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 F: Поточные шифры

Выполнила студентка 3 курса группы 171-341

Решетникова Дарья

Москва 2020 г.

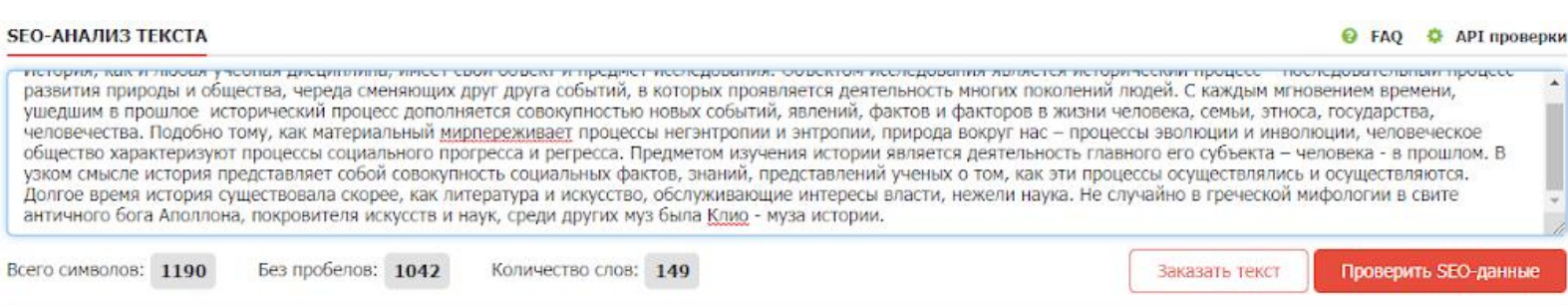
Аннотация

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.



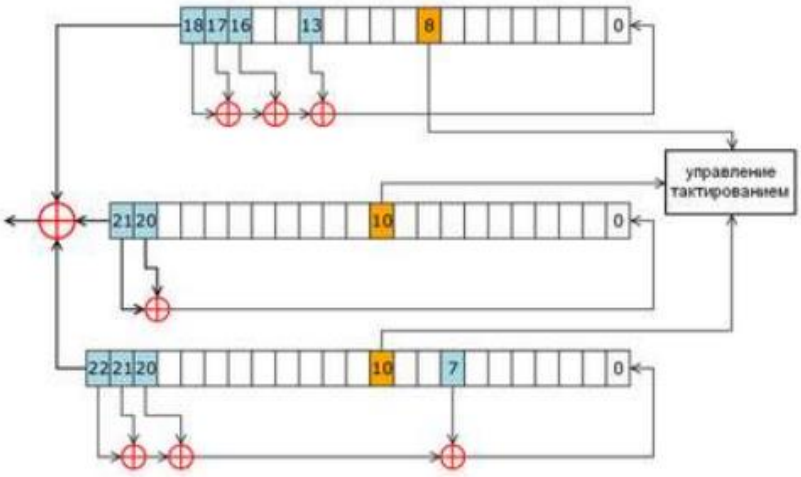
A5/1.

1. Описание шифра.

A5 — это поточный алгоритм шифрования, используемый для обеспечения конфиденциальности передаваемых данных между телефоном и базовой станцией в европейской системе мобильной цифровой связи GSM (*Groupe Special Mobile*).

2. Алгоритм шифра.

АЛГОРИТМ ШИФРОВАНИЯ A5/1



Система РСЛОС в **алгоритме A5/1**:

Три регистра(R1, R2, R3) имеют длины 19, 22 и 23 бита,

Многочлены обратных связей:

$\mathbf{X^{19} + X^{18} + X^{17} + X^{14} + 1}$ для R1

$\mathbf{X^{22} + X^{21} + 1}$ для R2

$\mathbf{X^{23} + X^{22} + X^{21} + X^{8} + 1}$ для R3

Управление тактированием:

биты синхронизации: 8 (R1), 10 (R2), 10 (R3)

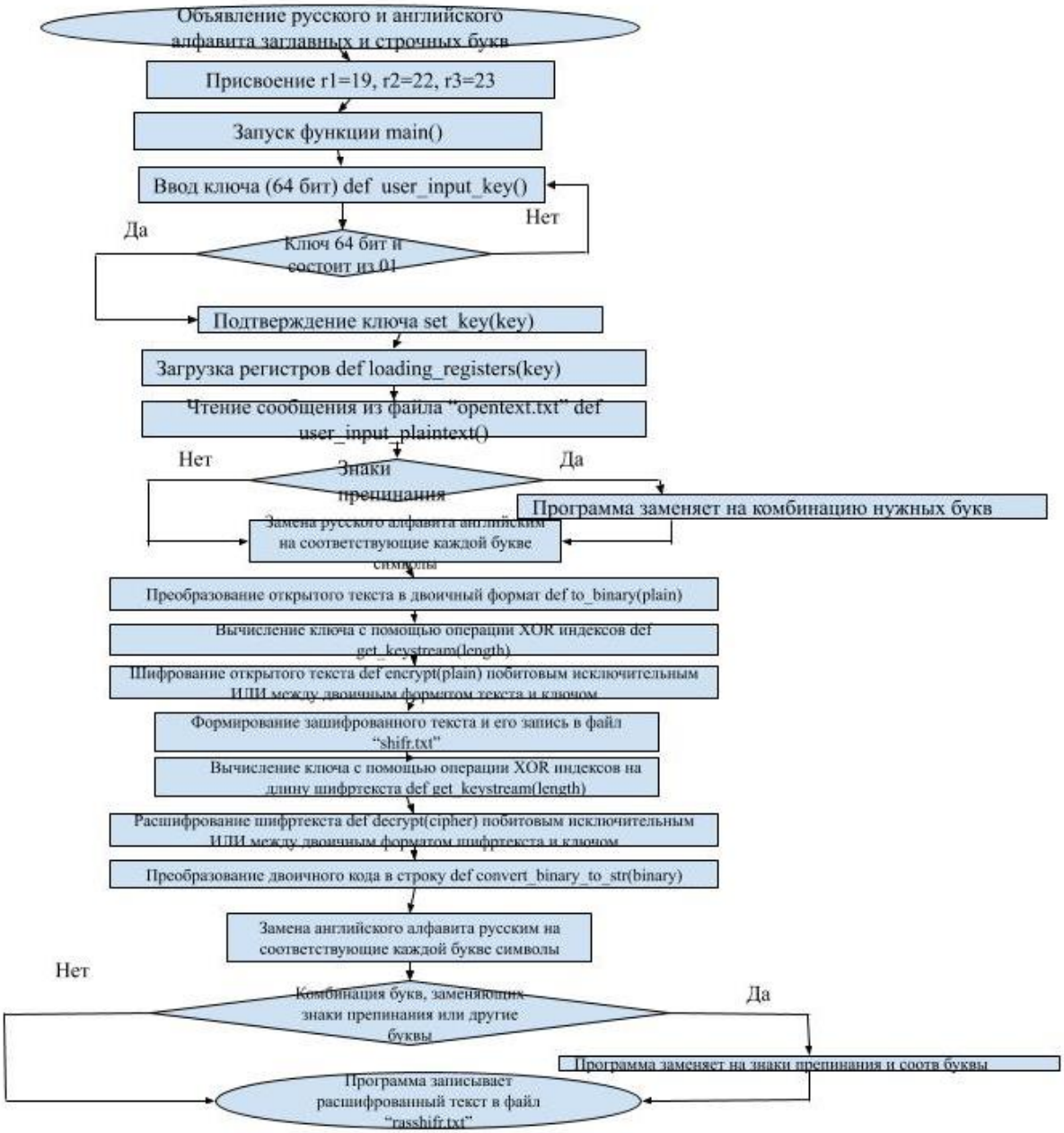
функция F = x&y|x&z|y&z,

где & — булево AND, | - булево OR, x, y и z — биты синхронизации R1, R2 и R3

- сдвигаются только те регистры, у которых бит синхронизации равен F.

Выходной бит системы — результат операции XOR над выходными битами регистров.

