Лабораторная работа 2

Попов Дмитрий Павлович, НФИмд-01-23

Содержание

List of Figures

List of Tables

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: Математические основы защиты информации и информационной безопасности

Преподователь: Кулябов Дмитрий Сергеевич

Студент: Попов Дмитрий Павлович

Группа: НФИмд-01-23

МОСКВА

2023 г.

1 Цель работы

Целью данной работы является приобретение практических навыков в шифрах перестановки.[1]

2 Выполнение лабораторной работы

Требуется реализовать:

1. Маршрутное шифрование.
2. Шифрование с помощью решеток.
3. Таблица Виженера

2.1 Маршрутное шифрование

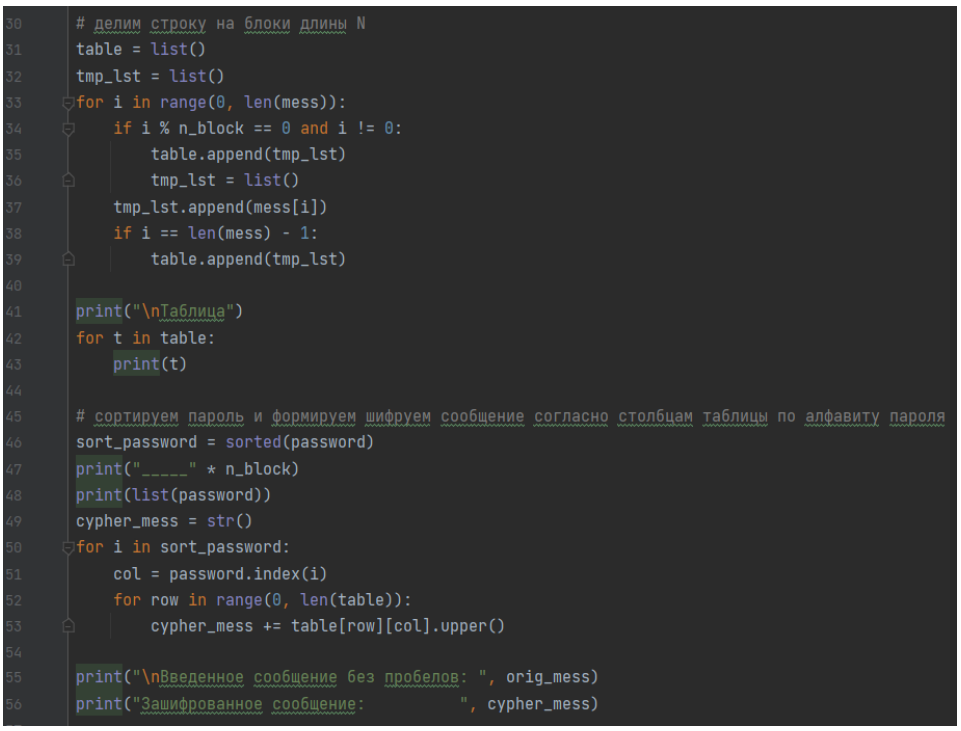
Текст разбивается на равные блоки N длиной M. Если в конце не хватает букв, то они добавляются в конец. Блоки записываются построчно в таблицу. Затем буквы выписываются по столбцам, которые упорядываются согласно паролю: внизу таблицы приписывается слово из n неповторяющихся букв и столбы нумеруются по алфавитному порядку букв пароля

Чтобы реализовать программу был написал код на Python(fig. 2.1)(fig. 2.2):

lab_2_marshrut.py

```python
print("-------------------------")
print(" Маршрутное шифрование")
print("-------------------------")

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"
alphabet = alphabet.split()
mess = str(input("Введите предложение, которое нужно зашифровать: ")).lower()
# mess = 'нельзя недооценивать противника'
mess = mess.replace(' ', '')
mess = mess.replace('\t', '')
orig_mess = mess
mess = list(mess)
# n_block = int(input("Введите длину блока N: "))
# n_block = 4
password = str(input("Введите пароль: ")).lower()
# password = 'пароль'
password = password.replace(' ', '')
password = password.replace('\t', '')

n_block = len(password)

print("\nВведенное сообщение без пробелов: ", orig_mess)
print("Длина блока N:                    ", n_block)
print("Введенный пароль без пробелов:    ", password)

# дополняем строку буквами "а" до кратного числу N
while len(mess) % n_block != 0:
    mess.append('а')
```

