Создаем файл caesar.py и реализовываем функции шифрования и расшифровывания числа и последовательности чисел:

```python
def encrypt(m, key):
    return (m + key) % 256

def decrypt(m, key):
    return (m - key) % 256

def encrypt_data(data, key):
    cypher_data = []
    for m in data:
        cypher_data.append(encrypt(m, key))
    return cypher_data

def decrypt_data(data_c, key):
    data = []
    for c in data_c:
        data.append(decrypt(c, key))
    return data
```

Рис. 1. Файл caesar.py

Пример работы шифра Цезаря:

```python
m = 24      # сообщение
key = 37    # ключ
c = caesar.encrypt(m, key)
print('c =', c)
m1 = caesar.decrypt(c, key)
print('m1 =', m1)
data = [34, 67, 123, 79, 201]
encrypt_data = caesar.encrypt_data(data, key)
print('encrypt_data =', encrypt_data)
decrypt_data = caesar.decrypt_data(encrypt_data, key)
print('decrypt_data =', decrypt_data)
```

```
c = 61
m1 = 24
encrypt_data = [71, 104, 160, 116, 238]
decrypt_data = [34, 67, 123, 79, 201]
```

Рис. 2. Пример работы шифра Цезаря

Формируем файл ‘f1.txt’:

He was a burly man of an exceedingly dark complexion, with an exceedingly large head and a corresponding large hand. He took my chin in his large hand and turned up my face to have a look at me by the light of the candle. He was prematurely bald on the top of his head, and had bushy black eyebrows that wouldn't lie down but stood up bristling. His eyes were set very deep in his head, and were disagreeably sharp and suspicious. He had a large watchchain, and strong black dots where his beard and whiskers would have been if he had let them. He was nothing to me, and I could have had no foresight then, that he ever would be anything to me, but it happened that I had this opportunity of observing him well.

Теперь зашифруем весь файл ‘f1.txt’. Результат шифрования запишем в файл ‘f1_encrypt.txt’.

```
data = read_write_file.read_data_1byte('data/f1.txt')
encrypt_data = caesar.encrypt_data(data, key=67)
txt = ''.join([chr(s) for s in encrypt_data])
read_write_file.write_data_1byte('encrypt_data/f1_encrypt.txt', encrypt_data)
```

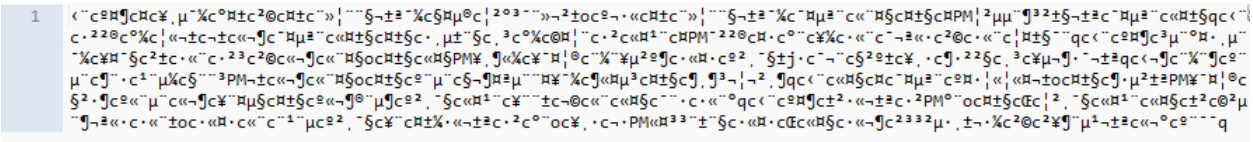


Рис. 3. Результат шифрования шифром Цезаря с ключом 67

Расшифруем обратно файл ‘f1_encrypt.txt’, используя известный ключ.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/f1_encrypt.txt')
decrypt_data = caesar.decrypt_data(encrypt_data, key=67)
txt = ''.join([chr(s) for s in decrypt_data])
read_write_file.write_data_1byte('decrypt_data/f1_decrypt_know_key.txt', decrypt_data)
```

```
He was a burly man of an exceedingly dark complexion, with an exceedingly large head and a
corresponding large hand. He took my chin in his large hand and turned up my face to have a
look at me by the light of the candle. He was prematurely bald on the top of his head, and had
bushy black eyebrows that wouldn't lie down but stood up bristling. His eyes were set very deep
in his head, and were disagreeably sharp and suspicious. He had a large watchchain, and strong
black dots where his beard and whiskers would have been if he had let them. He was nothing to
me, and I could have had no foresight then, that he ever would be anything to me, but it
happened that I had this opportunity of observing him well.
```

Рис. 4. Результат расшифровки

Расшифруем файл ‘f1_encrypt’, будем считать, что ключ нам неизвестен.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/f1_encrypt.txt')

for k in range(256):
    decrypt_data = caesar.decrypt_data(encrypt_data, key=k)
    txt = ''.join([chr(s) for s in decrypt_data])
    is_english = detectEnglish.isEnglish(txt)
    if is_english:
        print('key =', k)
        print('decrypt_data =', txt)
        break
read_write_file.write_data_1byte('decrypt_data/f1_decrypt_unknown_key.txt', decrypt_data)
```

```
key = 67
decrypt_data = He was a burly man of an exceedingly dark complexion, with an exceed
corresponding large hand. He took my chin in his large hand and turned up my face t
look at me by the light of the candle. He was prematurely bald on the top of his he
bushy black eyebrows that wouldn't lie down but stood up bristling. His eyes were s
in his head, and were disagreeably sharp and suspicious. He had a large watchchain,
black dots where his beard and whiskers would have been if he had let them. He was
me, and I could have had no foresight then, that he ever would be anything to me, b
happened that I had this opportunity of observing him well.
```

Рис. 5. Результат расшифровки, используя перебор ключа

Возьмем файл ‘f2.png’ (Рис. 6).