3. Блок-схема программы



4. Код программы

```
# модули
import re
import copy
import sys
rus = [ 'а','б','в','г','д','е','ж',
'з','и','й','к','л','м','н','о','п','р','с','т','у','ф','х','ц', 'ч', 'ш', 'щ',
'ъ','ы', 'ь','э','ю','я',' ']
eng = [
'a','b','v','g','d','e','ty','z','i','y','k','l','m','n','o','p','r','s','t','u','f','
ry','c','ry','ly','dy','sy','xy','fy','j','q','w','{']

rus_big = []
eng_big = []
for i in rus:
    rus_big.append(i.upper())

for i in eng:
    eng_big.append(i.upper())
```

```
r1 = 19
r2 = 22
r3 = 23

key_one = ""
reg_r1 = []
reg_r2 = []
reg_r3 = []

def loading_registers(key): # загрузка регистров
    i = 0
    while(i < r1):
           reg_r1.insert(i, int(key[i]))
           i = i + 1
    j = 0
    p = r1
    while(j < r2):
           reg_r2.insert(j,int(key[p]))
           p = p + 1
           j = j + 1
    r = r2 + r1
    k = 0
    while(k < r3):
           reg_r3.insert(k,int(key[r]))
           k = k + 1
           r = r + 1

def set_key(key): # устанавливает ключ и загружает регистры, если они содержат 0 и 1 и
длина ключа 64 бита
    if(len(key) == 64 and re.match("^([01])+", key)):
           key_one=key
           loading_registers(key)
           return True
    return False

def get_key(): # получен ключ
    return key_one

def to_binary(plain): # преобразование обычного текста в двоичный формат
    s = ""
    i = 0
    for i in plain:
           binary = str(' '.join(format(ord(x), 'b') for x in i))
           j = len(binary)
           while(j < 8):
                  binary = "0" + binary
                  s = s + binary
                  j = j + 1
    binary_values = []
    k = 0
    while(k < len(s)):
           binary_values.insert(k, int(s[k]))
           k = k + 1
    return binary_values


def get_majority(r1,r2,r3): # получение большинства
    if (r1 + r2 + r3 > 1):
           return 1
    else:
           return 0

def get_keystream(length): # вычисление ключа с помощью операции XOR индексов
    reg_r1_temp = copy.deepcopy(reg_r1)
    reg_r2_temp = copy.deepcopy(reg_r2)
```

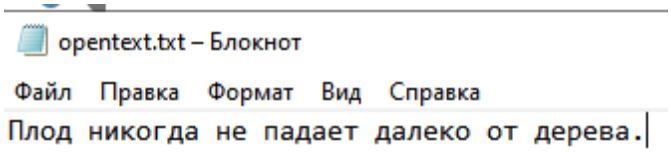
```
    reg_r3_temp = copy.deepcopy(reg_r3)
    keystream = []
    i = 0
    while i < length:
           majority = get_majority(reg_r1_temp[8], reg_r2_temp[10], reg_r3_temp[10])#
биты синхронизации
           if reg_r1_temp[8] == majority:
                  new = reg_r1_temp[13] ^ reg_r1_temp[16] ^ reg_r1_temp[17] ^
reg_r1_temp[18]# многочлены обратных связей
                  reg_r1_temp_two = copy.deepcopy(reg_r1_temp)
                  j = 1
                  while(j < len(reg_r1_temp)):
                         reg_r1_temp[j] = reg_r1_temp_two[j-1]
                         j = j + 1
                  reg_r1_temp[0] = new

           if reg_r2_temp[10] == majority:
                  new_one = reg_r2_temp[20] ^ reg_r2_temp[21]
                  reg_r2_temp_two = copy.deepcopy(reg_r2_temp)
                  k = 1
                  while(k < len(reg_r2_temp)):
                         reg_r2_temp[k] = reg_r2_temp_two[k-1]
                         k = k + 1
                  reg_r2_temp[0] = new_one

           if reg_r3_temp[10] == majority:
                  new_two = reg_r3_temp[7] ^ reg_r3_temp[20] ^ reg_r3_temp[21] ^
reg_r3_temp[22]
                  reg_r3_temp_two = copy.deepcopy(reg_r3_temp)
                  m = 1
                  while(m < len(reg_r3_temp)):
                         reg_r3_temp[m] = reg_r3_temp_two[m-1]
                         m = m + 1
                  reg_r3_temp[0] = new_two

           keystream.insert(i, reg_r1_temp[18] ^ reg_r2_temp[21] ^ reg_r3_temp[22])
           i = i + 1
    return keystream
# ^ - побитовое исключительное ИЛИ

def convert_binary_to_str(binary): # преобразует двоичный код в строку
    s = ""
    length = len(binary) - 8
    i = 0
    while(i <= length):
           s = s + chr(int(binary[i:i+8], 2))
           i = i + 8
    return str(s)

def encrypt(plain): # шифрование (открытый текст преобразуется в двоичный, получаем
полный ключ после ввода длины двоичного текста и получаем значение XOR полного ключа и
двоичного текста)
    s = ""
    binary = to_binary(plain)
    keystream = get_keystream(len(binary))
    i = 0
    while(i < len(binary)):
           s = s + str(binary[i] ^ keystream[i])
           i = i + 1
    return s

def decrypt(cipher): # дешифрование (принимает шифр, получает полный ключ от его
длины, XOR шифра с ключевым потоком и преобразуется в строку)
    s = ""
    binary = []
```

```
    keystream = get_keystream(len(cipher))
    i = 0
    while(i < len(cipher)):
           binary.insert(i,int(cipher[i]))
           s = s + str(binary[i] ^ keystream[i])
           i = i + 1
    return convert_binary_to_str(str(s))

def user_input_key(): # ввод ключа
    tha_key = str(input('Введите ключ (64 бита): '))
    if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
           return tha_key
    else:
           while(len(tha_key) != 64 and not re.match("^([01])+", tha_key)):
                  if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
                         return tha_key
                  tha_key = str(input('Введите ключ (64 бита): '))
    return tha_key

def user_input_plaintext(): # чтение открытого текста из файла
    f = open(r"opentext.txt", "rt", encoding='utf-8')
    someIn = f.read()
    someIn = someIn.replace('.', 'тчк') # Если в сообщении попадется точка, она
заменется на тчк
    someIn = someIn.replace(',', 'зпт') # Если в сообщении попадется запятая, она
заменется на зпт
    someIn = someIn.replace('-', 'тире') # Если в сообщении попадется тире, символ
заменется на тире
    #someIn = str(input('Введите текст для зашифрования: '))
    return someIn

def user_input_ciphertext(): # чтение шифртекста из файла
    f = open(r'shifr.txt', 'rt', encoding='utf-8')
    ciphertext = f.read()
    return ciphertext

def tha_main(): # главная функция записи в файл шифртекста и дешифрованного текста
    key = str(user_input_key())
    set_key(key)
    if True:
        plaintext = str(user_input_plaintext())
        print('Открытый текст: ', plaintext)
        plaintext2 = ''
        for i in plaintext:
            try:
                if i.isupper():
                    plaintext2 += eng_big[rus_big.index(i)]
                else:
                    plaintext2 += eng[rus.index(i)]
            except:
                plaintext2 += i
        f = open('shifr.txt', 'wt', encoding='utf-8')
        sh = encrypt(plaintext2)
        f.writelines(sh)
        f.close()
        print('Зашифрованный текст (двоичный вид): ', sh)
        plaintext = ''
        file = open('rasshifr.txt', 'wt', encoding='utf-8')
        rasshifr = decrypt(sh)
        for i in rasshifr:
            try:
                if i.isupper():
                    plaintext += rus_big[eng_big.index(i)]
                else:
                    plaintext += rus[eng.index(i)]
```

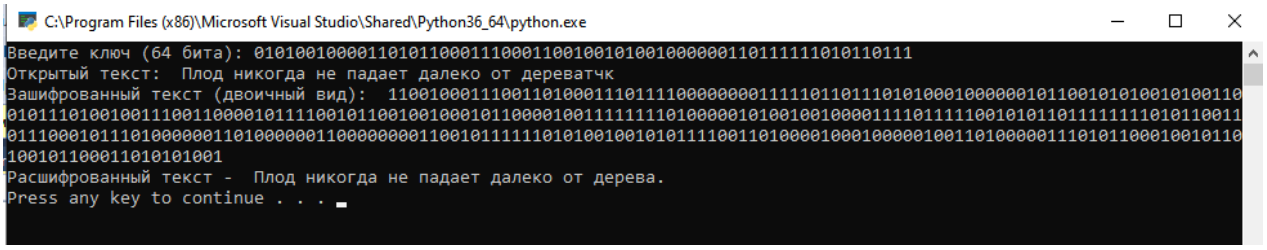
```
            except:
                plaintext += i
        plaintext = plaintext.replace('рй', 'ч')
        plaintext = plaintext.replace('РЙ', 'Ч')
        plaintext = plaintext.replace('тй', 'ж')
        plaintext = plaintext.replace('пй', 'х')
        plaintext = plaintext.replace('ПЙ', 'Х')
        plaintext = plaintext.replace('ТЙ', 'Ж')
        plaintext = plaintext.replace('хй', 'ы')
        plaintext = plaintext.replace('лй', 'ш')
        plaintext = plaintext.replace('ЛЙ', 'Ш')
        plaintext = plaintext.replace('дй', 'щ')
        plaintext = plaintext.replace('ДЙ', 'Щ')
        plaintext = plaintext.replace('сй', 'ъ')
        plaintext = plaintext.replace('СЙ', 'Ъ')
        plaintext = plaintext.replace('фй', 'ь')
        plaintext = plaintext.replace('ФЙ', 'Ь')
        plaintext = plaintext.replace('тчк', '.')
        plaintext = plaintext.replace('зпт', ',')
        plaintext = plaintext.replace('тире', '-')
        file.writelines(plaintext)
        file.close()
        print('Расшифрованный текст - ', plaintext)

#Example of 64-bit key:
0101001000011010110001110001100100101001000000110111111010110111
#Плод никогда не падает далеко от дерева.
tha_main()
```

5. Тестирование

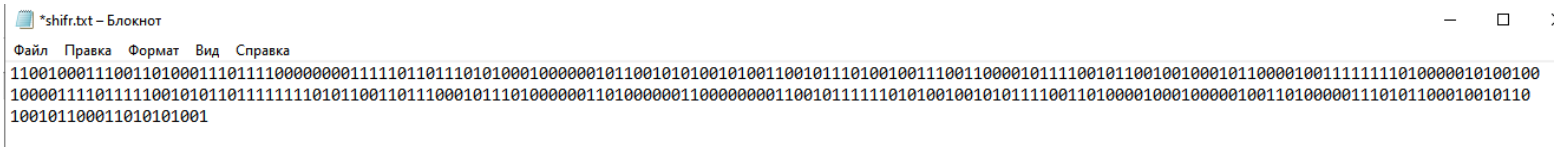
Перед началом работы программы в файл “opentext.txt” записываем исходный текст.

