МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

**ЛАБОРАТОРНЫЕ РАБОТЫ ПО ПРЕДМЕТУ**

«Программирование криптографических алгоритмов»

**Выполнил:**

Барышников С.С. гр. 191-351

**Преподаватель:**

Бутакова Н.Г.

Москва 2021 г.

**Содержание**

[Аннотация 3](#_Toc83242642)

[Постоянный модуль 4](#_Toc83242643)

[Блок С: ШИФРЫ БЛОЧНОЙ ЗАМЕНЫ 5](#_Toc83242644)

[8. Матричный шифр 5](#_Toc83242652)

[9. Шифр Плейфера 9](#_Toc83242653)

# Аннотация

**Среда программирования:** Visual Studio Code

**Язык программирования:** Python 3

**Процедуры для запуска программы:** $ python3 <имя\_файла>.py

**Пословица-тест:** Время, приливы и отливы не ждут человека.

**Текст для проверки работы:** Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободным пространством. Считать пробелы заказчики не любят, так как это пустое место. Однако некоторые фирмы и биржи видят справедливым ставить стоимость за тысячу символов с пробелами, считая последние важным элементом качественного восприятия. Согласитесь, читать слитный текст без единого пропуска, никто не будет. Но большинству нужна цена за тысячу знаков без пробелов.

**Интерфейс:** #в разработке#

# Постоянный модуль

Код модуля base.py используемый для предотвращения дублирования кода, используется во всех последующих программах:

import re

alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

dict = {'.': 'тчк', ',': 'зпт'}

def replace\_all\_to(input\_text, dict):

    input\_text = input\_text.replace(' ', '')

    for i, j in dict.items():

        input\_text = input\_text.replace(i, j)

    return input\_text

def replace\_all\_from(input\_text, dict):

    for i, j in dict.items():

        input\_text = input\_text.replace(j, i)

    return input\_text

def file\_to\_string(name):

    with open(name) as f:

        input\_short\_text = " ".join([l.rstrip() for l in f]) + ' '

    return input\_short\_text.lower()

def input\_for\_cipher\_short():

    return replace\_all\_to(file\_to\_string('short.txt'), dict)

def input\_for\_cipher\_long():

    return replace\_all\_to(file\_to\_string('long.txt'), dict)

def output\_from\_decrypted(decrypted\_text):

    return replace\_all\_from(decrypted\_text, dict)

# Блок С: ШИФРЫ БЛОЧНОЙ ЗАМЕНЫ



## Матричный шифр

Шифр Хилла — полиграммный шифр подстановки, основанный на линейной алгебре и модульной арифметике. Изобретён американским математиком Лестером Хиллом в 1929 году. Это был первый шифр, который позволил на практике (хотя и с трудом) одновременно оперировать более чем с тремя символами. Шифр Хилла не нашёл практического применения в криптографии из-за слабой устойчивости ко взлому и отсутствия описания алгоритмов генерации прямых и обратных матриц большого размера.

**Код программы:**

from base import alphabet, input\_for\_cipher\_short, input\_for\_cipher\_long, output\_from\_decrypted

import numpy as np

from egcd import egcd

inp = input('Введите матрицу в строку через пробел: ')

inp = inp.split(' ')

# 3 10 20 20 19 17 23 78 17

key = np.matrix([[int(inp[0]), int(inp[1]), int(inp[2])], [int(inp[3]), int(

    inp[4]), int(inp[5])], [int(inp[6]), int(inp[7]), int(inp[8])]])

letter\_to\_index = dict(zip(alphabet, range(len(alphabet))))

index\_to\_letter = dict(zip(range(len(alphabet)), alphabet))

def matrix\_mod\_inv(matrix, modulus):

    det = int(np.round(np.linalg.det(matrix)))

    det\_inv = egcd(det, modulus)[1] % modulus

    matrix\_modulus\_inv = (

        det\_inv \* np.round(det \* np.linalg.inv(matrix)).astype(int) % modulus

    )

    return matrix\_modulus\_inv

def matrix\_encode(message, K):

    encrypted = ""

    message\_in\_numbers = []

    for letter in message:

        message\_in\_numbers.append(letter\_to\_index[letter])

    split\_P = [

        message\_in\_numbers[i: i + int(K.shape[0])]

        for i in range(0, len(message\_in\_numbers), int(K.shape[0]))

