

Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(МОСКОВСКИЙ ПОЛИТЕХ)**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ  
КАФЕДРА «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

Программирование криптографических алгоритмов  
Блок G: Комбинационные шифры

Выполнила студентка 3 курса группы 171-341

Решетникова Дарья

Москва 2020 г.

## Аннотация

**Язык :** Python

**Программа:** Visual Studio 2017

**Пословица:** Плод никогда не падает далеко от дерева.

**Текст:** История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический процесс - последовательный процесс развития природы и общества, череда сменяющих друг друга событий, в которых проявляется деятельность многих поколений людей. С каждым мгновением времени, ушедшим в прошлое исторический процесс дополняется совокупностью новых событий, явлений, фактов и факторов в жизни человека, семьи, этноса, государства, человечества. Подобно тому, как материальный мир переживает процессы негэнтропии и энтропии, природа вокруг нас - процессы эволюции и инволюции, человеческое общество характеризуют процессы социального прогресса и регресса. Предметом изучения истории является деятельность главного его субъекта - человека - в прошлом. В узком смысле история представляет собой совокупность социальных фактов, знаний, представлений ученых о том, как эти процессы осуществлялись и осуществляются. Долгое время история существовала скорее, как литература и искусство, обслуживающие интересы власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств и наук, среди других муз была Клио - муза истории.

### SEO-АНАЛИЗ ТЕКСТА

FAQ API проверки

История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический процесс - последовательный процесс развития природы и общества, череда сменяющих друг друга событий, в которых проявляется деятельность многих поколений людей. С каждым мгновением времени, ушедшим в прошлое исторический процесс дополняется совокупностью новых событий, явлений, фактов и факторов в жизни человека, семьи, этноса, государства, человечества. Подобно тому, как материальный мир переживает процессы негэнтропии и энтропии, природа вокруг нас - процессы эволюции и инволюции, человеческое общество характеризуют процессы социального прогресса и регресса. Предметом изучения истории является деятельность главного его субъекта - человека - в прошлом. В узком смысле история представляет собой совокупность социальных фактов, знаний, представлений ученых о том, как эти процессы осуществлялись и осуществляются. Долгое время история существовала скорее, как литература и искусство, обслуживающие интересы власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств и наук, среди других муз была Клио - муза истории.

Всего символов: 1190    Без пробелов: 1042    Количество слов: 149

Заказать текст

Проверить SEO-данные

## AES.

### 1. Описание шифра.

Advanced Encryption Standard (AES), также известный как Rijndael — симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES. Этот алгоритм хорошо проанализирован и сейчас широко используется, как это было с его предшественником DES. В настоящее время из алгоритмов на основе SP-сетей широко используется AES и ГОСТ Р 34.12-2015 (Кузнечик).

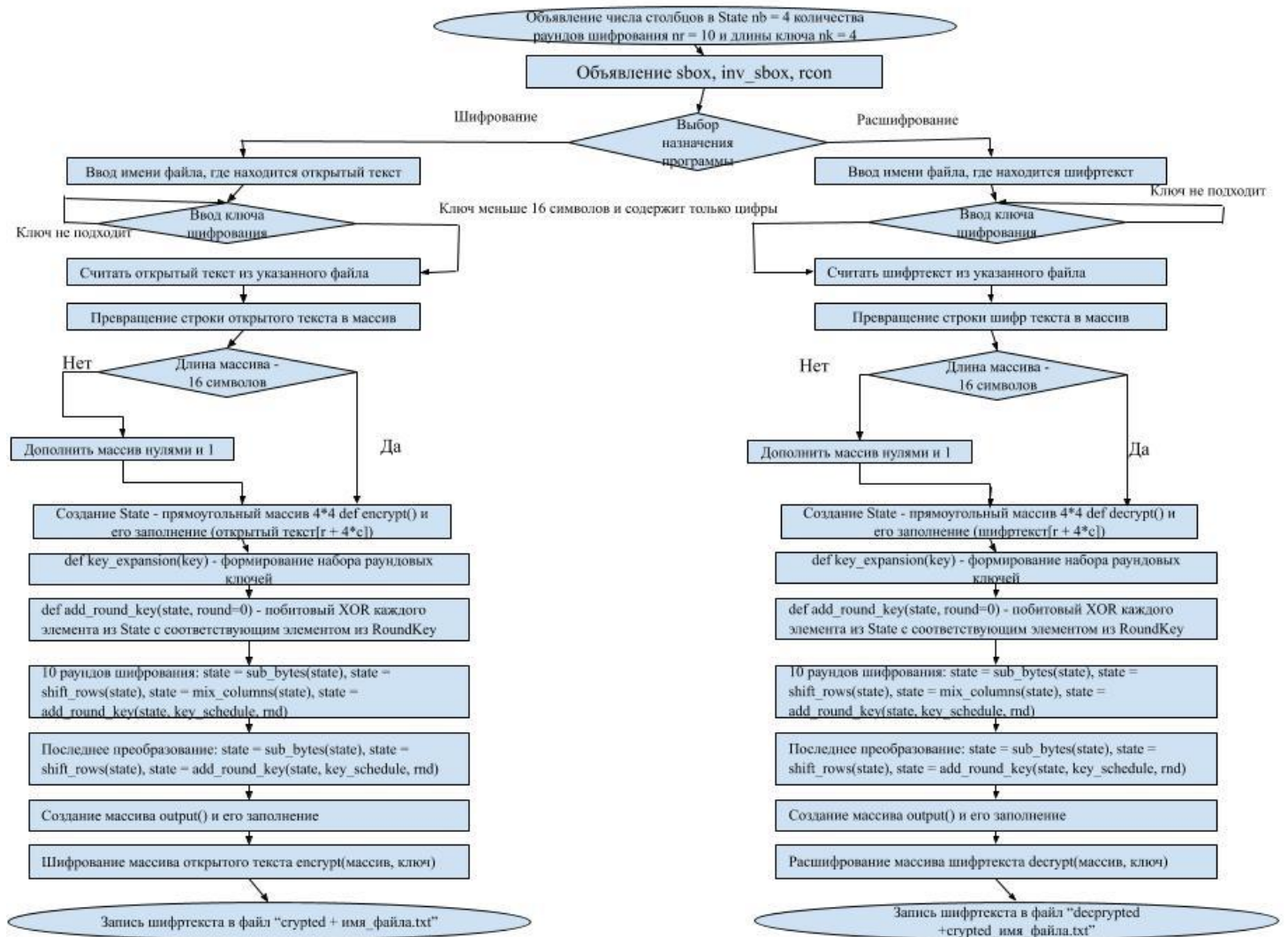
### 2. Алгоритм шифра.

Алгоритм шифрования получает на вход 128-битный блок данных  $input$  и расписание ключей  $w$ , которое получается после KeyExpansion. 16-байтовый  $input$  он записывает в виде матрицы  $s$  размера  $4 \times N_b$ , которая называется состоянием AES, и затем  $N_r$  раз применяет к этой матрице 4 преобразования. В конце он записывает матрицу в виде массива и подаёт его на выход — это зашифрованный блок. Каждое из четырёх преобразований очень простое.

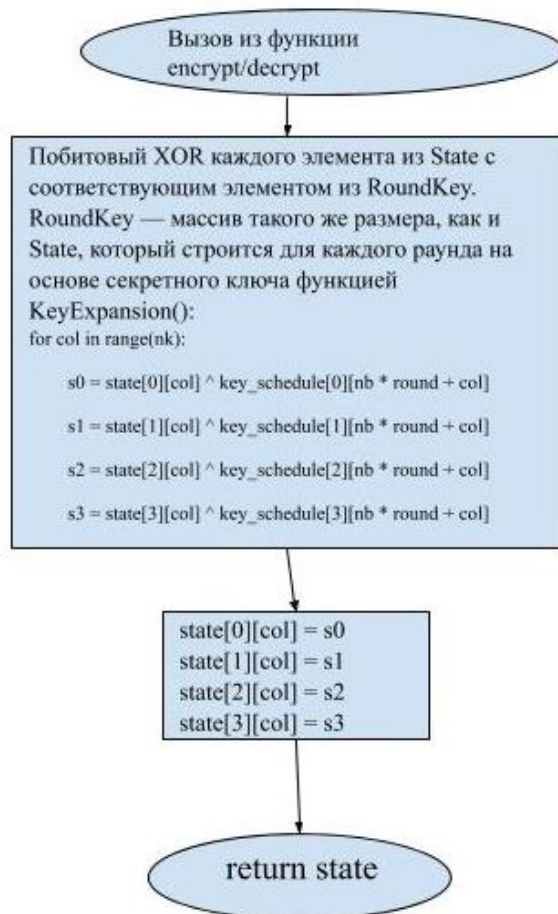
1. **AddRoundKey** берёт из расписания ключей одну матрицу размера  $4 \times N_b$  и поэлементно добавляет её к матрице состояния. Если два раза применить AddRoundKey, то ничего не изменится, поэтому преобразование обратное к AddRoundKey это оно само.
2. **SubBytes** заменяет каждый элемент матрицы состояния соответствующим элементом таблицы SBox:  $s_{ij} = SBox[s_{ij}]$ . Преобразование SubBytes обратимо. Обратное к нему находится с помощью таблицы InvSBox.
3. **ShiftRows** сдвигает  $i$ -ую строку матрицы  $s$  на  $i$  позиций влево, считая  $i$  с нуля. Обратное преобразование InvShiftRows сдвигает строки вправо.
4. **MixColumns** умножает каждый столбец матрицы  $s$  слева на особую матрицу размера  $4 \times 4$ .

### 3. Блок-схема программы

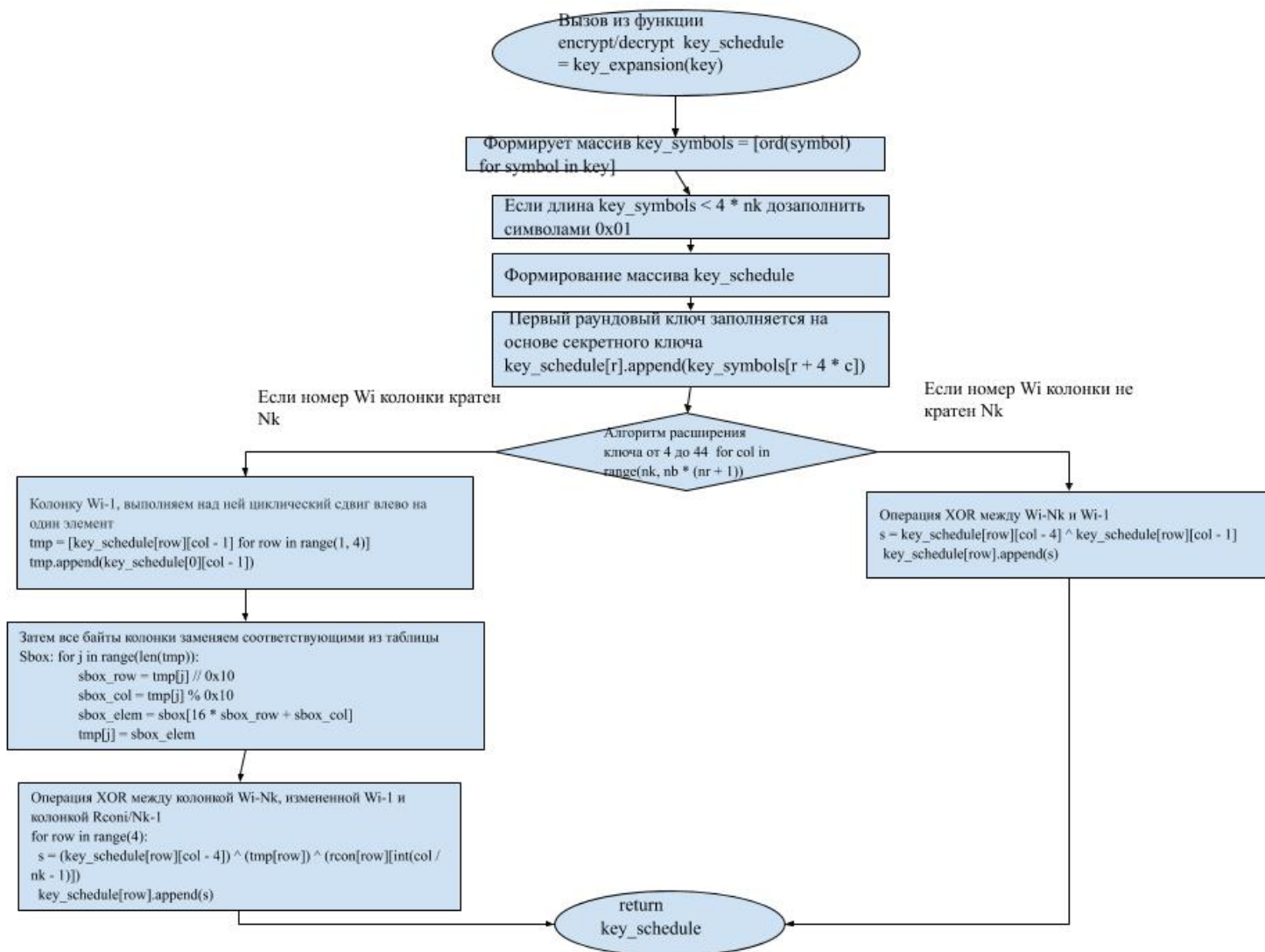
Общая блок-схема AES:



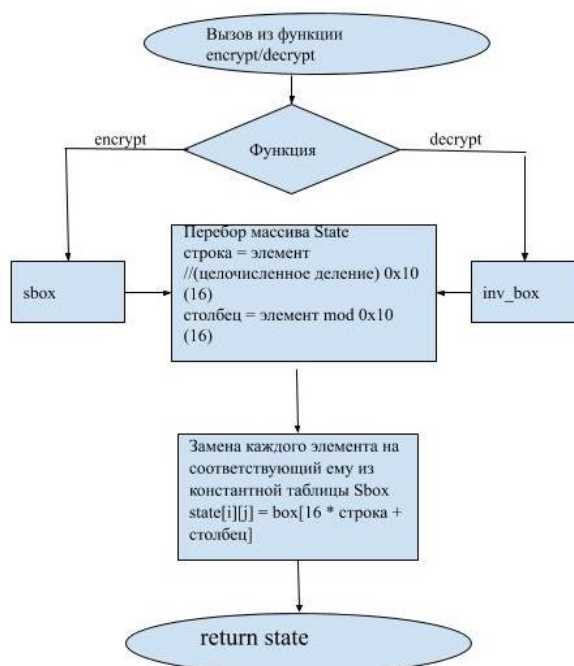
AddRoundKey



KeyExpansion

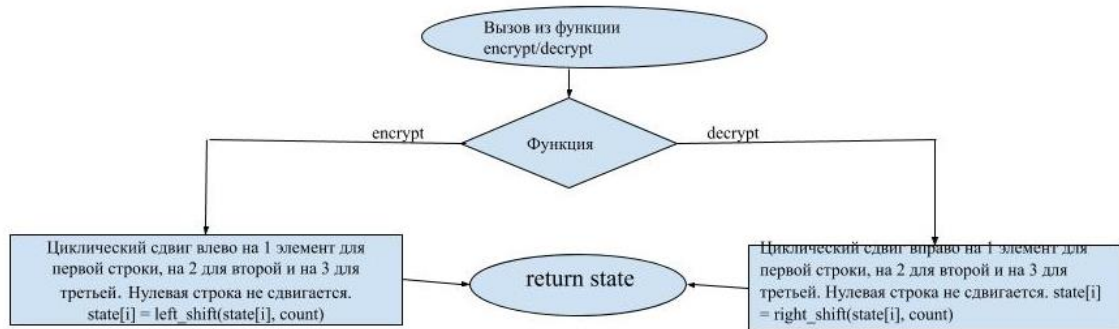


## SubBytes

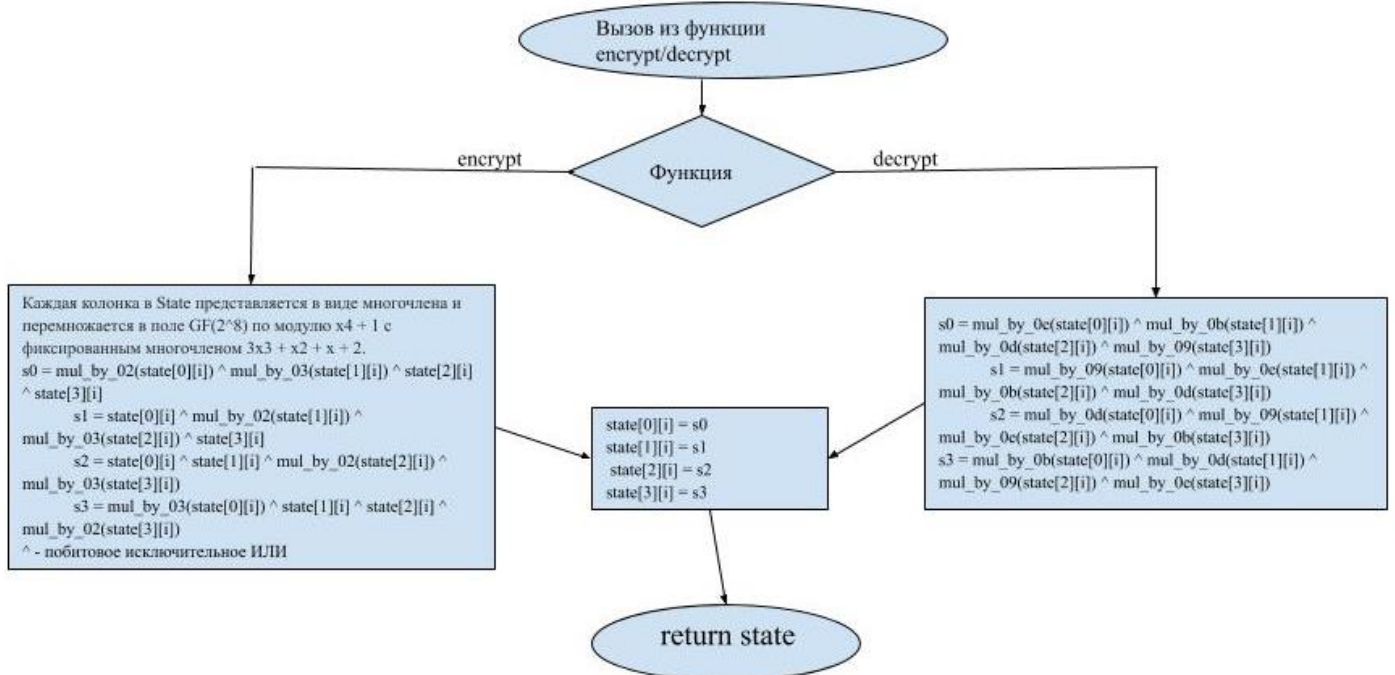




## ShiftRows



## MixColumns



$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \leq c < Nb.$$

$$\begin{aligned} s'_{0,c} &= (\{02\} \cdot s_{0,c}) \oplus (\{03\} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} &= s_{0,c} \oplus (\{02\} \cdot s_{1,c}) \oplus (\{03\} \cdot s_{2,c}) \oplus s_{3,c} \\ s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus (\{02\} \cdot s_{2,c}) \oplus (\{03\} \cdot s_{3,c}) \\ s'_{3,c} &= (\{03\} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \cdot s_{3,c}). \end{aligned}$$

## 4. Код программы

```
import os
```

```
import time
```

`nb = 4` # число столбцов (32-х битных слов), составляющих State. Для стандарта регламентировано `Nb = 4`

nr = 10 # количество раундов шифрования. В зависимости от длины ключа, Nr = 10, 12 или 14 (if nb = 4 nr = 10)

nk = 4 # длина ключа в 32-х битных словах. Для AES, Nk = 4, 6, 8. Nk = 4 (in 32-bit words)

# использовано в SubBytes().

hex\_symbols\_to\_int = {'a': 10, 'b': 11, 'c': 12, 'd': 13, 'e': 14, 'f': 15}

sbox = [

0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe,  
0xd7, 0xab, 0x76,

0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c,  
0xa4, 0x72, 0xc0,

0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71,  
0xd8, 0x31, 0x15,

0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb,  
0x27, 0xb2, 0x75,

0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29,  
0xe3, 0x2f, 0x84,

0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a,  
0x4c, 0x58, 0xcf,

0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50,  
0x3c, 0x9f, 0xa8,

0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10,  
0xff, 0xf3, 0xd2,

0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64,  
0x5d, 0x19, 0x73,

0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde,  
0x5e, 0x0b, 0xdb,

0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91,  
0x95, 0xe4, 0x79,

0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65,  
0x7a, 0xae, 0x08,



0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b,  
0xbd, 0x8b, 0x8a,  
0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86,  
0xc1, 0x1d, 0x9e,  
0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce,  
0x55, 0x28, 0xdf,  
0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0,  
0x54, 0xbb, 0x16  
]

inv\_sbox = [

0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81,  
0xf3, 0xd7, 0xfb,  
0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4,  
0xde, 0xe9, 0xcb,  
0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42,  
0xfa, 0xc3, 0x4e,  
0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d,  
0x8b, 0xd1, 0x25,  
0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d,  
0x65, 0xb6, 0x92,  
0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7,  
0x8d, 0x9d, 0x84,  
0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8,  
0xb3, 0x45, 0x06,  
0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01,  
0x13, 0x8a, 0x6b,  
0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0,  
0xb4, 0xe6, 0x73,  
0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c,  
0x75, 0xdf, 0x6e,  
0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa,  
0x18, 0xbe, 0x1b,

```

    0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78,
    0xcd, 0x5a, 0xf4,
    0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27,
    0x80, 0xec, 0x5f,
    0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93,
    0xc9, 0x9c, 0xef,
    0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83,
    0x53, 0x99, 0x61,
    0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55,
    0x21, 0x0c, 0x7d
]

```

```

rcon = [[0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36],
        [0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00],
        [0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00],
        [0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]
]

```

```

def encrypt(input_bytes, key):

```

```

    state = [[] for j in range(4)] # промежуточный результат шифрования,
    который может быть представлен как прямоугольный массив байтов
    имеющий 4 строки и Nb колонок.