Рис. 6.

Зашифруем изображение.

```
data = read_write_file.read_data_1byte('data/f2.png')
encrypt_data = caesar.encrypt_data(data, key=143)
read_write_file.write_data_1byte('encrypt_data/f2_encrypt.png', encrypt_data)
```

Теперь считаем, что ключ неизвестен, расшифруем ‘f2_encrypt.png’.

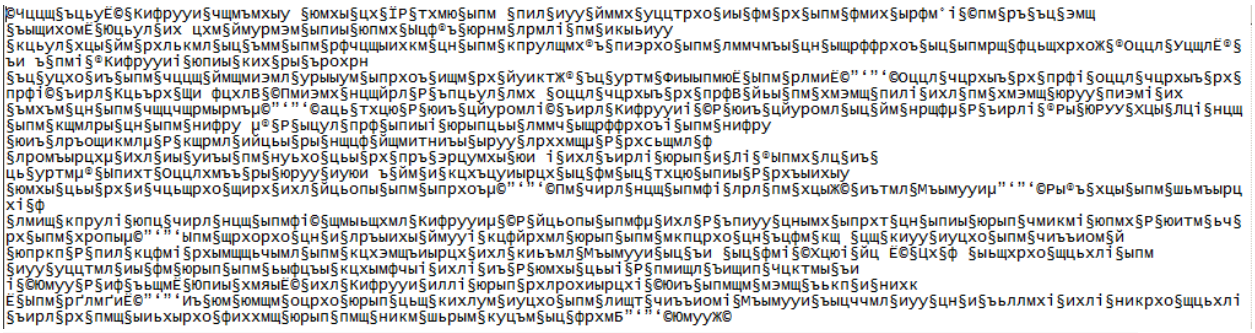
```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/f2_encrypt.png')
for k in range(256):
    decrypt_data = caesar.decrypt_data(encrypt_data, key=k)
    if decrypt_data[0] == 0x89 and decrypt_data[1] == 0x50:
        print('key =', k)
        break
read_write_file.write_data_1byte('decrypt_data/f2_decrypt.png', decrypt_data)
```

```
key = 143
```

Рис. 7. Результат расшифровки ‘f2_encrypt.png’

Как видно, получили тот же ключ.

Дешифруем файл ‘t3_caesar_c_all.txt’.




```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/t3_caesar_c_all.txt')
for k in range(256):
    decrypt_data = caesar.decrypt_data(encrypt_data, key=k)
    txt = ''.join([chr(s) for s in decrypt_data])
    is_english = detectEnglish.isEnglish(txt)
    if is_english:
        print('key =', k)
        print('decrypt_data =', txt)
        read_write_file.write_data_1byte('decrypt_data/t3_caesar_c_all_decrypt.txt', decrypt_data)
        break
```

Рис. 8.

Результат:

```
key = 135
```

```
"Poor soul!" Camilla presently went on (I knew they had all been looking at me in the mean time), "he is so very strange! Would anyone believe that when Tom's wife died, he actually could not be induced to see the importance of the children's having the deepest of trimmings to their mourning? 'Good Lord!' says he, 'Camilla, what can it signify so long as the poor bereaved little things are in black?' So like Matthew! The idea!"

"Good points in him, good points in him," said Cousin Raymond; "Heaven forbid I should deny good points in him; but he never had, and he never will have, any sense of the proprieties."

"You know I was obliged," said Camilla, "I was obliged to be firm. I said, 'It WILL NOT DO, for the credit of the family.' I told him that, without deep trimmings, the family was disgraced. I cried about it from breakfast till dinner. I injured my digestion. And at last he flung out in his violent way, and said, with a D, 'Then do as you like.' Thank Goodness it will always be a consolation to me to know that I instantly went out in a pouring rain and bought the things."

"He paid for them, did he not?" asked Estella.

"It's not the question, my dear child, who paid for them," returned Camilla. "I bought them. And I shall often think of that with peace, when I wake up in the night."

The ringing of a distant bell, combined with the echoing of some cry or call along the passage by which I had come, interrupted the conversation and caused Estella to say to me, "Now, boy!" On my turning round, they all looked at me with the utmost contempt, and, as I went out, I heard Sarah Pocket say, "Well I am sure! What next!" and Camilla add, with indignation, "Was there ever such a fancy! The i-de-a!"

As we were going with our candle along the dark passage, Estella stopped all of a sudden, and, facing round, said in her taunting manner with her face quite close to mine:

"Well?"
```

Рис. 9.

