**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.

**Теоретические сведения**

Для характеристики источника сообщений с различным алфавитом представляет интерес сравнение фактической энтропии источника с максимально возможной. В этом смысле введено понятие **избыточности источника сообщений, или избыточности алфавита.**

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

Одно и то же сообщение на основе алфавита русского и представленное в кодах ASCII будут характеризоваться различным количеством содержащейся в них информации. Дополнительная избыточность обусловлена переносом сообщения из одной среды в другую или, иначе говоря, кодированием символов исходного алфавита.

Ранее почтовый сервер при передаче письма обнулял старшие биты в каждом байте сообщения. Одним из решений проблемы стала кодировка (а точнее – алгоритм) **base64**. Алгоритм base64 используется для кодирования бинарных данных.

Кодирование base64 разработано для **представления произвольных последовательностей октетов в форме, позволяющей использовать строчные и прописные буквы.** Используется 65-символьное подмножество набора символов US-ASCII, обеспечивающее представление **одним печатным символом 6 битов данных** (дополнительный 65-й символ используется для обозначения функции специальной обработки).

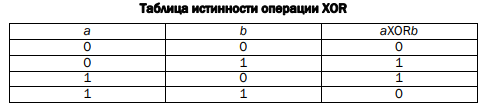
Процесс кодирования:

1. Группа из 24 последовательных битов в форме строки из 4 символов.
2. Обработка выполняется слева направо.
3. 24-битная исходная группа образуется конкатенацией трех 8-битных групп.
4. 24 бита после трактуются как 4 сцепленных группы по 6 битов, каждая группа – символ алфавита base64.
5. Каждая 6-битная группа используется в качестве индекса массива из 64 печатных символов.
6. Символы алфавита, соответствующие индексу, помещаются в выходную строку.

Если кодируются только один или два байта, в результате получаются только первые два или три символа строки, а выходная строка дополняется двумя или одним символами «=». Это предотвращает добавление дополнительных битов к восстановленным данным.



Операция XOR (вспомним, что эта операция называется также сложением по модулю 2, логическим сложением, исключающим «ИЛИ», строгой дизъюнкций, поразрядным дополнением).



Если а и b имеют длину более 1 бита, к примеру 1 байт, то рассматриваемая операция над ними выполняется побитово. Указанным байтам могут соответствовать символы в определенной кодировке. Положим, символу «М» (hex4d) соответствует 8-битный код 01001101 (см. табл. 3.2), а символу «а» (hex61) соответствует код 01100001, тогда операция сложения по модулю 2 этих двух бинарных кодов дает 00101100 ((hex2с), или символ «,».

**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. В качестве входных данных можно использовать указанный преподавателем вариант из списка: • входные параметры; • **текстовый файл (\*.txt)**; • документ Word (\*.doc); • документ Word (\*.docx); • документ PowerPoint (\*.ppt, \*.pptx); • архив (\*.zip); • текстовая строка; • случайное число (от 999999); • PDF-файл; • архив (\*.rar); • архив (\*.7z).

Программа проверяет существование входного файла с помощью fs.existsSync(filePath). Если файл найден, его содержимое считывается в переменную fileContent с помощью метода fs.readFileSync(filePath). Затем данные преобразуются в строку в формате Base64 с помощью метода toString('base64'). Полученная строка записывается в новый файл outputFile с помощью fs.writeFileSync(outputFile, base64String, 'utf-8').

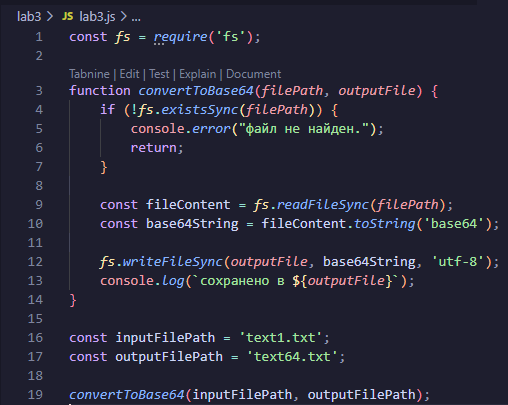


Рисунок 1.1 – Код программы 1



Рисунок 1.2 – Исходный текстовый документ

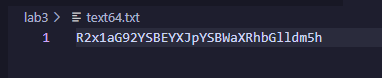


Рисунок 1.3 – Преобразованный в base64

Таким образом, исходный файл конвертируется в строку Base64 и сохраняется в текстовом формате.