```

```

    # Каждая ячейка State содержит значение размером в 1 байт

```

```

    for r in range(4):

```

```

        for c in range(nb):

```

```

            state[r].append(input_bytes[r + 4 * c])

```

```

#В начале заполняется массив State входными значениями по формуле State[r][c]
= input[r + 4c], r = 0,1...4; c = 0,1..Nb. То есть по колонкам. За раз шифруется блок
размером 16 байт.

```

```

    key_schedule = key_expansion(key)

```

```

    state = add_round_key(state, key_schedule)

```

```

for rnd in range(1, nr):
    state = sub_bytes(state)
    state = shift_rows(state)
    state = mix_columns(state)
    state = add_round_key(state, key_schedule, rnd)

state = sub_bytes(state)
state = shift_rows(state)
state = add_round_key(state, key_schedule, rnd + 1)

output = [None for i in range(4 * nb)]
for r in range(4):
    for c in range(nb):
        output[r + 4 * c] = state[r][c]

return output

```

```

def decrypt(cipher, key):
    state = [[] for i in range(nb)]
    for r in range(4):
        for c in range(nb):
            state[r].append(cipher[r + 4 * c])

    key_schedule = key_expansion(key)

    state = add_round_key(state, key_schedule, nr)

```

```

rnd = nr - 1
while rnd >= 1:
    state = shift_rows(state, inv=True)
    state = sub_bytes(state, inv=True)
    state = add_round_key(state, key_schedule, rnd)
    state = mix_columns(state, inv=True)

```

```

    rnd -= 1

```

```

state = shift_rows(state, inv=True)
state = sub_bytes(state, inv=True)
state = add_round_key(state, key_schedule, rnd)

```

```

output = [None for i in range(4 * nb)]

```

```

for r in range(4):

```

```

    for c in range(nb):

```

```

        output[r + 4 * c] = state[r][c]

```

```

return output

```

# замена каждого байта из State на соответствующий ему из константной таблицы Sbox. Значения элементов Sbox представлены в шестнадцатеричной системе счисления. Сама же таблица получена посредством преобразований поля  $GF(2^8)$ .

```

def sub_bytes(state, inv=False):

```

```

    if inv == False:

```

```

        box = sbox

```

```

    else:

```

```

        box = inv_sbox

```

```

for i in range(len(state)):
    for j in range(len(state[i])):
        row = state[i][j] // 0x10
        col = state[i][j] % 0x10

        box_elem = box[16 * row + col]
        state[i][j] = box_elem

return state

```

# Простая трансформация. Она выполняет циклический сдвиг влево на 1 элемент для первой строки, на 2 для второй и на 3 для третьей. Нулевая строка не сдвигается.

```

def shift_rows(state, inv=False):
    count = 1

    if inv == False:
        for i in range(1, nb):
            state[i] = left_shift(state[i], count)
            count += 1
    else:
        for i in range(1, nb):
            state[i] = right_shift(state[i], count)
            count += 1

    return state

```

# каждая колонка в State представляется в виде многочлена и перемножается в поле GF(28) по модулю  $x^4 + 1$  с фиксированным многочленом  $3x^3 + x^2 + x + 2$ .

```
def mix_columns(state, inv=False):
```

```
    for i in range(nb):
```

```
        if inv == False:
```

```
            s0 = mul_by_02(state[0][i]) ^ mul_by_03(state[1][i]) ^ state[2][i] ^  
state[3][i]
```

```
            s1 = state[0][i] ^ mul_by_02(state[1][i]) ^ mul_by_03(state[2][i]) ^  
state[3][i]
```

```
            s2 = state[0][i] ^ state[1][i] ^ mul_by_02(state[2][i]) ^  
mul_by_03(state[3][i])
```

```
            s3 = mul_by_03(state[0][i]) ^ state[1][i] ^ state[2][i] ^  
mul_by_02(state[3][i])
```

```
        else:
```

```
            s0 = mul_by_0e(state[0][i]) ^ mul_by_0b(state[1][i]) ^  
mul_by_0d(state[2][i]) ^ mul_by_09(state[3][i])
```

```
            s1 = mul_by_09(state[0][i]) ^ mul_by_0e(state[1][i]) ^  
mul_by_0b(state[2][i]) ^ mul_by_0d(state[3][i])
```

```
            s2 = mul_by_0d(state[0][i]) ^ mul_by_09(state[1][i]) ^  
mul_by_0e(state[2][i]) ^ mul_by_0b(state[3][i])
```

```
            s3 = mul_by_0b(state[0][i]) ^ mul_by_0d(state[1][i]) ^  
mul_by_09(state[2][i]) ^ mul_by_0e(state[3][i])
```

```
        state[0][i] = s0
```

```
        state[1][i] = s1
```

```
        state[2][i] = s2
```

```
        state[3][i] = s3
```

```
    return state
```

# вспомогательная трансформация формирует набор раундовых ключей — KeySchedule.



# KeySchedule представляет собой длинную таблицу, состоящую из  $N_b \cdot (N_r + 1)$  столбцов или  $(N_r + 1)$  блоков,

# каждый из которых равен по размеру State. Первый раундовый ключ заполняется на основе секретного ключа

```
def key_expansion(key):
```

```
    key_symbols = [ord(symbol) for symbol in key]
```

```
    if len(key_symbols) < 4 * nk:
```

```
        for i in range(4 * nk - len(key_symbols)):
```

```
            key_symbols.append(0x01)
```

```
    key_schedule = [[] for i in range(4)]
```

```
    for r in range(4):
```

```
        for c in range(nk):
```

```
            key_schedule[r].append(key_symbols[r + 4 * c])
```

```
    for col in range(nk, nb * (nr + 1)):
```

```
        if col % nk == 0:
```

```
            tmp = [key_schedule[row][col - 1] for row in range(1, 4)]
```

```
            tmp.append(key_schedule[0][col - 1])
```

```
            for j in range(len(tmp)):
```

```
                sbox_row = tmp[j] // 0x10
```

```
                sbox_col = tmp[j] % 0x10
```

```
                sbox_elem = sbox[16 * sbox_row + sbox_col]
```

```
                tmp[j] = sbox_elem
```

```
            for row in range(4):
```

```
                s = (key_schedule[row][col - 4]) ^ (tmp[row]) ^ (rcon[row][int(col / nk - 1)])
```

```

        key_schedule[row].append(s)

    else:

        for row in range(4):

            s = key_schedule[row][col - 4] ^ key_schedule[row][col - 1]

            key_schedule[row].append(s)

    return key_schedule

# производит побитовый XOR каждого элемента из State с соответствующим
элементом из RoundKey.

# RoundKey — массив такого же размера, как и State,

# который строится для каждого раунда на основе секретного ключа
функцией KeyExpansion()

def add_round_key(state, key_schedule, round=0):

    for col in range(nk):

        s0 = state[0][col] ^ key_schedule[0][nb * round + col]
        s1 = state[1][col] ^ key_schedule[1][nb * round + col]
        s2 = state[2][col] ^ key_schedule[2][nb * round + col]
        s3 = state[3][col] ^ key_schedule[3][nb * round + col]

        state[0][col] = s0
        state[1][col] = s1
        state[2][col] = s2
        state[3][col] = s3

    return state

#left_shift/right_shift(array, count) поворачивают входной array в
соответствующую сторону count раз

```

```
def left_shift(array, count):  
    res = array[:]  
    for i in range(count):  
        temp = res[1:]  
        temp.append(res[0])  
        res[:] = temp[:]  
  
    return res
```

```
def right_shift(array, count):  
    res = array[:]  
    for i in range(count):  
        tmp = res[:-1]  
        tmp.insert(0, res[-1])  
        res[:] = tmp[:]  
  
    return res
```

# Вспомогательные функции умножения

```
def mul_by_02(num):  
    if num < 0x80:  
        res = (num << 1)  
    else:  
        res = (num << 1) ^ 0x1b  
  
    return res % 0x100
```

```
def mul_by_03(num):
```

```

    return (mul_by_02(num) ^ num)

def mul_by_09(num):
    return mul_by_02(mul_by_02(mul_by_02(num))) ^ num

def mul_by_0b(num):
    return mul_by_02(mul_by_02(mul_by_02(num))) ^ mul_by_02(num) ^ num

def mul_by_0d(num):
    return mul_by_02(mul_by_02(mul_by_02(num))) ^
mul_by_02(mul_by_02(num)) ^ num

def mul_by_0e(num):
    return mul_by_02(mul_by_02(mul_by_02(num))) ^
mul_by_02(mul_by_02(num)) ^ mul_by_02(num)

print('Шаг 1:')

while True: # цикл с постусловием

    print('Нажмите 1, чтобы выполнить функцию зашифрования. Нажмите 2,
чтобы выполнить функцию расшифрования.')

    way = input()

    if way not in ['1', '2']:

        print('Действие отклонено.')

        continue

    else:

        break

print()

print('Шаг 2:')

while True:

```

```
print('Введите полное имя файла.')

input_path = os.path.abspath(input()) # os.path.abspath(path) - возвращает
нормализованный абсолютный путь

if os.path.isfile(input_path): # os.path.isfile(path) - является ли путь файлом.
    break
else:
    print('Такого файла не существует.')
    continue

print()
```

```
print('Шаг 3:')

while True:

    print('Введите ключ зашифрования/расшифрования. Ключ должен быть
меньше 16 символов.')

    key = input()

    if len(key) > 16:

        print('Слишком длинный ключ. Введите другой')

        continue

    for symbol in key:

        if ord(symbol) > 0xff:

            print('Этот ключ не подходит. Используйте цифры, либо латинские
буквы.')

            continue

    break

print("\r\nОжидайте...")
```

```
time_before = time.time()

# Входные данные

with open(input_path, 'rb') as f:

    data = f.read()
```

```

# при зашифровании
if way == '1':
    crypted_data = []
    temp = []
    for byte in data:
        temp.append(byte) # превращение строки из файла в массив
        if len(temp) == 16: # если длина ключа равна 1
            crypted_part = encrypt(temp, key) # зашифрование
            crypted_data.extend(crypted_part) # добавление в массив, без этого мы
получим пустой файл
            del temp[:] # очистить массив
        else:
            # если ключ короче 16 символов
            if 0 < len(temp) < 16:
                empty_spaces = 16 - len(temp)
                for i in range(empty_spaces - 1): # заполнить массив 0
                    temp.append(0)
                temp.append(1) # заполнить массив 1
                crypted_part = encrypt(temp, key) # зашифрование
                crypted_data.extend(crypted_part) # добавление в массив, без этого мы
получим пустой файл
            out_path = os.path.join(os.path.dirname(input_path) , 'crypted_' +
os.path.basename(input_path))
            # os.path.join(path1[, path2[, ...]]) - соединяет пути с учётом особенностей
операционной системы
            # os.path.dirname(path) - возвращает имя директории пути path
            # os.path.basename(path) - базовое имя пути
            # Выходные данные зашифрования
            with open(out_path, 'wb') as ff:
                ff.write(bytes(crypted_data)) # превращение в байтовую строку

```



```
# при расшифровании
```

```
else:
```

```
    decrypted_data = []
```

```
    temp = []
```

```
    for byte in data:
```

```
        temp.append(byte)
```

```
    if len(temp) == 16:
```

```
        decrypted_part = decrypt(temp, key)
```

```
        decrypted_data.extend(decrypted_part)
```

```
        del temp[:]
```

```
    else:
```

```
# если ключ короче 16 символов
```

```
    if 0 < len(temp) < 16:
```

```
        empty_spaces = 16 - len(temp)
```

```
        for i in range(empty_spaces - 1):
```

```
            temp.append(0)
```

```
        temp.append(1)
```

```
        decrypted_part = encrypt(temp, key)
```

```
        decrypted_data.extend(decrypted_part)
```

```
    out_path = os.path.join(os.path.dirname(input_path) , 'decrypted_' +  
os.path.basename(input_path))
```

```
# Выходные данные расшифрования
```

```
with open(out_path, 'wb') as ff:
```

```
    ff.write(bytes(decrypted_data))
```

```
time_after = time.time()
```

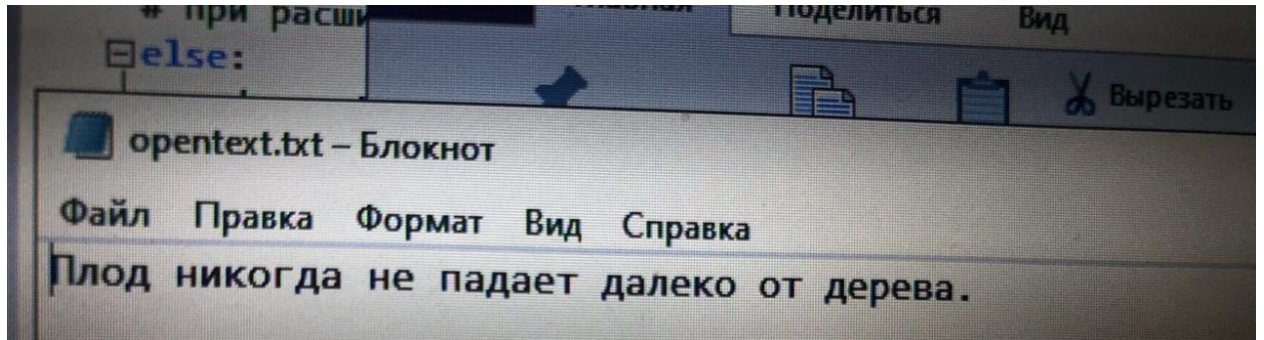
```
print('Новый файл:', out_path, '--', time_after - time_before, ' секунд.')
```

```
print('Если что-то не так проверьте ключ.')
```

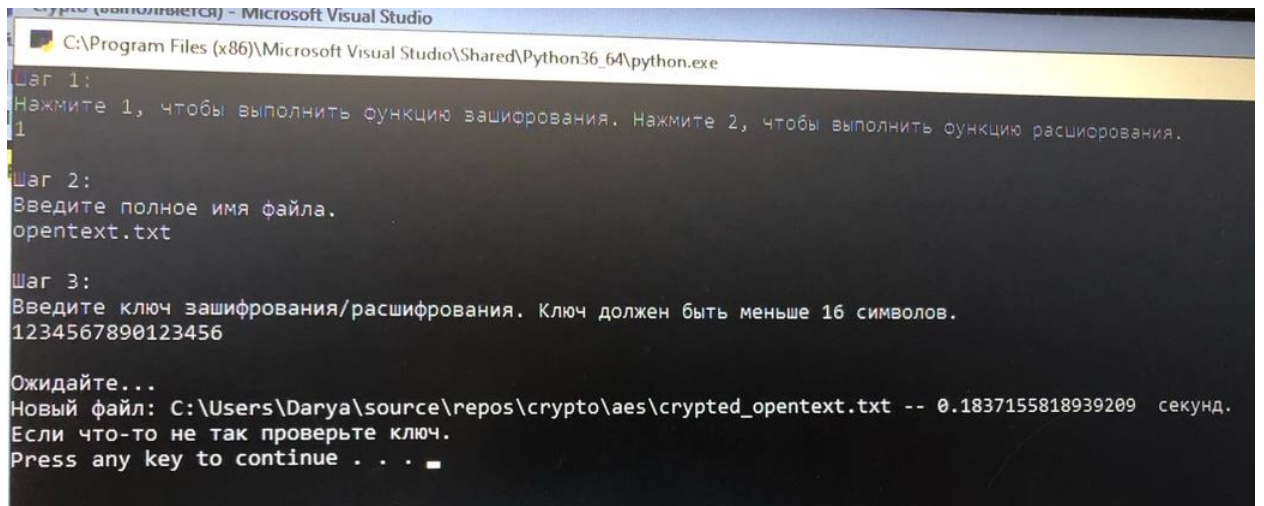
```
# 1054613298567235
```

## 5. Тестирование

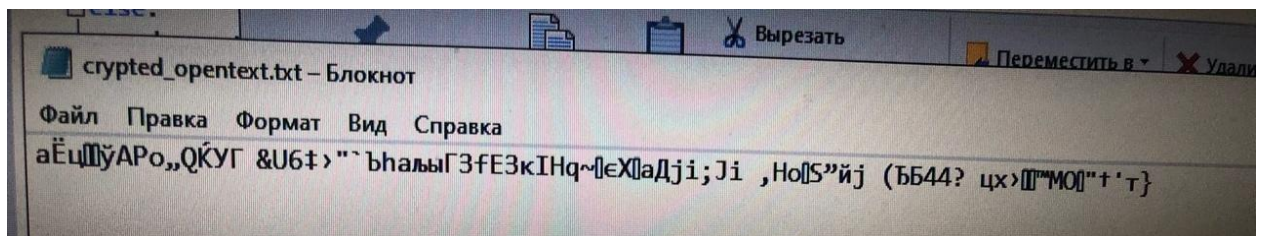
Перед началом работы программы в файл “opentext.txt” записываем исходный текст.



