МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

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

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

                                                                 Выполнил студент 3 курса 5 группы

Дмитрук Илья Игоревич

Минск 2024

# Задание 1.

Для выполнения данного задания, был выбран текст на турецком языке. Для преобразования данного текста в Base64, была разработана функция ConvertToBase64. Код функции ConvertToBase64 представлен в листинге 1.1.

public static string ConvertToBase64(string text)

{

byte[] bytes = Encoding.UTF8.GetBytes(text);

return Convert.ToBase64String(bytes);

}

Листинг 1.1 – Функция для преобразования в Base64

# Задание 2. Частотные свойства, энтропия, избыточность

После преобразования текста в Base64, были подсчитаны количества встречи разных символов в Base64. Гистограмма распределения символов в Base64 представлена на рисунке 2.1.

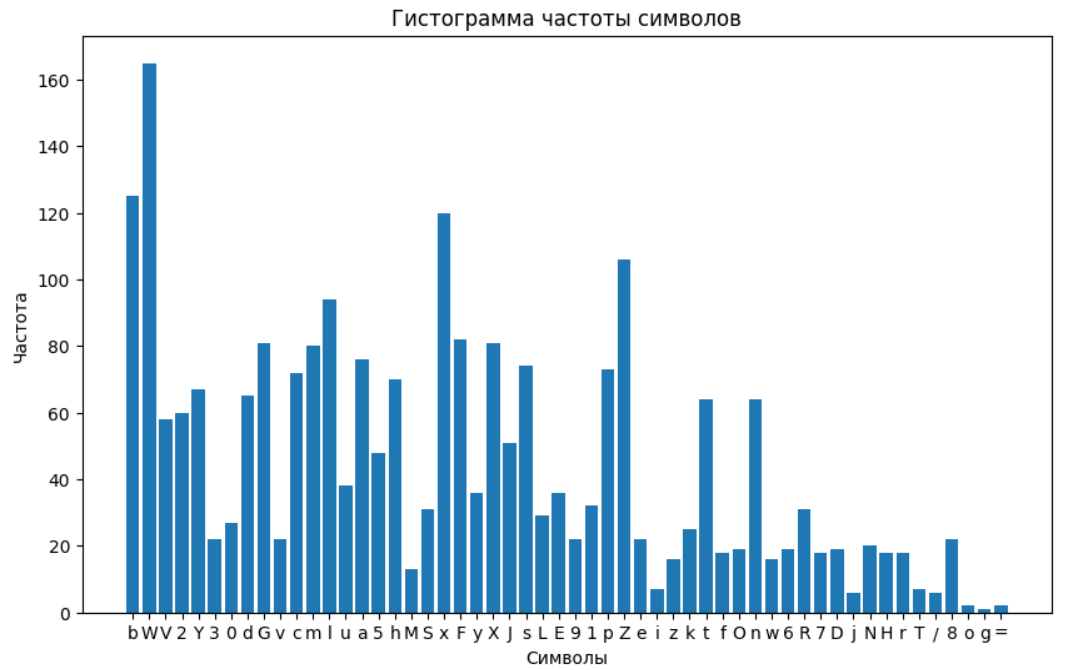


Рисунок 2.1 – Гистограмма распределения символов в Base64

Для нахождения энтропии Шеннона, была разработана функция GetShannonEntropy. Код данной функции представлен в листинге 2.1.

public static double GetShannonEntropy(string text, Dictionary<char, int>

numberSymbols)

{

double entropy = 0d;

foreach (var symbol in numberSymbols)

{

double P = (double)symbol.Value / text.Length;

entropy -= P \* Math.Log2(P);

}

return Math.Round(entropy, 3);

}

Листинг 2.1 – Функция для вычисления энтропии Шеннона

Для нахождения энтропии Хартли, была разработана функция GetHartlyEntropy. Код данной функции представлен в листинге 2.2.

public static double GetHartlyEntropy(string text, Dictionary<char, int>

numberSymbols)

{

return Math.Round(Math.Log2(numberSymbols.Count), 3);

}

Листинг 2.2 – Функция для вычисления энтропии Хартли

Избыточностью алфавита называют 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.