2. 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.

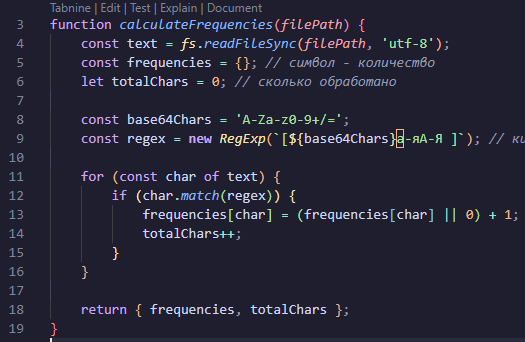


Рисунок 1.4 – Распределение частотных свойств алфавитов

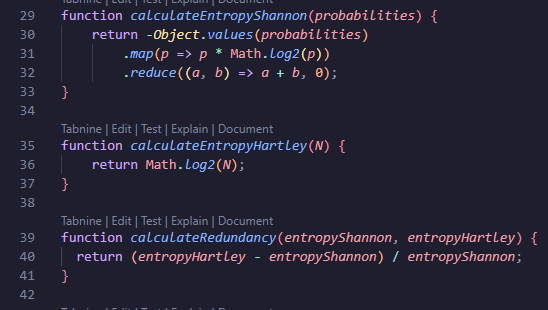


Рисунок 1.5 – Вычисление энтропий и избыточности

Программа сначала считывает содержимое файла и определяет частоту встречаемости символов, принадлежащих как Base64-алфавиту, так и кириллическим символам. На основе этих частот вычисляются вероятности появления каждого символа. Далее рассчитывается энтропия Шеннона, которая отражает среднее количество информации на символ, а также энтропия Хартли, зависящая от количества уникальных символов в алфавите. Итоговый коэффициент избыточности показывает, насколько информация в тексте избыточна относительно максимально возможного уровня неопределенности.

Результаты анализа были получены для двух документов: исходного (а) и преобразованного в Base64 (б). Из-за особенностей кодировки Base64 алфавит преобразованного файла становится больше, а распределение символов изменяется, что влияет на значение энтропии. Обычно **энтропия Шеннона для Base64-кодированного файла выше**, чем у исходного, так как символы распределены более равномерно. В то же время **избыточность алфавита у Base64-кода меньше**, так как его алфавит приближается к максимально возможному разнообразию символов. Разница между энтропией по Шеннону (4.2889 бит) и энтропией по Хартли (4.3923 бит) невелика, что говорит о достаточно равномерном распределении символов в анализируемом тексте. **Энтропия по Хартли (4.3923 бит)** – это теоретический максимум неопределенности, который зависит только от количества уникальных символов в алфавите. **Энтропия по Шеннону (4.2889 бит)** – это мера реальной неопределенности в тексте, учитывающая частоты появления символов. Разница в значениях небольшая (около **0.1 бита**), что свидетельствует о том, что текст **близок к равномерному распределению**, но имеет небольшие отклонения. Если бы символы встречались абсолютно равномерно, энтропии были бы равны. Это подтверждает, что в документе нет значительного преобладания отдельных символов, однако небольшая предсказуемость все же присутствует.

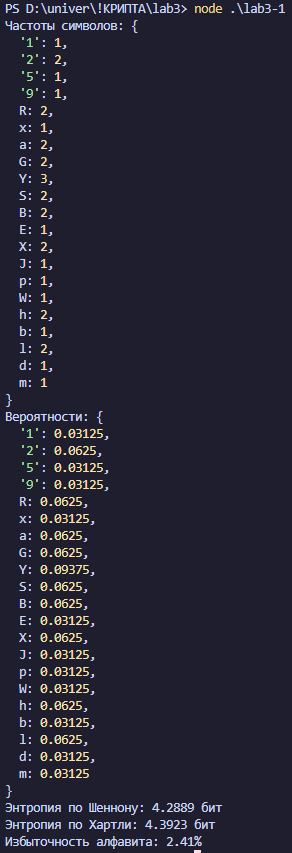
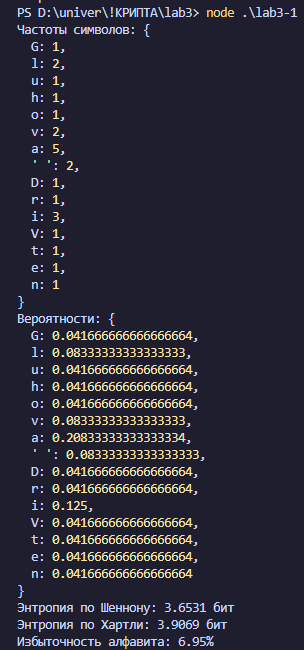