Дешифруем файл ‘c4_caesar_c_all.bmp’.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/c4_caesar_c_all.bmp')
for k in range(256):
    decrypt_data = caesar.decrypt_data(encrypt_data, key=k)
    if decrypt_data[0] == 0x42 and decrypt_data[1] == 0x4d:
        print('key =', k)
        print('decrypt_data =', decrypt_data[:10])
        read_write_file.write_data_1byte('decrypt_data/c4_caesar_c_all_decrypt.bmp', decrypt_data)
        break
```

Результат:

```
key = 231
```

Рис. 10.

Зашифруем рис.10, оставив первые 50 байт без изменения.

```
data = read_write_file.read_data_1byte('decrypt_data/c4_caesar_c_all_decrypt.bmp')
encrypt_data = data[:50] + caesar.encrypt_data(data[50:], key=231)
read_write_file.write_data_1byte('encrypt_data/c4_caesar_c_all_encrypted_50.bmp', encrypt_data)
```

Рис. 11. Шифрование с ключом 231

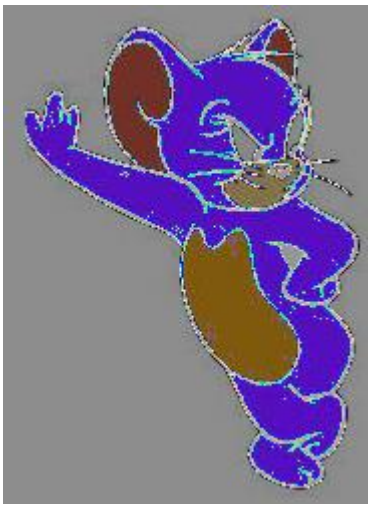


Рис. 12. Шифрование с ключом 142

Реализуем модуль substitution.py.

```
k=[179, 109, 157, 182, 126, 141, 251, 220, 169,
    237, 188, 131, 207, 22, 32, 242, 208, 68, 216,
    170, 249, 199, 44, 198, 206, 8, 148, 197, 136,
    195, 159, 98, 175, 53, 123, 212, 233, 150, 6,
    243, 38, 79, 156, 153, 2, 134, 47, 215, 102,
    15, 57, 110, 236, 24, 184, 72, 137, 113, 171,
    70, 161, 64, 252, 247, 49, 103, 105, 138, 119,
    213, 87, 130, 203, 90, 167, 238, 231, 116, 78,
    86, 173, 250, 200, 239, 178, 97, 114, 94, 166,
    142, 104, 31, 75, 89, 106, 56, 128, 69, 164,
    67, 26, 228, 61, 181, 125, 227, 54, 96, 168,
    107, 17, 14, 37, 190, 219, 211, 121, 112, 35,
    18, 143, 158, 193, 129, 71, 23, 101, 191, 41,
    241, 82, 201, 223, 120, 59, 177, 58, 63, 151,
    42, 36, 183, 226, 127, 172, 202, 84, 132, 3,
    45, 73, 30, 235, 50, 189, 4, 1, 43, 221, 205,
    83, 232, 46, 147, 93, 192, 124, 244, 12, 21,
    80, 55, 160, 145, 245, 209, 88, 204, 176, 13,
    253, 11, 99, 165, 140, 19, 224, 111, 27, 185,
    65, 62, 16, 163, 210, 115, 217, 34, 92, 187,
    152, 155, 108, 5, 122, 229, 174, 118, 162, 95,
    100, 7, 66, 29, 230, 144, 149, 52, 9, 91, 117,
    214, 76, 48, 33, 194, 254, 10, 234, 218, 40,
    133, 196, 139, 135, 240, 60, 25, 225, 85, 255,
    246, 51, 28, 146, 74, 222, 186, 39, 77, 0, 20,
    180, 154, 81, 248]

def encrypt_data(data):
    cypher_data = []
    for d in data:
        cypher_data.append(k[d])
    return cypher_data

def decrypt_data(data):
    decypher_data = []
    for d in data:
        decypher_data.append(k.index(d))
    return decypher_data
```

И с помощью таблицы замен k расшифруем файл ‘c3_subst_c_all.png’.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/c3_subst_c_all.png')
for k in range(256):
    decrypt_data = substitution.decrypt_data(encrypt_data)
    if decrypt_data[0] == 0x89 and decrypt_data[1] == 0x50:
        print('k =', k)
        break
read_write_file.write_data_1byte('decrypt_data/c3_subst_c_all_decrypt.png', decrypt_data)
```

Результат:

Рис.13.

Теперь зашифруем рис. 13 оставив первые 350 байт без изменения.

```
data = read_write_file.read_data_1byte('decrypt_data/c3_subst_c_all_decrypt.png')
encrypt_data = data[:350] + substitution.encrypt_data(data[350:])
read_write_file.write_data_1byte('encrypt_data/c3_subst_c_all_encrypted_350.png', encrypt_data)
```

Результат:

Рис. 14.

