**Основные алгоритмы синхронного шифрования**

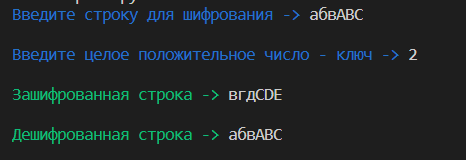
**Цель работы**

Познакомиться с механизмами реализации самых простых алгоритмов шифрования и дешифрования текста.

**Задания для выполнения**

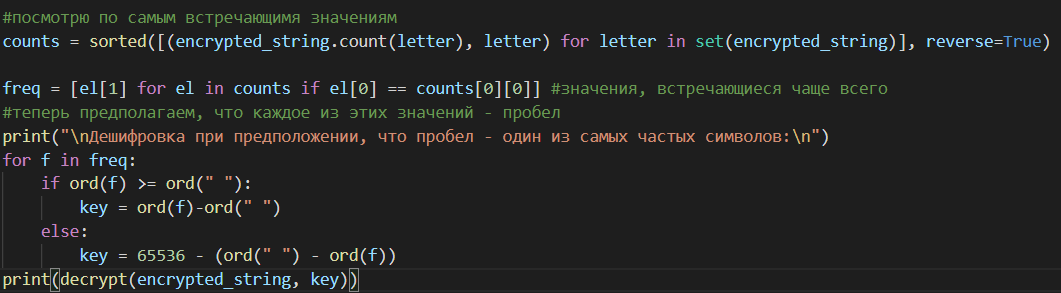
1. Написать функцию шифрования и дешифрования текста обобщенным шифром Цезаря.

Программа так и считает: получает на вход строку, которую надо зашифровать и целочисленный ключ и сдвигает каждую букву на целочисленный ключ (решение в одну строчку), затем аналогично дешифровывает результат. Пример работы:

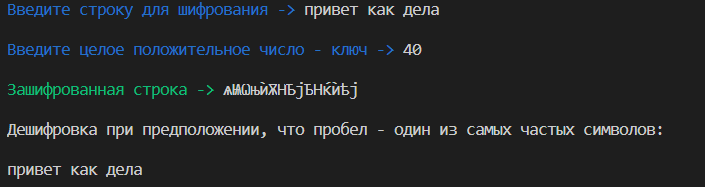


1. Написать функцию, принимающую шифротекст, зашифрованный шифром из предыдущего задания и восстанавливающий текст, без знания ключа.

Программа строится на том, что самый часто встречающийся символ – пробел (так было написано в методичке), поэтому я нашла все элементы, которые встретились больше всего раз (буквы в слове могут повторяться). Теперь я предполагаю, что каждый из этих самых часто встречающихся элементов – пробел и ищу смещение относительного его, а затем пытаюсь декодировать:

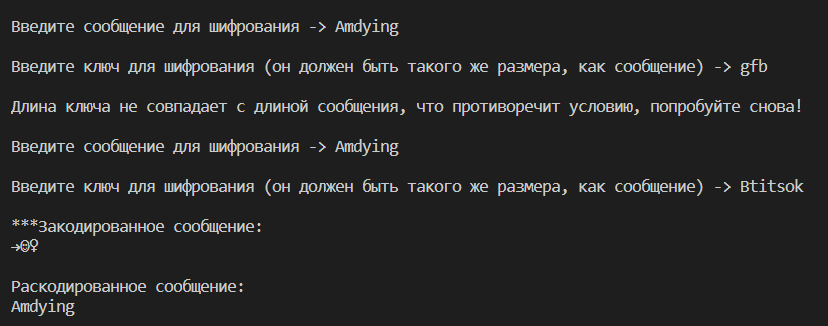


Тогда, когда пробел действительно один из самых встречающихся символов, программа определяет зашифрованную строку верно:



1. Реализовать в виде функций шифр Вернама.

Здесь каждый символ сообщения умножается на соответствующий символ ключа и путём xor’а получается код символа и затем отображается:



Также идёт проверка на соответствие длины ключа длине сообщения, ведь это обязательное условие.