Figure 2.1: route1



```
# делим строку на блоки длины N
table = list()
tmp_lst = list()
for i in range(0, len(mess)):
    if i % n_block == 0 and i != 0:
        table.append(tmp_lst)
        tmp_lst = list()
    tmp_lst.append(mess[i])
    if i == len(mess) - 1:
        table.append(tmp_lst)

print("\nТаблица")
for t in table:
    print(t)

# сортируем пароль и формируем шифруем сообщение согласно столбцам таблицы по алфавиту пароля
sort_password = sorted(password)
print("_____" * n_block)
print(list(password))
cypher_mess = str()
for i in sort_password:
    col = password.index(i)
    for row in range(0, len(table)):
        cypher_mess += table[row][col].upper()

print("\nВведенное сообщение без пробелов: ", orig_mess)
print("Зашифрованное сообщение:          ", cypher_mess)
```

Figure 2.2: route2

Затем я запустил программу, ввел пароль и исходное сообщение. Получил таблицу шифрования, далее получил зашифрованное сообщение. Вывод работы программы (fig. 2.3)

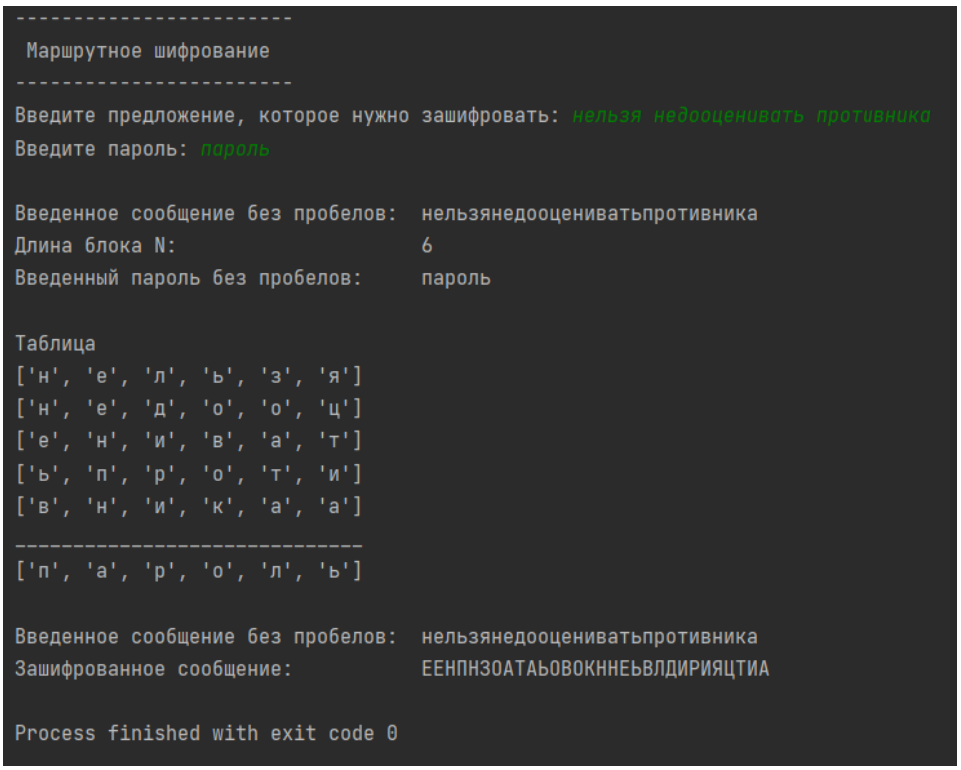


Figure 2.3: route_out

2.2 Шифрование с помощью решеток

Строится квадрат из k чисел. Затем к нему добавляются еще 3 квадрата, которые поворачиваются на 90 градусов и получается большой квадрат 2k размерностью. Дальше из большого квадрата вырезаются клетки и прорези записываются буквы. Когда заполнятся все прорези решето поворачивается на 90 градусов. И так продолжается пока не заполнится вся таблица. И буквы выписываются по алфивитному порядку пароля.