Для аффинных шифров реализуем модуль affine.py. с алгоритмом Евклида, расширенным алгоритмом Евклида и функциями шифрования и дешифрования.

```
def gcd(a, b):
    # Return the GCD of a and b using Euclid's Algorithm
    while a != 0:
        a, b = b % a, a
    return b

def findModInverse(a, m):
    # Returns the modular inverse of a % m, which is
    # the number x such that a*x % m = 1

    if gcd(a, m) != 1:
        return None

    # Calculate using the Extended Euclidean Algorithm:
    u1, u2, u3 = 1, 0, a
    v1, v2, v3 = 0, 1, m
    while v3 != 0:
        q = u3 // v3  # // is the integer division operator
        v1, v2, v3, u1, u2, u3 = (u1 - q * v1), (u2 - q * v2), (u3 - q * v3), v1, v2, v3
    return u1 % m

def encrypt(m, a, b):
    return (a * m + b) % 256

def decrypt(m, a, b):
    return ((m - b) * findModInverse(a, 256)) % 256

def encrypt_data(data, a, b):
    cypher_data = []
    for m in data:
        cypher_data.append(encrypt(m, a, b))
    return cypher_data

def decrypt_data(data, a, b):
    decypher_data = []
    for m in data:
        decypher_data.append(decrypt(m, a, b))
    return decypher_data
```

Зашифруем файл ‘f3.bmp’ аффинным шифром a = 167, b = 35.

Рис. 15. f3.bmp

```
data = read_write_file.read_data_1byte('data/f3.bmp')
a = 167
b = 35
encrypt_data = affine.encrypt_data(data, a, b)
read_write_file.write_data_1byte('encrypt_data/f3_affine_encrypt.bmp', encrypt_data)
```

Расшифруем ‘f3_affine_encrypt.bmp’.

```
a = 167
b = 35
encrypt_data = read_write_file.read_data_1byte('encrypt_data/f3_affine_encrypt.bmp')
decrypt_data = affine.decrypt_data(encrypt_data, a, b)
read_write_file.write_data_1byte('decrypt_data/f3_affine_decrypt.bmp', decrypt_data)
```

Теперь зашифруем оставив первые 50 байт без изменения.

```
encrypt_data = decrypt_data[:50] + affine.encrypt_data(decrypt_data[50:], a, b)
read_write_file.write_data_1byte('encrypt_data/f3_affine_c_encrypted_50.png', encrypt_data)
```

Результат:

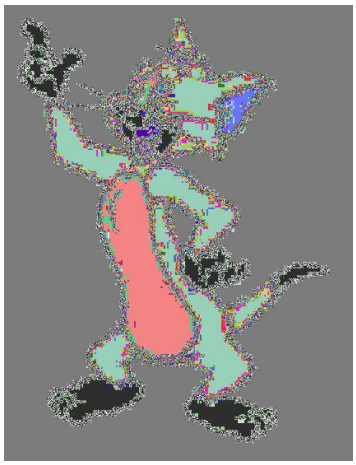


Рис. 16.

Дешифруем файл ‘text10_affine_c_all.txt’. Шифр аффинный, a и b неизвестны.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/text10_affine_c_all.txt')
count_keys = 0
is_english = False
for a in range(256):
    if affine.gcd(a, 256) == 1:
        for b in range(256):
            count_keys += 1
            decrypt_data = affine.decrypt_data(encrypt_data[:30], a, b)
            text = ''.join([chr(s) for s in decrypt_data[:30]])
            is_english = detectEnglish.isEnglish(text)
            if is_english:
                print('k =', count_keys)
                print('a =', a)
                print('b =', b)
                decrypt_data = affine.decrypt_data(encrypt_data, a, b)
                read_write_file.write_data_1byte('decrypt_data/text10_affine_c_all_decrypt.txt', decrypt_data)
                break
    if is_english:
        break
```

Результат:

3701 итерация, a = 29, b = 116.

```
k = 3701
a = 29
b = 116
```

```
For oft, when on my couch I lie
In vacant or in pensive mood,
They flash upon that inward eye
Which is the bliss of solitude;
And then my heart with pleasure fills,
And dances with the daffodils.
```

Дешифруем файл ‘b4_affine_c_all.png’. Шифр аффинный, a и b неизвестны.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/b4_affine_c_all.png')
count_keys = 0
is_PNG = False
for a in range(256):
    if affine.gcd(a,256) == 1:
        for b in range(256):
            count_keys += 1
            decrypt_data = affine.decrypt_data(encrypt_data[:2], a, b)
            is_PNG = decrypt_data[0] == 0x89 and decrypt_data[1] == 0x50
            if is_PNG:
                print('k =', count_keys)
                print('a =', a)
                print('b =', b)
                decrypt_data = affine.decrypt_data(encrypt_data, a, b)
                read_write_file.write_data_1byte('decrypt_data/b4_affine_c_all_decrypt.png', decrypt_data)
                break
    if is_PNG:
        break
```

Результат:

7955 итераций, a = 63, b = 18.

```
k = 7955
a = 63
b = 18
```

Рис. 17.