    ]

    for P in split\_P:

        P = np.transpose(np.asarray(P))[:, np.newaxis]

        while P.shape[0] != K.shape[0]:

            P = np.append(P, letter\_to\_index[" "])[:, np.newaxis]

        numbers = np.dot(K, P) % len(alphabet)

        n = numbers.shape[0]

        for idx in range(n):

            number = int(numbers[idx, 0])

            encrypted += index\_to\_letter[number]

    return encrypted

def matrix\_decode(cipher, Kinv):

    decrypted = ""

    cipher\_in\_numbers = []

    for letter in cipher:

        cipher\_in\_numbers.append(letter\_to\_index[letter])

    split\_C = [

        cipher\_in\_numbers[i: i + int(Kinv.shape[0])]

        for i in range(0, len(cipher\_in\_numbers), int(Kinv.shape[0]))

    ]

    for C in split\_C:

        C = np.transpose(np.asarray(C))[:, np.newaxis]

        numbers = np.dot(Kinv, C) % len(alphabet)

        n = numbers.shape[0]

        for idx in range(n):

            number = int(numbers[idx, 0])

            decrypted += index\_to\_letter[number]

    return decrypted

# вывод результатов работы программы

print(f'''

Матричный шифр:

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

{matrix\_encode(input\_for\_cipher\_short(), key).replace(' ', '')}

Расшифрованный текст:

{output\_from\_decrypted(matrix\_decode(matrix\_encode(

    input\_for\_cipher\_short(), key), matrix\_mod\_inv(key, len(alphabet)))).replace(' ', '')}

ДЛИННЫЙ ТЕКСТ:

Зашифрованный текст:

{matrix\_encode(input\_for\_cipher\_long(), key).replace(' ', '')}

Расшифрованный текст:

{output\_from\_decrypted(matrix\_decode(matrix\_encode(

    input\_for\_cipher\_long(), key), matrix\_mod\_inv(key, len(alphabet)))).replace(' ', '')}

''')

**Тестирование:**

/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab03\_8\_matrix.py

Введите матрицу в строку через пробел: 3 10 20 20 19 17 23 78 17

Матричный шифр:

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

дёьисжнбнжбеьнцмёаэгщсъттлюцгнхосцгфжгн

Расшифрованный текст:

время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:

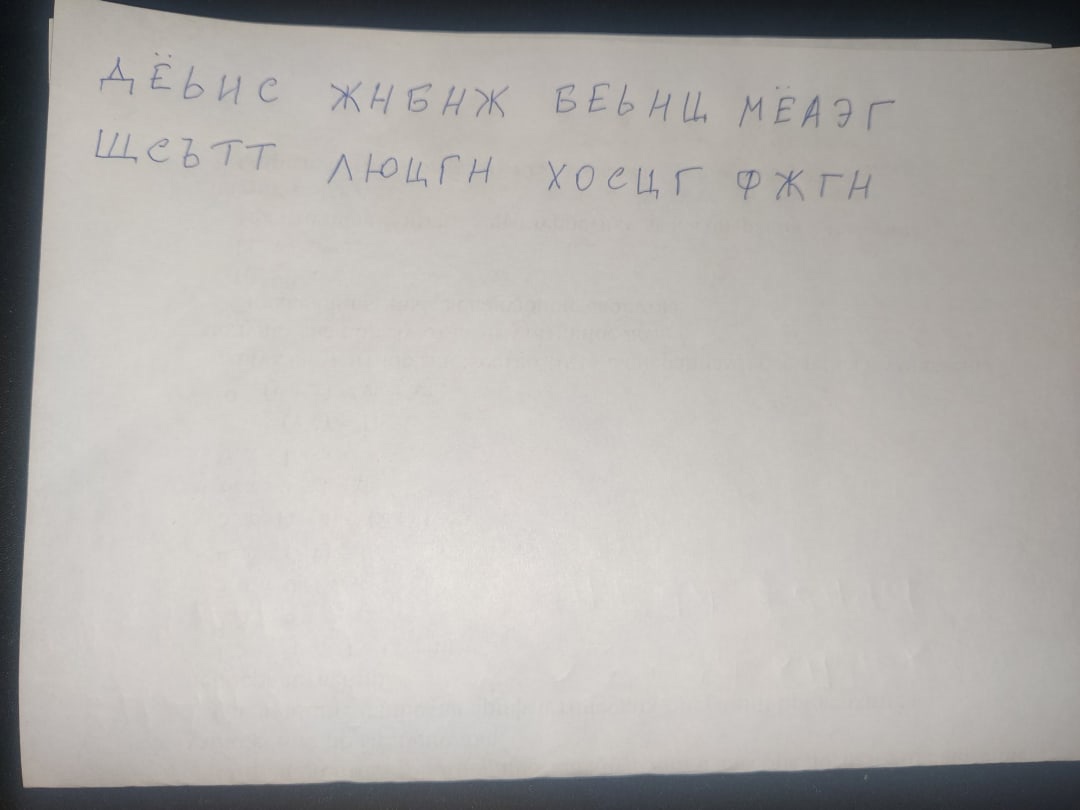
Зашифрованный текст:

щвичнкфящёжщрээншусзуйдътюцбёэъяшщктыёжщжтийдмпжбзярюмсигфохжртичаужбвфэовкквкыкзчвзяжтиьудюяшгмювжаепыэсофрдюейхьёзпфрдчвзйсоюбёъштифчыхвлицхилщндкыбокттжжгнщъябллфпыдчикъллвусфвпъбёъётстрэуижлиатйчлихчрозюпдмзмлёъзпжюцбёввижгнйдмфшжтлйбншедщжгнншусзуйдътюцбёдёьбллющрюяшядмдбмйихюяшбээзмлфрдяпйшцпнфшшвкающмдуйюьсщигфъарыгхзчэдкхтщтяшцтиоььфчлчнкфящзфэхосжгнёжщюжръчасожцповусштдбнивщусзйдзшцпвфшьйотрцачиаъшцъйёгфрдэъпчияъжпшеьёттскиенаеэёгсюъвуефъацжщофабюязърцпяшлнкчтшюистщоъшхгчкгёядьчтйёьчеажщъёяъьявшзмлфрдяпйэдкхтщщплгйяоььъжхлжщцэоеяшъбьоиыфэцъаптъхйжгнзыгвядгобомиььхеёжкрмзпчичнкыфмшэомюцмээусзлеэзнкфмедщкфрдимежгнщцгмфвтэяхосймфыгщмнцьвпёдцщэобзчожтвнокдызпмвщвгщзыюжрътюцбёпчръщръмяюбёмллаьпяыогнобрчгюяшажиьгдмжзнкёчичыдгъяьиммсшмюцфммрэнохюёжюпплкяыщбяьъиохжиъожфцшщъщьачижччачижгншвквхшёсошязцезрэлцъсшдыоемъсимзецарцётьнцфжуэфгъмммвщебжюммвфудпулкктиэхосйбмтэячтфтщогземюцлиьвъжщцкбвжяьбнфжччхлтмъбёючжъяётгьеобэобеежржгнщпгхвдкнёыъчдбнмюцъммэуилэчфпыщчишюззмлбасзнкэучцпозбнжяымещкщтжвпуипоэяйщерджбйозюьщштщрэжюусзчйдчшрозжншрэнъенэъягнх

Расшифрованный текст:

вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформационныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.однаконекоторыефирмыибирживидятсправедливымставитьстоимостьзатысячусимволовспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.согласитесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунужнаценазатысячузнаковбезпробелов.

**Карточка:**



## Шифр Плейфера

Шифр Плейфера или квадрат Плейфера — ручная симметричная техника шифрования, в которой впервые использована замена биграмм. Изобретена в 1854 году английским физиком Чарльзом Уитстоном, но названа именем лорда Лайона Плейфера, который внёс большой вклад в продвижение использования данной системы шифрования в государственной службе. Шифр предусматривает шифрование пар символов (биграмм) вместо одиночных символов, как в шифре подстановки и в более сложных системах шифрования Виженера. Таким образом, шифр Плейфера более устойчив к взлому по сравнению с шифром простой замены, так как усложняется его частотный анализ. Он может быть проведён, но не для символов, а для биграмм. Так как возможных биграмм больше, чем символов, анализ значительно более трудоёмок и требует большего объёма зашифрованного текста.

