

# **Отчёт по лабораторной работе №2**

**Шифры перестановки**

Алексей Бондарь

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Теоретические сведения</b>	<b>5</b>
2.1	Шифр маршрутной перестановки . . . . .	5
2.2	Шифр Кардано . . . . .	5
2.3	Шифр Виженера . . . . .	6
<b>3</b>	<b>Выполнение работы</b>	<b>7</b>
3.1	Реализация шифра маршрутной перестановки на языке Python .	7
3.2	Реализация шифра решеткой на языке Python . . . . .	8
3.3	Реализация шифра Виженера на языке Python . . . . .	12
3.4	Контрольный пример . . . . .	16
<b>4</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

# List of Figures

3.1	Работа алгоритма маршрутной перестановки . . . . .	16
3.2	Работа алгоритма решетки . . . . .	16
3.3	Работа алгоритма Виженера . . . . .	17

# 1 Цель работы

Изучение алгоритмов маршрутной перестановки, решеток и Виженера

## 2 Теоретические сведения

### 2.1 Шифр маршрутной перестановки

Широкое распространение получили шифры перестановки, использующие некоторую геометрическую фигуру. Преобразования из этого шифра состоят в том, что в фигуру исходный текст вписывается по ходу одного “маршрута”, а затем по ходу другого выписывается с нее. Такой шифр называют маршрутной перестановкой. Например, можно вписывать исходное сообщение в прямоугольную таблицу, выбрав такой маршрут: по горизонтали, начиная с левого верхнего угла поочередно слева направо и справа налево. Выписывать же сообщение будем по другому маршруту: по вертикали, начиная с верхнего правого угла и двигаясь поочередно сверху вниз и снизу вверх.

### 2.2 Шифр Кардано

Решётка Кардано — инструмент кодирования и декодирования, представляющий собой специальную прямоугольную (в частном случае — квадратную) таблицу-карточку, четверть ячеек которой вырезана.

Таблица накладывается на носитель, и в вырезанные ячейки вписываются буквы, составляющие сообщение. После переворачивания таблицы вдоль вертикальной оси, процесс вписывания букв повторяется. Затем то же самое происходит после переворачивания вдоль горизонтальной и снова вдоль вертикальной осей.

В частном случае квадратной таблицы, для получения новых позиций для

вписывания букв, можно поворачивать квадрат на четверть оборота.

Чтобы прочитать закодированное сообщение, необходимо наложить решётку Кардано нужное число раз на закодированный текст и прочитать буквы, расположенные в вырезанных ячейках.

Такой способ шифрования сообщения был предложен математиком Джероламо Кардано в 1550 году, за что и получил своё название.

## 2.3 Шифр Виженера

Шифр Виженера (фр. Chiffre de Vigenère) — метод полиалфавитного шифрования буквенного текста с использованием ключевого слова.

Этот метод является простой формой многоалфавитной замены. Шифр Виженера изобретался многократно. Впервые этот метод описал Джован Баттиста Беллазо (итал. Giovan Battista Bellaso) в книге *La cifra del. Sig. Giovan Battista Bellaso* в 1553 году, однако в XIX веке получил имя Блеза Виженера, французского дипломата. Метод прост для понимания и реализации, он является недоступным для простых методов криптоанализа.

В шифре Цезаря каждая буква алфавита сдвигается на несколько строк; например в шифре Цезаря при сдвиге +3, А стало бы D, В стало бы Е и так далее. Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифровывания может использоваться таблица алфавитов, называемая *tabula recta* или квадрат (таблица) Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 26 различных шифров Цезаря. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова.

## 3 Выполнение работы

### 3.1 Реализация шифра маршрутной перестановки на языке Python

```
def marhsrutshifr():
# задание 1. Маршрутное шифрование.
    text = input("Input anything").replace(' ', '')
# вводим текст и очищаем от пробелов
    n = int(input("Введите число n"))
# на сколько блоков надо разбить текст
    m = int(input("Введите число m"))
# на сколько блоков надо разбить текст
    parol = input("Введите слово-пароль")
# слово-пароль
    # мы не предусмотрели проверку на соответствие
    lists = [['a' for i in range(0, n)] for j in range(m)]
#создаем матрицу nxm
    it = 0 #итератор
    for i in range(m):
        for j in range(n):
            if it < len(text):
                lists[i][j] = text[it]
            #заполняем текстом
```

```

        it += 1

lis = list()
for i in range(n):
    lis.append(parol[i])
# добавляем пароль в список
lists.append(lis)
# а список в матрицу
pprint(lists)
result = ""
# сюда будем записывать результат
spisok = sorted(lists[len(lists) - 1])
# сортируем по буквам пароль
for i in spisok:
# и согласно сортировке выписывает столбцами результат
    print(i, " = ", lists[len(lists)-1].index(i))
    for j in range(len(lists)):
        if j==len(lists)-1:
            continue
        result += lists[j][lists[len(lists)-1].index(i)]
print(result)