Расшифруем файл ‘im6_vigener_c_all.bmp’. Шифр Виженера, ключ – magistr.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/im6_vigener_c_all.bmp')
key = 'magistr'
decrypt_data = []
for i in range(len(encrypt_data)):
    m = (encrypt_data[i] - ord(key[i % len(key)])) % 256
    decrypt_data.append(m)
read_write_file.write_data_1byte('decrypt_data/im6_vigener_decrypt.bmp', decrypt_data)
```

Результат:

Рис. 18

Зашифруем рис. 18, оставив первые 50 байт без изменения.

```
data = read_write_file.read_data_1byte('decrypt_data/im6_vigener_decrypt.bmp')
key = 'magistr'
encrypt_data = []

for i, d in enumerate(data[50:]):
    encrypt_data.append((d + ord(key[i % len(key)])) % 256)

read_write_file.write_data_1byte('encrypt_data/im6_vigener_encrypt_50.bmp', data[:50] + encrypt_data)
```

Результат:

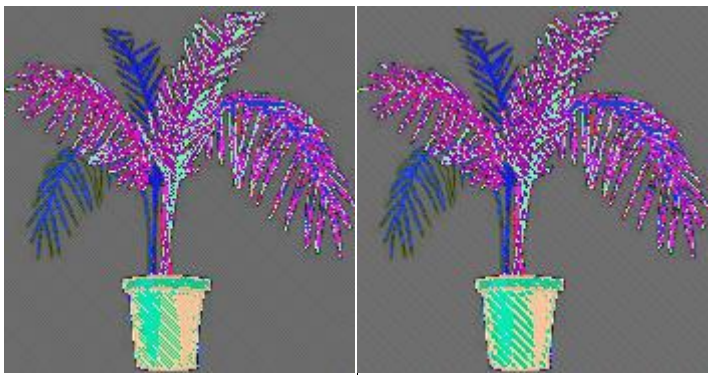


Рис. 19. Шифрование ключом magistr(слева), ключом crypto (справа)

Дешифруем файл ‘text4_vigener_c_all.txt’. Известно, что в файле есть слово housewives.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/text4_vigener_c_all.txt')
word = 'housewives'
for i in range(len(encrypt_data)):
    key = ''
    ww = encrypt_data[i:i+len(word)]
    for j, w in enumerate(ww):
        k = (w - ord(word[j % len(word)])) % 256
        key += chr(k)
    is_english = detectEnglish.isEnglish(key[:3])
    if is_english:
        print(key) # key = student

key = 'student'
decrypt_data = []
for i, e in enumerate(encrypt_data):
    decrypt_data.append((e - ord(key[i % len(key)])) % 256)
read_write_file.write_data_1byte('decrypt_data/text4_vigener_c_all_decrypt.txt', decrypt_data)
```

Результатом является ключ – student. И расшифрованный текст:

```
1
2   Long walks at night--
3   that's what good for the soul:
4   peeking into windows
5   watching tired housewives
6   trying to fight off
7   their beer-maddened husbands.
```

Расшифруем файл ‘text1_vigener_c.txt’. Известно, что в файле присутствуют слова it therefore.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/text1_vigener_c.txt')
word = 'it therefore'
for i in range(len(encrypt_data)):
    key = ''
    ww = encrypt_data[i:i+len(word)]
    for j, w in enumerate(ww):
        k = (w - ord(word[j % len(word)])) % 256
        key += chr(k)
    # print(key)
    is_english = detectEnglish.isEnglish(key[:3])
    if is_english:
        print(key)  # KEYBOARD

key = 'KEYBOARD'
decrypt_data = []
for i, e in enumerate(encrypt_data[48:]):
    decrypt_data.append((e - ord(key[i % len(key)])) % 256)
read_write_file.write_data_1byte('decrypt_data/text1_vigener_c_all_decrypt.txt', decrypt_data)
```

Результатом является слово – KEYBOARD. И расшифрованный текст:

```
Take this kiss upon the brow!
And, in parting from you now,
Thus much let me avow--
You are not wrong, who deem
That my days have been a dream;
Yet if hope has flown away
In a night, or in a day,
In a vision, or in none,
Is it therefore the less gone?
All that we see or seem
Is but a dream within a dream.

I stand amid the roar
Of a surf-tormented shore,
And I hold within my hand
Grains of the golden sand--
How few! yet how they creep
Through my fingers to the deep,
While I weep--while I weep!
O God! can I not grasp
Them with a tighter clasp?
O God! can I not save
One from the pitiless wave?
Is all that we see or seem
But a dream within a dream?
```

Расшифруем файл ‘text2_permutation_c.txt’. Текст зашифрован шифром перестановки длины 6.

