Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Элементы теории информации. Информативность данных в различных кодировках

Студент: Стрелковская В. А.

ФИТ 3 курс 2 группа

Преподаватель: Хартанович А. А.

Минск 2025

# Теоретические сведения

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

**Избыточность алфавита** – уменьшение информационной нагрузки на один символ вследствие 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:



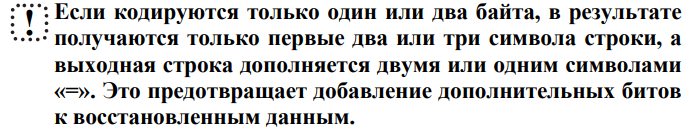
Дополнительная избыточность обусловлена переносом сообщения из одной среды в другую или, иначе говоря, кодированием символов исходного алфавита.

Утверждение восьмибитных кодировок (ASCII) как стандарта принесло некоторые проблемы. К этому моменту уже существовала определенная инфраструктура, использующая семибитные кодировки.

Утверждение восьмибитного символа дало 256 различных значений, что позволило уместить в одной кодовой таблице и общепринятые символы (цифры, знаки препинания, латиницу), и символы кириллицы.

Процесс кодирования представляет группу из 24 последовательных битов в форме строки из 4 символов. Обработка выполняется слева направо, а 24-битная исходная группа образуется конкатенацией трех 8-битных групп (байтов). Данные 24 бита после этого трактуются как 4 сцепленных группы по 6 битов, каждая из которых транслируется в один символ алфавита base64. Каждая 6-битная группа используется в качестве индекса массива из 64 печатных символов.

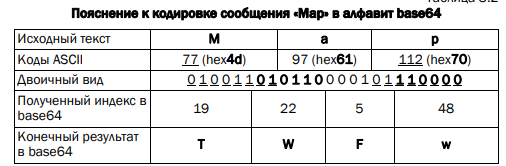
Каждые 6 битов буфера, начиная с самых старших, используются как индексы строки «ABCDEFGHIJKLMNOPQRSTUV WXYZabcdefghijklmnopqrstuvwxyz0123456789+/», и ее символы, на которые указывают индексы, помещаются в выходную строку.

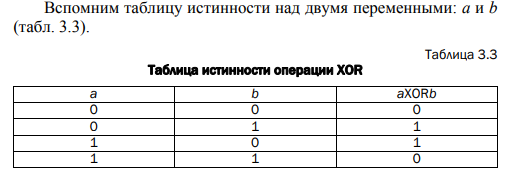


Если на входе доступно менее 24 битов, входная группа дополняется (справа) нулями до формирования целого числа 6-битных групп. Заполнение в конце данных осуществляется как раз с использованием символа «=». Поскольку входная информация base64 всегда включает целое число октетов, возможны лишь перечисленные ниже случаи:

* размер финального блока кодирования на входе кратен 24 битам, кодированный результат будет содержать целое число 4-символьных групп без заполнения символами «=»;
* размер финального блока кодирования на входе составляет 8 битов, выходной блок будет представлять 2 символа, дополненные последовательностью из двух символов заполнения «==»;
* размер финального блока кодирования на входе составляет 16 битов, выходной блок будет представлять 3 символа, дополненные символом заполнения «=».







# Практическое задание

# 2.1 Энтропия алфавита

Для чтения данных из файла мы сохраняем путь к исходному файлу и файлу после кодирования. Представлено на рисунке 2.1.

string filePath = "input.txt"; // Исходный файл

string base64FilePath = "output\_base64.txt"; // Файл после кодировки

// Чтение исходного файла

string inputText = File.ReadAllText(filePath, Encoding.UTF8);

Console.WriteLine("Исходный текст:\n" + inputText);

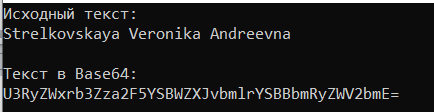


Рисунок 2.1 – Чтение данных из файла

Далее выполняем кодирование в Base64. Представлено на рисунке 2.2.

string base64Encoded = Convert.ToBase64String(Encoding.UTF8.GetBytes(inputText));

File.WriteAllText(base64FilePath, base64Encoded);

Console.WriteLine("\nТекст в Base64:\n" + base64Encoded);

Рисунок 2.2 – Кодирование в Base64

Для выполнения анализа частоты символов используется функция, представленная на рисунке 2.3.

static void AnalyzeFrequency(string text, string description)