```

## 3.2 Реализация шифра решеткой на языке Python

# функция для поворота матрицы. нужен для 2рого задания

```
def rot90(matrix):
```

```
    return[list(reversed(col)) for col in zip(*matrix)]
```

# функция удаления чисел из матрицы нужен для 2 рого задания

```
def udalenie(largelist, inn, k):
```



```

for i in range(k * 2):
    for j in range(k * 2):
        if largelist[i][j] == inn:
            largelist[i][j] = " "
    return

def cardangrille(): # второе задания
    k = int(input("Введите число k")) # вводим наше число k
    s=1
    lists = [[i for i in range(k)] for i in range(k)]
    # строим матрицу этого размера
    for i in range(k):
        for j in range(k):
            lists[i][j] = s # заполняем матрицу числами
            s += 1
    print(lists)
    lists1 = rot90(lists)
# заранее делаем повороты и сохраняем эти матрицы, чтоб потом приклеить друг к друг
    lists2 = rot90(lists1)
    lists3 = rot90(lists2)
    largelist = [[1 for i in range(2*k)] for i in range(2*k)]
    # создаем большую матрицу, сюда будем клеить
    for i in range(k):
        # тут уже каждый цикл клеит все: 1 верхний кубик, 2 правый верхний кубик и т.д.
        for j in range(k):
            largelist[i][j] = lists[i][j]
    i1 = 0
    j1 = 0

```

```

for i in range(0, k):
    for j in range(k, k*2):
        largelist[i][j] = lists1[i1][j1]
        j1 += 1
    j1 = 0
    i1 += 1
i1 = 0
j1 = 0
for i in range(k, k*2):
    for j in range(k, k * 2):
        largelist[i][j] = lists2[i1][j1]
        j1 += 1
    j1 = 0
    i1 += 1
i1 = 0
j1 = 0
for i in range(k, k * 2):
    for j in range(0, k):
        largelist[i][j] = lists3[i1][j1]
        j1 += 1
    j1 = 0
    i1 += 1
pprint(largelist)
text = "договорподписали"
# текст который мы шифруем. его можно менять и даже нужно
largelist_a = [[" " for i in range(2*k)] for i in range(2*k)]
# тут вторая матрица из букв. Сюда будем вписывать буквы
s = 0
li = [i for i in range(1,k**2+1)]

```

```

# список из чисел, которые надо удалить
for inn in li:
    udalenie(largelist, inn, k)
# удаляем по очереди. Да, согласен алгоритм удаления такое себе))
ind = 0
# а тут уже "выписываем" буквы. Если текста еще есть то делаем повороты и тд
for i in range(k * 2):
    for j in range(k * 2):
        if largelist[i][j] == largelist_a[i][j] and len(text) > 0:
            largelist_a[i][j] = text[0]
            text = text[1:]
largelist = rot90(largelist)
for i in range(k * 2):
    for j in range(k * 2):
        if largelist[i][j] == largelist_a[i][j] and len(text) > 0:
            largelist_a[i][j] = text[0]
            text = text[1:]
if len(text) > 0:
    largelist = rot90(largelist)
    for i in range(k * 2):
        for j in range(k * 2):
            if largelist[i][j] == largelist_a[i][j] and len(text) > 0:
                largelist_a[i][j] = text[0]
                text = text[1:]
if len(text) > 0:
    largelist = rot90(largelist)
    for i in range(k * 2):
        for j in range(k * 2):
            if largelist[i][j] == largelist_a[i][j] and len(text) > 0:

```

```

        largelist_a[i][j] = text[0]
        text = text[1:]
    pprint(largelist_a)
    stri = input("Введите пароль")
    # тут дописываем пароль или удаляем чтоб длина слова была норм и потом прибав
    if len(stri) > k*2:
        stri = stri[:k*2]
    elif len(stri) < k*2:
        while len(stri) != k*2:
            stri += "z"
    largelist_a.append(list(stri))
    pprint(largelist_a)
    result = ""
    #фактически эта часть кода, такая же как в первом задании.
    spisok = sorted(largelist_a[len(largelist_a) - 1])
    for i in spisok:
        print(i, " = ", largelist_a[len(largelist_a) - 1].index(i))
        for j in range(len(largelist_a)):
            if j==len(largelist_a)-1:
                continue
            result += largelist_a[j][largelist_a[len(largelist_a) - 1].index(i)]
    print(result.replace(" ", ""))