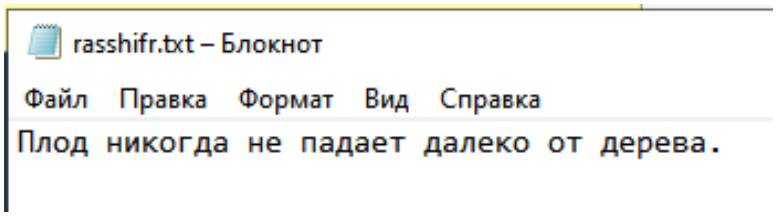


Так выглядит окно выполнения программы. Ключ вводит пользователь.



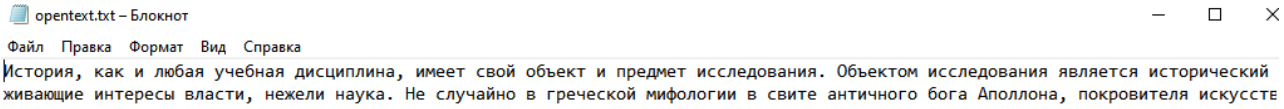
После выполнения программы в файл “shifr.txt” записывается зашифрованный текст, а в файл “rasshifr.txt” расшифрованный текст.



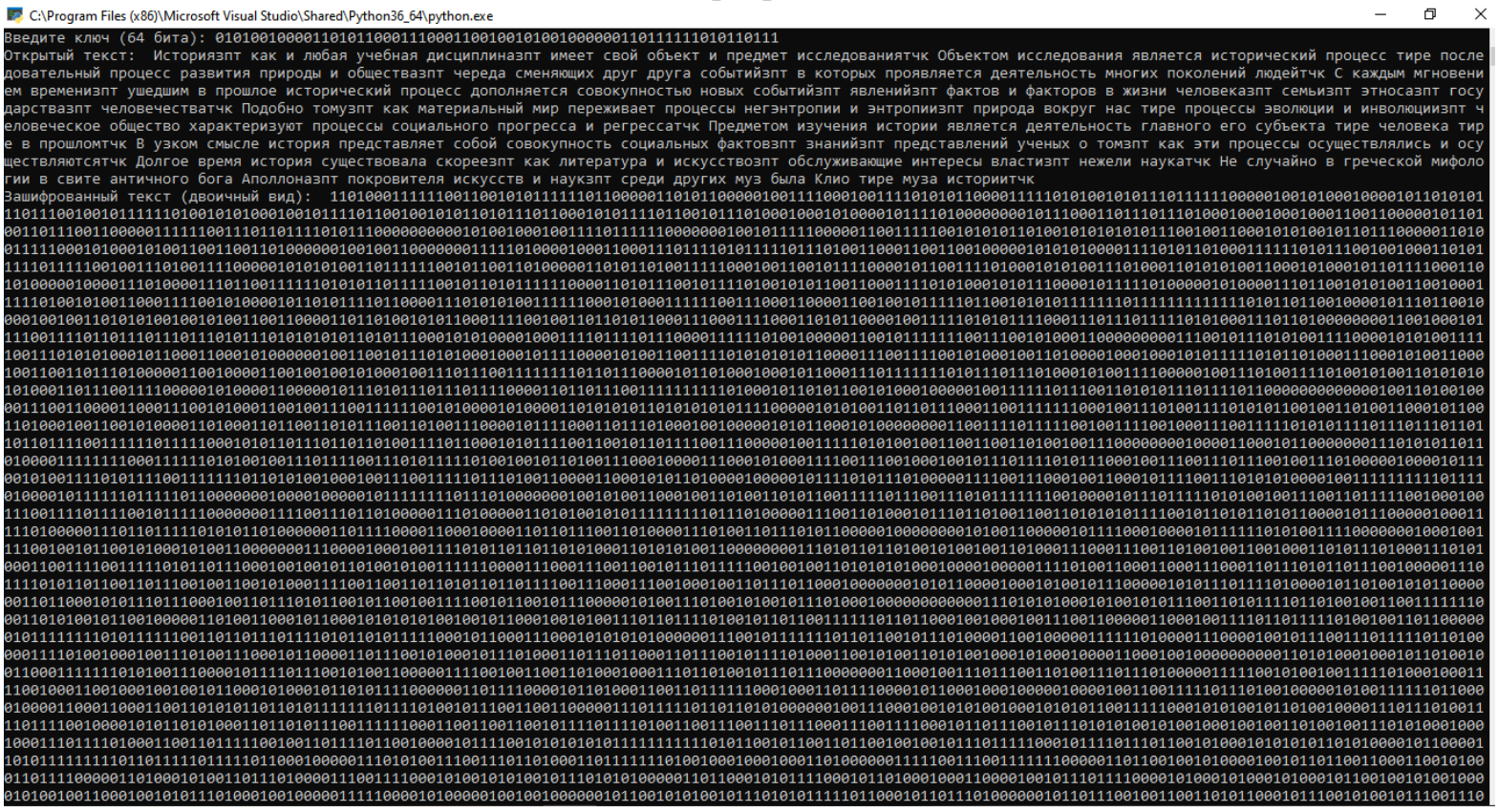


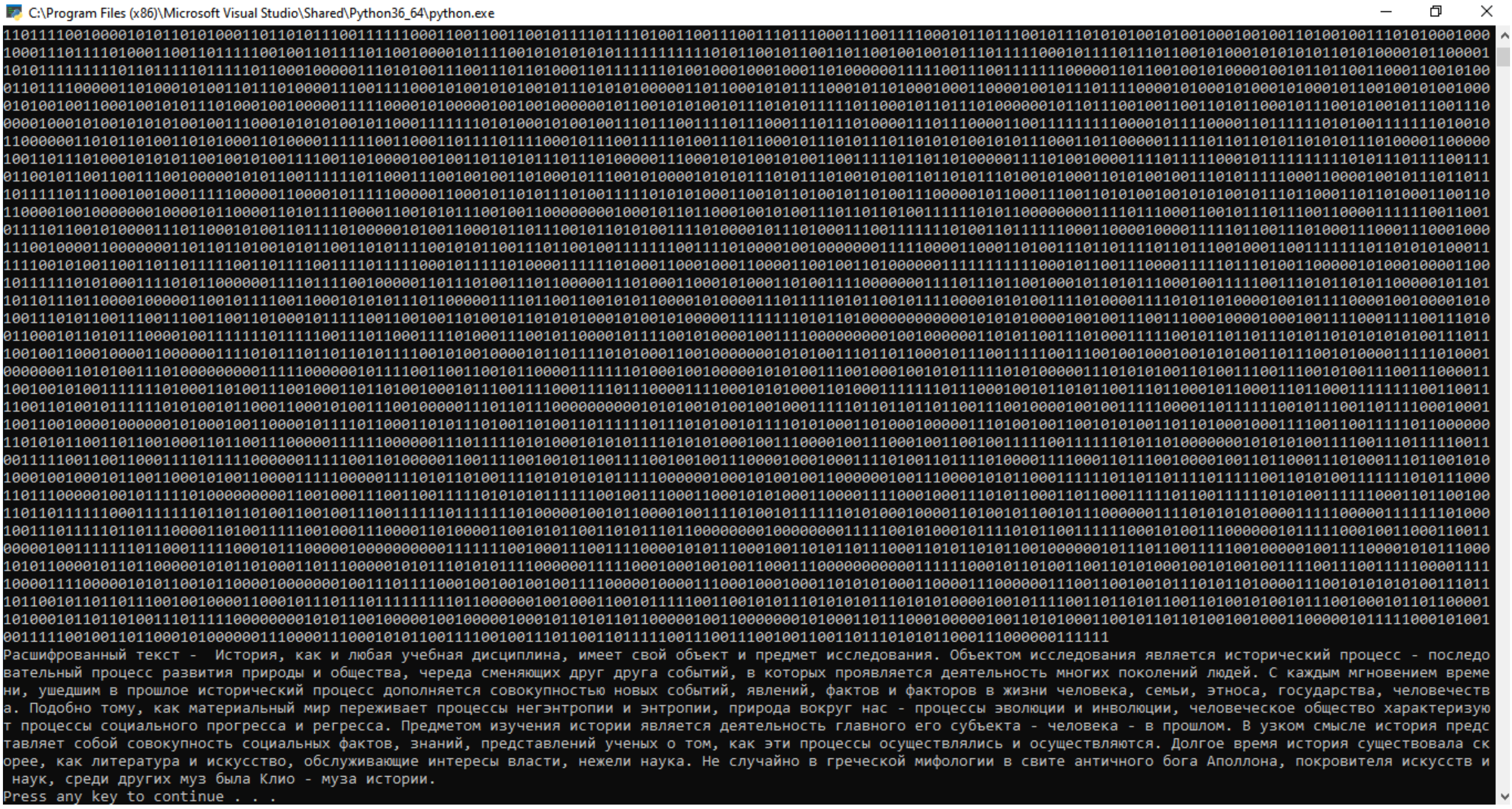
6. Работа с текстом не менее 1000 знаков