```
import itertools

encrypt_data = read_write_file.read_data_1byte('encrypt_data/text2_permutation_c.txt')
template = '123456'
ew = detectEnglish.ENGLISH_WORDS
for p in itertools.permutations(template, len(template)):
    perm = ''.join(p)
    decrypt_data = ''
    for i in range(0, 24, len(perm)):
        ww = encrypt_data[i:i+len(perm)]
        for j in range(len(perm)):
            decrypt_data += chr(ww[int(perm[j])-1])

    dd_split = decrypt_data.upper().split(' ')
    for d in dd_split:
        if d == '' or d == 'I': continue
        if d not in ew:
            break
    else:
        print('text =', decrypt_data)
        print('permutation =', perm)
        break
```

Результат:

```
text = Whose woods these are I
permutation = 231465
```

Расшифруем теперь весь файл.

```
encrypt_data = read_write_file.read_data_1byte('encrypt_data/text2_permutation_c.txt')
key = '231465'
decrypt_data = []
for i in range(0, len(encrypt_data), len(key)):
    ww = encrypt_data[i:i+len(key)]
    for j in range(len(key)):
        decrypt_data.append(ww[int(key[j]) - 1])
read_write_file.write_data_1byte('decrypt_data/text2_permutation_c_decrypt.txt', decrypt_data)
```

Результат:

```
Whose woods these are I think I know.
His house is in the village, though;
He will not see me stopping here
To watch his woods fill up with snow.
My little horse must think it queer
To stop without a farmhouse near
Between the woods and frozen lake
The darkest evening of the year.

He gives his harness bells a shake
To ask if there is some mistake.
The only other sound's the sweep
Of easy wind and downy flake.
The woods are lovely, dark and deep,
But I have promises to keep,
And miles to go before I sleep,
And miles to go before I sleep
```

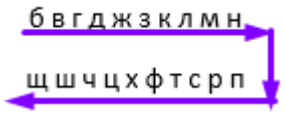
Простая литорея.

Литорея – тайнописание, употреблявшееся в древнерусской литературе. Литорея предполагает простую замену буквы на другую. Например, для русского языка берется алфавит, из него выбрасываются все гласные буквы, а согласные записываются в два ряда:

б в г д ж з к л м н

щ ш ч ц х ф т с р п

И как видно порядок следования букв идет следующим образом:



Замена производится по следующему правилу: верхние буквы заменяются на нижние, а нижние на верхние, букв, которых нет в таблице – не шифруются [1].

Например, для слова магистр шифр будет – рачилкм.

Напишем программу. Будем использовать вместо русского алфавита английский со следующей таблицей замен:

b c d f g h j k l m

z x v w t s r q p n

```
text = read_write_file.read_data_1byte('data/f5.txt')
encrypt = ''

cipher = ['bz', 'cx', 'dw', 'fv', 'gt', 'hs', 'jr', 'kq', 'lp', 'mn']

for t in text:
    flag = False
    for c in cipher:
        if chr(t).lower() in c:
            encrypt += c[1-c.index(chr(t).lower())]
            flag = True

    if not flag:
        encrypt += chr(t)

print(encrypt)
read_write_file.write_data_1byte_text('encrypt_data/f5_encrypt.txt', encrypt)
```

Исходные данные:

```
Hello darkness, my old friend
I have come to talk with you again
Because a vision softly creeping
Left its seeds while I was sleeping
And the vision that was planted in my brain
Still remains
Within the sound of silence
In restless dreams I walked alone
Narrow streets of cobblestone
Neath the halo of a street lamp
I turned my collar to the cold and damp
When my eyes were stabbed by the flash of a neon light
That split the night
And touched the sound of silence
```

Результат:

```
seppo wajqmehh, ny opw vjiemw
I safe xone go gapq digs you ataim
zexauhe a fihiom hovgpy xjeelimt
pevg igh heewh dsipe I dah hpeelimt
Amw gse fihiom gsag dah lpamgew im ny zjaim
hgipp jenaimh
digsim gse houmw ov hipemxe
Im jehgpehh wjeanh I dapqew apome
majjod hgjeegh ov xozzpehgome
meags gse sapo ov a hgjeeg panl
I gujmew ny xoppaj go gse xopw amw wanl
dsem ny eyeh deje hgazzew zy gse vpahs ov a meom pitsg
gsag hlpig gse mitsg
Amw gouxsew gse houmw ov hipemxe
```

Можно попробовать сделать и для картинки. Пусть есть алфавит из 256 символов от 0 до 255. Удалим из него все нечетные числа и запишем оставшиеся 128 значений в две строчки:

0 2 4 … 124 126

254 252 250 … 130 128