Чтобы реализовать программу был написал код на Python(fig. 2.4)(fig. 2.5)(fig. 2.6):

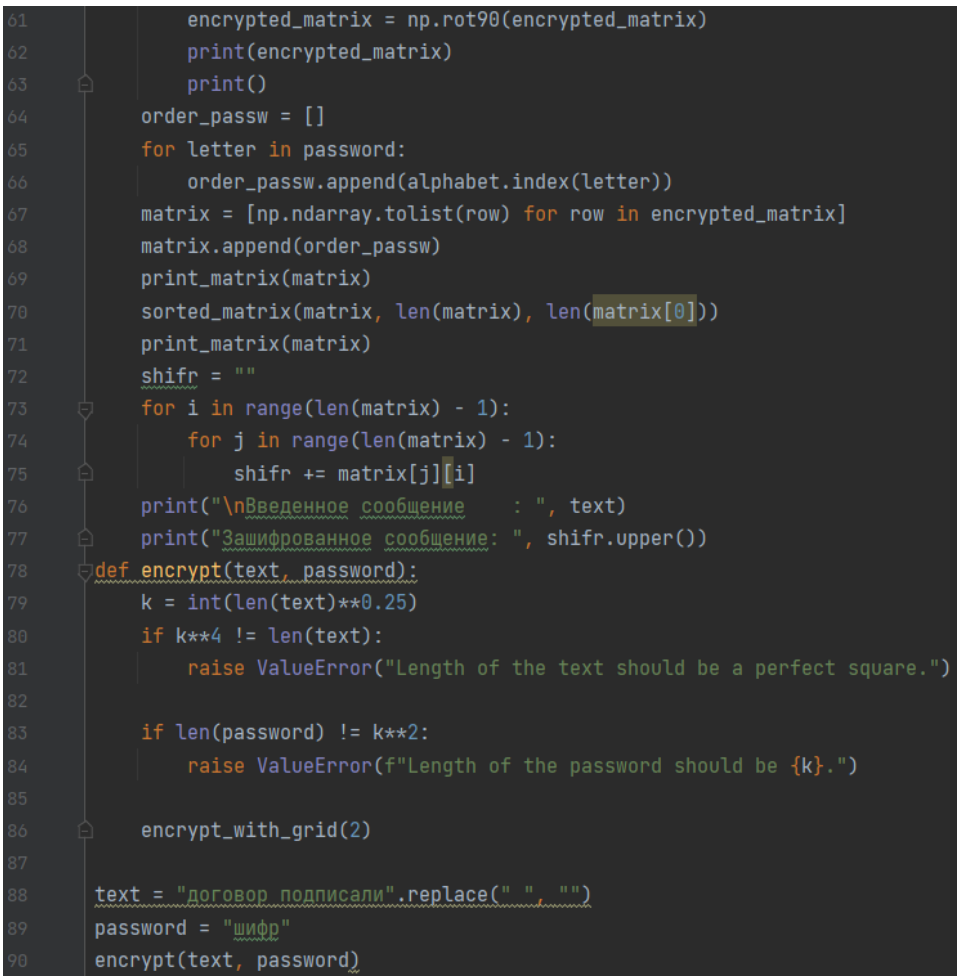
lab_2_reshetka.py

```python
print("-------------------------------------")
print(" Шифрование с использованием решеток")
print("-------------------------------------")
import numpy as np
from random import randint
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"
alphabet = alphabet.split()
def print_matrix(matrix):
    for row in matrix:
        print(row)
    print()
def sorted_matrix(a, N, M):
    k = M-1
    while k > 0:
        ind = 0
        for j in range(k+1):
            if a[N-1][j] > a[N-1][ind]:
                ind = j
        for i in range(N):
            m = a[i][ind]
            a[i][ind] = a[i][k]
            a[i][k] = m
        k -= 1
    return a
def encrypt_with_grid(k):
    encrypted = np.zeros((2*k, 2*k))
    for _ in range(4):
        for i in range(k):
            for j in range(k):
                encrypted[i][j] = i * k + j + 1
```

Figure 2.4: grid1

```
        encrypted = np.rot90(encrypted)
        print(encrypted)
        print()
    uniq = [num for num in range(1, k**2+1)]
    indexes = []
    while len(uniq) > 0:
        for i, lst in enumerate(encrypted):
            if len(uniq) == 0:
                break
            index = np.where(lst == uniq[0])
            index = index[0]
            if len(index) == 1:
                indexes.append((i, index[0]))
                uniq.pop(0)
            elif len(index) > 1:
                prob = randint(0, 1)
                if prob == 1:
                    indexes.append((i, index[-1]))
                else:
                    indexes.append((i, index[0]))
                uniq.pop(0)
    print("Индексы: ", indexes)
    print()
    encrypted_matrix = np.chararray((k * 2, k * 2), unicode=True)
    ind_text = 0
    for _ in range(k*2):
        for ind in indexes:
            encrypted_matrix[ind[0]][ind[1]] = text[ind_text]
            ind_text += 1
```

