaryDataFromFile(String filePath) throws IOException {  File file = new File(filePath);  byte[] data = new byte[(int) file.length()];  try (FileInputStream fis = new FileInputStream(file)) {  fis.read(data);  }  return data;  }  private static double calculateEntropy(byte[] data) {  Map<Byte, Integer> byteFrequency = new HashMap<>();  int totalCount = 0;  // Count byte frequencies  for (byte b : data) {  byteFrequency.put(b, byteFrequency.getOrDefault(b, 0) + 1);  totalCount++;  }  // Calculate probabilities and entropy  double entropy = 0.0;  for (byte b : byteFrequency.keySet()) {  double probability = (double) byteFrequency.get(b) / totalCount;  entropy -= probability \* (Math.log(probability) / Math.log(2));  }  return entropy;  }    public static String textToBinary(String text) {  StringBuilder binary = new StringBuilder();    for (int i = 0; i < text.length(); i++) {  char c = text.charAt(i);  String binaryChar = Integer.toBinaryString(c);    // Дополнить нулями до 8 бит  while (binaryChar.length() < 8) {  binaryChar = "0" + binaryChar;  }    binary.append(binaryChar);  }    return binary.toString();  }    public static double calculateInformation(String binaryMessage) {  int length = binaryMessage.length();  double information = length \* Math.log(2);    return information;  }    } |

Листинг 5 – Полный код приложения

**Вывод.**

В ходе выполнения данной задачи по расчету количества информации и анализу параметров и информативных характеристик дискретных информационных систем, мы приобрели практические навыки, которые могут быть полезны в реальных сценариях.  
 Мы изучили понятие бинарной энтропии и ее связь с количеством информации. Бинарная энтропия позволяет оценить степень неопределенности или разнородности в сообщении, а количество информации выражает количество бит, необходимых для представления этой информации.

Мы также учли вероятность ошибочной передачи единичного бита сообщения. Это позволяет учесть потери информации при передаче данных и получить более реалистичную оценку количества информации.

Таким образом, в результате лабораторной работы, были приобретены практические навыки расчета и анализа параметров и информативных характеристик дискретных ИС.