Формула избыточности:

Для вычисления избыточности алфавита, была разработана функция GetRedundancy. Её код представлен в листинге 2.3.

public static double GetRedundancy(string text, Dictionary<char, int>

numberSymbols)

{

return Math.Round(((GetHartlyEntropy(text, numberSymbols) -

GetShannonEntropy(text, numberSymbols)) / GetHartlyEntropy(text,

numberSymbols)) \* 100, 3);

}

Листинг 2.3 – Код вычисления избыточности

Результаты вычисления двух энтропий для турецкого и английского текста, а также их избыточности, представлены на рисунке 2.2.

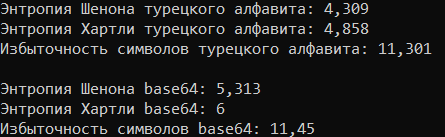


Рисунок 2.2 – Результаты вычислений

Как мы видим, избыточность Base 64 выше, чем избыточность турецкого алфавита, следовательно турецкий алфавит имеет большую информационную нагрузку на один символ, чем Base64.

# Задание 3.

Для выполнения данного задания, была разработана функция XOR. Она принимает два строковых параметра, преобразует их в байтовое представление, заполняет нулями байты в том представлении, которое меньше, чтобы длины обоих представлений были одинаковыми, затем проводит между ними операцию XOR. Результат данной операции, возвращается в виде строки. Функция представлена на листинге 3.1

public static string XOR(string a, string b)

{

byte[] bytesA = Encoding.UTF8.GetBytes(a);

byte[] bytesB = Encoding.UTF8.GetBytes(b);

byte[] result;

if (bytesA.Length > bytesB.Length)

{

Array.Resize(ref bytesB, bytesA.Length);

result = new byte[bytesA.Length];

}

else

{

Array.Resize(ref bytesA, bytesB.Length);

result = new byte[bytesB.Length];

}

for (int i = 0; i < result.Length; i++)

{

result[i] = (byte)(bytesA[i] ^ bytesB[i]);

}

return Encoding.UTF8.GetString(result);

}

Листинг 3.1 – Функция для выполнения операции XOR

Результат приводится к кодам ascii в бинарном виде. Для этого была разработана функция ToAsciiBinary. Её код представлен в листинге 3.2.

public static string ToAsciiBinary(string text)

{

string asciiBinary = "";

foreach (var c in text)

{

string bits = Convert.ToString((int)c, 2);

while (bits.Length < 8)

{

bits = bits.Insert(0, "0");

}

asciiBinary += bits;

}

return asciiBinary;

}

Листинг 3.2 – Функция для преобразования строки в представление ascii в бинарном виде

Так же необходимо было найти XOR между двумя строками в Base64. Для преобразования строк к Base64, использовалась функция ConvertToBase64, которая была описана ранее.

Строки, передаваемые функции XOR, представляют с собой мою фамилию и имя. Так же необходимо было выполнить операцию XOR с результатом выполнения данной функции и именем. Для этого первым параметром в функцию XOR, передаём результат выполнения, а вторым имя. Результат данных вычислений представлен на рисунке 3.1.

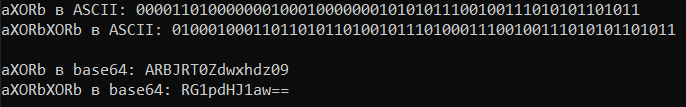


Рисунок 3.1 – Результат работы функции XOR

В результате повторного выполнения операции XOR, мы получили обратно текст, представляющий с собой мою фамилию.

# Вывод

В ходе лабораторной работы были приобретены практические навыки кодирования данных и нахождения избыточности алфавита. А также была изучена кодировка Base64 и освоен оператор XOR.