Figure 2.5: grid2



```
            encrypted_matrix = np.rot90(encrypted_matrix)
            print(encrypted_matrix)
            print()
        order_passw = []
        for letter in password:
            order_passw.append(alphabet.index(letter))
        matrix = [np.ndarray.tolist(row) for row in encrypted_matrix]
        matrix.append(order_passw)
        print_matrix(matrix)
        sorted_matrix(matrix, len(matrix), len(matrix[0]))
        print_matrix(matrix)
        shifr = ""
        for i in range(len(matrix) - 1):
            for j in range(len(matrix) - 1):
                shifr += matrix[j][i]
        print("\nВведенное сообщение    : ", text)
        print("Зашифрованное сообщение: ", shifr.upper())
def encrypt(text, password):
        k = int(len(text)**0.25)
        if k**4 != len(text):
            raise ValueError("Length of the text should be a perfect square.")

        if len(password) != k**2:
            raise ValueError(f"Length of the password should be {k}.")

        encrypt_with_grid(2)

text = "договор подписали".replace(" ", "")
password = "шифр"
encrypt(text, password)
```

Figure 2.6: grid3

Затем я запустил программу. Получил матрицы шифрования, далее получил зашифрованное сообщение. Вывод работы программы (fig. 2.7)(fig. 2.8)(fig. 2.9)