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



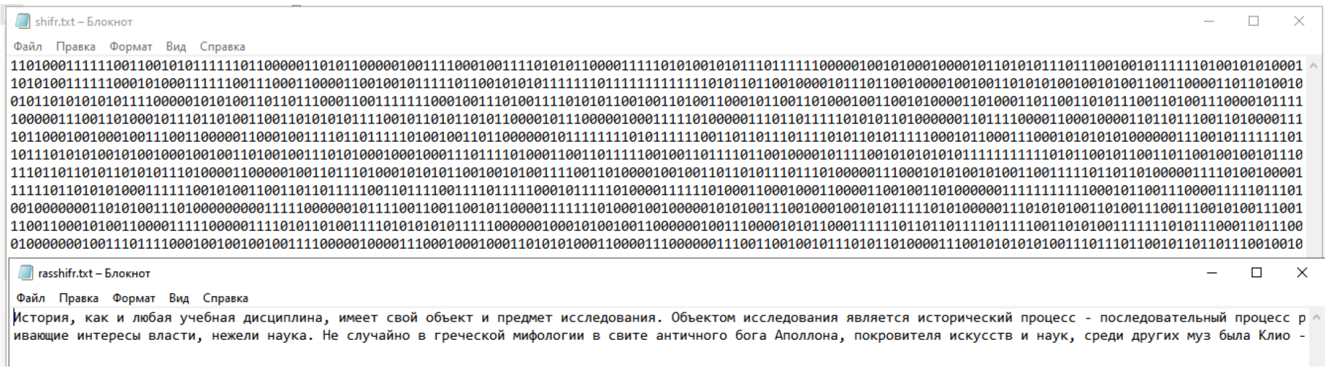
Так выглядит окно выполнения программы. Ключ вводит пользователь.





Полный ключ:
0101001000011010110001110001100100101001000000110111111010110111

После выполнения программы в файл “shifr.txt” записывается зашифрованный текст, а в файл “rasshifr.txt” расшифрованный текст.



Полный зашифрованный текст:

1101000111111001100101011111101100000110101100000100111100010011110101011000011111010100101011101111111000000100101000100001011010101110111001001011111101001010100010010111101100100101011010111011000101011110110010111010001000101000010111101000000000101110001101110111010001000100010001100110000010110100110111001100000111111001110110111101011100000000001010010001001111011111100000000100101111100000110011111001010101101001010101010111001001100010101001011011100000110100111110001010001010011001100110100000001001001100000000111110100001000110001110111101011111011101001100011001100100000101010100000

11110101101000111111010111001001000110101111101111100100111010011
10000010101010011011111100101100110100000110101101001111100010011 0
01011110000101100111101000101010011101000110101010011000101000101 1
01111000110101000001000011101000011101100111111010101101111100101 1
01011111100001101011100101111010010101100110001111010100010101110 0
00101111010000010100001110110010101001100100011111010010100110001
11100101000010110101111011000011101010100111111000101000111111001 1
10001100001100100101111101100101010111111101111111111111010110110 0
10000101110110010000100100110101010010010100110011000011011010010 1
01100011110010011011010110001110001111000110101100001001111101010 1
11100011101110111110101000111011010000000011001000101111001111011 0
11101110111010111010101011010111000101010000100011110111011110001 1
01111110100100000110010111111100111001010001100000000011100101110 1
01001111000010101001111100111010101000101100011000101000000100110 0
10111010100010001011110000101001100111101010101011000011100111100 1
01000100110100001000100010101111101011010001110001010011000100110 0
11011101000001100100001100100100101000100111011100111111110110111 0
00010110100010001011000111011111110101110111010001010011110000010 0
11101001111010010100110101010101000110111001111000001010000110000 0
10111010111011101111000011011011100111111111010001011010110010100
01000001001111110111001101010111011110110000000000000100110100100 0
01110011000011000111001010001100100111001111110010100001010000110 1
01010110101010101111000001010100110110111000110011111110001001110 1
00111101010110010011010011000101100110100010011001010000110100011 0
11001101011100110100111000010111100011011101000100100000101011000 1
01000000011001111011111001001111001000111001111101010111011101110 1
01101101101111001111110111110001010110111011011010011110110001010 1
11100110010110111001110000010011111010100100110011001101001001110
00000001000011000101100000001110101011011010000111111110001111110 1
01001001110111100111010111110100100101101001110001000011100010100 0
11110011100100010010111011110101110001001110011101110010011101000 0
01000010111001010011110101111001111111011010100100010011100111110 1
11010011000011000101011010000100000101111010111010000011110011100 0
10011000101111001110101010000100111111111011110100001011111101111 1
10110000000100001000001011111111011101000000010010100110001001101 0
01101011001111101110011101011111110010000101110111110101001001110 0
11011111001000100111001111011110010111110000000111100111011010000 0
11101000001101010010101111111111011101000001110011010001011101101 0
11001101010101111001011010110101100001011100000100011111010000011 1
01101111101010110100000011011110000110001000011011011100110100001 1
10100110111010110000010000000010100110000010111100010000101111110 1
01001111000000010001001111001001011001010001010011000000011100001 0
00100111101011011011010100011010101001100000000111010110110100101 0
01001101000111000111001101001001100100011010111010001110101000110 0
11110011111010110111000100100101101001010011111100001110001110011 0

010111011111001001001101010101000100001000001111010011000110001110
001101110101101110010000011101111010110110011011100100110010100011

0011111000001100001011111000001100010110101110100111110101010001100
0101101001011010011100000101100011100110101001001010100101110110000
1101101000110011011000010010000000100001011000011010111100001100100
1011100100110000000010001011011000100101001110110110100111111010110
0000000011110111000110010111011100110000111111001100101111011001010
0000111011000101001101111010000010100110001011011100101101010011110
0100001011101000111001111111010011011111100011000010000111110110010
1101000111000111000100011100100001100000001101101101001010110011010
0111100101011001110110010011111110011110100001001000000001111000010
1000110100111011011110110111001000110011111110110101010001111110010
0100110011011011111001101111001111011111000101111101000011111101000
0110001000110000110010011010000000111111111110001011001110000111110
1110100110000001010001000011001011111101010001111010110000000111101
1100100000110111010011101100000011101000110001010001101001111000000
0111101110110010001011010111000100111110011101011010110000001011011
0110111011000010000011001011110011000101010111011000000111101100110
0101011000001010000111011111010110010111100001010100111101000011110
1011010000100101111000010010000101010011101011001110011100110011010
0001011111001100100110100101101010100010100101000001111111101011010
0000000000001010101000010010011100111000100001000100111100011110010
1101001100010110101110000100111111101111110011101100011110100011100
1011000010111100101000010011110000000000100100000011010110011101000
1111100101101101110101101010101010011101110010011000100001100000010
1110101110110110101111001010010000101101111010100011001000000010100
1001110110110001011100111110011100100100010010101001101110010100000
1111101000100000001101010011101000000000011111000000101111001100110
0101100001111111010001001000000101010011100100010010101111101010000
0111010101001101001110011100101001110011100001110010010100111111100
1000110100111001000110110100100010111001111000111101110000111100010
0101000110100011111110111000100101101011001110110001011000111011000
0111111110011001111001101001011111101010010110001100010100111001000
0001110110111000000000001010100101001001000111110110110110011100000
1000010010011111000011011111100101110011011110001000110011001000010
0000001010001001100001011110110001101011101001101001101111110111010
0100101111010100011010001000001110100100110010101001101101000100010
1110011001111101100000011010101100110110010001101100111000000111111
0000001110111111010100010101011110101010001001110000100111000100110
0100111110011111101011010000000101010100111100111011111001100111110
0011001100011110111111000000111110011010000001100111100100101100111
0010010011100001000100011110100110111101000011110001101110010000100
0110110001110100011101100101010001001000101100110001010011000011110
1000001111010110100111101010101011111000000010001010010011000000100
1110000101011000111111011011011110111110011010100111111101011100010
1011100000100101111101000000000110010001110011001111101010101111110
0010011100011000101010001100001111000100011101011000110110001111100

