УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Лабораторная работа №1.1

Дисциплина «Информационная безопасность»

Вариант 3

Выполнил: студент группы P34131

Кузнецов Максим Александрович

Преподаватель:

Маркина Татьяна Анатольевна

Санкт-Петербург, 2023 г.

Цель работы

Изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

Задание

Реализовать шифрование и дешифрацию файла с использованием метода биграмм. Ключевое слово вводится.

Листинг разработанной программы

encode.py

```
from tools import *


def encode(keyword, origin_phrase, should_extend=False, print_steps=False, print_table=False) ->
str:
    """
    @:param keyword -- ключевое слово, по которому будет строиться матрица и шифроваться слово

    @:param origin_phrase -- оригинальное слово, которое будет зашифровано

    @:param should_extend -- False по-умолчанию, флаг, который отвечает за разделение биграммы из
двух одинаковых букв

    @:param print_steps -- False по-умолчанию, флаг, который отвечает за показ пошагового
конвертирования биграмм

    @:param print_table -- False по-умолчанию, флаг, который отвечает за отображение матрицы по
ключевому слову
    """
    origin_phrase = ''.join(re.findall('[А-ЯЁ .,]', origin_phrase))
    if should_extend:
        if len(origin_phrase) > 1:
            for i in range(0, len(origin_phrase), 2):
                if origin_phrase[i] == origin_phrase[i + 1]:
                    origin_phrase = origin_phrase[:i + 1] + 'Я' + origin_phrase[i + 1:]

    if len(origin_phrase) % 2 == 1:
        origin_phrase = origin_phrase + " "

    print(f'Итоговая фраза: "{origin_phrase}"')

    enc_phrase = ""
    matrix = build_table(keyword, print_table)
    for i in range(0, len(origin_phrase), 2):
        first_y, first_x = find_position(matrix, origin_phrase[i])
        second_y, second_x = find_position(matrix, origin_phrase[i + 1])

        if first_x == second_x:
            enc_phrase = enc_phrase + matrix[0 if first_y == 5 else first_y + 1][first_x]
            enc_phrase = enc_phrase + matrix[0 if second_y == 5 else second_y + 1][second_x]
```

```
        elif first_y == second_y:
            enc_phrase = enc_phrase + matrix[first_y][0 if first_x == 5 else first_x + 1]
            enc_phrase = enc_phrase + matrix[second_y][0 if second_x == 5 else second_x + 1]

        else:
            enc_phrase = enc_phrase + matrix[first_y][second_x]
            enc_phrase = enc_phrase + matrix[second_y][first_x]
        if print_steps:
            print(f'{origin_phrase[i]}{origin_phrase[i + 1]}-->{enc_phrase[i]}{enc_phrase[i + 
1]}')

    print(f'Зашифрованная итоговая фраза: {enc_phrase}')
    return enc_phrase
```

decode.py

```
from tools import *


def decode(keyword, origin_phrase, print_steps=False, print_table=False) -> str:
    """
    @:param keyword -- ключевое слово, по которому будет строиться матрица и дешифроваться слово

    @:param origin_phrase -- оригинальное слово, которое будет дешифровано

    @:param print_steps -- False по-умолчанию, флаг, который отвечает за показ пошагового 
конвертирования биграмм

    @:param print_table -- False по-умолчанию, флаг, который отвечает за отображение матрицы по 
ключевому слову
    """
    temp = len(origin_phrase)

    origin_phrase = ''.join(re.findall('[А-ЯЁ ,.]', origin_phrase))
    if temp != len(origin_phrase) or temp % 2 == 1:
        raise Exception(f'ОШИБКА: нечетное количество символом, либо недопустимые элементы')

    print(f'Итоговая фраза: "{origin_phrase}"')

    dec_phrase = ""
    matrix = build_table(keyword, print_table)
    for i in range(0, len(origin_phrase), 2):
        first_y, first_x = find_position(matrix, origin_phrase[i])
        second_y, second_x = find_position(matrix, origin_phrase[i + 1])

        if first_x == second_x:
            dec_phrase = dec_phrase + matrix[5 if first_y == 0 else first_y - 1][first_x]
            dec_phrase = dec_phrase + matrix[5 if second_y == 0 else second_y - 1][second_x]

        elif first_y == second_y:
            dec_phrase = dec_phrase + matrix[first_y][5 if first_x == 0 else first_x - 1]
            dec_phrase = dec_phrase + matrix[second_y][5 if second_x == 0 else second_x - 1]

        else:
            dec_phrase = dec_phrase + matrix[first_y][second_x]
            dec_phrase = dec_phrase + matrix[second_y][first_x]
        if print_steps:
            print(f'{origin_phrase[i]}{origin_phrase[i + 1]}-->{dec_phrase[i]}{dec_phrase[i + 
1]}')

    print(f'Дешифрованная итоговая фраза: {dec_phrase}')
    return dec_phrase
```