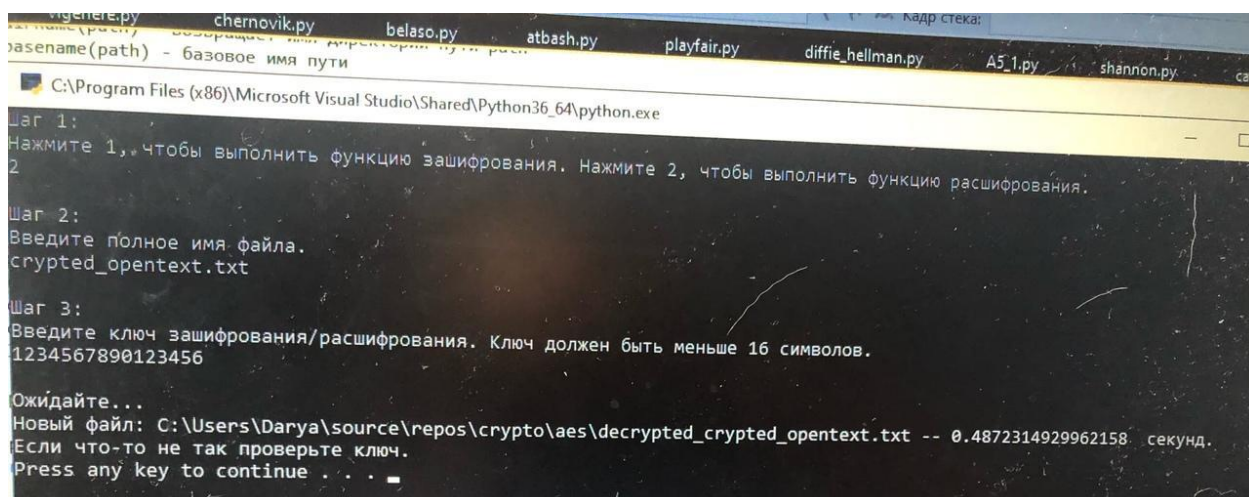
Результат выполнения программы:



После выполнения программы в файл “ crypted\_opentext.txt” записывается зашифрованный текст.



Далее перезапускаем программу:



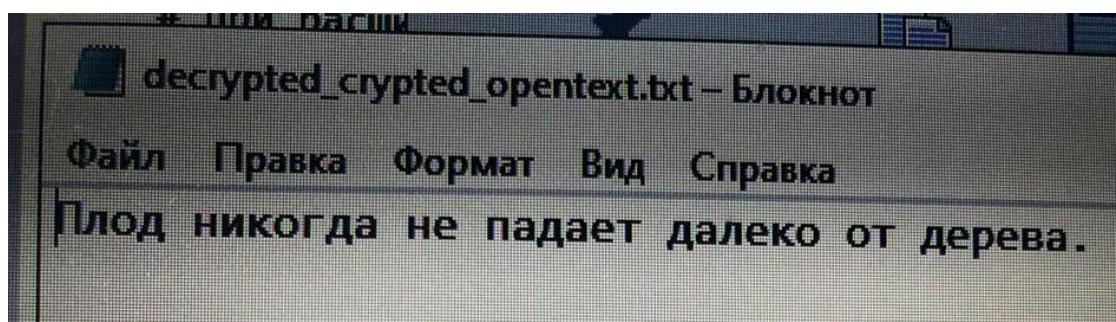
```
Step 1:
Press 1, to perform the encryption function. Press 2, to perform the decryption function.
2

Step 2:
Enter the full file name.
decrypted_opentext.txt

Step 3:
Enter the encryption/decryption key. The key must be less than 16 characters.
1234567890123456

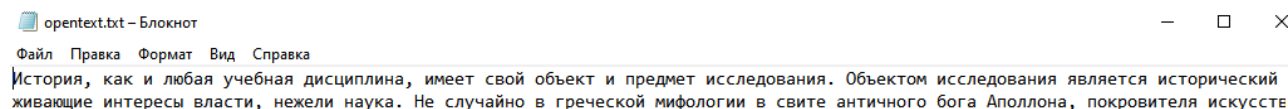
Waiting...
New file: C:\Users\Darya\source\repos\crypto\aes\decrypted_decrypted_opentext.txt -- 0.4872314929962158 seconds.
If something is not right, check the key.
Press any key to continue . . .
```

In the file “decrypted\_decrypted\_opentext.txt” the decrypted text.

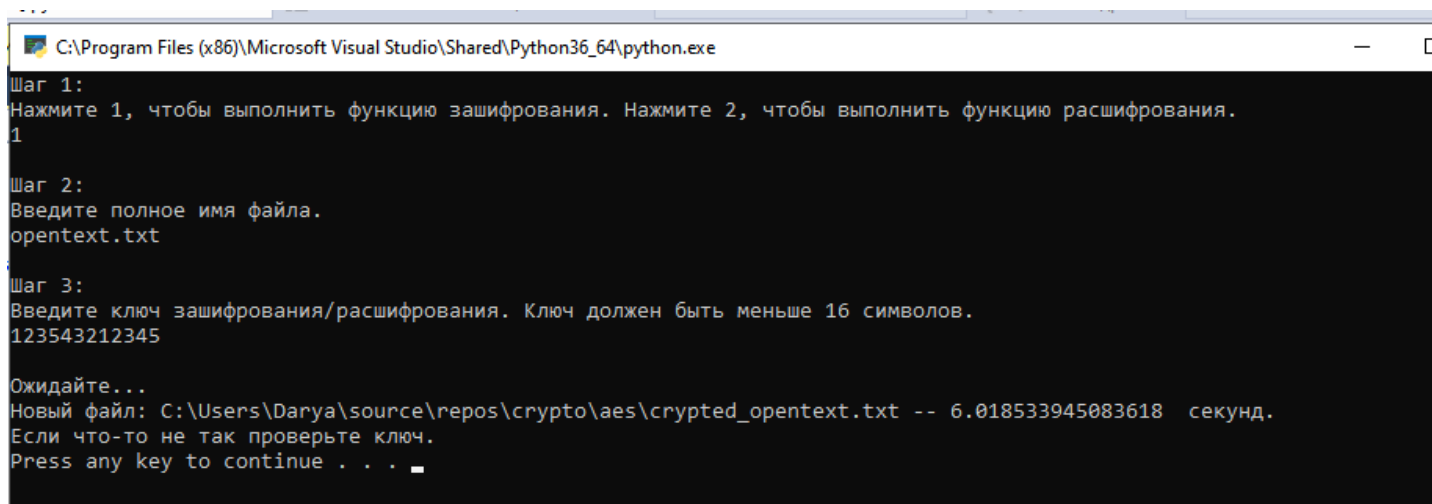


## 6. Work with text not less than 1000 characters

Before the start of the program work in the file “opentext.txt” we record the original text. (The full original text is in the annotation)



Thus the execution window of the program.



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Step 1:
Press 1, to perform the encryption function. Press 2, to perform the decryption function.
1

Step 2:
Enter the full file name.
opentext.txt

Step 3:
Enter the encryption/decryption key. The key must be less than 16 characters.
123543212345

Waiting...
New file: C:\Users\Darya\source\repos\crypto\aes\crypted_opentext.txt -- 6.018533945083618 seconds.
If something is not right, check the key.
Press any key to continue . . .
```

После выполнения программы в файл “crypted\_opentext.txt” записывается зашифрованный текст.



Полный зашифрованный текст:

В\*7ђ&я¶Uαπ!G+MEbЮОдDW©4°йф6°ь-|x=sJЌЌЙ"..."mУЯНОЮЇјα-"...тЩС  
;ьy!lXMaпJюйцбK?Hб#гхььU1sDњo9\$гУЦЫl#m:я‘эw}xX>крUzi...psA,w3Q:K  
ЫIV±ΠiB@...ЫА†ЇαБэыиА%o

...\$>9x мFпKyJзчж%o.УμЄκ?’g+(eA^5сТлО—  
†vb^ьo~льы<Т ЫIwпCпмь2!~б™“ИшуГ38\*ЌЖЃ\Мf06—  
I@z]Из±BгαGЩаг†Тoщ

mwЌ,(@)w!•Пђ-Ќ°ь-ећРльб—№өМФЕ“мцG><y!  
ыиOA\_<ѓєHцЮд3Fi”™9\њOUGNLчoмэpK(sH!‘oЁgrЁЮс9±6DİдюErвчJbkѓE  
q\_њ•i,щ]K>]O±\*© йт{йV2\_иBf†C=aTX{—PjлтsBO

|гфWlЦ8lhЧ!кFІp2TW

fЧ,7Iw8u\сXS,6!aЉз{3И=ЌЩюO—jQQ°Щ7el,,‡eTlC‘ИЕЉ

Вмн±п”Su\$s™Л-  
-п7ў’јгZћ\$miУФрц1№15Sү€№иЩ.33oЛкЭ;bz¶HJЉdbц&@...Эп<ћ8uP7N»Oөј  
уαхOтцёЮЦТЬКЕЩи5хшZсЯЩ•Y>”щбё\*є—[ЕЯ—HR«h®®°>м+  
kXишхЙobrrI4езРйрш&etYќ

ДаDзЌгУ»Ь6μр~.с@Ф07`C/цd\$"ђФХуκμC”iЕуЛћf6ЯХцТ(Є

+»W24ц  
vFBФ)t VрБдя†фK¶ШуKj№{Љ\_αф-н\*l\$:<чзП-\$5

V...`\_ЃТд<vТ стЦ’\$ИР2Oј+пл9dгЃЃЌ“  
!6кцЌZ`Щ`<%орЙкЭфI†Qє,ФЦd«Е-ўѓэ}ћ7ў`°™

ibaЄJUшJљF,,...л“һуи0гњLШ!Идсп%?N\SeЦщSү9њЁјЬ  
t©pR&bгdd>Књ...exЁх-  
~qщТ-њ/

||B7O.гь\_і`M\$ш...GuяVЄЭЖФт{μє%yS—  
VΠЖ,пXBтие0ФсЃМмGUћж.€€sjРйеOј«SbЦH™

%o^сЃуфњ\*?i,,nmbEИХ© №®<=й%‰Ж2%\«KJ0зk;?



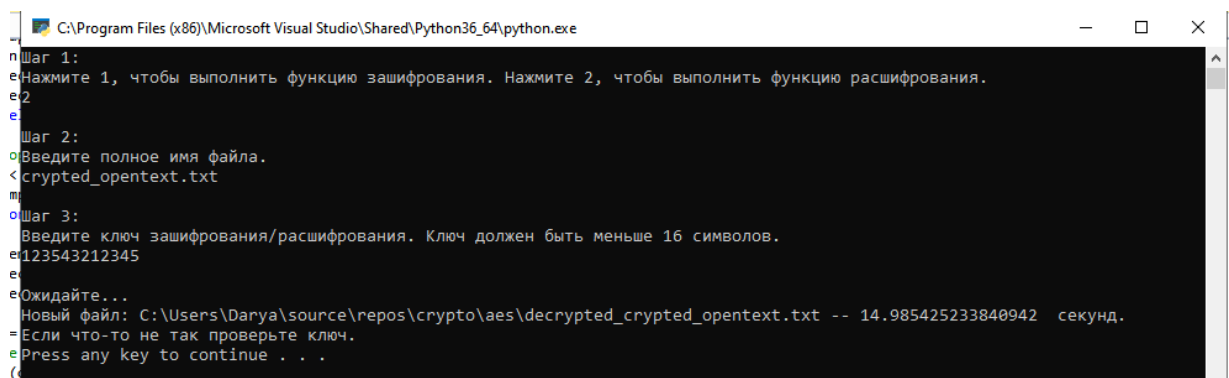
бхРіічЩс]чх%/Э,8ЖЄRMk€еЗ|Н%ТГЬбхйр™Льq(S,†§єЙЮР}Нь“qt(uYГ‘#9бль  
Nc 3цcq¶jбzhу ЪНЁ™)^X^Ё,qH–  
уйСѠаКэщО=л2kK3щIаЁыP^ЧsЫбтУћГйPфжЩNГF6ќ@Їяf=ЪМ alf‘–  
”кWc[ЬьGuwaMcЇ+] 1Ќj«SbЦН™%о^сЪЦуфНь±Q¶7эђ[“#IШЗЭCkhg

ЈиПќF0E;Ќы Ж-IёMAf-SзйS¶ї-w\*(&bЫ”Ыy@Ъ!x”ђ3NssYEфv  
=<^Ex§jZГ;ЄЌm°миь-ckфwp-ц|Fj1†‡о  
ia‡”ш’TM|ФPQLp® iN[Ї†бj‡#1иё%одЕо•ЎуНiЪ#УWёl<1%оЛ<6ПЬи-  
X|xuvћpМФ4ЩsѓПт¶Ньет3;K-†-t\*’q  
ЖХЌ  
qь ПквѠЄ-KG7¶UZЪ¶АтыICY  
йPE®ёMv’ћ /®b

Ч5I‘%оТ  
%os0=F\_ќ%оG<DћУлЎфАМУ2vмъQqОэЛ§ж.тс]]Л–Сэ»,Кў9^<№0μЭ\*Sё§  
QкЧ?>и  
sKЄMdj‘ЫN9=i^ђ°†ЕЎёв¶Q ryko©D|ћ–(#\x]ВИЛЬJ.АхБЯ  
\\nWHгнщ,,Ёрь§OеDBh|њх9д-haJeТ–  
МЎ,,†ЖАНРТе”уЖjj»,®,УЌАЬЌ Ие&ђiС{А;іX~3“В~dfЎuьv|XJ(АІйшГЬьх  
ЭЧN,  
™S"=<5i|Т<\_сШ!еЕ,,и<ьГФЧ\*п>льЫйБЖ¶ЬеЄвв{3Ph9\j5§9;)хЭ%с5МСрў  
\\шЭхКП>G-і[Xћзуг—P7п^Фц°,WJ|#§ќћv)ЮммаЙ  
ЕцЕЮz¶тГЮцсЭє4,,ЌJIZ°/Г ХЎючуђ§5|жяћ–и°:¶—|–wѓюёe[1

Сьqыэ#С”Ё(-.%о.–«КОHμ ѓ§[X-pJэнЙс]Й¶wбУруНиТ°Г>ёѠ;ж–k—  
<b>иСбьљу!mn>РЩ@\*t;Ц\*±ЫШЄЖщijй“КщГJ™—  
кіъщПѓWK§OVlAg?Љаг°SpГЖWдгаюВХгpw]aЄI–:ѓНрК^Е—чhЙль\Ц  
і|ђћ\_<™!Ъль gsf\$|:X-S) –Ы%оGM,Ф%оы^ly©ГшљуьEwtЌ'нЛ с;FIЛ  
b™<C-Ej}S1Flr,FћхЎ\S,±№PS7ЯЙ=гЙ?ЙJ\$мГ©һи€wBA-  
АЧ^,,х}5Б(ђћВО`rPS’Z¶r,Ѡш°©є‘.«д^Г 1ш  
Тtr\_ќшlbщАкЎж2црC[SdmРЖУ4с]Є–љ7\Нд,ЫТеВ}®-ЯU™Те—;в

Далее перезапускаем программу:



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Шаг 1:
e Нажмите 1, чтобы выполнить функцию зашифрования. Нажмите 2, чтобы выполнить функцию расшифрования.
e 2
e
Шаг 2:
o Введите полное имя файла.
< crypted_opentext.txt
m
Шаг 3:
o Введите ключ зашифрования/расшифрования. Ключ должен быть меньше 16 символов.
e 123543212345
e
e Ожидайте...
Новый файл: C:\Users\Darya\source\repos\crypto\aes\decrypted_crypted_opentext.txt -- 14.985425233840942 секунд.
= Если что-то не так проверьте ключ.
e Press any key to continue . . .
e
```

В файле “decrypted\_crypted\_opentext.txt” расшифрованный текст.



Полный расшифрованный текст:

История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический процесс - последовательный процесс развития природы и общества, череда сменяющих друг друга событий, в которых проявляется деятельность многих поколений людей. С каждым мгновением времени, ушедшим в прошлое исторический процесс дополняется совокупностью новых событий, явлений, фактов и факторов в жизни человека, семьи, этноса, государства, человечества. Подобно тому, как материальный мир переживает процессы негэнтропии и энтропии, природа вокруг нас - процессы эволюции и инволюции, человеческое общество характеризуют процессы социального прогресса и регресса. Предметом изучения истории является деятельность главного его субъекта - человека - в прошлом. В узком смысле история представляет собой совокупность социальных фактов, знаний, представлений ученых о том, как эти процессы осуществлялись и осуществляются. Долгое время история существовала скорее, как литература и искусство, обслуживающие интересы власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств и наук, среди других муз была Клио - муза истории.

7. Вся работа происходит в файлах: "opentext.txt", "crypted\_opentext.txt", "decrypted\_crypted\_opentext.txt".

## **ГОСТ 28147-89.**

### **1. Описание шифра.**

ГОСТ 28147-89 «Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования» — устаревший государственный стандарт союза ССР (а позже межгосударственный стандарт СНГ), описывающий алгоритм симметричного блочного шифрования и режимы его работы.

Является примером DES-подобных криптосистем, созданных по классической итерационной схеме Фейстеля.

ГОСТ 28147-89 — симметричный блочный алгоритм шифрования с 256-битным ключом, оперирует блоками данных по 64 бита.

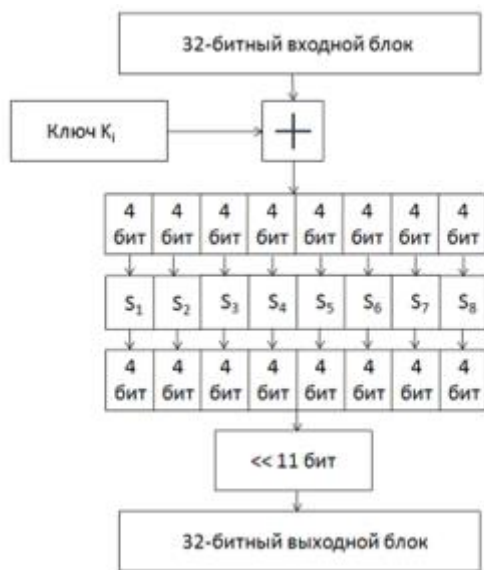
Один из режимов его работы, гаммирования с обратной связью, является потоковым режимом блочного шифра.

