Математические основы защиты информации и информационной безопасности. Отчет по лабораторной работе №1

Шифры простой замены

Терентьев Егор Дмитриевич 1132236902

Содержание

List of Figures

List of Tables

1 Цель работы

Освоить на практике шифры простой замены.

2 Выполнение лабораторной работы

Требуется реализовать:

1. Шифр Цезаря с произвольным ключом K.
2. Шифр Атбаш.

2.1 Шифр цезаря

Шифр Цезаря (также он является шифром простой замены) — это моноалфавитная подстановка, т.е. каждой букве открытого текста ставится в соответствие одна буква шифртекста. На практике при создании шифра простойзамены в качестве шифроалфавита берется исходный алфавит, но с нарушеннымпорядком букв (алфавитная перестановка). Для запоминания нового порядкабукв перемешивание алфавита осуществляется с помощью пароля. В качествепароля могут выступать слово или несколько слов с неповторяющимися буквами. Шифровальная таблица состоит из двух строк: в первой записывается стандартный алфавит открытого текста, во второй — начиная с некоторой позицииразмещается пароль (пробелы опускаются), а далее идут в алфавитном порядке оставшиеся буквы, не вошедшие в пароль. В случае несовпадения начала пароля с началом строки процесс после ее завершения циклически продолжается с первой позиции. Ключом шифра служит пароль вместе с числом, указывающим положение начальной буквы пароля.

Чтобы реализовать программу был написал след. код на python:

Функция проверяющая правильность введенного ключа: это значение int и если оно больше значения алфавита, то получаем остаток fig. 2.1.

```
def is_key(key):
    try:
        key = int(key)
        if key > len(alphabet):
            key %= len(alphabet)
        return key
    except ValueError:
        print("key is not int")
        raise ValueError
```

Figure 2.1: is_key

Функция проверяющая введенное значения для пароля и слова для шифрования: нужно чтобы каждая буква входила в алфавит fig. 2.2.

```
def is_correct_input_word_value(word, alphabet):
    for letter in word:
        if letter not in alphabet:
            print("Incorrect input word value")
            raise ValueError
```

Figure 2.2: correct_input_word

Функция добавляющая только уникальные буквы из слова (нужно для пароля) и функция создающая шифр-алфавит fig. 2.3.

```python
def unique_letters(password):
    unique = []
    for letter in password:
        if letter not in unique:
            unique.append(letter)
    return unique

def create_code_alpabet(alphabet, uniquePassLetters, key):
    codeAlphabet = []
    alphabet_without_upl = []
    for letter in alphabet:
        if letter not in uniquePassLetters:
            alphabet_without_upl.append(letter)

    codeAlphabet = alphabet_without_upl[-key:]
    codeAlphabet += uniquePassLetters
    codeAlphabet += alphabet_without_upl[:-key]
    return codeAlphabet
```

Figure 2.3: unique_let_and_cipher_alphabet

Функция шифрующая слово по шифр алфавиту fig. 2.4

```python
def word_to_code(word, alphabet, codeAlphabet):
    codeWord = ""
    for letter in word:
        for i in range(len(alphabet)):
            if alphabet[i] == letter:
                codeWord += codeAlphabet[i]
                break
    return codeWord
```

Figure 2.4: word_to_code

Основная фунция запуска где получаем входные значения и шифруем слово fig. 2.5

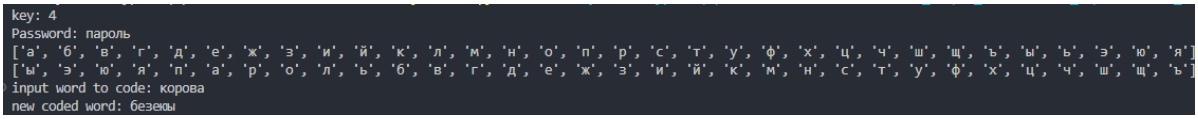
```
alphabet = ['а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п',
            'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']
key = input("key: ")
key = is_key(key=key)
password = input("Password: ").lower()
is_correct_input_word_value(word=password, alphabet=alphabet)
uniquePassLetters = unique_letters(password=password)
codeAlphabet = create_code_alpabet(alphabet=alphabet, uniquePassLetters=uniquePassLetters, key=key)

print(alphabet)
print(codeAlphabet)

wordToCode = input("input word to code: ")
is_correct_input_word_value(wordToCode, alphabet)
codeWord = word_to_code(word=wordToCode, alphabet=alphabet, codeAlphabet=codeAlphabet)
print("new coded word: " + codeWord)
```

Figure 2.5: main_func_caesar

Пример работы шифра как было показано в материалах к лабораторной работе fig. 2.6



```
key: 4
Password: пароль
['а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']
['ы', 'э', 'ю', 'я', 'п', 'а', 'р', 'о', 'л', 'ь', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'м', 'н', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ']
input word to code: корова
new coded word: безекы
```

Figure 2.6: output_caesar

2.2 Шифр атбаш

Данный шифр является шифром сдвига на всю длину алфавита, состоящего из русских букв и пробела.

Чтобы реализовать программу был написал след. код на python:

Здесь реализована функция для проверки что каждая буква входит в алфавит (для входных значений), разворот алфавита, и кодировка слова с помощью шифр-алфавита fig. 2.7

```python
def are_letters_from_alphabet(password):
    for letter in password:
        if letter not in alphabet:
            print("Incorrect input value")
            raise ValueError

def atbash_reverse(alphabet):
    return alphabet[::-1]

def word_to_code(word, alphabet, codeAlphabet):
    codeWord = ""
    for letter in word:
        for i in range(len(alphabet)):
            if alphabet[i] == letter:
                codeWord += codeAlphabet[i]
                break
    return codeWord
```

Figure 2.7: atbash_func

Основная фунция запуска где получаем входные значения и шифруем слово fig. 2.8

```python
alphabet = ['а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п',
            'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я', ' ']
atbashAlphabet = atbash_reverse(alphabet=alphabet)

print(alphabet, "\n", atbashAlphabet)

wordToCode = input("input word to code: ")
are_letters_from_alphabet(wordToCode)

codedword = word_to_code(word=wordToCode, alphabet=alphabet, codeAlphabet=atbashAlphabet)
print("coded word: " + codedword)
```

Figure 2.8: mainFunc_atbash

Пример работы шифра fig. 2.9

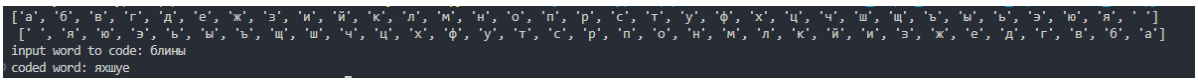


Figure 2.9: output_atbash

3 Выводы

В результате выполнения работы я освоил на практике применение шифров простой замены.

4 Список литературы

1. Методические материалы курса