1100111111010100111111000110110010011011011111100011111111011011010
0110010011100111111011111110100000100101100001001111010010111111101
0100010000110100101100101110000001111010101010000111110000011111111
0100010011101111101101110000110100111110010001110000110100001100101
1011001101011101100000000100000000011111001010001011110101100111111
0001010011100000010111110001001100011001100000100111111101100011111
1000101110000010000000000111111100100011100111100001010111000100111
0101101110001101011010110010000001011101100111110010000010011110000
0101011100010101100001011011000001010110100011011100000101011101010
0111100000011111000100010010011000111000000000001111110001011010011
1001101010001001010010011110011100111110000111110000111100000101010
1001011000010000000100111011110001001001001001111000001000011100010
0001000110101010001100001110000001110011001001011101011010000111000
1010101010011101110110010110110111001001000011000101110111011111111
1101100000010010001100101111100110010101110101010111010101000010010
0111100110110101100110100101001011100100010110110000110100010110110
0100111011111000000001010110010000010010000001000101101011011000001
0011000000010100011011100010000010011010100011001011011010010010000
1100000101111100010100100111110010011011000101000000111000011100010
0101100111100100111011001101111100111001110010011001101110101011000
0111000000111111

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”, “shifr.txt”, “rasshifr.txt”.

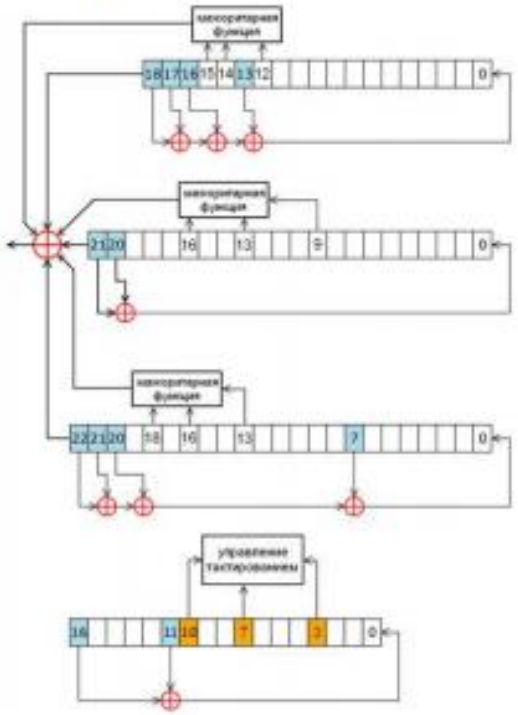
A5/2.

1. Описание шифра.

A5 — это поточный алгоритм шифрования, используемый для обеспечения конфиденциальности передаваемых данных между телефоном и базовой станцией в европейской системе мобильной цифровой связи GSM (*Groupe Special Mobile*).

2. Алгоритм шифра.

АЛГОРИТМ ШИФРОВАНИЯ A5/2



- добавлен регистр R4 длиной 17 бит

Многочлен обратной связи для R4:

$$X^{17} + X^{12} + 1 ,$$

Управление тактированием осуществляет R4:

биты синхронизации: 3, 7, 10

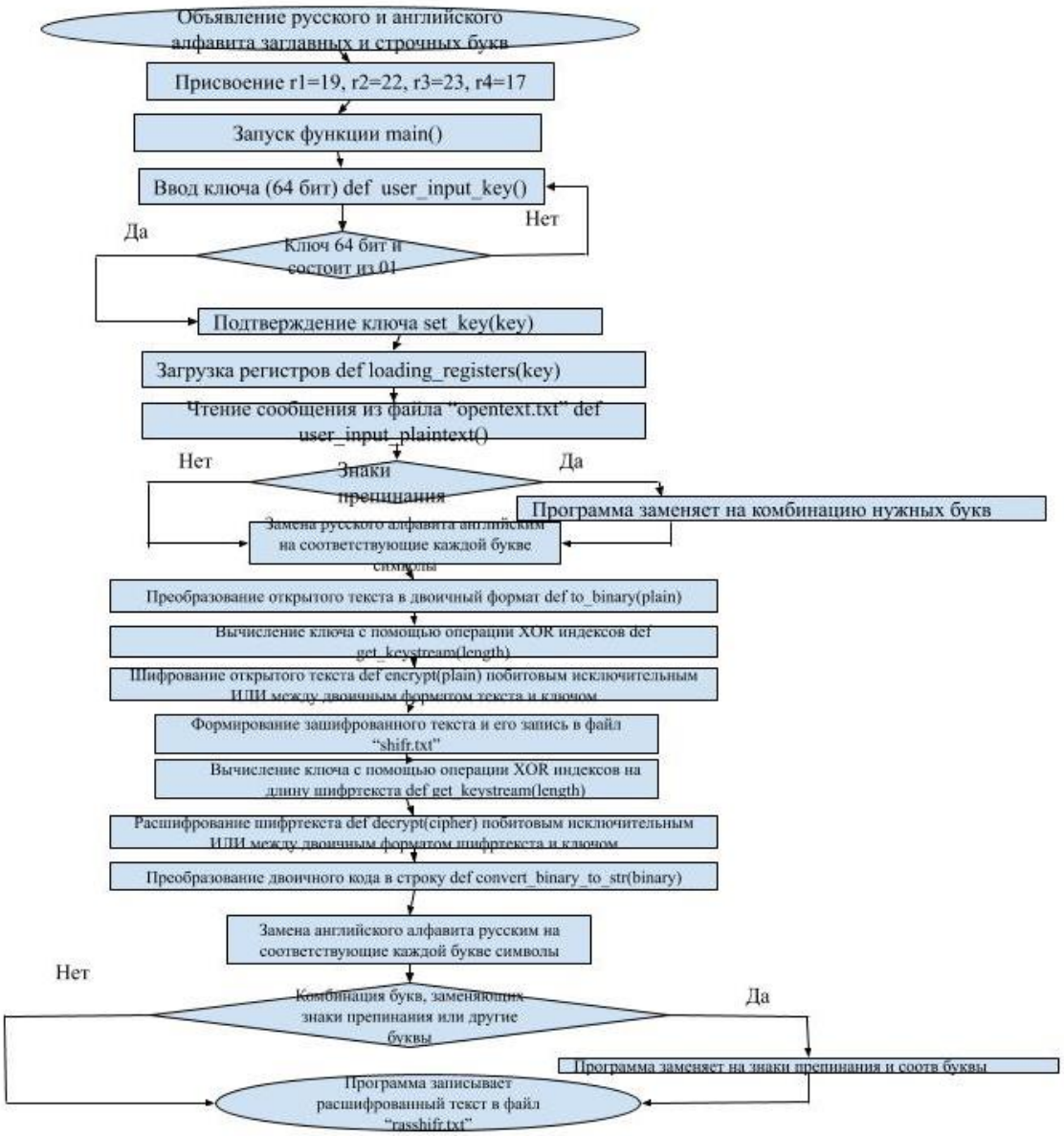
мажоритарная функция F = x&y|x&z|y&z (равна большинству), где & — булево AND, | - булево OR, а x, y и z — биты синхронизации R4(3), R4(7) и R4(10) соответственно,

R1 сдвигается если R4(10) = F
R2 сдвигается если R4(3) = F
R3 сдвигается если R4(7) = F