Рисунок 1.6 – Результат работы для двух документов(исходный – слева, преобразованный – справа)

3. 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?

При написании не использовать стандартные функции языка программирования. Итоговые данные сравнить с результатами использования стандартных функций языка программирования (если они есть).

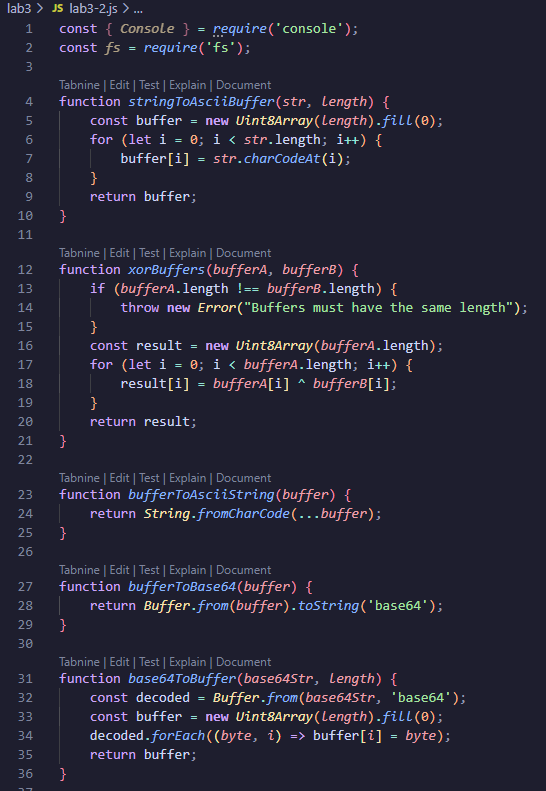


Рисунок 1.7 – Функция для выполнения операции XOR

Код выполняет операцию побитового XOR над двумя строками (фамилией и именем) в формате ASCII и Base64. Вначале строки конвертируются в массивы байтов, где меньшая строка дополняется нулями. Затем выполняется XOR между двумя массивами, полученный результат снова подвергается XOR с одним из исходных массивов, что по свойству операции приводит к восстановлению первоначального значения. Этот процесс повторяется и для кодировки Base64. Итоговые данные выводятся в виде числовых массивов, строк и закодированных значений. Код демонстрирует, что применение XOR дважды с одним и тем же аргументом приводит к исходному значению, что подтверждает обратимость операции.

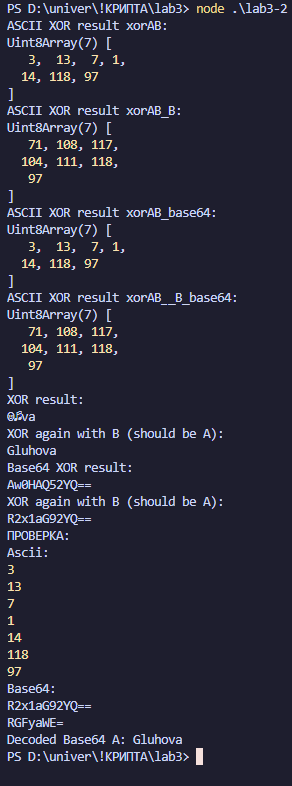


Рисунок 1.8 – Результат работы кода

**Вывод**

Операция A⊕B создает новый буфер (xorAB), содержащий побитовую разницу между символами фамилии и имени.

Операция (A⊕B) восстанавливает исходное значение A, так как при повторном применении XOR с тем же ключом 0происходит обратное преобразование. В обоих случаях (ASCII и Base64) фамилия была восстановлена.

Base64-кодирование не меняет суть XOR-операции, оно лишь представлено в другом формате, а результат XOR в Base64 (Aw0HAQ52YQ==) аналогичен ASCII-операции.