```python
table = list(range(0, 256, 2))
t1 = table[:len(table)//2]
t2 = table[len(table)//2:]
table = t1 + t2[::-1]

data = read_write_file.read_data_1byte('data/f4.bmp')

encrypt = []
for d in data:
    if d in table:
        idx = table.index(d)
        encrypt.append(table[(idx+len(table)//2) % len(table)])
    else:
        encrypt.append(d)

read_write_file.write_data_1byte('encrypt_data/f4_encrypt_1.bmp', encrypt)
```

Шифрующие таблицы Трисемуса.

Для получения такого шифра замены обычно использовались таблица для записи букв алфавита и ключевое слово (или фраза). В таблицу сначала вписывалось по строкам ключевое слово, причём повторяющиеся буквы исключались. Затем эта таблица дополнялась не вошедшими в неё буквами алфавита по порядку. Т.к. ключевое слово или фразу легко хранить в памяти, то такой подход упрощал процесс шифрования и дешифрования. Например, для русского алфавита можно взять таблицу размера 4x8. В качестве ключа возьмём слово БАНДЕРОЛЬ и получим шифрующую таблицу:

Б А Н Д Е Р О Л

Ь В Г Ж З И Й К

М П С Т У Ф Х Ц

Ч Ш Щ Ы Ъ Э Ю Я

При шифровании находят в этой таблице очередную букву открытого текста и записывают в шифр букву, расположенную ниже её в том же столбце. Буквы нижней строки таблицы шифруем буквами верхней строки из того же столбца. Например, при шифровании с помощью этой таблицы сообщения: ПРИЛЕТАЕМПЯТОГО получим шифр ШИФКЗЫВЗЧШЛЫЙСЙ [2].

Напишем программу. Будем использовать буквы английского алфавита и для простоты дополнительные символы, чтобы заполнить таблицу до конца.

Например, пусть ключевое слово будет magistr. Тогда шифрующая таблица примет вид:

m	a	g	i	s	t	r
b	c	d	e	f	h	j
k	l	n	o	p	q	u
v	w	x	y	z	<пробел>	.

Напишем программу.

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
extra_symbol = ' .,!?:;-@#$%^&*'
keyword = 'magistr'
m = list(keyword) + sorted(list(set(list(alphabet)) - set(list(keyword))))

table = []
for i, a in enumerate(m):
    if not i % len(keyword):
        table.append(''.join(m[i:i+len(keyword)]))
if len(table[-1]) != len(keyword):
    table[-1] = table[-1] + extra_symbol[:(len(keyword) - len(table[-1]))]

print(table)

text = 'fedotov alexandr alexandrovich'
encrypt = ''
for t in text:
    for idx, st in enumerate(table):
        if t in st:
            encrypt += table[(idx+1) % len(table)][st.index(t)]
print(encrypt)
```

Результат:

```
['magistr', 'bcdefhj', 'klnopqu', 'vwxyz .']
ponyhymtcwogcxnjtcwogcxnjymelq
fedotov alexandr alexandrovich
```

Для шифрования картинки сформируем таблицу замен из 256 значений:

```
from random import randint
from random import choice
alphabet = list(range(256))
keyword = []
len_keyword = choice([8, 16, 32, 64, 128])
k = 0
for _ in range(len_keyword + k):
    r = randint(0, 256)
    while r in keyword:
        r = randint(0, 256)
    keyword.append(r)

m = keyword + sorted(list( set(alphabet) - set(keyword) ))

table = []
for idx, t in enumerate(m):
    if not idx % (len_keyword):
        table.append(m[idx:idx+len_keyword])
print(table)
```

Получим следующую таблицу:

```
table = [[43, 119, 178, 55, 169, 255, 214, 200, 171, 192, 237, 227, 230, 245, 181, 151],
         [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15],
         [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31],
         [32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48],
         [49, 50, 51, 52, 53, 54, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65],
         [66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81],
         [82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97],
         [98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113],
         [114, 115, 116, 117, 118, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130],
         [131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146],
         [147, 148, 149, 150, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163],
         [164, 165, 166, 167, 168, 170, 172, 173, 174, 175, 176, 177, 179, 180, 182, 183],
         [184, 185, 186, 187, 188, 189, 190, 191, 193, 194, 195, 196, 197, 198, 199, 201],
         [202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 215, 216, 217, 218],
         [219, 220, 221, 222, 223, 224, 225, 226, 228, 229, 231, 232, 233, 234, 235, 236],
         [238, 239, 240, 241, 242, 243, 244, 246, 247, 248, 249, 250, 251, 252, 253, 254]]
```

Теперь можно шифровать картинку:

```
data = read_write_file.read_data_1byte('data/f4.bmp')
encrypt_data = []
for d in data:
    for idx, t in enumerate(table):
        if d in t:
            encrypt_data.append(table[(idx+1) % len(table)][t.index(d)])

read_write_file.write_data_1byte('encrypt_data/f4_encrypt.bmp', encrypt_data)
```

Шифр табличной маршрутной перестановки.