```

### 3.3 Реализация шифра Виженера на языке Python

```

# Вижинер
# https://habr.com/ru/post/140820/

def form_dict():

```

```

d = {}
iter = 0
for i in range(0,127):
    d[iter] = chr(i)
    iter = iter +1
return d

def encode_val(word):
    list_code = []
    lent = len(word)
    d = form_dict()

    for w in range(lent):
        for value in d:
            if word[w] == d[value]:
                list_code.append(value)
    return list_code

def comparator(value, key):
    len_key = len(key)
    dic = {}
    iter = 0
    full = 0

    for i in value:
        dic[full] = [i,key[iter]]
        full = full + 1
        iter = iter +1
        if (iter >= len_key):

```

```

        iter = 0
    return dic

def full_encode(value, key):
    dic = comparator(value, key)
    print('Compare full encode', dic)
    lis = []
    d = form_dict()

    for v in dic:
        go = (dic[v][0]+dic[v][1]) % len(d)
        lis.append(go)
    return lis

def decode_val(list_in):
    list_code = []
    lent = len(list_in)
    d = form_dict()

    for i in range(lent):
        for value in d:
            if list_in[i] == value:
                list_code.append(d[value])
    return list_code

def full_decode(value, key):
    dic = comparator(value, key)

```

```

print('Deshifre=', dic)
d = form_dict()
lis =[]

for v in dic:
    go = (dic[v][0]-dic[v][1]+len(d)) % len(d)
    lis.append(go)
return lis

def vijer():
    word = "Hello world"
    key = "key"
    sys.stdout.write(word)
    sys.stdout.write(key)
    key_encoded = encode_val(key)
    value_encoded = encode_val(word)
    sys.stdout.write(str(key_encoded))
    sys.stdout.write(str(value_encoded))
    shifre = full_encode(value_encoded, key_encoded)
    print('Шифр=', ''.join(decode_val(shifre)))

    decoded = full_decode(shifre, key_encoded)
    print('Decode list=', decoded)
    decode_word_list = decode_val(decoded)
    print('Word=', ''.join(decode_word_list))

```

### 3.4 Контрольный пример

```
20 print(result)

In [3]: 1 marhsrutshifr()

Введите текст тест
Введите число n 3
Введите число m4
Введите слово-пароль код
т е с
т а а
а а а
а а а
к о д
д = 2
к = 0
о = 1
саааттааеааа
```

Figure 3.1: Работа алгоритма маршрутной перестановки

```
In [9]: 1 cardangrille()

Введите число k3
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
1 2 3 7 4 1
4 5 6 8 5 2
7 8 9 9 6 3
3 6 9 9 8 7
2 5 8 6 5 4
1 4 7 3 2 1
т е с т т е
  е с т с т
    т

Введите паролькод
т е с т т е
  е с т с т
    т

к о д з з з
з = 3
з = 3
з = 3
д = 2
к = 0
о = 1
тттттсстте
```

Figure 3.2: Работа алгоритма решетки



```

In [11]: 1  vijer()

Hello worldkey[107, 101, 121][72, 101, 108, 108, 111, 32, 119, 111, 114, 108, 100]Compare full encode {0: [72, 107], 1: [101, 1
01], 2: [108, 121], 3: [108, 107], 4: [111, 101], 5: [32, 121], 6: [119, 107], 7: [111, 101], 8: [114, 121], 9: [108, 107], 10:
[100, 101]}
Шифр= 4KFXU8CU1XJ
Deshifre= {0: [52, 107], 1: [75, 101], 2: [102, 121], 3: [88, 107], 4: [85, 101], 5: [26, 121], 6: [99, 107], 7: [85, 101], 8:
[108, 121], 9: [88, 107], 10: [74, 101]}
Decode list= [72, 101, 108, 108, 111, 32, 119, 111, 114, 108, 100]
Word= Hello world

```

Figure 3.3: Работа алгоритма Виженера

## **4 Выводы**

Изучили алгоритмы шифрования с помощью перестановок

# Список литературы

1. Шифр маршрутной перестановки
2. Шифр Кардано
3. Шифр Виженера