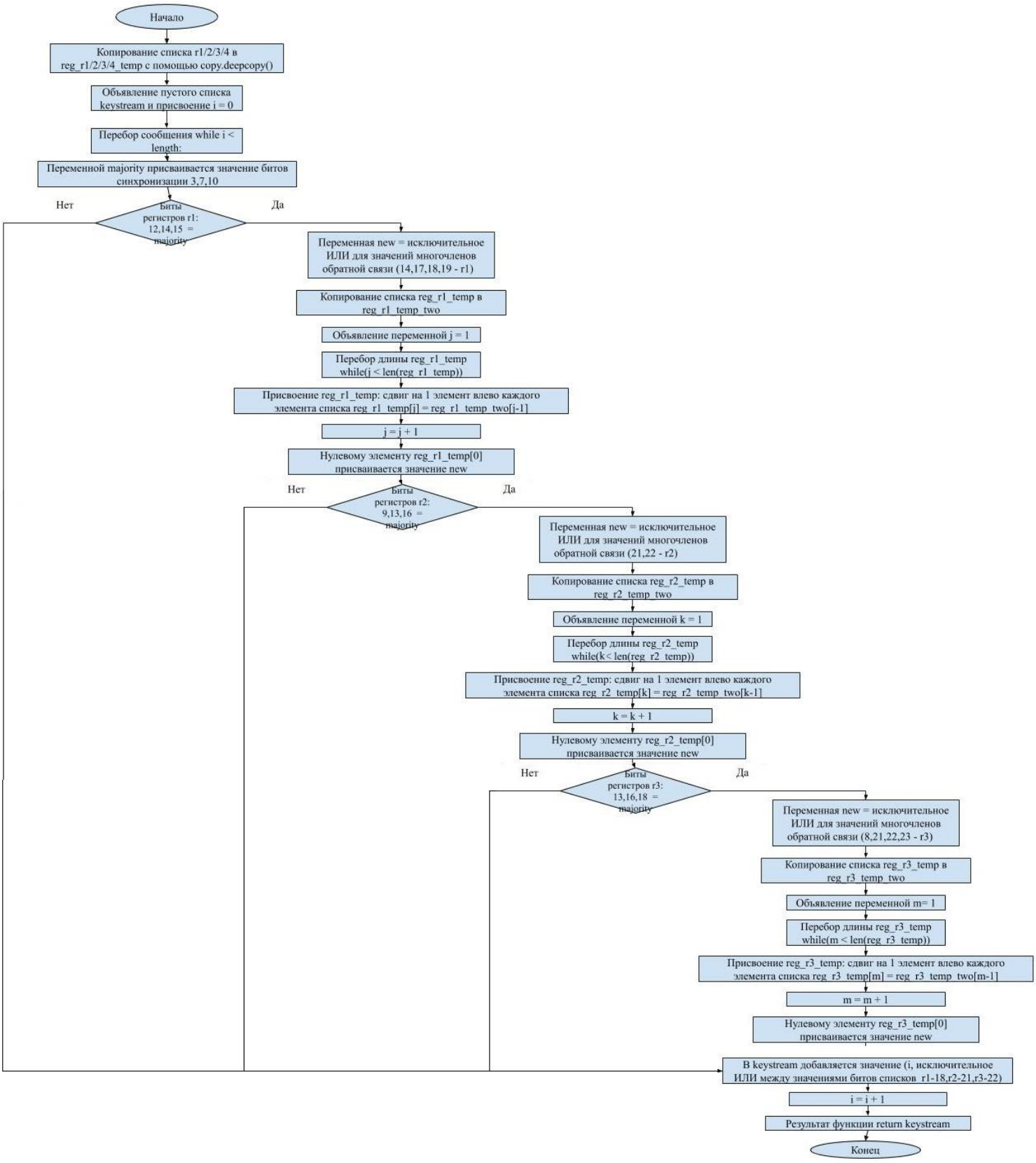
Выходной бит системы —XOR над старшими битами регистров и мажоритарных функций от определённых битов регистров:

R1 — 12, 14, 15,
R2 — 9, 13, 16,
R3 — 13, 16, 18.

3. Блок-схема программы



def get_keystream() - это функция вычисления ключа с помощью операции XOR индексов.



4. Код программы

```
import re
import copy
import sys
```

```
rus = [ 'а','б','в','г','д','е','ж', 'з','и','й','к','л','м','н','о','п','р','с','т','у','ф','х','ц', 'ч', 'ш', 'щ', 'ъ','ы', 'ь','э','ю','я',' ']
eng = [
'a','b','v','g','d','e','ty','z','i','y','k','l','m','n','o','p','r','s','t','u','f','py','c','ry','ly','dy','sy','xy','fy','j','q','w','{']

rus_big = []
eng_big = []
for i in rus:
    rus_big.append(i.upper())

for i in eng:
    eng_big.append(i.upper())

r1 = 19
r2 = 22
r3 = 23
r4 = 17

key_one = ""
reg_r1 = []
reg_r2 = []
reg_r3 = []
reg_r4 = []

def loading_registers(key): # загрузка регистров
    i = 0
    while(i < r1):
        reg_r1.insert(i, int(key[i]))
        i = i + 1
    j = 0
    p = r1
    while(j < r2):
        reg_r2.insert(j,int(key[p]))
        p = p + 1
        j = j + 1
    r = r2 + r1
    k = 0
    while(k < r3):
        reg_r3.insert(k,int(key[r]))
        k = k + 1
        r = r + 1
    i = 0
    while(i < r4):
        reg_r4.insert(i, int(key[i]))
```

```
        i = i + 1

def set_key(key): # устанавливает ключ и загружает регистры, если они содержат 0 и 1 и длина ключа 64 бита
    if(len(key) == 64 and re.match("^([01])+", key)):
        key_one=key
        loading_registers(key)
        return True
    return False

def get_key(): # получен ключ
    return key_one

def to_binary(plain): # преобразование обычного текста в двоичный формат
    s = ""
    i = 0
    for i in plain:
        binary = str(' '.join(format(ord(x), 'b') for x in i))
        j = len(binary)
        while(j < 8):
            binary = "0" + binary
            s = s + binary
            j = j + 1
    binary_values = []
    k = 0
    while(k < len(s)):
        binary_values.insert(k, int(s[k]))
        k = k + 1
    return binary_values


def get_majority(r1,r2,r3): # получение большинства
          if (r1 + r2 + r3 > 1):
                    return 1
          else:
                    return 0

def get_keystream(length): # вычисление ключа с помощью операции XOR индексов
    reg_r1_temp = copy.deepcopy(reg_r1)
    reg_r2_temp = copy.deepcopy(reg_r2)
    reg_r3_temp = copy.deepcopy(reg_r3)
    reg_r4_temp = copy.deepcopy(reg_r4)
    keystream = []
    i = 0
```

```
    while i < length:
        majority = get_majority(reg_r4_temp[3], reg_r4_temp[7], reg_r4_temp[10]) #
биты синхронизации
        if get_majority(reg_r1_temp[12], reg_r1_temp[14], reg_r1_temp[15]) ==
majority: # определённые биты регистров
            new = reg_r1_temp[13] ^ reg_r1_temp[16] ^ reg_r1_temp[17] ^
reg_r1_temp[18]
            reg_r1_temp_two = copy.deepcopy(reg_r1_temp)
            j = 1
            while(j < len(reg_r1_temp)):
                reg_r1_temp[j] = reg_r1_temp_two[j-1]
                j = j + 1
            reg_r1_temp[0] = new

        if get_majority(reg_r2_temp[9], reg_r2_temp[13], reg_r2_temp[16]) ==
majority: # определённые биты регистров
            new_one = reg_r2_temp[20] ^ reg_r2_temp[21]
            reg_r2_temp_two = copy.deepcopy(reg_r2_temp)
            k = 1
            while(k < len(reg_r2_temp)):
                reg_r2_temp[k] = reg_r2_temp_two[k-1]
                k = k + 1
            reg_r2_temp[0] = new_one

        if get_majority(reg_r3_temp[13], reg_r3_temp[16], reg_r3_temp[18]) ==
majority:  # определённые биты регистров
            new_two = reg_r3_temp[7] ^ reg_r3_temp[20] ^ reg_r3_temp[21] ^
reg_r3_temp[22]
            reg_r3_temp_two = copy.deepcopy(reg_r3_temp)
            m = 1
            while(m < len(reg_r3_temp)):
                reg_r3_temp[m] = reg_r3_temp_two[m-1]
                m = m + 1
            reg_r3_temp[0] = new_two

        keystream.insert(i, reg_r1_temp[18] ^ reg_r2_temp[21] ^ reg_r3_temp[22])
        i = i + 1
    return keystream

def convert_binary_to_str(binary): # преобразует двоичный код в строку
    s = ""
    length = len(binary) - 8
    i = 0
    while(i <= length):
        s = s + chr(int(binary[i:i+8], 2))
```