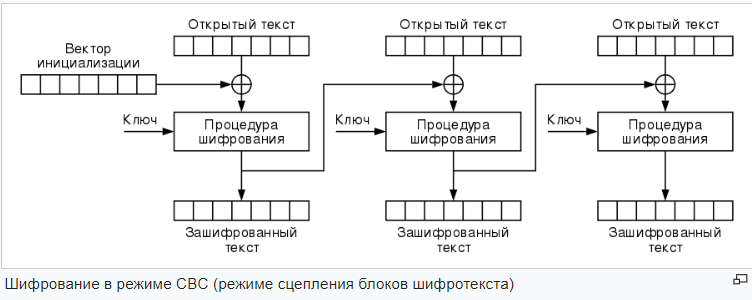
**Дополнительные задания**

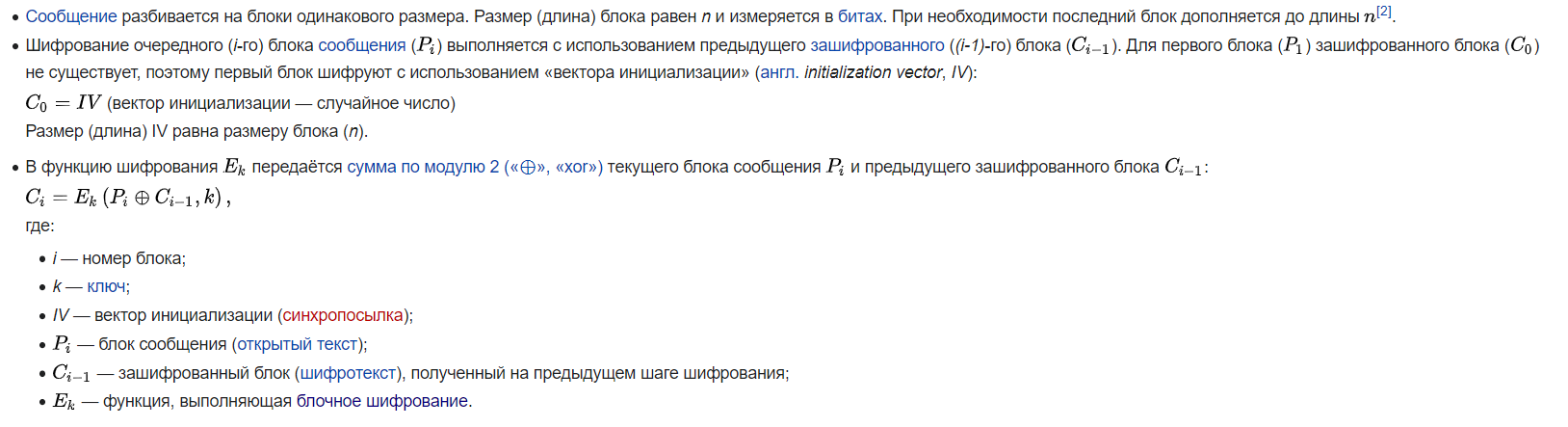
1. Реализуйте алгоритм шифрования OTP (one time pad).



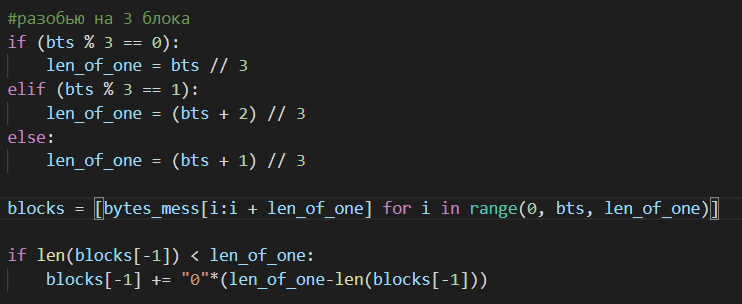
(это тот же Вернам)

1. Реализуйте алгоритм цепочки блоков (Cipher block chaining)

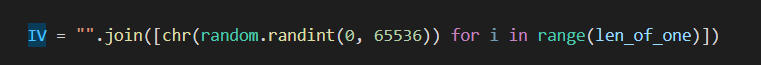




Я разбиваю исходное сообщение на блоки одинакового размера, на 3 штуки, и дополняю последний блок нулями спереди, если не хватает символов:



Вектор инициализации генерируется рандомно:



Дальше для каждого блока считывается ключ, всё преобразуется к одному размеру и вычисляется (я использую в качестве функции предыдущее задание) и получается так:

