Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Защита информации и надёжность информационных систем

Студент: Лопатнюк П.В.

ФИТ 3 курс 1 группа

Преподаватель: Нистюк О.А.

Минск 2025

**Лабораторная работа № 3**

**ЭЛЕМЕНТЫ ТЕОРИИ ИНФОРМАЦИИ. ИНФОРМАТИВНОСТЬ ДАННЫХ В РАЗЛИЧНЫХ КОДИРОВКАХ**

**Цель:** приобретение практических навыков трансформации данных и сопоставление энтропийных свойств используемых при этом алфавитов.

Теория сюда:

**Задачи:**

1. Закрепить теоретические знания по взаимной конвертации данных, представленных в кодах ASCII и base64.

2. Разработать приложение для конвертации произвольного документа в формат base64 и обратно.

3. Исследовать энтропийные характеристики используемых в конвертерах алфавитов.

4. Изучить особенности практической реализации операции XOR над данными, представленными в разных форматах.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

# Теоретические сведения

Алфавит, А – это набор знаков или символов (N), используемых для генерации или передачи сообщений.

Энтропия алфавита – это количественная мера неопределенности, связанная с появлением символов из данного алфавита в сообщении.

С физической точки зрения энтропия алфавита показывает, какое количество полезной информации приходится в среднем на один символ алфавита.

Энтропия **максимальна** при **равномерном** появлении букв на любом месте сообщения. Для характеристики источника сообщений с различным алфавитом представляет интерес сравнение фактической энтропии источника с максимально возможной.

**Избыточностью** алфавита называют уменьшение информационной нагрузки на один символ засчёт разной вероятности и взаимозависимости появления его символов в сообщениях

В наиболее общем виде избыточность алфавита R можно оценить отношением энтропии по Хартли и по Шеннону:



# Практическое задание

1. Создать собственное приложение (приветствуется!) или воспользоваться Base64-онлайн-кодировщиком, с помощью которого конвертировать произвольный документ (а) на латинице (можно использовать документ из лабораторной работы № 1) в документ (б) формата base64.

Исходный документ:

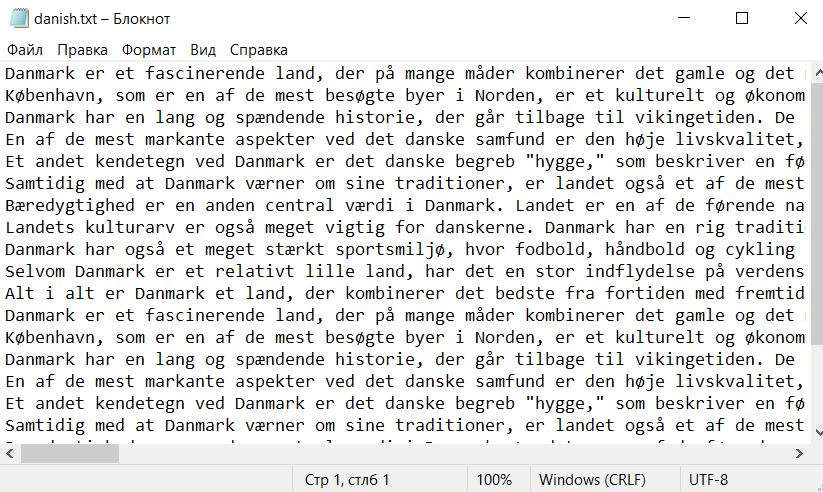


Рисунок 1.1 – Файл исходного документа

Используем сервис для кодирования файла:

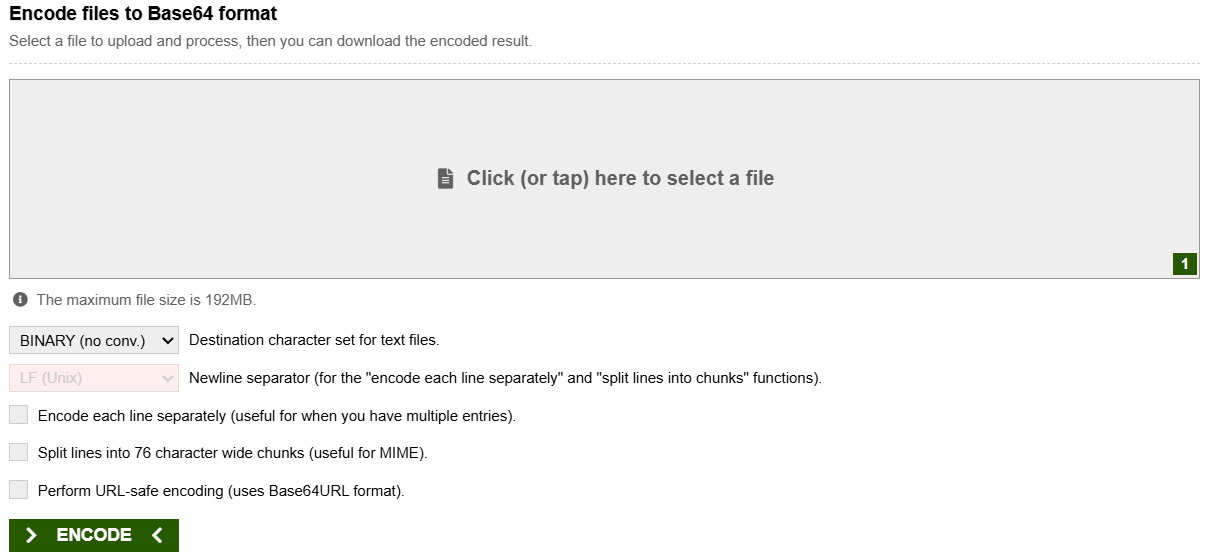


Рисунок 1.2 – Страница сервиса для кодирования

Результат:

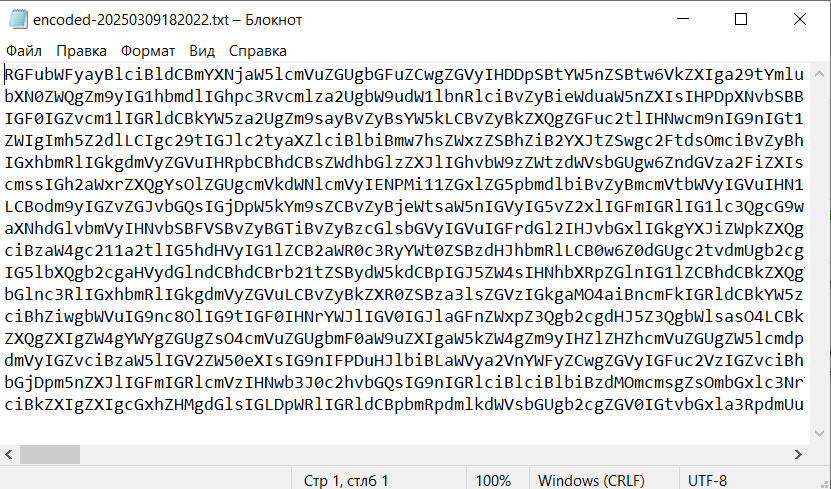


Рисунок 1.3 – Файл содержащий закодированную информацию

1. С помощью приложения, созданного в лабораторной работе № 1, получить распределение частотных свойств алфавитов по документам (а) и (б). Вычислить энтропию Хартли и Шеннона, а также избыточность алфавитов. Объяснить полученный результат.

Вычисление энтропии по Хартли:

|  |
| --- |
| public static double CalculateEntropy(string text, char[] alphabet)  {  text = new string(text.ToLower().Where(c => alphabet.Contains(c)).ToArray());  int textLength = text.Length;  if (textLength < 100)  {  Console.WriteLine("Текст слишком маленький для того, чтобы рассчитать энтропию.");  return -1;  }  var frequency = new Dictionary<char, int>();  foreach (var c in text)  {  if (frequency.ContainsKey(c))  frequency[c]++;  else  frequency[c] = 1;  }  double entropy = 0;  foreach (var kvp in frequency)  {  double probability = (double)kvp.Value / textLength;  entropy += probability \* Math.Log2(probability);  }  return -entropy;  } |

Листинг 1.1 – Вычисление энтропии по Хартли

Вычисление энтропии по Шеннону:

|  |
| --- |
| public static double CalculateEntropyHartly(char[] alphabet)  {  return Math.Log2(alphabet.Length);  } |