### **2. Алгоритм шифра.**

1. Исходное сообщение разбивается на блоки по 64 бита

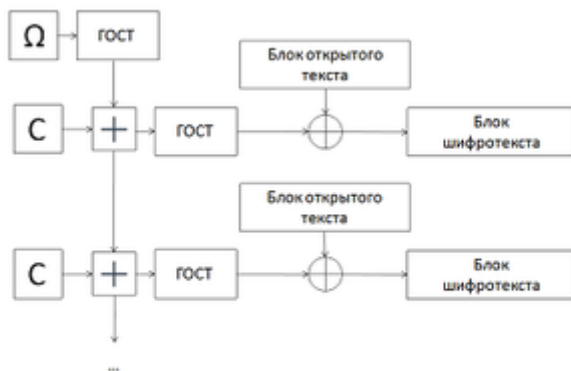


2. На каждый блок XOR'ом «накладывается» гамма, тоже длиной 64 бита
3. Гамма формируется шифрованием 64-битного блока «состояния» с помощью ключа в режиме простой замены
  - a. В момент начала шифрования сообщения блок принимается равным синхропосылке или вектору инициализации
  - b. В следующей итерации вместо синхропосылки используется зашифрованный блок текста из предыдущей



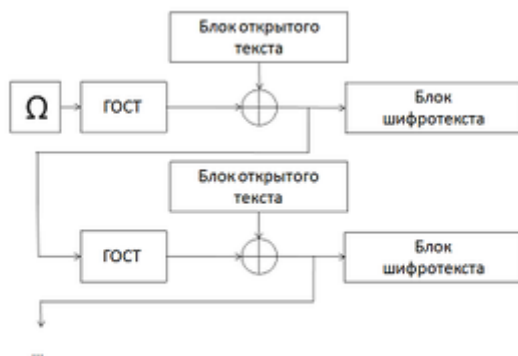
Алгоритм выработки:

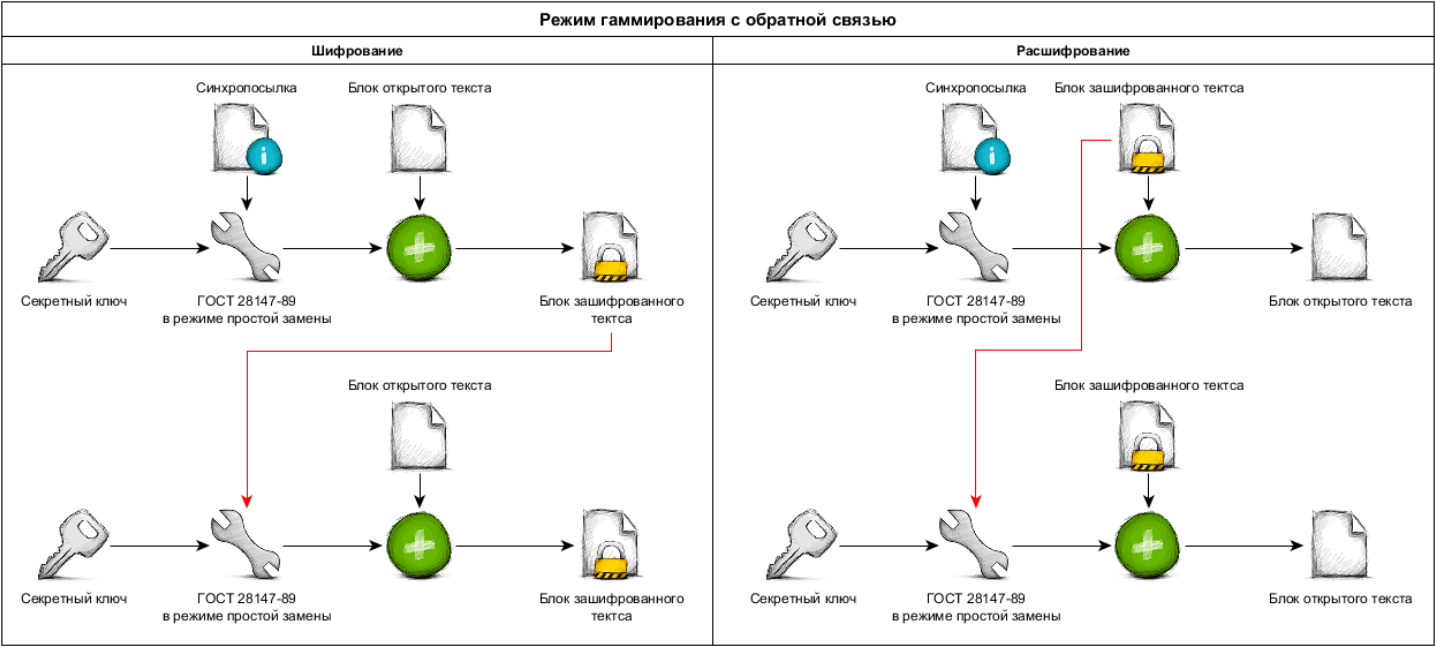
1. Синхропосылка шифруется с использованием описанного алгоритма простой замены, полученные значения записываются во вспомогательные 32-разрядные регистры  $N_3$  и  $N_4$  - младшие и старшие биты соответственно.
2.  $N_3$  суммируется по модулю  $2^{32}$  с константой  $C_2 = 1010101_{16}$
3.  $N_4$  суммируется по модулю  $2^{32}-1$  с константой  $C_1 = 1010104_{16}$
4.  $N_3$  и  $N_4$  переписываются соответственно в  $N_1$  и  $N_2$ , которые затем шифруются с использованием алгоритма простой замены. Полученный результат является 64 битами гаммы.
5. Шаги 2-4 повторяются в соответствии с длиной шифруемого текста.



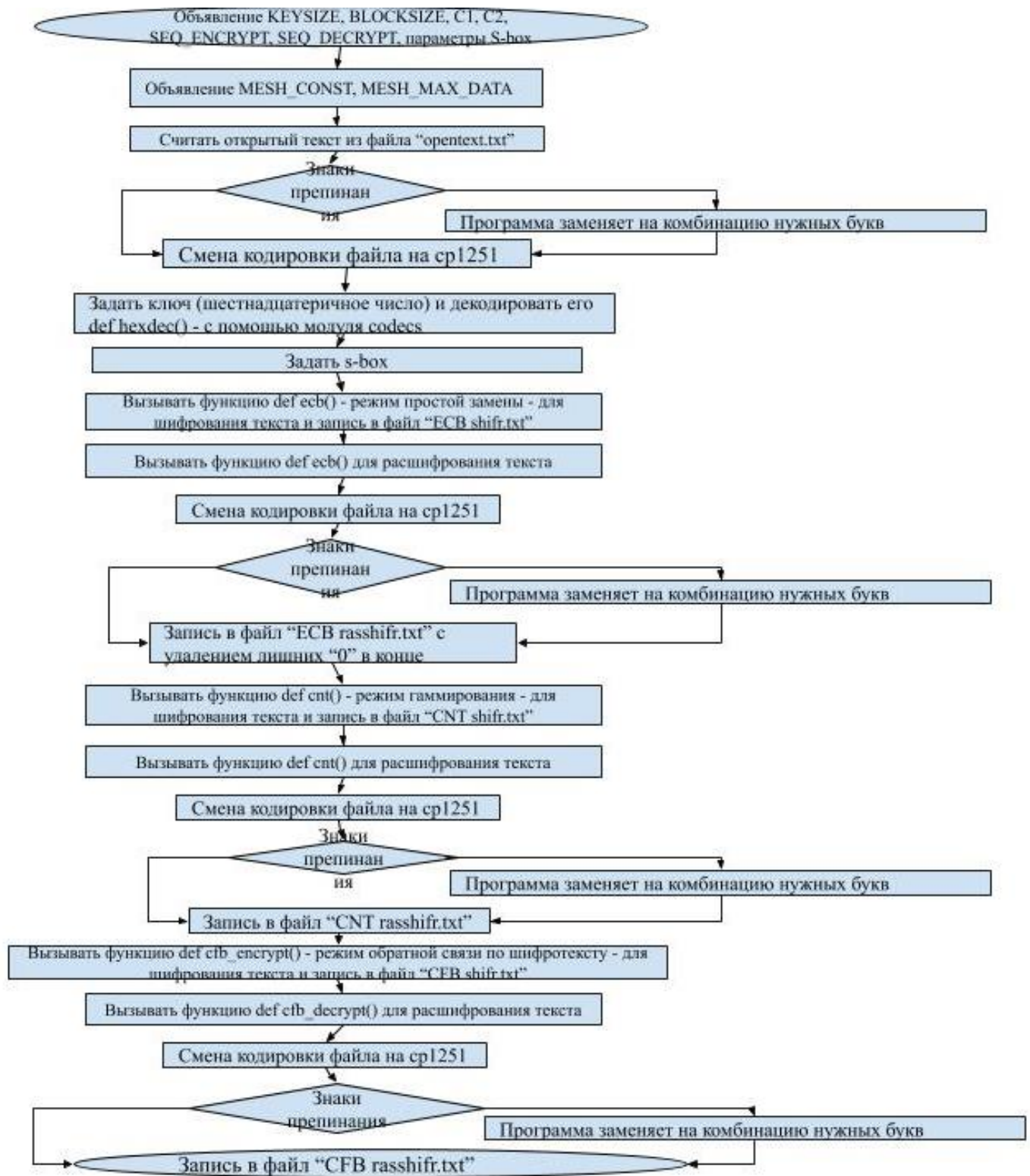
Алгоритм:

1. Синхропосылка заносится в регистры  $N_1$  и  $N_2$
2. Содержимое регистров  $N_1$  и  $N_2$  шифруется в соответствии с алгоритмом простой замены. Полученный результат является 64-битным блоком гаммы.
3. Блок гаммы побитно складывается по модулю 2 с блоком открытого текста. Полученный шифротекст заносится в регистры  $N_1$  и  $N_2$
4. Операции 2-3 выполняются для оставшихся блоков требующего шифрования текста.

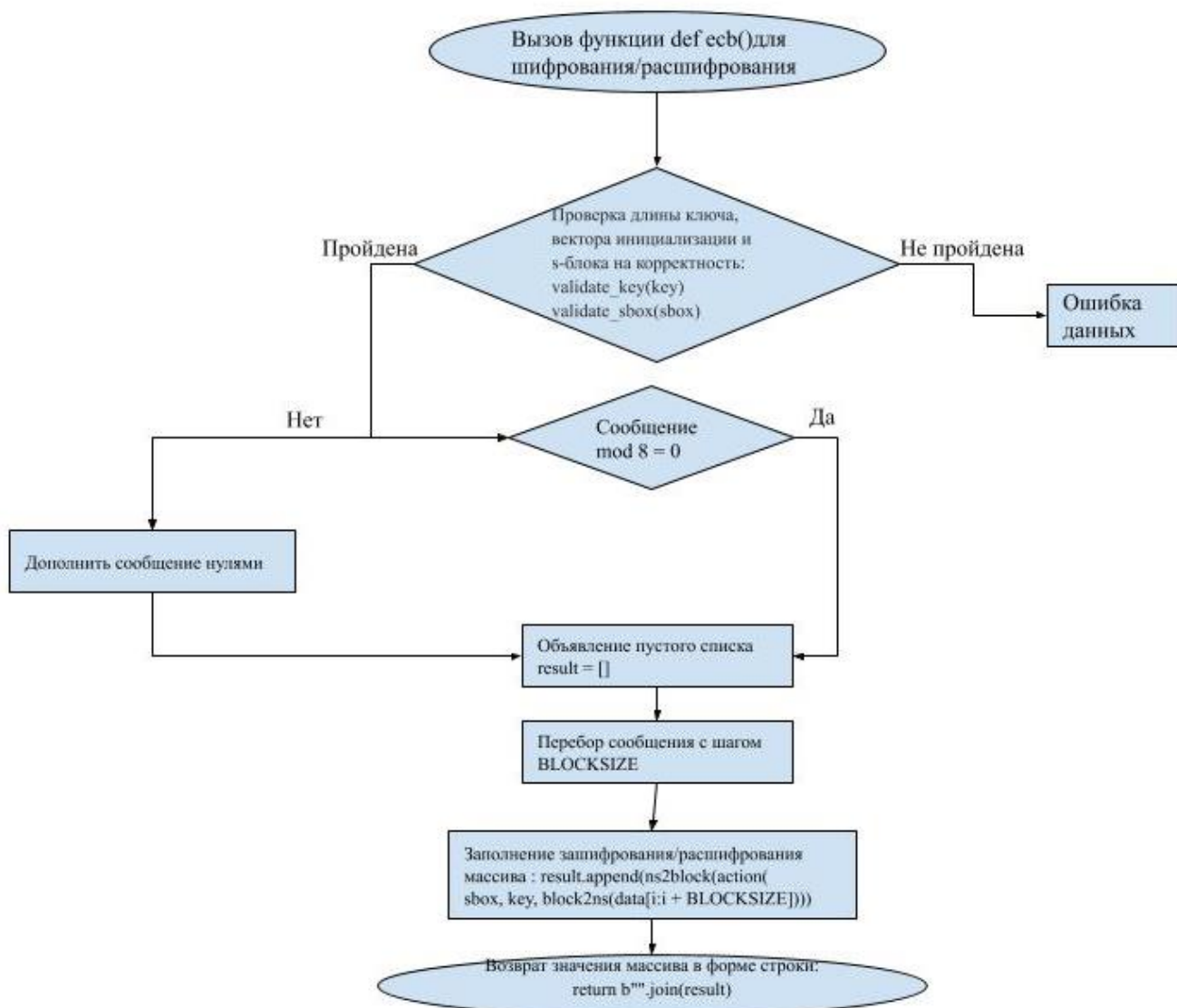




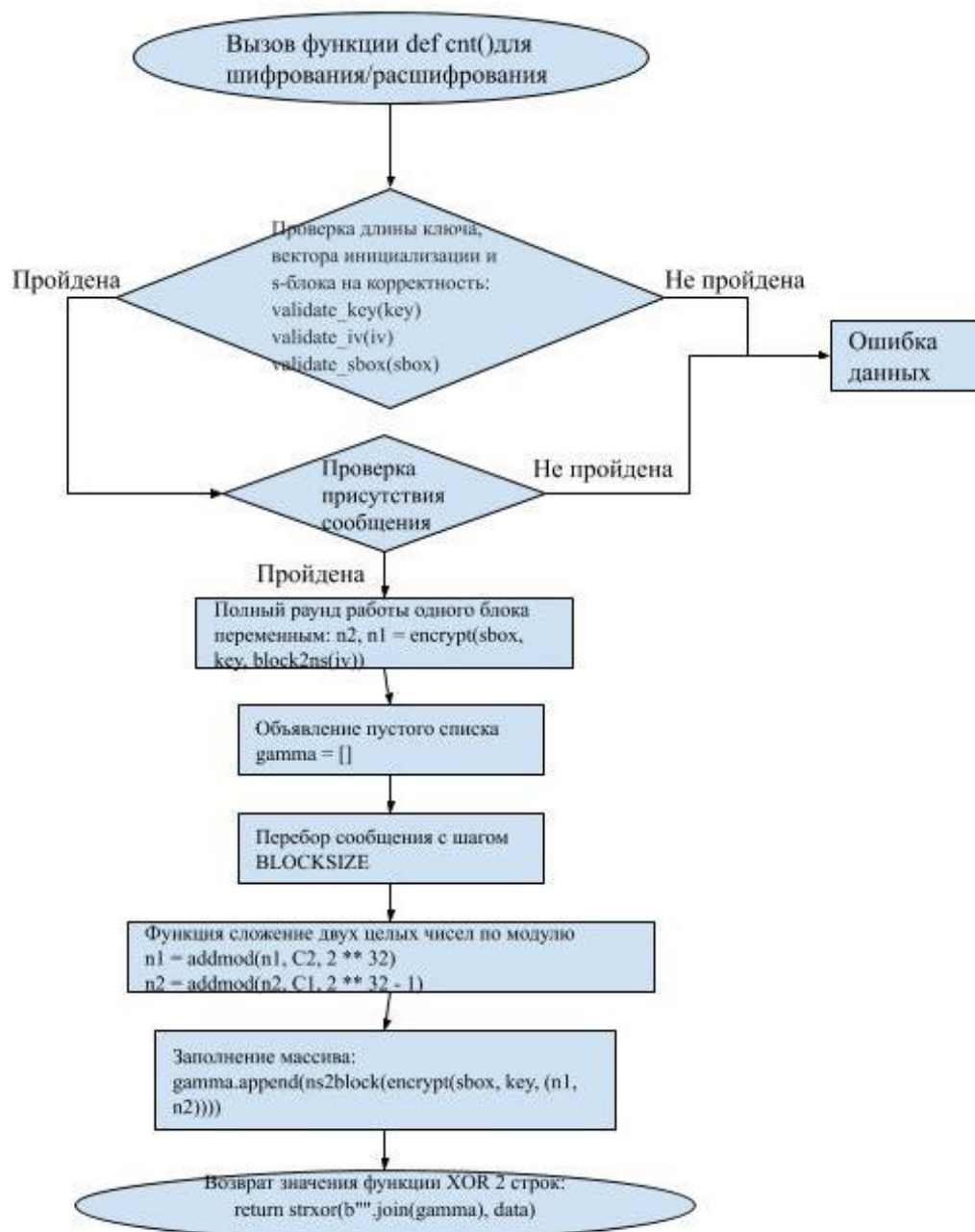
3. Блок-схема программы  
Общая блок-схема ГОСТ 28147-89:



ГОСТ 28147-89 ECB - режим простой замены:

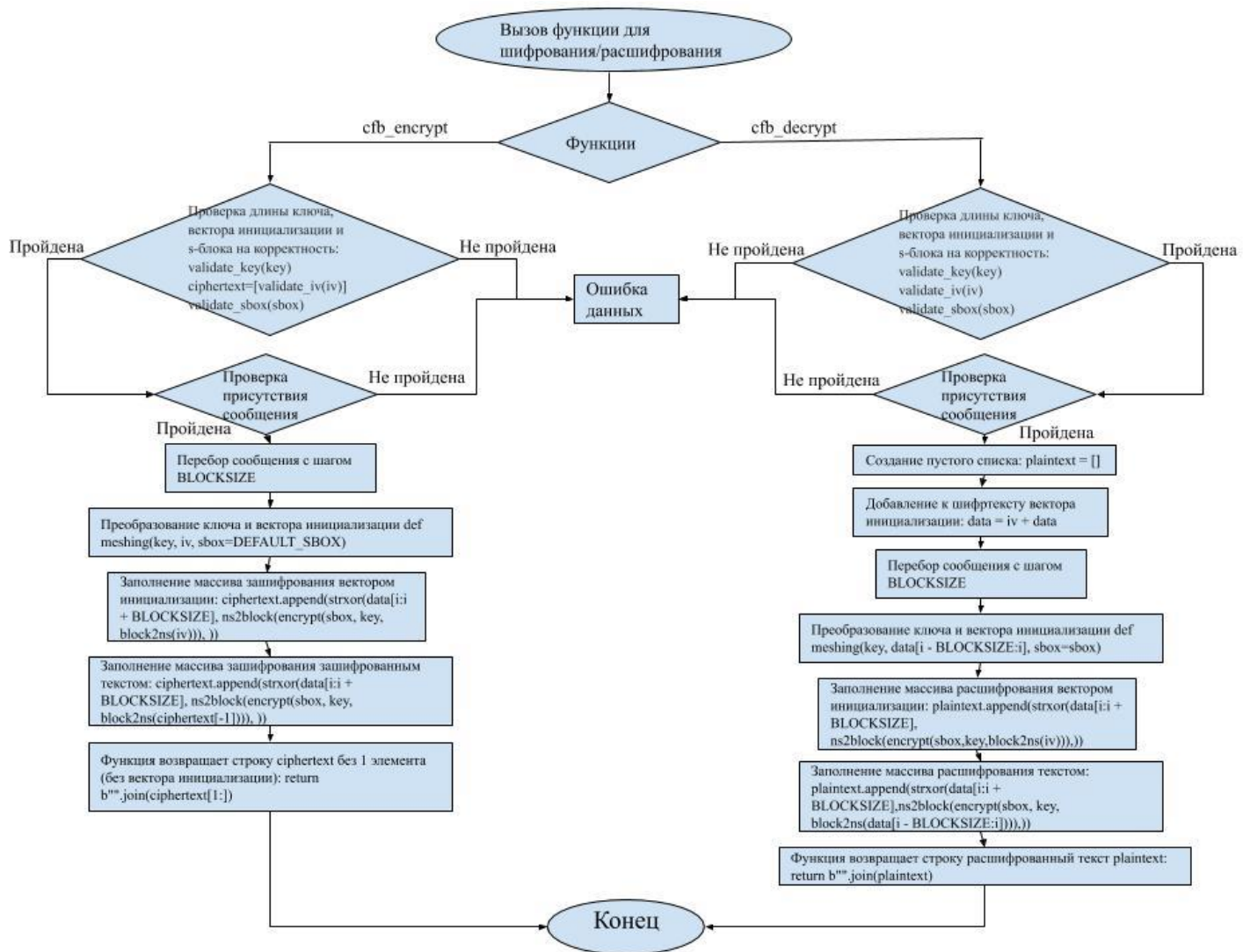


ГОСТ 28147-89 CNT - режим гаммирования:

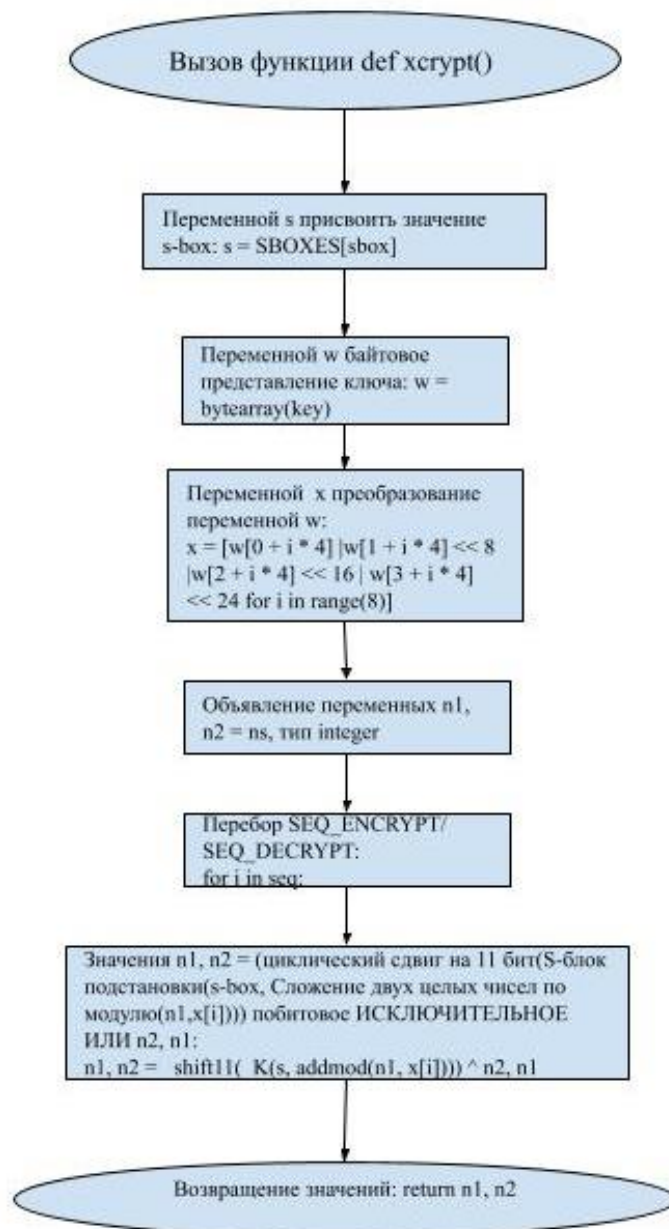


ГОСТ 28147-89 CFB - режим обратной связи по шифротексту:





ГОСТ 28147-89 функция `def xcrypt()` - полный раунд работы блока - эта функция используется в функциях `def encrypt()` и `def decrypt()`, которые используются в функциях `def cnt()` - режим гаммирования и `def cfb_encrypt()/def cfb_decrypt()` - режим обратной связи по шифротексту.



#### 4. Код программы

```

from functools import partial
from codecs import getdecoder
from codecs import getencoder

```

```
KEYSIZE = 32
```

```
BLOCKSIZE = 8
```

```
C1 = 0x01010104
```

```
C2 = 0x01010101
```

# Последовательность применения  $K_i$  S-box для шифрования и дешифрования

SEQ\_ENCRYPT = (

0, 1, 2, 3, 4, 5, 6, 7,

0, 1, 2, 3, 4, 5, 6, 7,

0, 1, 2, 3, 4, 5, 6, 7,

7, 6, 5, 4, 3, 2, 1, 0,

)

SEQ\_DECRYPT = (

0, 1, 2, 3, 4, 5, 6, 7,

7, 6, 5, 4, 3, 2, 1, 0,

7, 6, 5, 4, 3, 2, 1, 0,

7, 6, 5, 4, 3, 2, 1, 0,

)

# Параметры S-box

DEFAULT\_SBOX = "Gost28147\_CryptoProParamSetA"

SBOXES = {

"Gost2814789\_TestParamSet": (

(4, 2, 15, 5, 9, 1, 0, 8, 14, 3, 11, 12, 13, 7, 10, 6),

(12, 9, 15, 14, 8, 1, 3, 10, 2, 7, 4, 13, 6, 0, 11, 5),

(13, 8, 14, 12, 7, 3, 9, 10, 1, 5, 2, 4, 6, 15, 0, 11),

(14, 9, 11, 2, 5, 15, 7, 1, 0, 13, 12, 6, 10, 4, 3, 8),

(3, 14, 5, 9, 6, 8, 0, 13, 10, 11, 7, 12, 2, 1, 15, 4),

(8, 15, 6, 11, 1, 9, 12, 5, 13, 3, 7, 10, 0, 14, 2, 4),

(9, 11, 12, 0, 3, 6, 7, 5, 4, 8, 14, 15, 1, 10, 2, 13),

(12, 6, 5, 2, 11, 0, 9, 13, 3, 14, 7, 10, 15, 4, 1, 8),

),

"Gost28147\_CryptoProParamSetA": (  
    (9, 6, 3, 2, 8, 11, 1, 7, 10, 4, 14, 15, 12, 0, 13, 5),  
    (3, 7, 14, 9, 8, 10, 15, 0, 5, 2, 6, 12, 11, 4, 13, 1),  
    (14, 4, 6, 2, 11, 3, 13, 8, 12, 15, 5, 10, 0, 7, 1, 9),  
    (14, 7, 10, 12, 13, 1, 3, 9, 0, 2, 11, 4, 15, 8, 5, 6),  
    (11, 5, 1, 9, 8, 13, 15, 0, 14, 4, 2, 3, 12, 7, 10, 6),  
    (3, 10, 13, 12, 1, 2, 0, 11, 7, 5, 9, 4, 8, 15, 14, 6),  
    (1, 13, 2, 9, 7, 10, 6, 0, 8, 12, 4, 5, 15, 3, 11, 14),  
    (11, 10, 15, 5, 0, 12, 14, 8, 6, 2, 3, 9, 1, 7, 13, 4),  
),

"Gost28147\_CryptoProParamSetB": (  
    (8, 4, 11, 1, 3, 5, 0, 9, 2, 14, 10, 12, 13, 6, 7, 15),  
    (0, 1, 2, 10, 4, 13, 5, 12, 9, 7, 3, 15, 11, 8, 6, 14),  
    (14, 12, 0, 10, 9, 2, 13, 11, 7, 5, 8, 15, 3, 6, 1, 4),  
    (7, 5, 0, 13, 11, 6, 1, 2, 3, 10, 12, 15, 4, 14, 9, 8),  
    (2, 7, 12, 15, 9, 5, 10, 11, 1, 4, 0, 13, 6, 8, 14, 3),  
    (8, 3, 2, 6, 4, 13, 14, 11, 12, 1, 7, 15, 10, 0, 9, 5),  
    (5, 2, 10, 11, 9, 1, 12, 3, 7, 4, 13, 0, 6, 15, 8, 14),  
    (0, 4, 11, 14, 8, 3, 7, 1, 10, 2, 9, 6, 15, 13, 5, 12),  
),

"Gost28147\_CryptoProParamSetC": (  
    (1, 11, 12, 2, 9, 13, 0, 15, 4, 5, 8, 14, 10, 7, 6, 3),  
    (0, 1, 7, 13, 11, 4, 5, 2, 8, 14, 15, 12, 9, 10, 6, 3),  
    (8, 2, 5, 0, 4, 9, 15, 10, 3, 7, 12, 13, 6, 14, 1, 11),  
    (3, 6, 0, 1, 5, 13, 10, 8, 11, 2, 9, 7, 14, 15, 12, 4),  
    (8, 13, 11, 0, 4, 5, 1, 2, 9, 3, 12, 14, 6, 15, 10, 7),  
    (12, 9, 11, 1, 8, 14, 2, 4, 7, 3, 6, 5, 10, 0, 15, 13),  
    (10, 9, 6, 8, 13, 14, 2, 0, 15, 3, 5, 11, 4, 1, 12, 7),  
    (7, 4, 0, 5, 10, 2, 15, 14, 12, 6, 1, 11, 13, 9, 3, 8),

),

"Gost28147\_CryptoProParamSetD": (

(15, 12, 2, 10, 6, 4, 5, 0, 7, 9, 14, 13, 1, 11, 8, 3),  
(11, 6, 3, 4, 12, 15, 14, 2, 7, 13, 8, 0, 5, 10, 9, 1),  
(1, 12, 11, 0, 15, 14, 6, 5, 10, 13, 4, 8, 9, 3, 7, 2),  
(1, 5, 14, 12, 10, 7, 0, 13, 6, 2, 11, 4, 9, 3, 15, 8),  
(0, 12, 8, 9, 13, 2, 10, 11, 7, 3, 6, 5, 4, 14, 15, 1),  
(8, 0, 15, 3, 2, 5, 14, 11, 1, 10, 4, 7, 12, 9, 13, 6),  
(3, 0, 6, 15, 1, 14, 9, 2, 13, 8, 12, 4, 11, 10, 5, 7),  
(1, 10, 6, 8, 15, 11, 0, 4, 12, 3, 5, 9, 7, 13, 2, 14),

),

"GostR3411\_94\_TestParamSet": (

(4, 10, 9, 2, 13, 8, 0, 14, 6, 11, 1, 12, 7, 15, 5, 3),  
(14, 11, 4, 12, 6, 13, 15, 10, 2, 3, 8, 1, 0, 7, 5, 9),  
(5, 8, 1, 13, 10, 3, 4, 2, 14, 15, 12, 7, 6, 0, 9, 11),  
(7, 13, 10, 1, 0, 8, 9, 15, 14, 4, 6, 12, 11, 2, 5, 3),  
(6, 12, 7, 1, 5, 15, 13, 8, 4, 10, 9, 14, 0, 3, 11, 2),  
(4, 11, 10, 0, 7, 2, 1, 13, 3, 6, 8, 5, 9, 12, 15, 14),  
(13, 11, 4, 1, 3, 15, 5, 9, 0, 10, 14, 7, 6, 8, 2, 12),  
(1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12),

),

"GostR3411\_94\_CryptoProParamSet": (

(10, 4, 5, 6, 8, 1, 3, 7, 13, 12, 14, 0, 9, 2, 11, 15),  
(5, 15, 4, 0, 2, 13, 11, 9, 1, 7, 6, 3, 12, 14, 10, 8),  
(7, 15, 12, 14, 9, 4, 1, 0, 3, 11, 5, 2, 6, 10, 8, 13),  
(4, 10, 7, 12, 0, 15, 2, 8, 14, 1, 6, 5, 13, 11, 9, 3),  
(7, 6, 4, 11, 9, 12, 2, 10, 1, 8, 0, 14, 15, 13, 3, 5),  
(7, 6, 2, 4, 13, 9, 15, 0, 10, 1, 5, 11, 8, 14, 12, 3),  
(13, 14, 4, 1, 7, 0, 5, 10, 3, 12, 8, 15, 6, 2, 9, 11),

(1, 3, 10, 9, 5, 11, 4, 15, 8, 6, 7, 14, 13, 0, 2, 12),  
)

"AppliedCryptography": (

(4, 10, 9, 2, 13, 8, 0, 14, 6, 11, 1, 12, 7, 15, 5, 3),  
(14, 11, 4, 12, 6, 13, 15, 10, 2, 3, 8, 1, 0, 7, 5, 9),  
(5, 8, 1, 13, 10, 3, 4, 2, 14, 15, 12, 7, 6, 0, 9, 11),  
(7, 13, 10, 1, 0, 8, 9, 15, 14, 4, 6, 12, 11, 2, 5, 3),  
(6, 12, 7, 1, 5, 15, 13, 8, 4, 10, 9, 14, 0, 3, 11, 2),  
(4, 11, 10, 0, 7, 2, 1, 13, 3, 6, 8, 5, 9, 12, 15, 14),  
(13, 11, 4, 1, 3, 15, 5, 9, 0, 10, 14, 7, 6, 8, 2, 12),  
(1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12),  
)

"Gost28147\_tc26\_ParamZ": (

(12, 4, 6, 2, 10, 5, 11, 9, 14, 8, 13, 7, 0, 3, 15, 1),  
(6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15),  
(11, 3, 5, 8, 2, 15, 10, 13, 14, 1, 7, 4, 12, 9, 6, 0),  
(12, 8, 2, 1, 13, 4, 15, 6, 7, 0, 10, 5, 3, 14, 9, 11),  
(7, 15, 5, 10, 8, 1, 6, 13, 0, 9, 3, 14, 11, 4, 2, 12),  
(5, 13, 15, 6, 9, 2, 12, 10, 11, 7, 8, 1, 4, 3, 14, 0),  
(8, 14, 2, 5, 6, 9, 1, 12, 15, 4, 11, 0, 13, 10, 3, 7),  
(1, 7, 14, 13, 0, 5, 8, 3, 4, 15, 10, 6, 9, 12, 11, 2),  
)

"EACParamSet": (

(11, 4, 8, 10, 9, 7, 0, 3, 1, 6, 2, 15, 14, 5, 12, 13),  
(1, 7, 14, 9, 11, 3, 15, 12, 0, 5, 4, 6, 13, 10, 8, 2),  
(7, 3, 1, 9, 2, 4, 13, 15, 8, 10, 12, 6, 5, 0, 11, 14),  
(10, 5, 15, 7, 14, 11, 3, 9, 2, 8, 1, 12, 0, 4, 6, 13),  
(0, 14, 6, 11, 9, 3, 8, 4, 12, 15, 10, 5, 13, 7, 1, 2),  
(9, 2, 11, 12, 0, 4, 5, 6, 3, 15, 13, 8, 1, 7, 14, 10),

```

(4, 0, 14, 1, 5, 11, 8, 3, 12, 2, 9, 7, 6, 10, 13, 15),
(7, 14, 12, 13, 9, 4, 8, 15, 10, 2, 6, 0, 3, 11, 5, 1),
),
}

```

```
def pad_size(data_size, blocksize):
```

```
    # Рассчитать необходимый размер пэда для полного размера блока
```

```
    if data_size < blocksize: # длина сообщ < 8
```

```
        return blocksize - data_size # вернуть 8 - длина сообщ
```

```
    if data_size % blocksize == 0: # длина сообщ mod 8 = 0
```

```
        return 0 # вернуть 0
```

```
    return blocksize - data_size % blocksize # вернуть 8 - длина сообщ mod 8
```

```
def pad2(data, blocksize):
```

```
    # Метод заполнения 2 (также известен как ISO/IEC 7816-4)
```

```
    # Добавить один бит, а затем заполняет нулями
```

```
    return data + b"\x80" + b"\x00" * pad_size(len(data) + 1, blocksize)
```

```
def strxor(a, b):
```

```
    # Операции XOR двух строк
```

```
    # Эта функция будет обрабатывать только самую короткую длину обеих
    # строк, игнорируя оставшиеся
```

```
    mlen = min(len(a), len(b)) # поиск самой короткой строки
```

```
    a, b, xor = bytearray(a), bytearray(b), bytearray(mlen) # массивы байтов
```

```
    for i in range(mlen): # перебираем массив байтов mlen (короткой строки)
```

```
        xor[i] = a[i] ^ b[i] # побитовое исключительное или каждого элемента
```

```
    return bytes(xor) # тип байты от xor
```

```
_hexdecoder = getdecoder("hex")
```

```
def hexdec(data):
```

```
    #Декодировать шестнадцатеричное число
```

```
    return _hexdecoder(data)[0]
```

```
def _K(s, _in):
```

```
    # S-блок подстановки
```

```
    """ S-блок подстановки
```

```
    :param s: S-box
```

```
    :param _in: 32-bit word
```

```
    :returns: substituted 32-bit word
```

```
    """
```

```
    return (
```

```
        (s[0][(_in >> 0) & 0x0F] << 0) +
```

```
        (s[1][(_in >> 4) & 0x0F] << 4) +
```

```
        (s[2][(_in >> 8) & 0x0F] << 8) +
```

```
        (s[3][(_in >> 12) & 0x0F] << 12) +
```

```
        (s[4][(_in >> 16) & 0x0F] << 16) +
```

```
        (s[5][(_in >> 20) & 0x0F] << 20) +
```

```
        (s[6][(_in >> 24) & 0x0F] << 24) +
```

```
        (s[7][(_in >> 28) & 0x0F] << 28)
```

```
    )
```

```
def block2ns(data):
```

```
    # Преобразование блока в целые числа N1 и N2
```

```
    data = bytearray(data)
```

```
    return (
```



```
data[0] | data[1] << 8 | data[2] << 16 | data[3] << 24,  
data[4] | data[5] << 8 | data[6] << 16 | data[7] << 24,  
)
```

```
def ns2block(ns):
```

```
# Преобразование целых чисел N1 и N2 в 8-байтовый  
n1, n2 = ns  
return bytes(bytearray((  
    (n2 >> 0) & 255, (n2 >> 8) & 255, (n2 >> 16) & 255, (n2 >> 24) & 255,  
    (n1 >> 0) & 255, (n1 >> 8) & 255, (n1 >> 16) & 255, (n1 >> 24) & 255,  
)))
```

```
def addmod(x, y, mod=2 ** 32):
```

```
# Сложение двух целых чисел по модулю  
r = x + y  
return r if r < mod else r - mod
```

```
def _shift11(x):
```

```
# 11-битный циклический сдвиг  
return ((x << 11) & (2 ** 32 - 1)) | ((x >> (32 - 11)) & (2 ** 32 - 1))  
# | - побитовое ИЛИ  
# << >> - сдвиги  
# & - побитовое И
```

```
def validate_key(key):
```

```
if len(key) != KEYSIZE: # если длина ключа не соответствует 32  
    raise ValueError("Неподходящий размер ключа.")
```

```
def validate_iv(iv):
    if len(iv) != BLOCKSIZE: # если длина вектора инициализации не
        соответствует 8
        raise ValueError("Неподходящий размер ключа инициализации.")
```

```
def validate_sbox(sbox):
    if sbox not in SBOXES: # если задан другой sbox
        raise ValueError("Неизвестный sbox.")
```

```
def xcrypt(seq, sbox, key, ns):
    # Полный раунд работы одного блока
    """
    :param seq: sequence of K_i S-box applying (either encrypt or decrypt)
    :param sbox: S-box parameters to use
    :type sbox: str, SBOXES'es key
    :param bytes key: 256-bit encryption key
    :param ns: N1 and N2 integers
    :type ns: (int, int)
    :returns: resulting N1 and N2
    :rtype: (int, int)
    """
    s = SBOXES[sbox] # s = заданный sbox
    w = bytearray(key) # байтовое представление ключа
    x = [
        w[0 + i * 4] |
        w[1 + i * 4] << 8 |
        w[2 + i * 4] << 16 |
        w[3 + i * 4] << 24 for i in range(8)
    ] # ключ 256-bit
```

```

n1, n2 = ns
for i in seq: # перебор SEQ_EN/DECRYPT
    n1, n2 = _shift11(_K(s, addmod(n1, x[i]))) ^ n2, n1 # (Ф-сдвиг на 11)(S-
    блок подстановки(s-box, сложение по mod(n1 и x[i])^ n2, n1
    # ^ - побитовое ИСКЛЮЧИТЕЛЬНОЕ ИЛИ
return n1, n2

```

```

def encrypt(sbox, key, ns):
    # Шифрование одного блока

    return xcrypt(SEQ_ENCRYPT, sbox, key, ns)