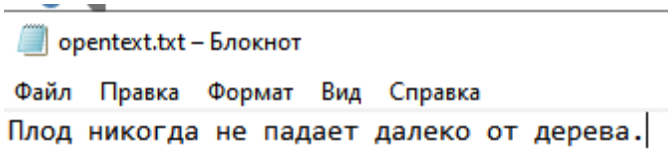
```
        i = i + 8
    return str(s)

def encrypt(plain): # шифрование (открытый текст преобразуется в двочный, получаем полный ключ после ввода длины двоичного текста и получаем значение XOR полного ключа и двоичного текста)
    s = ""
    binary = to_binary(plain)
    keystream = get_keystream(len(binary))
    i = 0
    while(i < len(binary)):
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return s

def decrypt(cipher): # дешифрование (принимает шифр, получает полный ключ от его длины, XOR шифра с ключевым потоком и преобразуется в строку)
    s = ""
    binary = []
    keystream = get_keystream(len(cipher))
    i = 0
    while(i < len(cipher)):
        binary.insert(i,int(cipher[i]))
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return convert_binary_to_str(str(s))

def user_input_key(): # ввод ключа
        tha_key = str(input('Введите ключ (64 бита): '))
        if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
                return tha_key
        else:
                while(len(tha_key) != 64 or re.match("^([01])+", tha_key)):
                        if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
                                return tha_key
                        tha_key = str(input('Введите ключ (64 бита): '))
        return tha_key


def user_input_plaintext(): # чтение открытого текста из файла
    f = open(r"opentext.txt", "rt", encoding='utf-8')
    someIn = f.read()
    someIn = someIn.replace('.', 'тчк') # Если в сообщении попадется точка, она заменется на тчк
```

```
    someIn = someIn.replace(',', 'зпт') # Если в сообщении попадется запятая, она заменется на зпт
    someIn = someIn.replace('-', 'тире') # Если в сообщении попадется тире, символ заменется на тире
    #someIn = str(input('Введите текст для зашифрования: '))
    return someIn

def user_input_ciphertext(): # чтение шифртекста из файла
    f = open(r'shifr.txt', 'rt', encoding='utf-8')
    ciphertext = f.read()
    return ciphertext


def tha_main(): # главная функция записи в файл шифртекста и дешифрованного текста
    key = str(user_input_key())
    set_key(key)
    if True:
        plaintext = str(user_input_plaintext())
        print('Открытый текст: ', plaintext)
        plaintext2 = ''
        for i in plaintext:
            try:
                if i.isupper():
                    plaintext2 += eng_big[rus_big.index(i)]
                else:
                    plaintext2 += eng[rus.index(i)]
            except:
                plaintext2 += i
        f = open('shifr.txt', 'wt', encoding='utf-8')
        sh = encrypt(plaintext2)
        f.writelines(sh)
        f.close()
        print('Зашифрованный текст (двоичный вид): ', sh)
        plaintext = ''
        file = open('rasshifr.txt', 'wt', encoding='utf-8')
        rasshifr = decrypt(sh)
        for i in rasshifr:
            try:
                if i.isupper():
                    plaintext += rus_big[eng_big.index(i)]
                else:
                    plaintext += rus[eng.index(i)]
            except:
                plaintext += i
```

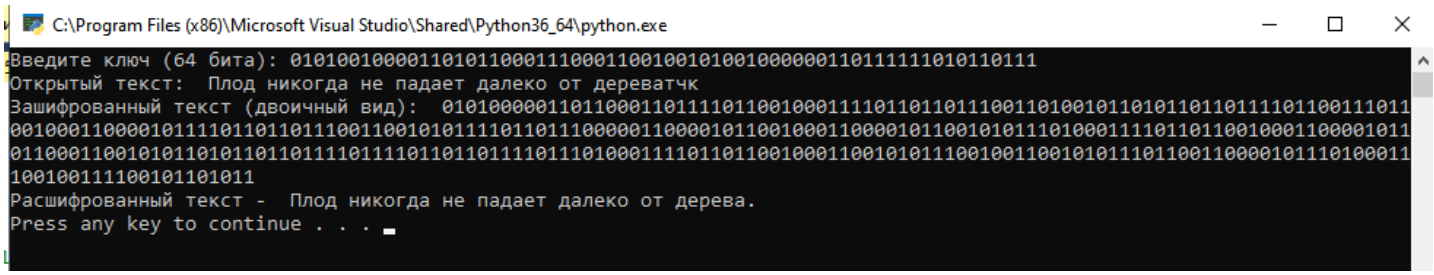
```
    plaintext = plaintext.replace('рй', 'ч')
    plaintext = plaintext.replace('РЙ', 'Ч')
    plaintext = plaintext.replace('тй', 'ж')
    plaintext = plaintext.replace('пй', 'х')
    plaintext = plaintext.replace('ПЙ', 'Х')
    plaintext = plaintext.replace('ТЙ', 'Ж')
    plaintext = plaintext.replace('хй', 'ы')
    plaintext = plaintext.replace('лй', 'ш')
    plaintext = plaintext.replace('ЛЙ', 'Ш')
    plaintext = plaintext.replace('дй', 'щ')
    plaintext = plaintext.replace('ДЙ', 'Щ')
    plaintext = plaintext.replace('сй', 'ъ')
    plaintext = plaintext.replace('СЙ', 'Ъ')
    plaintext = plaintext.replace('фй', 'ь')
    plaintext = plaintext.replace('ФЙ', 'Ь')
    plaintext = plaintext.replace('тчк', '.')
    plaintext = plaintext.replace('зпт', ',')
    plaintext = plaintext.replace('тире', '-')
    file.writelines(plaintext)
    file.close()
    print('Расшифрованный текст - ', plaintext)

#Example of 64-bit key:
0101001000011010110001110001100100101001000000110111111010110111
#Плод никогда не падает далеко от дерева.
tha_main()
```

5. Тестирование

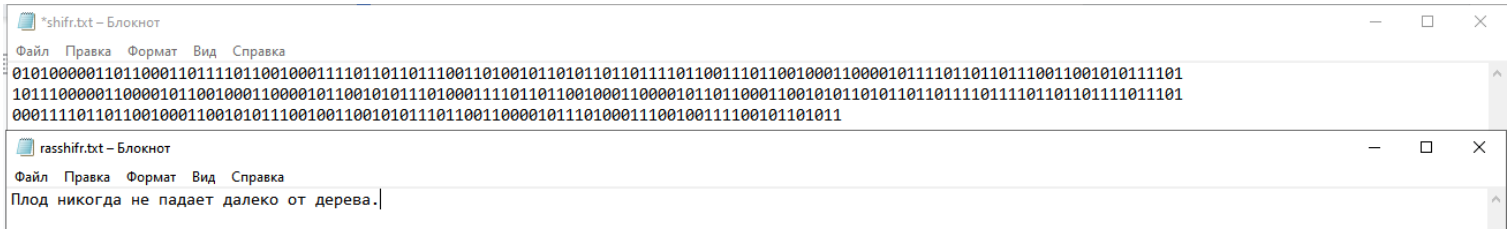
Перед началом работы программы в файл “opentext.txt” записываем исходный текст.



Так выглядит окно выполнения программы. Ключ вводит пользователь.

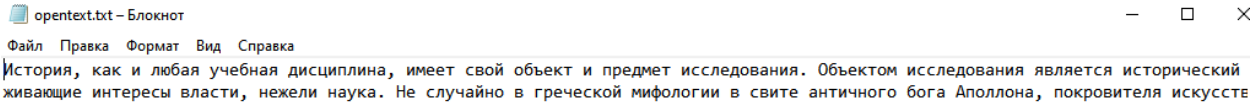


После выполнения программы в файл “shifr.txt” записывается зашифрованный текст, а в файл “rasshifr.txt” расшифрованный текст.

