Отчёт по лабораторной работе №2

Шифры перестановки

**Семмар вассим**

Содержание

[1 Цель работы 1](#_Toc178279729)

[2 Теоретические сведения 1](#_Toc178279730)

[2.1 Шифр маршрутной перестановки 1](#_Toc178279731)

[2.2 Шифр Кардано 2](#_Toc178279732)

[2.3 Шифр Виженера 2](#_Toc178279733)

[3 Выполнение работы 3](#_Toc178279734)

[3.1 Реализация шифра маршрутной перестановки на языке Python 3](#_Toc178279735)

[3.2 Реализация шифра решеткой на языке Python 3](#_Toc178279736)

[3.3 Реализация шифра Виженера на языке Python 6](#_Toc178279737)

[3.4 Контрольный пример 8](#_Toc178279738)

[4 Выводы 9](#_Toc178279739)

[Список литературы 10](#_Toc178279740)

# 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 Bellasо 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, A стало бы D, B стало бы E 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 соотвествие  
 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  
 prrint(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 другу  
 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 клеет все: 1 верхний кубик, 2 правый верхний кубик и тд  
 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  
 prrint(largelist)  
 text = "examplemessage"  
 # 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:]  
 prrint(largelist\_a)  
 stri = input("Введите пароль")  
 # тут дописываем пароль или удаляем чтоб длина слова была норм и потом прибавляем к матрице  
 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))  
 prrint(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 = "data Science"  
 key = "cipher"  
 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

A screenshot of a computer

Description automatically generated

Figure 1: Работа алгоритма маршрутной перестановки

A screenshot of a computer code

Description automatically generated

Figure 2: Работа алгоритма решетки

A screenshot of a computer code

Description automatically generated

Figure 3: Работа алгоритма Виженера

# 4 Выводы

Изучили алгоритмы шифрования с помощью перестановок

# Список литературы

1. [Шифр маршрутной перестановки](https://life-prog.ru/2_89965_marshrutnie-perestanovki.html)
2. [Шифр Кардано](https://kabinfo.ucoz.ru/index/shifr_reshetka_kardano/0-374)
3. [Шифр Виженера](https://habr.com/ru/post/103055/)