```

```

def decrypt(sbox, key, ns):
    # Расшифрование одного блока

    return xcrypt(SEQ_DECRYPT, sbox, key, ns)

```

```

def ecb(key, data, action, sbox=DEFAULT_SBOX):
    # Режим простой замены
    """ ECB - Режим простой замены

    :param bytes key: encryption key
    :param data: plaintext
    :type data: bytes, multiple of BLOCKSIZE
    :param func action: "encrypt"/"decrypt"
    :param sbox: S-box parameters to use
    :type sbox: str, SBOXES'es key
    :returns: ciphertext
    :rtype: bytes

```

```

"""

validate_key(key)
validate_sbox(sbox)

if not data or len(data) % BLOCKSIZE != 0:
    for i in range(8 - len(data) % 8):
        data += b'0' # прибавить 0 если сообщение не делится на 8

result = []
for i in range(0, len(data), BLOCKSIZE):
    result.append(ns2block(action( # action - зашифрование или
расшифрование
        sbox, key, block2ns(data[i:i + BLOCKSIZE])
    )))
return b"".join(result)

# functools.partial(func, *args, **keywords) - возвращает partial-объект (по
сути, функцию), который при вызове вызывает func
# как функция func, но дополнительно передают туда позиционные
аргументы args, и именованные аргументы kwargs.
# Если другие аргументы передаются при вызове функции, то позиционные
добавляются в конец, а именованные расширяют и перезаписывают.

ecb_encrypt = partial(ecb, action=encrypt)
ecb_decrypt = partial(ecb, action=decrypt) # ecb меняется постоянно

def cnt(key, data, iv=8 * b"\x00", sbox=DEFAULT_SBOX):
    # CNT - Режим гаммирования - Режим работы счетчика
    # Для расшифровки используется эта же функция
    """

:param bytes key: encryption key
:param bytes data: plaintext
:param iv: initialization vector

```

:type iv: bytes, BLOCKSIZE length  
:param sbox: S-box parameters to use  
:type sbox: str, SBOXES'es key  
:returns: ciphertext  
:rtype: bytes

For decryption you use the same function again.

```
"""  
  
validate_key(key)  
validate_iv(iv)  
validate_sbox(sbox)  
if not data:  
    raise ValueError("Сообщение отсутствует.")  
n2, n1 = encrypt(sbox, key, block2ns(iv))  
gamma = []  
for _ in range(0, len(data) + pad_size(len(data), BLOCKSIZE), BLOCKSIZE):  
    n1 = addmod(n1, C2, 2 ** 32)  
    n2 = addmod(n2, C1, 2 ** 32 - 1)  
    gamma.append(ns2block(encrypt(sbox, key, (n1, n2))))  
return strxor(b"".join(gamma), data)
```

```
MESH_CONST =  
hexdec("6900722264C904238D3ADB9646E92AC418FEAC9400ED0712C086D  
CC2EF4CA92B") # MESH_CONST = декодированное 16-ричное число
```

```
MESH_MAX_DATA = 1024
```

```
# сцепление
```

```
def meshing(key, iv, sbox=DEFAULT_SBOX):
```

```
    """rfc: 4357 key meshing  
    """
```

```

key = ecb_decrypt(key, MESH_CONST, sbox=sbox)
iv = ecb_encrypt(key, iv, sbox=sbox)
return key, iv

def cfb_encrypt(key, data, iv=8 * b"\x00", sbox=DEFAULT_SBOX, mesh=False):
    # CFB - Режим обратной связи по шифротексту шифрование
    """ CFB - Режим обратной связи по шифротексту шифрование

    :param bytes key: encryption key
    :param bytes data: plaintext
    :param iv: initialization vector
    :type iv: bytes, BLOCKSIZE length
    :param sbox: S-box parameters to use
    :type sbox: str, SBOXES'es key
    :param bool mesh: enable key meshing
    :returns: ciphertext
    :rtype: bytes
    """
    validate_key(key)
    validate_iv(iv)
    validate_sbox(sbox)
    if not data:
        raise ValueError("Сообщение отсутствует.")
    ciphertext = [iv]
    # перебор сообщения от 0 до (длина сообщ+пэд(len(data),8), шаг 8) 0,8,16...
    for i in range(0, len(data) + pad_size(len(data), BLOCKSIZE), BLOCKSIZE):
        if mesh and i >= MESH_MAX_DATA and i % MESH_MAX_DATA == 0: #
            шаг меньше 1024 и шаг mod 1024 = 0
            key, iv = meshing(key, ciphertext[-1], sbox=sbox) # ключ,вектор
            инициализации = meshing(ключ,посл элемент шифртекста,s-блок)

```

```

        ciphertext.append(strxor( # заполнить массив (функция
            data[i:i + BLOCKSIZE], # i:i - вывод пустого массива
            ns2block(encrypt(sbox, key, block2ns(iv))),
            # ф-ия целые в 8-байтовые(зашифр(sbox, ключ, 8-байтовые в
            # целые(вектор инициализации)))
        ))
        continue
    ciphertext.append(strxor(
        data[i:i + BLOCKSIZE],
        ns2block(encrypt(sbox, key, block2ns(ciphertext[-1]))), # [-1] - только
        # последний элемент массива
    ))
    return b"".join(ciphertext[1:]) # вернуть строку (шифротекст без 1 элемента)

def cfb_decrypt(key, data, iv=8 * b"\x00", sbox=DEFAULT_SBOX, mesh=False):
    # CFB - Режим обратной связи по шифротексту расшифрование
    """ CFB - Режим обратной связи по шифротексту расшифрование

    :param bytes key: encryption key
    :param bytes data: plaintext
    :param iv: initialization vector
    :type iv: bytes, BLOCKSIZE length
    :param sbox: S-box parameters to use
    :type sbox: str, SBOXES'es key
    :param bool mesh: enable key meshing
    :returns: ciphertext
    :rtype: bytes
    """
    validate_key(key)
    validate_iv(iv)

```

```

validate_sbox(sbox)

if not data:
    raise ValueError("Сообщение отсутствует.")

plaintext = []

data = iv + data

for i in range(BLOCKSIZE, len(data) + pad_size(len(data), BLOCKSIZE),
BLOCKSIZE):
    if (
        mesh and
        (i - BLOCKSIZE) >= MESH_MAX_DATA and
        (i - BLOCKSIZE) % MESH_MAX_DATA == 0
    ):
        key, iv = meshing(key, data[i - BLOCKSIZE:i], sbox=sbox) # i -
BLOCKSIZE:i    i - 8:i
        plaintext.append(strxor(
            data[i:i + BLOCKSIZE],
            ns2block(encrypt(sbox, key, block2ns(iv))),
        ))
        continue
    plaintext.append(strxor(
        data[i:i + BLOCKSIZE],
        ns2block(encrypt(sbox, key, block2ns(data[i - BLOCKSIZE:i]))),
    ))

return b"".join(plaintext)

f = open(r"opentext.txt", "rt", encoding='utf-8')
text = f.read()

text = text.replace('.', 'тчк') # Если в сообщении попадетсa точка, она заменетсa
на тчк

```



```

text = text.replace(',', 'зпт') # Если в сообщении попадется запятая, она
заменется на зпт

text = text.replace('-', 'тире') # Если в сообщении попадется - (тире), оно
заменется на тире

text = text.replace(' ', 'прбл') #

text = text.encode('cp1251') # изменить кодировку файла на cp1251

key =
hexdec("59841235AE85621BE099F857A5621037C9DAF578032A2A56CD6307
4BF501C623") # декодировать ключ

sbox = "Gost2814789_TestParamSet" # задать sbox

# Режим шифрования - ECB - Режим простой замены
# в этот файл записывается зашифрованный текст ecb
f = open('ECB shifr.txt', 'wt', encoding='utf-8')
ecb_sh = ecb(key, text, action=encrypt, sbox=sbox)
f.writelines(str(ecb_sh))
f.close()

# в этот файл записывается расшифрованный текст
f = open('ECB rasshifr.txt', 'wt', encoding='utf-8')
ecb_rassh = ecb(key, ecb_sh, action=decrypt, sbox=sbox)
ecb_rassh = ecb_rassh.decode('cp1251')

ecb_rassh = ecb_rassh.replace('тчк', '.') # Если в сообщении попадется точка,
она заменется на тчк

ecb_rassh = ecb_rassh.replace('зпт', ',') # Если в сообщении попадется запятая,
она заменется на зпт

ecb_rassh = ecb_rassh.replace('тире', '-') # Если в сообщении попадется - (тире),
оно заменется на тире

ecb_rassh = ecb_rassh.replace('прбл', ' ') # Если в сообщении попадется - (тире),
оно заменется на тире

f.writelines(ecb_rassh[:ecb_rassh.find('00')])
f.close()

# Режим шифрования - cnt - Режим простой замены

```

```
# в этот файл записывается зашифрованный текст esb
f = open('CNT shifr.txt', 'wt', encoding='utf-8')
cnt_sh = cnt(key, text,iv=hexdec(b'0102030405060708'),sbox=sbox)
f.writelines(str(cnt_sh))
f.close()

# в этот файл записывается расшифрованный текст
f = open('CNT rasshifr.txt', 'wt', encoding='utf-8')
cnt_rassh = cnt(key,cnt_sh,iv=hexdec(b'0102030405060708'),sbox=sbox)
cnt_rassh = cnt_rassh.decode('cp1251')
cnt_rassh = cnt_rassh.replace('тчк', '.') # Если в сообщении попадетс я точка, она
заменетс я на тчк
cnt_rassh = cnt_rassh.replace('зпт', ',') # Если в сообщении попадетс я запятая,
она заменетс я на зпт
cnt_rassh = cnt_rassh.replace('тире', '-') # Если в сообщении попадетс я - (тире),
оно заменетс я на тире
cnt_rassh = cnt_rassh.replace('прбл', ' ') # Если в сообщении попадетс я - (тире),
оно заменетс я на тире
f.writelines(cnt_rassh)
f.close()

# Режим шифрования - Режим обратной связи по шифротексту
# в этот файл записывается зашифрованный текст cfb
f = open('CFB shifr.txt', 'wt', encoding='utf-8')
cfb_sh = cfb_encrypt(key, text,iv=hexdec(b'0102030405060708'),sbox=sbox)
f.writelines(str(cfb_sh))
f.close()

# в этот файл записывается расшифрованный текст
file = open('CFB rasshifr.txt', 'wt', encoding='utf-8')
cfb_rassh = cfb_decrypt(key,cfb_sh,iv=hexdec(b'0102030405060708'),sbox=sbox)
cfb_rassh = cfb_rassh.decode('cp1251')
cfb_rassh = cfb_rassh.replace('тчк', '.') # Если в сообщении попадетс я точка,
она заменетс я на тчк
```

```

cfb_rassh = cfb_rassh.replace('зпт', ',') # Если в сообщении попадетсся запятая,
она заменется на зпт

cfb_rassh = cfb_rassh.replace('тире', '-') # Если в сообщении попадетсся - (тире),
оно заменется на тире

cfb_rassh = cfb_rassh.replace('прбл', ' ') # Если в сообщении попадетсся - (тире),
оно заменется на тире

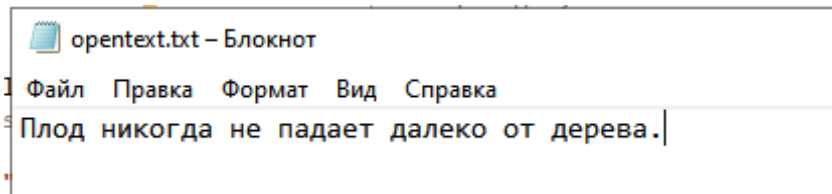
file.writelines(cfb_rassh)

file.close()

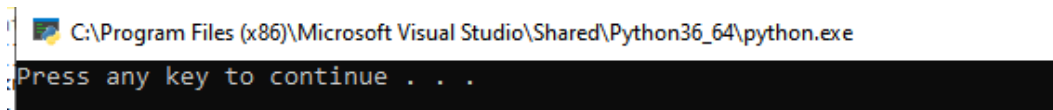
```

## 5. Тестирование

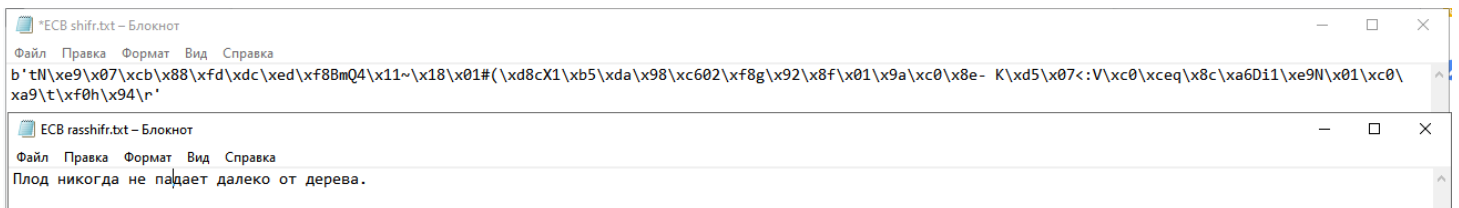
Перед началом работы программы в файл “opentext.txt” записываем исходный текст.



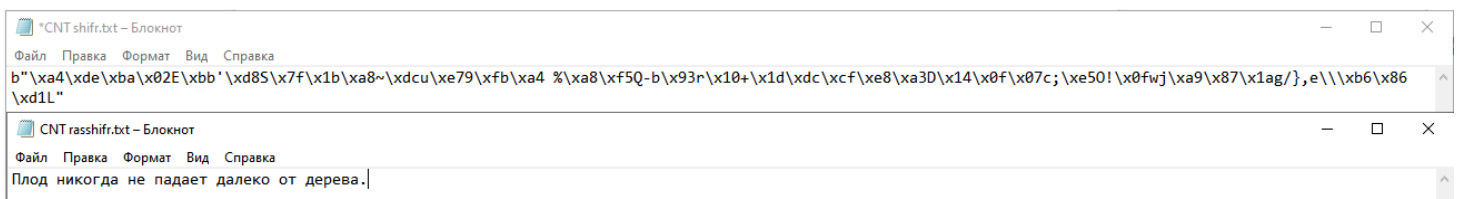
Результат выполнения программы:



После выполнения программы в файл “ECB shifr.txt” записывается зашифрованный текст - режим простой замены. В файл “ECB rasshifr.txt” - записывается расшифрованный текст - режим простой замены.

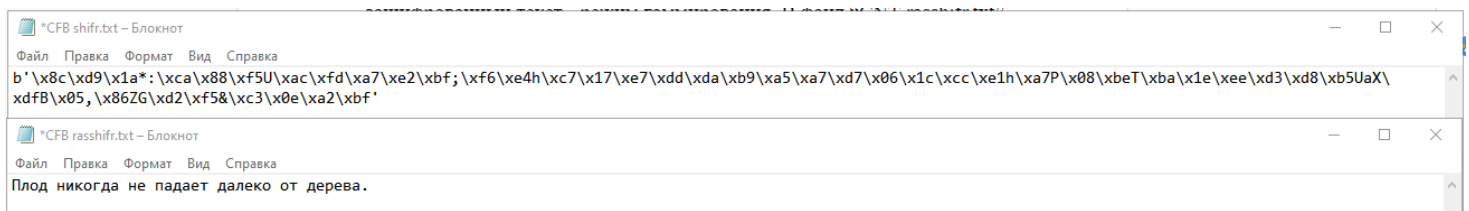


После выполнения программы в файл “CNT shifr.txt” записывается зашифрованный текст - режим гаммирования. В файл “CNT rasshifr.txt” - записывается расшифрованный текст - режим гаммирования.



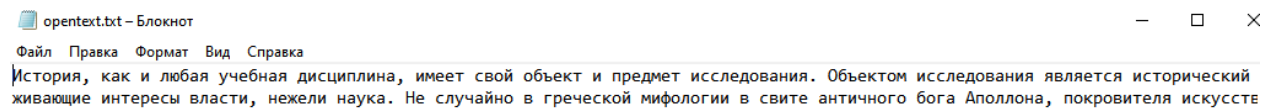
После выполнения программы в файл “CFB shifr.txt” записывается зашифрованный текст - режим обратной связи по шифротексту. В файл “CFB

rasshifr.txt” - записывается расшифрованный текст - режим обратной связи по шифротексту.

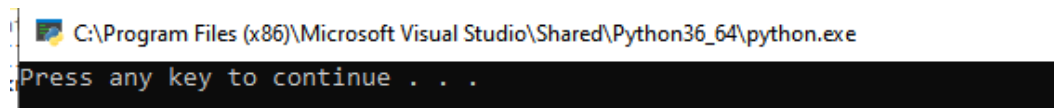


## 6. Работа с текстом не менее 1000 знаков

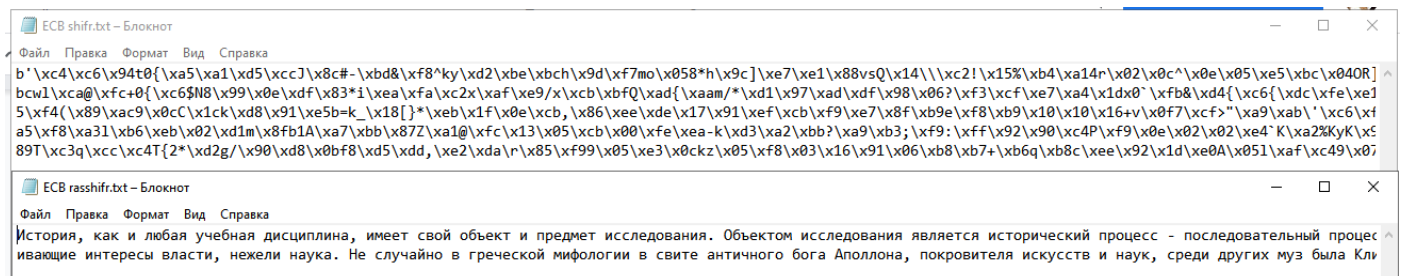
Перед началом работы программы в файл “opentext.txt” записываем исходный текст. (Полный исходный текст лежит в аннотации)



Так выглядит окно выполнения программы.



После выполнения программы в файл “ECB shifr.txt” записывается зашифрованный текст - режим простой замены. В файл “ECB rasshifr.txt” - записывается расшифрованный текст - режим простой замены.