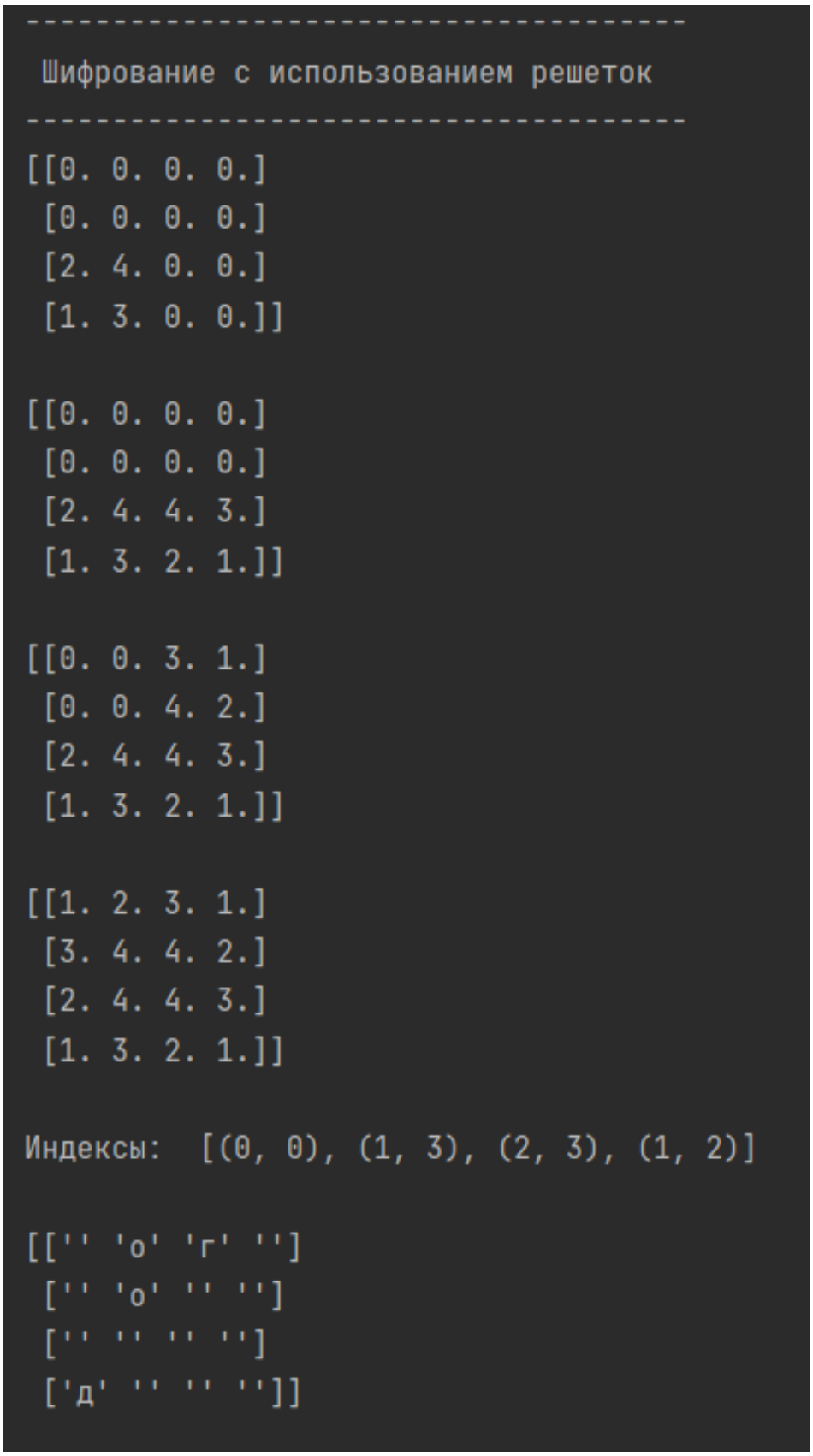


Figure 2.7: grid_out1

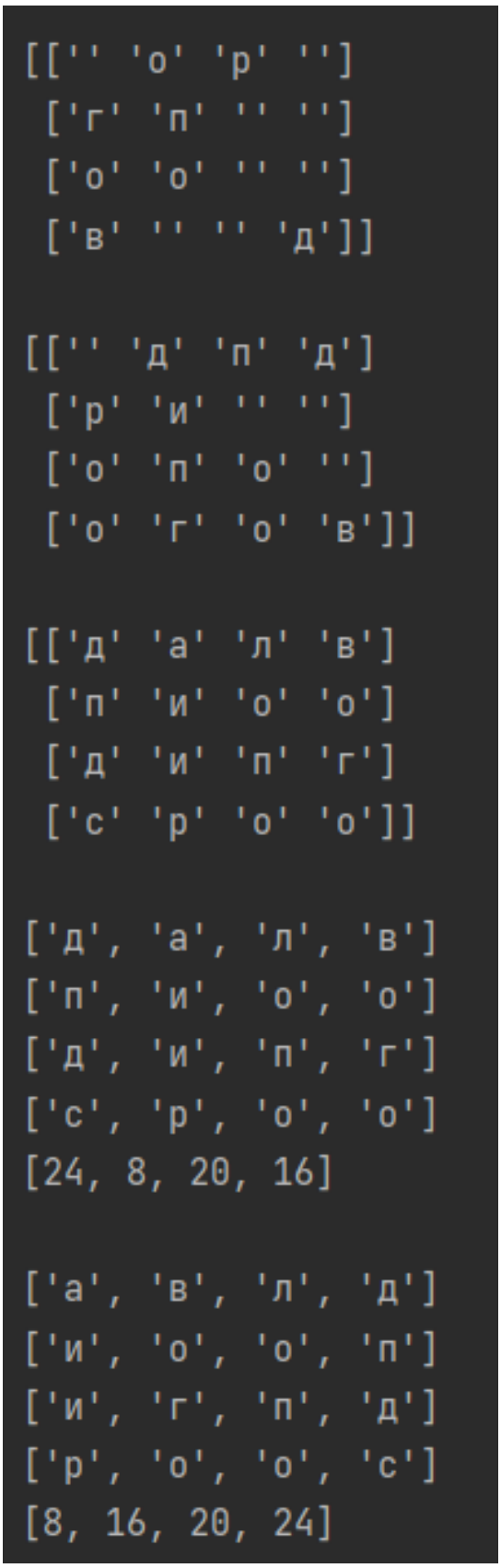


Figure 2.8: grid_out2

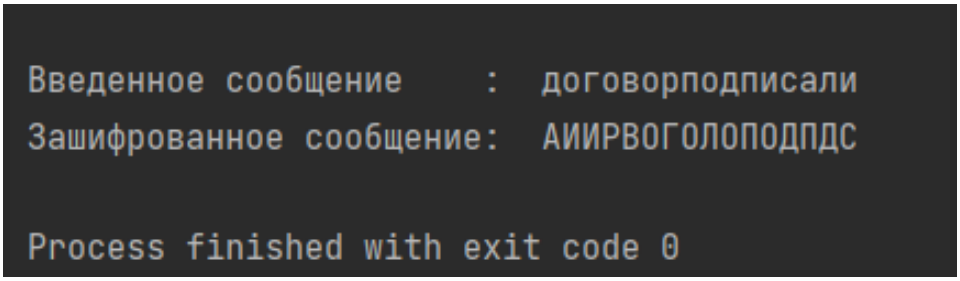


Figure 2.9: grid_out3

2.3 Таблица Виженера

В таблице записаны буквы русского алфавита. При переходе от одной строке к другой происходит циклический сдвиг на одну позицию. Пароль записывается с повторениями над буквами сообщения. В горизонтальном алфавите ищем букву нашего текста, а в вертикальном букву пароля и на их пересечении будет нужная нам буква.

Чтобы реализовать программу был написал код на Python(fig. 2.10)(fig. 2.11):

lab_2_vizhenera.py

```python
print("--------------------")
print(" Таблица Виженера")
print("--------------------")

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"
alphabet = alphabet.split()
mess = str(input("Введите предложение, которое нужно зашифровать: ")).lower()
# mess = 'криптография серьезная наука'
mess = mess.replace(' ', '')
mess = mess.replace('\t', '')
orig_mess = mess
password = str(input("Введите пароль: ")).lower()
# password = 'математика'
password = password.replace(' ', '')
password = password.replace('\t', '')

print("Введенное сообщение без пробелов: ", orig_mess)
print("Введенный пароль без пробелов:    ", password)

# записываем пароль такой же длины, как и предложение
mess = list(mess)
passw = list()
for i in range(0, len(mess)):
    index = i % len(password)
    passw.append(password[index])

print("\nПароль над предложением:")
print(passw)
print(mess)
```

Figure 2.10: viginere1

```
# создаем таблицу для шифрования
table = list()