tools.py

```
import re


def find_position(square, letter) -> (int, int):
    for y in range(0, 6):
        for x in range(0, 6):
            if square[y][x] == letter:
                return y, x


def build_table(keyword, print_table=False) -> [[], [], [], [], [], []]:
    keyword = ''.join(re.findall('[А-ЯЁ ,.]', keyword))
    print(f'Итоговое ключевое слово: "{keyword}"')

    encryption_square = [[], [], [], [], [], []]

    keyword = ''.join(sorted(set(keyword), key=keyword.index))

    total = keyword + "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ .,"

    encryption_word = ''.join(sorted(set(total), key=total.index))
    if print_table:
        print("\nМатрица 6x6:")
    for i in range(0, 6):
        for j in range(0, 6):
            encryption_square[i].append(encryption_word[j + i * 6])
        if print_table:
            print(encryption_square[i])

    return encryption_square
```

main.py

```
from encode import *
from decode import *

while 1:
    print("Введите режим шифрования: 1 -- файл, 2 -- консоль.")
    type = input()
    phrase = ""
    if type == '1':
        with open('input.txt', 'r') as f:
            phrase = f.read().upper()
    elif type == '2':
        print("Фраза:")
        phrase = input().upper()
    else:
        print("Неверный ввод. Отмена операции.")
        continue

    print("Введите ключевое слово (допуст. символы: [А-Я ,.]):")
    keyword = input().upper()

    print("Введите 1 -- для кодирования, 2 -- для декодирования:")
    mode = input()
    if mode == '1':
        output = encode(keyword, phrase, print_steps=True, print_table=True)
    elif mode == '2':
        output = decode(keyword, phrase, print_steps=True, print_table=True)
    else:
        print("Неверный ввод. Отмена операции.")
        continue
    if type == '1':
        with open('output.txt', 'w') as f:
            f.write(output)
```

Примеры работы:

Работа с текстом в файле:

```
Введите режим шифрования: 1 -- файл, 2 -- консоль.
1
Введите ключевое слово (допуст. символы: [А-Я ,.]):
занятие
Введите 1 -- для кодирования, 2 -- для декодирования:
1
Итоговая фраза: "Я УЧУСЬ В УНИВЕРСИТЕТЕ ИТМО."
Итоговое ключевое слово: "ЗАНЯТИЕ"
```

```
Матрица 6x6:
['З', 'А', 'Н', 'Я', 'Т', 'И']
['Е', 'Б', 'В', 'Г', 'Д', 'Ё']
['Ж', 'Й', 'К', 'Л', 'М', 'О']
['П', 'Р', 'С', 'У', 'Ф', 'Х']
['Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы']
['Ь', 'Э', 'Ю', ' ', '.', ',']
```

```
Зашифрованная итоговая фраза: ГЯРЩФУЭ.ГЮСЯНЁБПХНЗДЗД,ЯДФМ,
```

Input.txt

```
Я учусь в университете ИТМО.
```

Output.txt

```
ГЯРЩФУЭ.ГЮСЯНЁБПХНЗДЗД,ЯДФМ,
```

Работа из консоли:

```
Введите режим шифрования: 1 -- файл, 2 -- консоль.
2
Фраза:
кошка
Введите ключевое слово (допуст. символы: [А-Я ,.]):
собака
Введите 1 -- для кодирования, 2 -- для декодирования:
1
Итоговая фраза: "КОШКА "
Итоговое ключевое слово: "СОБАКА"

Матрица 6x6:
['С', 'О', 'Б', 'А', 'К', 'В']
['Г', 'Д', 'Е', 'Ё', 'Ж', 'З']
['И', 'Й', 'Л', 'М', 'Н', 'П']
['Р', 'Т', 'У', 'Ф', 'Х', 'Ц']
['Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'Ь']
['Э', 'Ю', 'Я', ' ', '.', ',']
КО-->ВБ
ШК-->ЫО
А -->ЁА
Зашифрованная итоговая фраза: ВБЫОЁА
```

Вывод

В результате выполнения лабораторной работы я:

- Узнал о методе шифрования как «биграммный» (шифр Плейфера)
- Разработал программную реализацию алгоритма шифрования на языке Python