Полный зашифрованный текст:

```
b"\xc4\xc6\x94t0{\xa5\xa1\xd5\xccJ\x8c#-
\xbd&\xf8^ky\xd2\xbe\xbc\x9d\xf7mo\x058*h\x9cJ\xe7\xe1\x88vsQ\x14\\\xc2!\x
15%\xb4\xa14r\x02\x0c^\x0e\x05\xe5\xbc\x04ORJ\xb8\x9e\xa8\x08\xd7\x18\xdc\
xef\x8a\xaa\xf7fy\xaa\x1f/f\x80L%5\xe5o\xf4Z\xd0\x12f\xfd\x8bV\xe8x\xdc?\x89
\x91\xa0\x86\x04\x97\xd0\xf3:\xae\x1d\xb3\xb0;\x9b\xfc!\xbb\xe0i\x1df;$\xeeh\x
8d\xe2\x07\x18\xfcj\xbe\x8bb\xed\t\x08\xfd(Sk\xc85\x96\xb7Ds\x12Q\xe3h\xec\x
91\t\xb1\x0f\x98YI\xea\xab\x8f\re\x9eYqM\xf7\x06s[6^/>G\xe8\xa8\xbf\n!\xff\xfd
>\xd2k\x10\x95\xa0\x1a:n\x18\xfd\xca\x1d\xb4\x04\r\xecZ\xd4\xf8\xb9,J\x9e\xd0
G\x81Gf\x17#\xe1V\x8cO\xa0\xc2:\xceU\xbe\x82[\xce\xbf\xa1\xea\xdf\xcb?\x9c\
x8cpX^\xf7\x12&=\xc2?\xd6X\|aa\x8f\x8e5+fF:\x1b\xd3+\x89\x96l\x0f\xdfz.U;I\
x0c\x8d\xe3U\x8bo\x08\xf1J\x1c\x17b\xc2<\xfc.\xe3\xba|T\x8dwV\x8da\x7f\x80\
x98\x987\x89[\tp1sw\xbe\x10
|z\x15\xdd\xb2\xee\x6\x81\xf7\xfd\xd1\xf7\x13,\xbfe\x99\xa0\xc0\x8eJY6\|aaV
```

\xd3%3(\x16\xdcj\xa2\x07\x95d\x98\x95\xea\xd2\xe0[\xb9\xab\x92\xf2\x82\xd2i\xf2\$\xb7f)8\xc9\xb5\x0c\x1d\xfa>\xef)[\xf1z\xbcwl\xca@\xfc+0{\xc6\$N8\x99\x0e\xdf\x83\*i\xea\xfa\xc2x\xaf\xe9/x\xcb\xbfQ\xad{\xaam/\*\xd1\x97\xad\xdf\x98\x06?\xf3\xcf\xe7\xa4\x1dx0`\xfb&\xd4{\xc6{\xdc\xfe\xe1B\xfe\x82\xcbM\$\xc2\x11MfA\x3n^P\xe0\xd9\xbeAL\xa7m\x86\xb6\x8f\xa2\xd3H\x87\xfeC%9B\xa1\*W MV\x15\xd9?}\xdb\[\x1b\x8b\xdb\xfe\xad\x9e{\x029\xb0\*}'~G\xfcck\x81\xc0i\xb92p7W\x904\x8bF\xbb>(\_T%\xecq\xa2!\x93\xe4~\x8a\xae\xde<\xe2\xa7{\xc0\xe a6.T\xbam\xbd`\x073\x16\xf8N\x0c\x0f\x02\x06\xb0B\x03\xd1\x1b0m^\xf5\xe1q M}\x/ec\x08\x01a\x8c\x1b\xa0\xab\xb6\x97E\x99\xa0\xc0\x8eJY6Op5Y\xda6+\x9c\x95\xe7Q\x1d\xb2\*y\xd1\x94\xc0\xb3F\x199\xd5\x8a@\x96eW\x03%\nN\xa5\x03M\x14\x1cZ:\xd6\xdc\x0c]\x1c\xf2>d\r\xab\x03rF\x10&,"x19\xa9\x16\r-R\xa8\xd6\x90\x80M\xb1\xff;L\x07F\xa3\xe2\xa5d\xc4\xd7\x15\xcb\xc8ZLy\x9c\ xf9C\x95?\x94\x8d\x90V2gE\xec\x14\x02\xfbu\x96W\xdd\xe6\xef\x88\xb5\x96\x92\x992\xc3\xb1&Z\x12m\x13N\x0c\x9f\xee\x8f\x8f\_\xb5\x1c\xa7\x95\x85\x98\x f9\xa6\xaa\xe3+\x08\xcd\x0b\x17\xe5\xf45\xde\xc0\x9f\xd5CS\x07\xce\xe9A\x04 1\xa8\x9f]^ \xd1\xc1\x87I\x97\_\xd1\x9dQSc\xc5\xf4(\x89\xac9\x0cC\x1ck\xd8\x9 1\xe5b=k\_\x18[]\*\xeb\x1f\x0e\xcb,\x86\xee\xde\x17\x91\xef\xcb\x9f\xe7\x8f\xb9 e\x8f\xb9\x10\x10\x16+v\x0f7\xcf>"\xa9\xab\x'xc6\xfb\xd3\x85\x05\xb2"\xd6v\x9 3Zr9QI\xb6\x12\r\x98dn\x10\x8a\x13F\xd1\x92\xf4\x8fH\xa3F)\x8b\xdc\x86%\xf 4\xd9z\x9el\xe1\xb9+U\x96\x85b'\xa2^u4\xbeB\x04!Y2\x94#\xb9e\x8f\xb9\x10\ x10\x16+\'z\xd8\xa8-aU)\xf1h\x16\x1e\xe0\xb2p\xf6\xd4\xaa\xdbz\xc6\xe7\xffi\x15\xcf\x05\x82\xc2\x ba\xc8\xa8`\x88\xcf\x00\xeaA\x83^\x03\x00\x10HY\xb8\xcb\xda9\x19\xddo\xd0\ xb8u\x1a\x16\xb3\xed\*CC7\x03\xfe\xbe\xdf\x92\x8bZ\xca\xf6O\xecf4\x801\x91 Y\xa3\x8c\xd0\xa9J\x9e\xf0K\xc5\xf6\x1e<Yo\x9aR\xf9\x05\xec\xbe\x94\x9a\'8q v\xf9\'?\xfb\xef\xe4tik\xa4h\xe8\xafr\xc2YAH\xa8\x01t\xc6\xca\x88\x0f\x03\x91 z\xde\nk\xe9s\x85\x8d\x10\x8b@\x13\x08r\x8d"\xea\xaf\x0b2\x8c\xbc\xdb\xf35U B#\xcf\xf7\xa2\xce`\xcf\x03\x83sq\x8d\xf6\x84p\xb8O,7Q\x1f\xad\x99\x85#\xb5p 9\xd0\xca\x1d\xd8n o\xd6\x13\xbc(\$\xb0\xe69\xb2<\x89U\xca\xb8\xf84K\xcbH\xd2\xca\x1d\xb4\x04\ r\xecZ\xd4\x1a\x13@f\x88\xf9\xa0\x9f\xddm\*\x0e\xa5\xf8\xa3l\xb6\xeb\x02\xd1 m\x8fb1A\xa7\xbb\x87Z\xa1@\xfc\x13\x05\xcb\x00\xfe\xea- k\xd3\xa2\xbb?\xa9\xb3;\xf9:\xff\x92\x90\xc4P\xf9\x0e\x02\x02\xe4`K\xa2%KyK \x9c@;{6\x04\xd7#8\x83)\xd5r\xed5\x86\xfc\xb4\xca\xbb\xc5\x96\x0cD\x92\x85\ xc1r\x9d\xb4\xa8\xc3`\xa1\x05h5L5#\xee\xderTPj\x158\xed\xfe\_\xfe\xe3R\xceH\ xf1^;\xe0\xf9\xb6\xdbz%\xc7{\x0c\xe8\xa7\x95s\xf5?\xc8\xec\xe0~O\xb2\xc4\xd 0\xa0i<`\xa6)\xcc\xde\xe9\xcb\xa5\x02\xbf\xb0<\xc7\xad\xd9m\xb5DQ,\xc1\$\x9d \xf8\x9d\xd3\xb5\x93^\x96Ux\xa2\x8f\xf3\x7fi\xb2\xe5\x01\xfa\x9f\xdc?L\xb8|8{ C7\x9f\x08\xf5\xe8\x8f\xc3z\x9el\xe1\xb9+U\x96\xe7o\xc4^\xdbSG\xb2\xd3\x98 A\xc8f\xdc9\xca\xc6^1\x84\x8fn\t\xd5X\xe7L\x96\xc4Em\x91}\xd4 I;\xf1\xbdJ\x01\x1b5\xe3\xf3F\x8dR"'OG\xd6h6)=\xc7\trm\x83\xb3\x00/;`\xe0)\xd 7m\xa2XC\xc5/\x98\_FC\xac\x10\xf1\x81zR\x13\xafyw\x8c\xbd\xad!v\xc90\x89\x ab\x85V\xcfB\xa29\xd5\xccJ\x8c#- \xbd&>\xac4\xaa\xb7c\xa6eL\xe8U\xabw\xe0\x90\x99\xb9e\x8f\xb9\x10\x10\x16 +z\x0b\rF\x18.\xf8\x84\x99;\xf0h\xd3\x08XS\xd2\x07\xbfD\xf9\xed\xec\xdd\xd3\

xd4\x90\x94SW)&\xa4Z>\x89T\xc3q\xcc\xc4T{2\*\xd2g/\x90\xd8\x0bf8\xd5\xdd,  
\xe2\xda\r\x85\xf99\x05\xe3\x0ckz\x05\xf8\x03\x16\x91\x06\xb8\xb7+\xb6q\xb8c  
\xee\x92\x1d\xe0A\x05l\xaf\xc49\x07q\xff\xcf\x82\x12\xd2\xf44\x04o\xec+\xd0\x  
10\xd7\x80R\xbf\"xdc\x8e\xf2\x17\x1c\xf2\xde]\x07\xeb\x11\xb0\xfeG\xc4\x96\t\  
xf9\x953\xf7)\x7f%\x1a\xb7\xafi\xa6\x19@e=\xa3kq\xe2~\xe8\xf9\re7\x89\xab\xee  
e\xcfq\xa6\x0e\x84j\x10\x08\x9e\xbe\xd2\xe1\x86\x99\xff\x95\xbar\xc5\x02\xf4ks  
\x1b\xfd\xa8\*/z\x9c\xf2\xc6\xb0%\x17\xc1\x9f\xcf\xad\x8f5h\xea\x14Jj;\xee\xbb2\  
xf6\x1b\xd7o\x8b\x93\xf1\x9e\xac\xdc\x93\xd4\x1e\xe7z\xe4\xed\xd6)K\$\xb0o\x  
7fta\xf3P\xbc(\x9d\xaa\x9c)\x96\xb3\x95Ij~\x88>\xb4qyX{b\xc9\xdeQ6\x81\xe6\  
xb6\xdb\xed\xc1\xca\x1d\xb4\x04r\xecZ\xd4\x18@V\xf5e\xc1\xf1\x935\xalxY\x  
9c\x00R\x8de\x1e\xfa9\xe1\xfd>K'

Полный расшифрованный текст:

История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический процесс - последовательный процесс развития природы и общества, череда сменяющих друг друга событий, в которых проявляется деятельность многих поколений людей. С каждым мгновением времени, ушедшим в прошлое исторический процесс дополняется совокупностью новых событий, явлений, фактов и факторов в жизни человека, семьи, этноса, государства, человечества. Подобно тому, как материальный мир переживает процессы негэнтропии и энтропии, природа вокруг нас - процессы эволюции и инволюции, человеческое общество характеризуют процессы социального прогресса и регресса. Предметом изучения истории является деятельность главного его субъекта - человека - в прошлом. В узком смысле история представляет собой совокупность социальных фактов, знаний, представлений ученых о том, как эти процессы осуществлялись и осуществляются. Долгое время история существовала скорее, как литература и искусство, обслуживающие интересы власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств и наук, среди других муз была Клио - муза истории.

После выполнения программы в файл "CNT shifr.txt" записывается зашифрованный текст - режим гаммирования. В файл "CNT rasshifr.txt" - записывается расшифрованный текст - режим гаммирования.

Полный зашифрованный текст:

b"\x0f\x1e\xd6q\x8e\xb1T\x9d;kD\x99\x07\x8c\xf0\x13o\x10\xb6\xa0\x0f\x92\x9b  
QG\xbc\x04\\\x90\x96\x88\x94a\\\xca\xf7Dt\xac\xc3\x9d\xd3\xf4\x1cO\x01\x97{\  
xdd\_+\x97\xf6^\xe5\xdb!\xb4\\\xcc\xfc\xc7\x1f\xa6j\xbe\xc4\x19m\xb4N4%(\xe3\  
x9b\x11\xdd{\xaa\x0f\xe0G:\x81\xbf\x94\x87\x8b\xd9\x80\xf45)\xb1Gisy\x02\xaf  
\n+o#\xbb\xd3\xd5\x10\xd9\x8d,\xe1+\xf9\x81\x19uLL\xd5\xd2\xef;\x98\xef"\xb  
0^^Yu\xaf\x12\xe1\xb9\x1a.\xbc\xc9\xdd\xa1\x05\xe8|\x1f\xfcg\x85\x84\x8b!K3\  
xe8\xfd~\xbb\x12\xb4:o3Z\x9f\x89\xeeLR\xf1\xd4\xdf\xbb:\xe5\xb6-  
\xe42r\xd6\x86\xfa^0\x18\x83B\x15c\xa1\x89\$2\xafV\x9f\x91\x99\xee\x08\xc7\x

fc\xb0 \xb6{\x14\x8f"\xc1\$\x1bI\x9cE\x9d\xec-  
\xf6.<\x12\x0f\x84\x19)\x0e\x0b9I\r\x01\x00\xa2\x9d\x04\xbd\x01\x00\x01L\xcbfy;  
\xa6\xd5v\xcbby:\x10(\xe0(\xb07\xdb\x1a?\xd5\xec  
\xa0n\xca\x19K\x1f%\r\x084\x7fO\xfe)\x1d\xcbY(\x00\xd9\xa6"\xa3\xa9Q\x0f0r\xb  
3,!bMa\xa0vE7\x05\xaa\x9b\x0b\xed\x09\xea\xcc\x03\xa9\x98\x86\x04\xaeJd\xe1\  
\x961V\xe3\xcf\xae\xac\xd8\x01{\x97\xcc\xed\x1f\xaf\x02\xb7\xe8Zx\x99"\x01\xa  
9\xcc\x16\x02\xb3\x01,BM\xe4\x08\x06\x02\x9c\xa00x\x9d\xda\t\x03\x9d\xafj\xae  
\xbd\x08B5\x1dA\x94||\xb3\x06<%5\xaa\x0f0yx\x9d\xfa\x06lU\xed\xee\xe9\x14{\  
\xe0L\x15\x8c\x8f\x017\x03\x0c\xe4\xd2+8\x17\xd6\xe8\xe0\x88||\x1c\xb6Z-  
\xb0\x07"\x92\xe9\x02\xd4!f\xb3k\x09Mj\x06||\x85\xe1Z\xfd\xedNi}\x0f\x0e\x13  
\xf5\x80"\xc3J&\x0f\x070n&\xf8\xff\xcd\x03\xd2\xddw\xddb\x0f^\xa6\x07\xb4q[l  
\xb1)\xcb\x90T\x97\x0b\xd5pv\xe1\xcf\xb8\xcb\x1b\x00&K\x01\x01\xde\xeb\x0  
6\x0b\x85\x8c\x91o\x0b8\x89\xd7'\xc9t\xa2\xa9x\xa9\x97\x9f\x8e}\x94\x9a\x17\  
\xe3kB\xce\xd6\x12\xafhV|\xa4\x04\xe1~\xd8\xa0\xa5;\x1c\xcfI\x94s\xd1D\xd9\x7  
f\xfd||\xf5\x9e3m\x91T\x12rc\xabdh\x85R\x8bs\x9b\xee\x07?\x96\x09\x033\x08\x  
b1\xef\x94C{\xc2}\xfb\xd6\x03n\x81\x88\xd6r\xaa\n\xbc@Ku(g\xaa\x19y\xde7s\  
\x9fw\xcd\x00w\x0fah\x13\xda\x07\x81;\xf8\x09\x06\xeaN\x18.: \xdb\xae\x91{\x1e\_  
\x9c\x02\xd0\x89\xbe\x05,kZ:h\x84\x11\x0eE\xe7\xe7\x0f1,\xd5A\x1c)\x12\x99\n\  
\xe3\x9b\xa4\x07\x04g\xa7\xaa\x96M\xec#\x17\x9d:\x9bs\x98HV\xddne\x06\xdf\x  
dc\x08\x04\x0bsl\xd0\x03\x93S\x17\x07x\xa3\x8b\xa5\xbc\x03|\xdf\x8d6|\xb6T\x0  
6\xa9\xd2\xe0\x0e\xe6\xec\x01\x0eE\x16\xea\_ \xaaQ\xe3\xd2\x01\xe6r\xad\x87z\  
\x9ex(?S\x00\x8d\xaa\x02\xd3SI\x18%\x0c\xa6\x0b\xfd\xea\x01h\xa4;M\x9dq\x08  
7u\xfb,\x025<\x8c\x01\xe5\x06KgB.j\x01\x81\n\x1c\x07f#\x11CC\x88v%\xcc?\x89  
\x0bs\x0f\x0abE-\x81\xe6\x9a\x08-  
\xea\x09G\x05\xa3\xa3\x17\xdf\x85{\xb9!\xf2K\x82\xcb\x07\x1d(\xcfx11,-  
\xae\x8b\x09F\$\xb0\x10\x02%\xa0\xd6m\x9b\x09\xaa\xd7\xa6\x0e/\xb9\xcc\xd2q\  
\xd3fj\xe0\xbd\x03\x14Wp\xa9\xec\x92\ta<\xe5\x0c\x00^Q3\xa8'\xc9\xa0\x04\xd  
8\xac\x99\x0fj\x8a\x974Rx\x1c\x04\xd8:\x88V\x88\x18||\x0b\x0cvB\x1e8LE\x0b\  
\x9f\x19\xe5sc\x04\x8ev\x9c\xa7\x9fF\xdb\x02\x02+q?\x07\xaa\x06\x16\x02K\x0  
8\x8b\xa8&\xf8\x94d6\x88\xcd!\xe4\x87\x03\x07\x9e\x88I-  
XBpP\xd0\x96\x88\xec\x03~\xb5\x00\xa7\xff\x17\x85\xe4\x08\x041\x19\xe1Z\x  
01-  
\xb3X\x08\xd5\x86~\xddq\x8fA\xd4\xd9\x1c%%\xd2\xff\xfd\x91\x83x\xeb\x01\x0  
e\x9b\x10\x15\xdeQ\x04?z[\x1f:s+\'0\xa5\x8cO\x12+L07\x8c\x07tX\x19\xfe\x87i\  
\x91\x11\x8e+M\x05\xfdX^\x1c\xfb\x05\xa0\x03\x0e\x01\x04\x82\x07fCl\xd4\x08\x  
82\x82\x08\xa0jB(\xc1y  
\x86\xba&\x1f+|=|\xd7\x025\xd9\x06\n\x0b0R\x0eg\x8e\x19\xde\x01x\x08M\x80(  
mw\_\x16\x8b-  
e\x08\x03}Q\x09\x90/5\x0cs:\xa3\x00QN\xbc\x9aJ3\x9e\x09\x8e\x1az\x0e\xd4[S\x  
ed?/\xbfxde\x82\xe77\xe8\xa6\x97\xbf\xeb\x03`\x03\xda\xdbuo\x0e\x10\xe4\x1d  
\xcc\x0b\x05?\x08\xa8H\x02\x00\x9c\xe2^\x9e+\xbc\x07\x15\x9e\n\x08\x93\xea\x  
ae\x8f\xad\x1e\x03\x0b\x1b\x04\x05\x8c\xe5(\x97\x1d\xe5te\x05w\xa4\x06\x1f\x9  
e\x035M\x02\xe3\xbc\x93v\x83\x1bG\x03\x07\xe2\x06Q\x04<Y\x1d\x18\x08&\xf8  
j\x069L\x0e\x88d,\xb0\x9a\xe7R\xec|\x89\xe0\xac|\x9e\xa6\x07\x09\x1dW\xaf\x  
e1\x88\xbc\xa2\xdb\x04/\x84\x94\x97/\xab\xa1\x07\*)F\xe6\x02ct\x8ea|\xac\x0e\

xc8\xa1\xf12\xf7\xa4\$V\x01k\xec\x1f\xa8-  
\xbc\xdefalbx\x89\xc2\xb3\xf8\x1e\x8b\xc3t"E\xb3x\x9d\xc7\xf7\xa6\xd6eN\xc8\  
xa5\x9d\xda\xd2  
\xf0d\x08GE\xf6\xd3\xb4\t\xa0\x08\xa6\x15V7\xef\x83S\xe6\x9e\xfc\|s\xe3\x07\  
daQ\x8d  
,\xc72\xbd'\U\x0b\x95j\x93\xf5N\x81\x8a9U\xad\xba^\xd176\x9e]%\xee\x1d\x18  
=\x18\xe7\xcdk~\x9b\x1b\xb3\xf8\xc0YP\x1cI8\x08\xd1\xf3\tk\x99h\xf2@\x8d\  
d1\xd2\x84\x08\x08'\xde\xe3\x9dz\xa1b\xac\x8a\xdb\x907\xb1\x02\xb3s\x16\x11\  
xb7\xa0\xcb\xf9\x99\xe2\xb0\xf6~]\xbc\xc6\xc0\x10\xc9\x81\xc4\xec\x9c\xc9\xdc  
\xce\xc9\xfa\$\x89\x8f\xdf\xfe\xba-  
J\x99\xeb\xef\xc9\x87\xbd\x9eT\x90\xaa\xac\xf4Sy\xa5\x11Z#\xfdsXt!\xc5^\xae\x  
e1"\xd0M\xba\xbb\xbd\x8\xab`\xd1\xe3\xe5\x83\xa6\xfc\xd6\xeeA0\x94^9C\x03O\x  
f\x83\xc1\xcdV\$\xc4q\x19B\xd1J^\x93\xdcv\x8a\xf53\xf2\xa5\xa3F!b\xcc\xa3\x0  
e,\x81\xf5\x9bk\xc6\xd3\x02\xf5\x9c\x1a6t\x0c\xb9\x8b\x07Arp\xeeu\|3\x1bet\x  
e\x7\xe3\xdbW\xd3\xa7\xce\xfdU\x9e\xe4\xebd\x13\xdb\xe3\xdb\xb9t\xffZ\x95\  
xdcuP\xf9\xe46\xa5\x04\x12\x86\x98\xd3\x0fi\xdc\x95\xac\x9c\x04\x07q\xcc\x2  
\x0c\xd3\xb11eV\xca\x6t\x9c\xa0\x7f\xcd\x8\xaeiT`\xfc\x87p&{u?s\x8e\x90}\  
xda\x90\xdb\x20J\xdd\x1a,\xd7H\x83\x12~\x85\xb5-  
\xbcz9\xb7I\x0c\x0\xa2T\x83\x82\x8a\xd4\xa3\xc4\xc3\xe8\x08\x88\x03,J\x99^\  
xd6<\xdf\xbf\xbb,\xe6\x19'\xbb}\xeb\x01\xc8{\xee\x7f\xbb\xcf\x7\x8d\xd9\xce\  
xfe\x02\xbcHAOf\xa2\xe6\x9d\xb9U\x87\x7fV9\xb2~\xc0\xd1\xf1\x8bp\xba\xdb\  
x1a\xbb\xea\x82\xd6Gaby)[P\xdc'