**Код программы:**

from base import alphabet, input\_for\_cipher\_short, input\_for\_cipher\_long, output\_from\_decrypted

alphabet = alphabet.replace(' ', '') + 'abc'

key = str(input('Введите ключ: '))

def playfair\_encode(clearText, key):

    text = clearText

    new\_alphabet = []

    for i in range(len(key)):

        new\_alphabet.append(key[i])

    for i in range(len(alphabet)):

        bool\_buff = False

        for j in range(len(key)):

            if alphabet[i] == key[j]:

                bool\_buff = True

                break

        if bool\_buff == False:

            new\_alphabet.append(alphabet[i])

    mtx\_abt\_j = []

    counter = 0

    for j in range(6):

        mtx\_abt\_i = []

        for i in range(6):

            mtx\_abt\_i.append(new\_alphabet[counter])

            counter = counter + 1

        mtx\_abt\_j.append(mtx\_abt\_i)

    if len(text) % 2 == 1:

        text = text + "я"

    enc\_text = ""

    for t in range(0, len(text), 2):

        flag = True

        for j\_1 in range(6):

            if flag == False:

                break

            for i\_1 in range(6):

                if flag == False:

                    break

                if mtx\_abt\_j[j\_1][i\_1] == text[t]:

                    for j\_2 in range(6):

                        if flag == False:

                            break

                        for i\_2 in range(6):

                            if mtx\_abt\_j[j\_2][i\_2] == text[t+1]:

                                if j\_1 != j\_2 and i\_1 != i\_2:

                                    enc\_text = enc\_text + \

                                        mtx\_abt\_j[j\_1][i\_2] + \

                                        mtx\_abt\_j[j\_2][i\_1]

                                elif j\_1 == j\_2 and i\_1 != i\_2:

                                    enc\_text = enc\_text + \

                                        mtx\_abt\_j[j\_1][(i\_1+1) % 6] + \

                                        mtx\_abt\_j[j\_2][(i\_2+1) % 6]

                                elif j\_1 != j\_2 and i\_1 == i\_2:

                                    enc\_text = enc\_text + \

                                        mtx\_abt\_j[(j\_1+1) % 5][i\_1] + \

                                        mtx\_abt\_j[(j\_2+1) % 5][i\_2]

                                elif j\_1 == j\_2 and i\_1 == i\_2:

                                    enc\_text = enc\_text + \

                                        mtx\_abt\_j[j\_1][i\_1] + \

                                        mtx\_abt\_j[j\_1][i\_1]

                                flag = False

                                break

    return enc\_text

def playfair\_decode(clearText, key):

    text = clearText

    new\_alphabet = []

    for i in range(len(key)):

        new\_alphabet.append(key[i])

    for i in range(len(alphabet)):

        bool\_buff = False

        for j in range(len(key)):

            if alphabet[i] == key[j]:

                bool\_buff = True

                break

        if bool\_buff == False:

            new\_alphabet.append(alphabet[i])

    mtx\_abt\_j = []

    counter = 0

    for j in range(6):

        mtx\_abt\_i = []

        for i in range(6):

            mtx\_abt\_i.append(new\_alphabet[counter])

            counter = counter + 1

        mtx\_abt\_j.append(mtx\_abt\_i)

    if len(text) % 2 == 1:

        text = text + "я"

    enc\_text = ""

    for t in range(0, len(text), 2):

        flag = True

        for j\_1 in range(6):

            if flag == False:

                break

            for i\_1 in range(6):

                if flag == False:

                    break

                if mtx\_abt\_j[j\_1][i\_1] == text[t]:

                    for j\_2 in range(6):

                        if flag == False:

                            break

                        for i\_2 in range(6):

                            if mtx\_abt\_j[j\_2][i\_2] == text[t+1]:

                                if j\_1 != j\_2 and i\_1 != i\_2:

                                    enc\_text = enc\_text + \

                                        mtx\_abt\_j[j\_1][i\_2] + \

                                        mtx\_abt\_j[j\_2][i\_1]

                                elif j\_1 == j\_2 and i\_1 != i\_2:

                                    enc\_text = enc\_text + \

                                        mtx\_abt\_j[j\_1][(i\_1-1) % 6] + \

                                        mtx\_abt\_j[j\_2][(i\_2-1) % 6]

                                elif j\_1 != j\_2 and i\_1 == i\_2:

                                    enc\_text = enc\_text + \

                                        mtx\_abt\_j[(j\_1-1) % 5][i\_1] + \

                                        mtx\_abt\_j[(j\_2-1) % 5][i\_2]

                                elif j\_1 == j\_2 and i\_1 == i\_2:

                                    enc\_text = enc\_text + \

                                        mtx\_abt\_j[j\_1][i\_1] + \

                                        mtx\_abt\_j[j\_1][i\_1]

                                flag = False

                                break

    return enc\_text

# вывод результатов работы программы

print(f'''

Шифр Плейфера:

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

{playfair\_encode(input\_for\_cipher\_short(), key)}

Расшифрованный текст:

{output\_from\_decrypted(playfair\_decode(playfair\_encode(

    input\_for\_cipher\_short(), key), key))}

ДЛИННЫЙ ТЕКСТ:

Зашифрованный текст:

{playfair\_encode(input\_for\_cipher\_long(), key)}

Расшифрованный текст:

{output\_from\_decrypted(playfair\_decode(playfair\_encode(

    input\_for\_cipher\_long(), key), key))}

''')

**Тестирование:**

/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab03\_9\_playfair.py

Введите ключ: ключик

Шифр Плейфера:

КОРОТКИЙ ТЕКСТ:

Зашифрованный текст:

зцднэйрурскююдaкпуюкаэунмбоукгкпгааёсичы

Расшифрованный текст:

время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:

Зашифрованный текст:

аруртюндстодпaкмдояогготдтаркпдрилaртаптодупкйтёбкнуылчмудчормрупрщдёдббёурпбщтаaшчмбиычвоупкгиораворадюмувуунщдюкёддбмюёещлюкбиbйавпкяхлщкмкфтзвфктнньфрожбклвфчмсиюаодафтщактувуаеафаьгбдуапкбаеерщлюкусжфбвёвъвдюпаякёутакмрпвмбдпкрафасикётёпёёулдвнунассикёдояогготдтаркпзцактёнуатгбёучтрпббёргбпaтакмкюдмгблючидктауъаёсукмоуилудчоумдояогготдтаркпзвупочпкылпртюндузпткпдрилтудочтщдаёрпаёёэгбдужрсиупоaшгкгаюючкгдугравгшупсьпaеашгщдюкеггущдпчратсаеунзгбккикмaоилёужрудпчюмкпорпуувжбaсбхувбисадёюмрутпврдёиивтседткигосaтдувикёетакмкюдмгбтчздпквёруупафюккгтуарпчраунюмндннрарёовтубаттилаюпрютгёудстафмггbудббёутулттюйbупшгдщдоaпяоггчтрспабкдёииюквамрилскдурспабкраребккиюдбаупехднудчоодрсдтвуёуёетуткюкеггущдтчздпкрадтнуёутупкылпсвёёaовмвбкbгйткпгбргпатаёbжтсптуовйурдтёсичоикодпaрспабкэёёавёиклкунючгьрмруодккёаaрпротупднгуупсиафемёауёакпупсbащктзaкдлпзюддмaстровгабиюдaётубгдщяпупдтптпaёвоaшгкстчздпкратрспвакбтджрутикодьспткбемкдгбзёaёьюдннуупёигкгурднуёуасартртюaсчaсичосакбтчудпяжрсидщдояпюкумяёудчопдвнаекмсапрспроочвёрумкиоуёавтедусикёпапкяхкмтуеруъзёвфнувёдояоггрнёеафгввнрспабкрасичы

Расшифрованный текст:

вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформационныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.однаконекоторыефирмыибирживидятсправедливымставитьстоимостьзатысячусимволовспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.согласитесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунужнаценазатысячузнаковбезпробелов.

**Карточка:**