table.append(alphabet)
for i in range(1, len(alphabet)):
    tmp_alph = alphabet[i:] + alphabet[:i]
    table.append(tmp_alph)

print("\nТаблица шифрования")
for t in table:
    print(t)

# шифруем сообщение по таблице
cypher_mess = str()
for i in range(0, len(passw)):
    row = alphabet.index(passw[i])
    col = alphabet.index(mess[i])
    cypher_mess += table[row][col].upper()

print("\nВведенное сообщение без пробелов: ", orig_mess)
print("Зашифрованное сообщение:          ", cypher_mess)
```

Figure 2.11: viginere2

Затем я запустил программу, ввел пароль и исходное сообщение. Получил пароль над предложением, таблицу шифрования (алфавит), далее получил зашифрованное сообщение. Вывод работы программы (fig. 2.12)(fig. 2.13)(fig. 2.14)

```
------------------
 Таблица Виженера
------------------
Введите предложение, которое нужно зашифровать: криптография серьезная наука
Введите пароль: математика
Введенное сообщение без пробелов:  криптографиясерьезнаянаука
Введенный пароль без пробелов:     математика

Пароль над предложением:
['м', 'а', 'т', 'е', 'м', 'а', 'т', 'и', 'к', 'а', 'м', 'а', 'т', 'е', 'м', 'а', 'т', 'и', 'к', 'а', 'м', 'а', 'т', 'е', 'м', 'а']
['к', 'р', 'и', 'п', 'т', 'о', 'г', 'р', 'а', 'ф', 'и', 'я', 'с', 'е', 'р', 'ь', 'е', 'з', 'н', 'а', 'я', 'н', 'а', 'у', 'к', 'а']
```

Figure 2.12: viginere_out1

Figure 2.13: viginere_out2

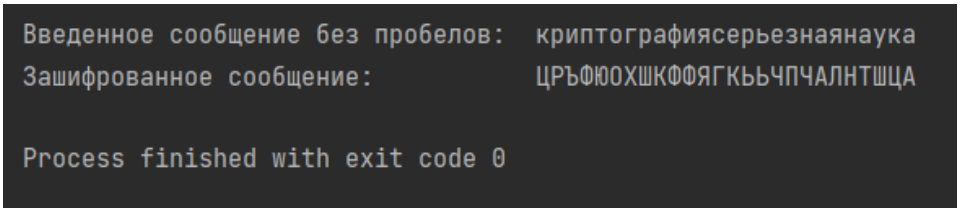


Figure 2.14: viginere_out3

3 Выводы

В результате выполнения работы я освоил на практике применение шифров перестановки.

4 Список литературы

1. Методические материалы курса