Листинг 1.2 – Вычисление энтропии по Шеннону

Вычисление избыточности:

|  |
| --- |
| public static double AlphabetRedundancy(double entropyShannon, double entropyHartly)  {  return (1 - entropyShannon / entropyHartly) \* 100;  } |

Листинг 1.3 – Вычисление избыточности информации

Вызов методов:

|  |
| --- |
| string danishText = FileReader.ReadTextFromFile("danish.txt");  string base64Text = FileReader.ReadTextFromFile("encoded\_danish.txt");  char[] DanishAlphabet = "abcdefghijklmnopqrstuvwxyzæøå".ToCharArray();  char[] Base64Alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=".ToCharArray();  Console.WriteLine("\n------- Энтропия датского текста -------");  double danishEntropy = EntropyCalculator.CalculateEntropy(danishText, DanishAlphabet);  double danishEntropyHartly = EntropyCalculator.CalculateEntropyHartly(DanishAlphabet);  Console.WriteLine($"Шеннон: {danishEntropy:F4}, Хартли: {danishEntropyHartly:F4}");  Console.WriteLine($"Избыточность: {EntropyCalculator.AlphabetRedundancy(danishEntropy, danishEntropyHartly):F4}%");  Console.WriteLine("----------------------------------------\n");  Console.WriteLine("------- Энтропия base64-текста -------");  double base64Entropy = EntropyCalculator.CalculateEntropy(base64Text, Base64Alphabet);  double base64EntropyHartly = EntropyCalculator.CalculateEntropyHartly(Base64Alphabet);  Console.WriteLine($"Шеннон: {base64Entropy:F4}, Хартли: {base64EntropyHartly:F4}");  Console.WriteLine($"Избыточность: {EntropyCalculator.AlphabetRedundancy(base64Entropy, base64EntropyHartly):F4}%");  Console.WriteLine("----------------------------------------\n"); |

Листинг 1.4 – Использование методов

Результат:

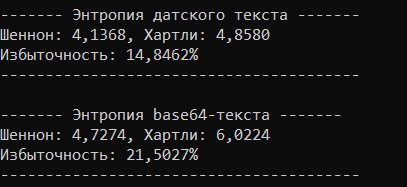


Рисунок 1.4 – Результаты вычисления энтропии

1. Написать функцию, которая принимает в качестве аргументов два буфера (а и b) одинакового размера и возвращает XOR (собственная фамилия (а) и имя (b); при разной длине меньшую дополнить нулями). Входные аргументы представлять: 1) в кодах ASCII; 2) в кодах base64. Что будет результатом операции аXORbXORb?

Вычисление XOR:

|  |
| --- |
| namespace Lab3  {  class XOR  {  public static byte[] XORBuffers(byte[] a, byte[] b)  {  byte[] result = new byte[Math.Max(a.Length, b.Length)];  for (int i = 0; i < result.Length; i++)  {  result[i] = (byte)((i < a.Length ? a[i] : 0) ^ (i < b.Length ? b[i] : 0));  }  return result;  }  public static string ToBinary(byte[] data)  {  if (data == null || data.Length == 0) return string.Empty;  return string.Join(" ", data.Select(b => Convert.ToString(b, 2).PadLeft(8, '0')));  }  }  } |

Листинг 1.5 – Вычисление XOR

Использование:

|  |
| --- |
| string surname = "Lopatniuk";  string firstname = "Polina";  byte[] surnameASCII = Encoding.ASCII.GetBytes(surname);  byte[] firstnameASCII = Encoding.ASCII.GetBytes(firstname);  string surnameBase64 = Convert.ToBase64String(surnameASCII);  string firstnameBase64 = Convert.ToBase64String(firstnameASCII);  byte[] surnameBase64Bytes = Convert.FromBase64String(surnameBase64);  byte[] firstnameBase64Bytes = Convert.FromBase64String(firstnameBase64);  byte[] resultASCII = XOR.XORBuffers(surnameASCII, firstnameASCII);  byte[] resultBase64 = XOR.XORBuffers(surnameBase64Bytes, firstnameBase64Bytes);  byte[] resultASCIIReversed = XOR.XORBuffers(resultASCII, firstnameASCII);  byte[] resultBase64Reversed = XOR.XORBuffers(resultBase64, firstnameBase64Bytes);  Console.WriteLine("------- ASCII Кодирование -------");  Console.WriteLine($"{surname}: \t {string.Join(" ", surnameASCII)}");  Console.WriteLine($"{firstname}: \t {string.Join(" ", firstnameASCII)}");  Console.WriteLine($"{surname} (2): \t {XOR.ToBinary(surnameASCII)}");  Console.WriteLine($"{firstname} (2): \t {XOR.ToBinary(firstnameASCII)}");  Console.WriteLine($"XOR (2): \t {XOR.ToBinary(resultASCII)}");  Console.WriteLine($"XOR: \t\t {string.Join(" ", resultASCII)}");  Console.WriteLine($"Re-XOR (2): \t {XOR.ToBinary(resultASCIIReversed)}");  Console.WriteLine($"Re-XOR: \t {string.Join(" ", resultASCIIReversed)}");  Console.WriteLine("----------------------------------------\n");  Console.WriteLine("------- Base64 Кодирование -------");  Console.WriteLine($"{surname}: \t {surnameBase64}");  Console.WriteLine($"{firstname}: \t {firstnameBase64}");  Console.WriteLine($"a (2): \t \t {XOR.ToBinary(surnameBase64Bytes)}");  Console.WriteLine($"b (2): \t \t {XOR.ToBinary(firstnameBase64Bytes)}");  Console.WriteLine($"a XOR b (2): \t {XOR.ToBinary(resultBase64)}");  Console.WriteLine($"a XOR b: \t {Convert.ToBase64String(resultBase64)}");  Console.WriteLine($"a XOR b XOR b2:\t {XOR.ToBinary(resultBase64Reversed)}");  Console.WriteLine($"a XOR b XOR b: \t {Convert.ToBase64String(resultBase64Reversed)}");  Console.WriteLine("----------------------------------------\n");  } |

Листинг 1.6 – Использование методов

Результат:

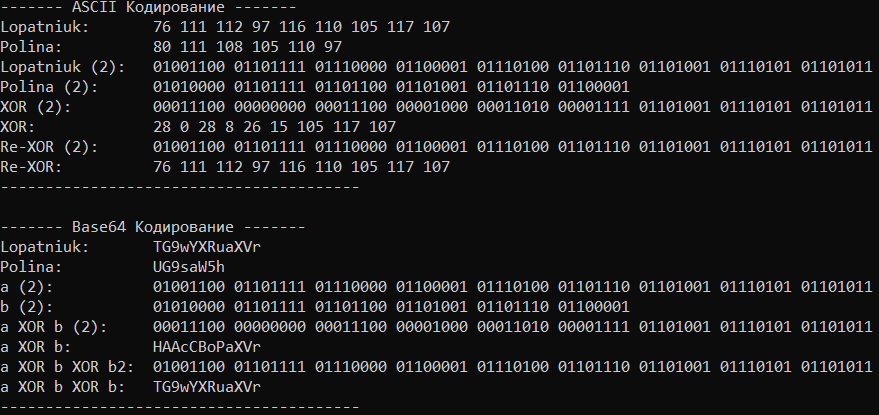


Рисунок 1.5 – Результат кодирования

# Вывод

В ходе выполнения данной лабораторной работы были закреплены теоретические знания и получены практические навыки, связанные с преобразованием данных между текстовым форматом и кодировкой Base64. Исходный текст, представленный на латинице, был успешно закодирован в формат Base64.

С использованием программных средств, разработанных в рамках предыдущей лабораторной работы, был проведён частотный анализ символов и вычислены основные энтропийные характеристики: энтропия по Шеннону, энтропия по Хартли и избыточность. Расчёты были выполнены как для исходного алфавита, так и для алфавита Base64, что позволило провести сравнительный анализ информативности и структуры данных до и после кодирования.

Дополнительно была реализована функция побитовой операции XOR, применяемая к данным, представленным в виде байтовых массивов — как на основе ASCII-кодов исходного текста, так и на основе байтов, полученных после преобразования в формат Base64. Это расширило возможности анализа и стало основой для дальнейшей реализации методов простейшего симметричного шифрования.