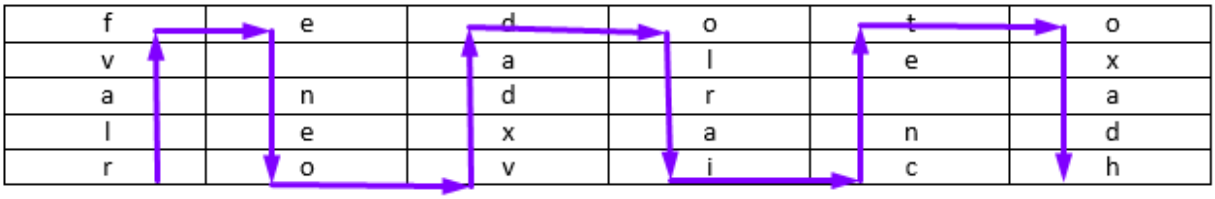
В этом шифре можно заменять как отдельные символы, так и пары, тройки букв и любую другую комбинацию. При шифровании символы текста перемещаются с исходных позиций в новые один раз. Текст записывается в таблицу, например, по строкам, а выписывается по столбцам [1].

Шифров маршрутной перестановки много, реализуем к примеру следующую перестановку – текст в таблицу записывается по строкам, а выписывается с первого столбца – снизу вверх, со второго столбца – сверху вниз, с третьего столбца – снизу вверх и т.д.

Рассмотрим на примере. Пусть есть текст ‘fedotov alexandr alexandrovich’.

Пусть будет таблица с 6 столбцами. Запишем в таблицу:

f	e	d	o	t	o
v		a	l	e	x
a	n	d	r		a
l	e	x	a	n	d
r	o	v	i	c	h



Тогда шифр будет таким: rlavfe neovxdadolraicn etoxadh

Напишем программу.

```
text = 'fedotov alexandr alexandrovich'
st = 6
table = []
for i, t in enumerate(text):
    if not i % st:
        table.append(list(text[i:i+st]))
print(table)
table_transpose = list(zip(*table))
print(table_transpose)
for i, t in enumerate(table_transpose):
    if not i % 2:
        table_transpose[i] = t[::-1]

print(table_transpose)

encrypt = ''
for t in table_transpose:
    for s in t:
        encrypt += s
print(encrypt)
```

Результат:

```
rlavfe neovxdadolraicn etoxadh
```

Шифр вертикальной перестановки.

Шифр похож на маршрутный, за исключением того, что столбцы выписываются по вертикалям (сверху вниз) в определенном порядке. Этот порядок записан в ключе[2].

Например. Пусть есть текст ‘fedotov alexandr alexandrovich’, пусть есть ключ – 52134.

Тогда имеем таблицу с пятью столбцами:

5	2	1	3	4
f	e	d	o	t
o	v		a	l
e	x	a	n	d
r		a	l	e
x	a	n	d	r
o	v	i	c	h

Зашифрованный текст будет – d aanievx av oanldctlderhfoerxo.

Напишем программу.

```
key = '52134'
text = 'fedotov alexandr alexandrovich'
table = []
for i, t in enumerate(text):
    if not i % len(key):
        table.append(text[i:i+len(key)])

print(table)
encrypt = ''
for k in range(len(key)):
    i = key.index(str(k+1))
    for t in table:
        if i < len(t):
            encrypt += t[i]
print(encrypt)
```

Результат:

```
['fedot', 'ov al', 'exand', 'r ale', 'xandr', 'ovich']
d aanievx avoanldctlderhfoerxo
```

Можно попробовать зашифровать картинку.

Сформируем ключ:

```
from random import shuffle
key = list(range(0, 64))
shuffle(key)
print(key)
```

Получим следующий ключ:

```
key = [20, 23, 35, 29, 10, 53, 41, 31, 59, 63, 60, 44, 38, 52, 30, 18,
       5, 14, 32, 22, 17, 36, 26, 15, 47, 24, 54, 11, 39, 27, 4,
      46, 58, 42, 43, 51, 57, 45, 28, 37, 40, 8, 3, 2, 0, 33, 1, 7,
      9, 49, 50, 16, 34, 55, 56, 19, 21, 25, 61, 6, 48, 62, 12, 13]
```

Программа:

```
data = read_write_file.read_data_1byte('data/f4.bmp')
table = []
for i, d in enumerate(data):
    if not i % len(key):
        table.append(data[i:i+len(key)])
encrypt = []
for k in range(len(key)):
    i = key.index(k)
    for t in table:
        if i < len(t):
            encrypt.append(t[i])
read_write_file.write_data_1byte('encrypt_data/f4_encrypt_v.bmp', encrypt)
```

Литература.

[1] Саймон Сингх. Книга шифров. Астрель. Москва 2006.

[2] Н.А. Ларина Защита информации. Криптология. Методическое пособие. Рубцовск 2014.