Полный расшифрованный текст:

История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический процесс - последовательный процесс развития природы и общества, череда сменяющих друг друга событий, в которых проявляется деятельность многих поколений людей. С каждым мгновением времени, ушедшим в прошлое исторический процесс дополняется совокупностью новых событий, явлений, фактов и факторов в жизни человека, семьи, этноса, государства, человечества. Подобно тому, как материальный мир переживает процессы негэнтропии и энтропии, природа вокруг нас - процессы эволюции и инволюции, человеческое общество характеризуют процессы социального прогресса и регресса. Предметом изучения истории является деятельность главного его субъекта - человека - в прошлом. В узком смысле история представляет собой совокупность социальных фактов, знаний, представлений ученых о том, как эти процессы осуществлялись и осуществляются. Долгое время история существовала скорее, как литература и искусство, обслуживающие интересы власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств и наук, среди других муз была Клио - муза истории.



После выполнения программы в файл “CFB shifr.txt” записывается зашифрованный текст - режим обратной связи по шифротексту. В файл “CFB rasshifr.txt” - записывается расшифрованный текст - режим обратной связи по шифротексту.

CFB shiffr.txt – Блокнот

ФайлПравкаФорматВидСправка

b' \xc4\xbd\x9b\xf5\xcd=f\xa5\x81^c\xff\xf9%\x8f' 'P-\x19c5j\x91\x85\xf7\xfda"xe5;\x99\xce\xe0={\xf3\x97\xf8\xf2\xac\xb1\xc7\xfap\x922Y""\xf5x\x17M\xb0G\xe7\xb8\xbe\>!  
!\x94\xe4\xa5K\xfd\xa3A\x85P3I\xc5\xac\xdd\xfc\x94\xc2\xab\x9fH|b|'\x189\xdb\xa0c\xf6\x84" w\x99\x03W\xbak\x92\x9b\xdb=D\x1c\xea\xe9\xbe\x10\xb8\xb26\xdahzD\x6e!  
f6s\x1f\x2a2\|y\w\xcb6\x86\xb3\*'\t'\xc2\xb5\xc0\x12\xf6g\x12\x82\xca\xe3\x7f\xa2puF\x10\x94\x8d\x80\x0c\xf5\x99[\xa90~\xc4\x0e\x17\xcb=\x13\xa2e\x9c29L\x84a\xee  
\xb0\x90z\xa1#\xa9a\x07\n\x9d9\xa2\xad@a\x55\x|u\x5b\x98sQ0\xcf2\x3f\xff\xeb\x2d\x11\x01\x0d7\x1cp\xa4\x96K\x0f\xad\x9bF\x88""\x08[\x83\x0d>\xd3\x1c)s\x17JY\x17' \xc  
xb\x1d\xa5\xf6\x10\x97\xda]I\x97\xc2\x174c\xe0W\xee2i=\xe6. \xc0\xee\x16K\xcc\xnov\xce\xe6\xccdG>a\x8f\x0b7\xbbY\x1eyG\xc8k'\x0bi\x7fb\x86>\xf9G\x1d5t\r\x1b\x88\>

CFB rasshifr.txt – Блокнот

ФайлПравкаФорматВидСправка

История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический процесс - последовательный процес  
ивание интересов власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств и наук, среди других муз была Кли

Полный зашифрованный текст:

b\xcaV\xb9\xf5\xcd=F\xa5\x01^c\xff\xf9%\x8f\`P-  
 \x19<Sj\x91\x85\xf7\xfd^\xe5;\x99\xce\xe0={\xf3\x97\xf8\xf2\xac\xb1\xc7\xfa\x922Y"\xf5x\x17M\xb0G\xe7\x8b\xbe\x04T\xd8&`\xfd\xe4\x03p\xf1~\x13\x8aJ\x87Ta\xc1\xb1l\xfeGw\xfb\x8e\xf5\xcb\x14\xb7H\x00\xf1\xdf0\xf8\xd8~\x90\xf7E\xaa\xad7\x9e\x97\xe8,l\x1a\x03I\x08p\x12\x99x|\`|\xb8|\xf3|\x8dH|\x072|\x93|\xb1|\xa7|\x03\*|\xa9F|\x82|\x84\$|\xbc\xab(\xac\xdd\xaa.\xc3c|\xb5t|\x03|\xf1R|\x0c|\x90|\xbc|\x8d|\xb4q|\x92|\xfc54|\xd3|\x0eH|\xaf\xef\xe5|\x85|\x8c|\x1e2|\xe8|\x1b|\x06|\xf2|\x11|\xe2|\xd8|\x0eQd|\x8dk|\xfeH|\x19|\x00|\x96|\xd5|\x8d|\xa8|\xae|\xea8|\xa4=|\x86|\xc1|\xbc|\xf1|\xa1-|\xb3|\x9d|\x8c>|\x0e|\x93|\x1b:|\x02|\x881|\xfc|\x99|\xe4|\xf3|\xb1|\x80|\xa9X|\x84|\xa1|\x9e|\xab  
 J\xe2W\xfc|\xbc|\xa7|\xb3|\xae|\xf2|\xb3|\xd6e|\xbe|\xec|\xba|\xce@\xcfZ8g\xdc|\x04|\xb4^|\x80|\xa36|\x05|\xb0<|\x9d|\x8e|\xbc|\x1a~3M|\xb7B|\xf97|\xddV|\x1a|\x14i|\xa6A|\xc2|\xa1r|\x0f|\xdb|\xdam\*5y|\xb7|\x87|\xacg8p|\xdd|\x12x|\x04N|\xa3|\x11Y|\x01#|\xe3|\x94T|\x8cb|\xaf|\xee|\xb1|\x89|\xdb|\x07,|\x92|\xbco|\xb0|\xba|\x9d"\xd3|\xe2Q9|\xc4|\xde|\xd8|\xfb|\x02@|\xd9D|\x13m|\xb2Q|\x00|\xbdMXy|\x9f|\`|\xcd|\xae|\xe8|\x1c|\xf9|\x98|\xf2|\xc0!|\x94|\xe4|\xa5K|\xfd|\xa3A|\x85P3I|\xc5|\xac|\xdd|\xfc|\x94|\xc2|\xab|\x9fH|b|\`|\x189|\xdb|\xa0c|\xf6|\x84`w|\x99|\x03W|\xbak|\x92|\x9b|\xdb=D|\x1c|\xea|\xe9|\xbe|\x10|\xb8|\xb26|\xdahzD|\xe6|\x1a|\xdbb|\x13l(Cg|\xb9|\xc4X|\x0br;|\x7f|\x16|\x8cnN;|\xefm|\xffs|\x9d`|\xb1|\x16|\x8c0g5pR\$\*r=|\x8d|\xfd|\x98|\x94|\xb7ti|\x17P1-  
 \xf2xu|\x86j|\xc5|\xaa|\xc0|\x08NbCg|\xcf|\x1b\_N|\xea>|\xae|\x0e|\xf4|\xf2|\x01h|\x8e|\xa8|\xc7|\x99+u||\xf1|\xf2|\xf3|\x86|\x9e|\x95|\xaa|\xc9b8|\xb5T|\xda|\x8c|\x10|\xd2|\x19|\xac|\x8e|\xc9P|\xd9X0|\xbc`|\x1f2|\x81|\xaa|\xc1|\xcd|\x0b|\xed|\x10|\xe6q|\x9c|\xe1|\xca%m|\x11|\xc3|\x1bZ>|n-  
 k|\xfd|\x9b|\xbd?|\x86|\x1e|\xb1Fv|\x84|\xe9h\$|\x14z|\xd5|\x04|\xad|\xb5|\x17|\x96<,\x1e|\xac|\xd4u|\xfdga|\x1b|\x9a|\xb7D,|\x97!|\xf4d|\x8b|\x1ac|\x8a|\x94|\xabB|\x82|\x88|\xf4|\x0f|\xb0q|\xe2M7|\x90=Z-  
 a|\xac|\x85|\xf1|\xd91|\xeb|\x9d|\xcfa|\xd8|\xf8RpdXo|\xad|\x0f|\xc1J|\x81|\x06C|\xbc|\xeb|\xd7|\xcd|\n|\xf9|\xed6|\xf9|\xdb|\xae{\`wJ|\xc8|\xb9|\xfd5{\`|xaajn|\xdd|\xa7E|\xa2|r|\xb6|\x1c|\xe2G|\xbe|\xc1|\xe5|\xa5|\x1b|\x03`|\xafYjP|\xf8j|\xd0|\x15|\x87|\xeci|\x16|\xde|\xaa|\x10?|\x12|\xbe{\`|\x88^!Y|\xc9|\x7f~|\xf2|\x84|\xf4|\x99|\x9bwA&^@|\xa8|\xfbE|\x9a|\x03|\xbdH"

x18xR\x8dt\xfb\xe1\x80\x9f\x83\xbb\xa2\xd0\xf6s\x1f\xa2\|yw\xcb6\x86\xb3\*\"t\|  
xc2\xb5\xc0\x12\xf6g\xd1\xc2\xca\xe3\xf7\xaa2puF\xb1\x94\x8d\x80\x0c\xf5\x99  
[\xa90~\xc4\x0e\x17\xcb=\x13\xa2e\x9c29L\x84a\xee\x1a\xdfnO\x00kh\x18\x19\  
xa7\x13@e\x05O\xa0\xd2=d\xcf\xa0\x14E\xc8\xb6  
|x13\xf4\xde0\xfcGJw\x83G\xf3\xdf\xb0DdIn\x0c\xae,)W\xb5\x18\xe3\xfb\xb1|x  
dc\xbf\*\x85\x046\xdb\xc2'\x93\*\x1a\xdb\x99\xef\x11]\xe8\xd3\xe74P\xb2\x8c(\x  
e2\x0f\x07\xca\xcaMnGX\xb0\*\x8b\x8d:\xbc\xb5Fni\xedb\$\x06D.\xf2\xdaa\xdb|x  
08\xc8\xbd\xd7\x94K\xec\xcdz\x1bh:\x14\xb6oj\x91\xd0\xc9{6sT\xa5\_=m\xda\x9  
9\x93\xde\xfa\xab\x97\xc0B{\x1cF\xd5\xa9\xd52!/\x0c=\x1bIO\x17\xa0\x0bL\xdf  
f\x93\x83\xc4\xa1\x93E\x15@\x19[,xe0\xd7\n&h\xb6\x97\xa5\x8a\$\xc4\xec\xee\  
xf4\xc7\x05'o\xe5\x85\xa3\x18}\x8e\xa3\xe1\xc7\xfd\xd8\x9a\x07B||\xd6\xde\xad  
d\x02\xe5\xd1~!+|n||\x8b\xe1Mb\x8d\xa7\xe3\xf6\x16|r+\x03\x91\xe6|r\x1b\xeeQ  
|xf04y#X\xe49\x99\xd2\xa6|r\x8b\xe4\xef\x97\x8c\x12\x8aq\x97wdj\xa6Fr\xa8F.|  
xf9\x93t\xe5\xe1\xff\xe7o\xbcH0\xa1\xf1\x08\xe6p\x1eX\xc2At\x8b\x88P\x12\x8  
6\x9d&\xa9\xb0\xdfM\xc0\xa1\xb1P\x95\x02\xbc\x90z\xa1#\xa9\x07\n\xd9\xa2\xad  
@a\xb5x|u\xb5B\x98sQ&|f2|f3|ff|eb|\xd2\x11\x01\xd7\x1cp\xa4\x96K\x0f  
|xad\x9bF\x88"|x08[|x83\xd0>|\xd3\x1c)s|x17]Y\x17'\xc0&\xc4\x8e\x03Y|xcc|ff  
|x00!|xf66\xa7uj=|eb|x85\xa1\xf5p|e3|ba|c9|x1dcf|\xd9\xf0|x85D|x1b|e8|ff  
|xed|bd|\xd0<+|rp|x9fy|b8|x9d|f2|x14a|r%|n0|\xd6|b7|bbx"|xf2|x8fM||Z|xacu  
0|xea^X|\xd8|\xd5|\xc4|\x90|\xa8'\xc9|\xf3|\x94|\xfdl5|\xa2|\x859|h|e2|\xc8KI|\x971|\xd8  
|\xda|\xd7|I|\x96a\_9|\xa8|e3|\x90Y|\x87Z|\xd735|\xc02|\x00A|bd|cb|\x9dZf9T|\xda|x  
e1|\x12|b7|\x92|\x0e+B|\xa4-  
|xa4|\x17|\xe3|\x89|bf}|s|x99|\xf1|\x82t|t|fd|f0Z\$|\xa9|r|\xd7|\x98Wy|\x13|\x9d|\x97|x  
7fQ|\xa0"|x91DC|tA|df&=|\x8d%|\xa6|b3|xea6|x9e^|\xac|\x0c|\x9e|\xa8|aaaf&|df|  
|bb)G]|xeeo9|cf|\x99|\x85P|\x82|\xb5vV|\x93:{|\x17|\x95i|ab|f6|x04|\x8d|\x87|b2  
Za5#|\x1f|b7|fd|b9|\x81|f7C|\x9f|fa|\x19|\x8c|\xc65F|\x07|\xed|\xde|\x10|\x07,wt)g  
vo|f2w|ee|\x19|cb|ea|ecM|\xd5|b9|\x97|f7|\xa2|f5|\xcd|ab|fd|\x84]|x8cpc|  
xd5`v|dfns|\x90|\x96|fd|ee|x8f|r|ca+|\x99|\xe8d|e5|\x9c|bd|bbe|\x15|ff|f3|x  
fa|e2|\x1eZ|\xc5|\x8e|\x97zNr|afq|ccM\*||\xac|eb|a5|x9e|f9|bb|\x1d|a5|f6|x  
10|\x97|da]I|\x97|\xc2|\x174c|e0W|ee2i=|e6.\xc0|ee|\x16K|ccC|nov|xce|e6|x  
ccdG>a|x8f|b7|bbY|\x1eyG|\xc8k}|x0bi|x7fb|x86@|f9G|\xd15t|r|b1|x88|cf|f  
f|\x1d|b7|\x97|f4|f7|\xc1WK|\x01|\x1aS|\xc2|ba|c6|\xd5KK|\xa4U;|b6|\xc8|\x0e  
@|bf!|ffT#c|e7|\x0b|fa|c7|\x9e|x8f|\xd8y|x8cN|bd|tj|ca|e6|\x05+<t|\x1f|\x0  
1|\x98|\xc6e|\x15;|\x17|\xde|aeT|te|\x90|cb|\x92|\xc8|af|e3#HS7|\x17i|eb  
D|\x8cWI|df[p|\x08|\xd6|f6|\x82O#|bd|\xb8|bf|\xa8+|\xd5|B|\x81,|e2m|\xa9A|\x9  
a|\x0bh|\x94|eb)|x02X|\x1f|b6{||\x1b|\xc9|\x8b|e5a|\x87|\x9dp)|e9|\x89|\x8b|\xd7|\x8  
4|bcY|b4i|\x84|\xa6y|ca|aep|\x0c:\x86|\x12|f1|\x8b|f6|\x83|\x12f|\x15|\x04|\xd2a|  
x0e|\x84|b2|\xd3|\x00L|bf|\xc0|\x9e|\xd3|\x8d|b8|fc|\xde|\xf8|\x011|b9|be|f5~L  
b|\xc1|t3|\xcd|\x81|cb|\x17|\x8ad~|\x86|\xa6|efJ|cb|\x84|\x0e`Rm|\x1ct^|\x90|e12|b  
0|\xccbK|\xa88R|\xc9{|\x9d=|b7|ee|a8|xedEDy|\xe148d|\x90~U|a|\xa0h8|\x01|f3x(  
,

Полный расшифрованный текст:

История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический процесс - последовательный процесс развития природы и общества, череда сменяющих друг друга событий, в которых проявляется деятельность многих поколений людей. С каждым мгновением времени, ушедшим в прошлое исторический процесс дополняется совокупностью новых событий, явлений, фактов и факторов в жизни человека, семьи, этноса, государства, человечества. Подобно тому, как материальный мир переживает процессы негэнтропии и энтропии, природа вокруг нас - процессы эволюции и инволюции, человеческое общество характеризуют процессы социального прогресса и регресса. Предметом изучения истории является деятельность главного его субъекта - человека - в прошлом. В узком смысле история представляет собой совокупность социальных фактов, знаний, представлений ученых о том, как эти процессы осуществлялись и осуществляются. Долгое время история существовала скорее, как литература и искусство, обслуживающие интересы власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств и наук, среди других муз была Клио - муза истории.

7. Вся работа происходит в файлах: "opentext.txt", "ECB shifr.txt", "ECB rasshifr.txt", "CNT shifr.txt", "CNT rasshifr.txt", "CFB shifr.txt", "CFB rasshifr.txt".