{

var frequency = text.GroupBy(c => c)

.ToDictionary(g => g.Key, g => g.Count());

Console.WriteLine($"\nЧастотный анализ ({description}):");

foreach (var kvp in frequency.OrderByDescending(x => x.Value))

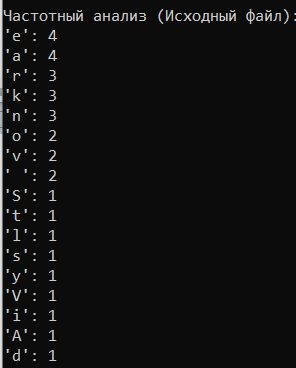
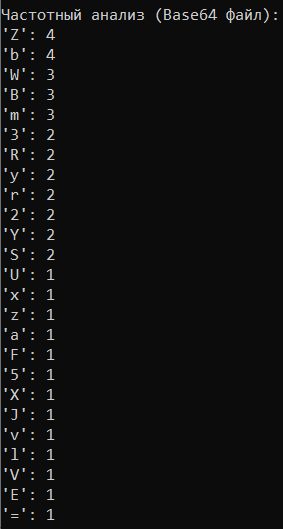
{

Console.WriteLine($"'{kvp.Key}': {kvp.Value}");

}

}

Рисунок 2.3 – Функция анализа частоты символов

Результат

Следующий шаг – вычисление энтропии по Шеннону и Хартли. Функции представлены на рисунке 2.4.

static double CalculateEntropy(string text)

{

var frequency = text.GroupBy(c => c)

.ToDictionary(g => g.Key, g => (double)g.Count() / text.Length);

return -frequency.Values.Sum(p => p \* Math.Log2(p));

}

static double CalculateHartleyEntropy(int length)

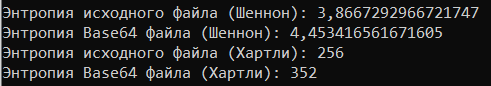
{

// Максимальная энтропия для ASCII (256 символов)

return Math.Log2(256) \* length;

}

Рисунок 2.4 – Функции вычисление энтропии по Шеннону и Хартли



Далее было выполнено вычисление избыточности. Представлено на рисунке 2.5.

static double CalculateRedundancy(double entropy, double maxEntropy)

{

return maxEntropy - entropy;

}

Рисунок 2.5 – Вычисление избыточности



Также было необходимо выполнить операцию XOR. Функция для её вычисления представлена на рисунке 2.6:

static string XORStrings(string str1, string str2)

{

int length = Math.Max(str1.Length, str2.Length); // Определяем длину, равную большей из двух строк

char[] result = new char[length];

for (int i = 0; i < length; i++)

{

char char1 = i < str1.Length ? str1[i] : '\0'; // Дополнение нулями

char char2 = i < str2.Length ? str2[i] : '\0'; // Дополнение нулями

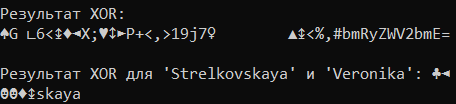
result[i] = (char)(char1 ^ char2);

}

return new string(result);

}

Рисунок 2.6 – Функция выполнения операции XOR



Последнее выполненное задание заключалось в функции поиске результата операции aXORbXORb. Результат представлен на рисунке 2.7.

// Пример использования функции XOR для фамилии и имени

string surname = "Strelkovskaya"; // Фамилия

string name = "Veronika"; // Имя

string xorNameResult = XORStrings(surname, name);

Console.WriteLine($"\nРезультат XOR для '{surname}' и '{name}': {xorNameResult}");

// Результат операции a XOR b XOR b

string finalResult = XORStrings(xorNameResult, name);

Console.WriteLine($"Результат операции '{surname} XOR {name} XOR {name}':

{finalResult}");



Рисунок 2.7 – Результат операции aXORbXORb

**Вывод:** В данной лабораторной работе были изучены основы теории информации и проведен анализ энтропийных свойств алфавитов. Мы рассмотрели процесс кодирования данных в Base64 и выяснили, как это влияет на информационную нагрузку. Выполнение частотного анализа, а также вычисление энтропии по Шеннону и Хартли позволило оценить избыточность алфавита. Кроме того, была реализована операция XOR для сравнения строк и исследования их взаимосвязи. Результаты показали, что различные кодировки влияют на эффективность передачи информации, что важно учитывать при разработке систем обработки данных.