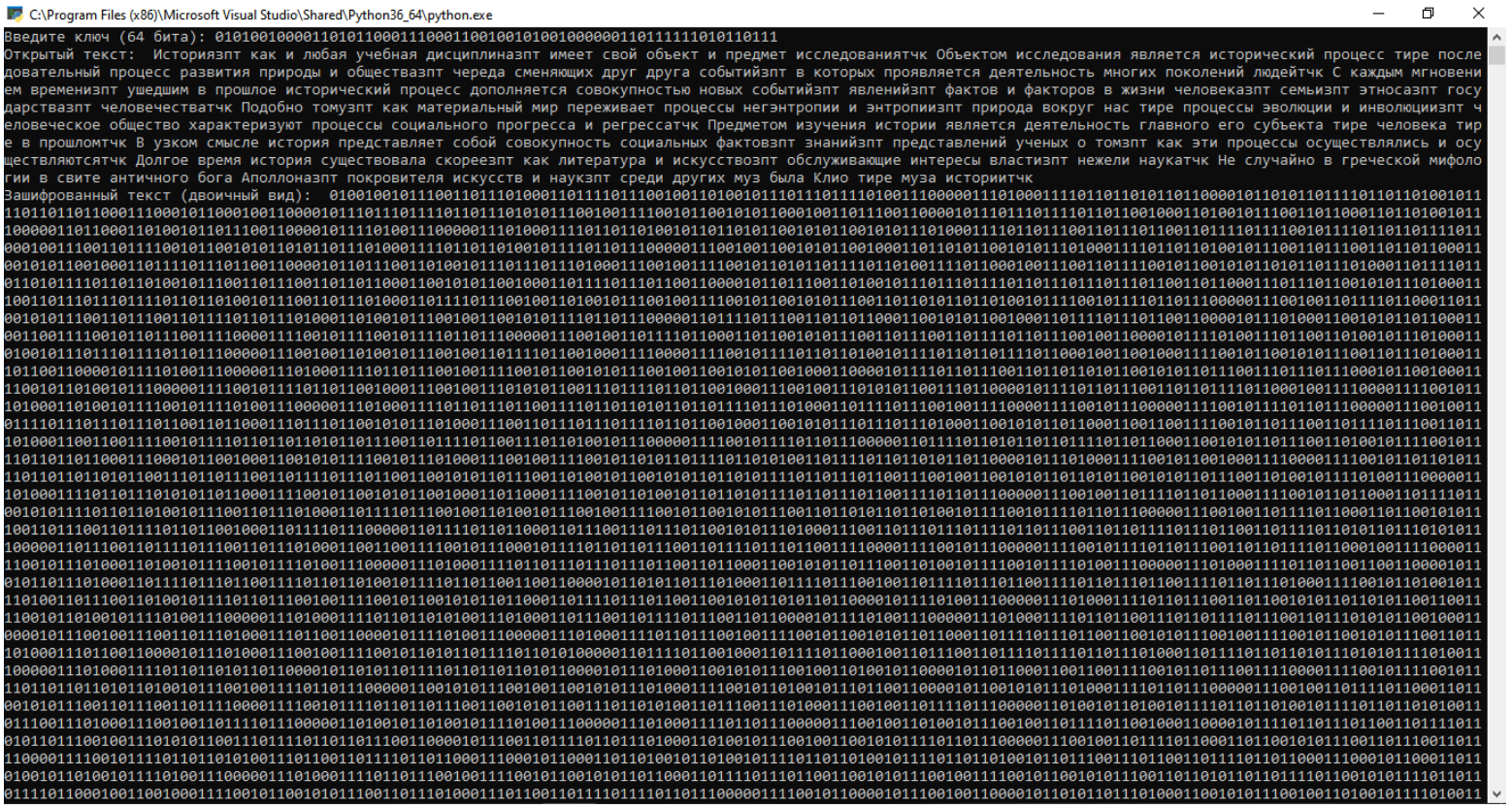


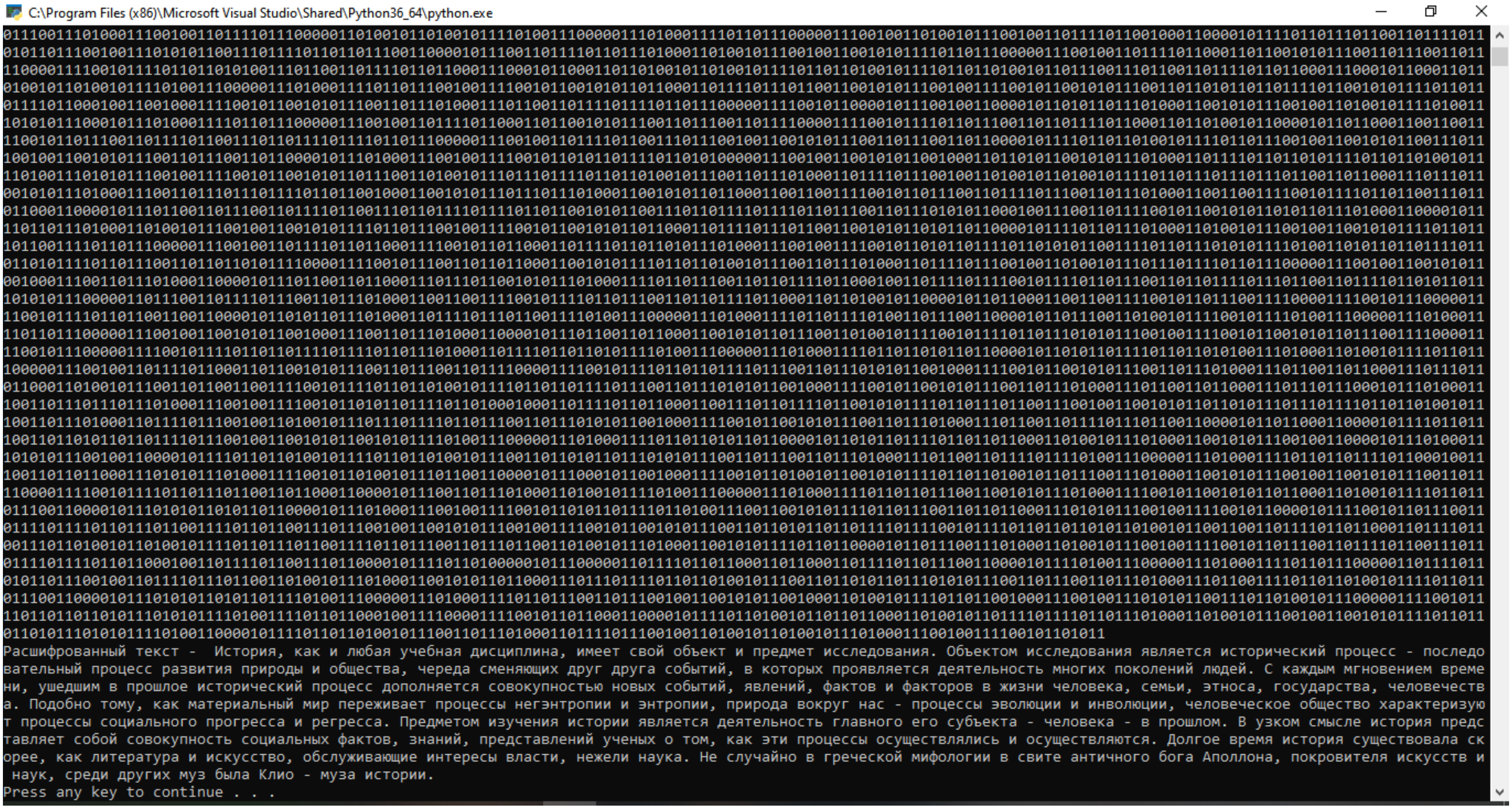
6. Работа с текстом не менее 1000 знаков

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



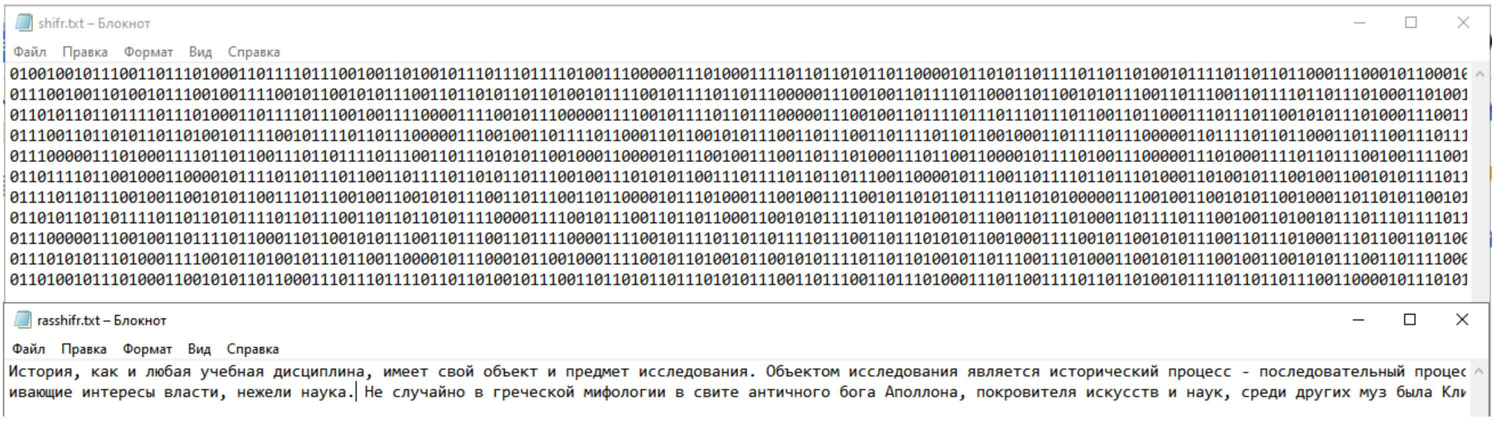
Так выглядит окно выполнения программы. Ключ вводит пользователь.





Полный ключ:
0101001000011010110001110001100100101001000000110111111010110111

После выполнения программы в файл “shifr.txt” записывается зашифрованный текст, а в файл “rasshifr.txt” расшифрованный текст.



Полный зашифрованный текст:

0100100101110011011101000110111101110010011010010111011011101001
1100000111010001111011011010110110000101101011011110110110100101111
1011011011000111000101100010011000010111011101111011011101010111001
1001111001011001010110001001101110011000010111011101111011011001001
0110100101110011011000110110100101110000011011000110100101101110011
1000010111101001110000001110100011110110110100101101101011001010110
0101011101000111101101110011011101100110111101111001011110110110111
1101100010011100110111100101100101011010110111010001111011011010011

01111011011100000111001001100101011001000110110101100101011101000
11101101101001011100110111001101101100011001010110010001101111011
1
01100110000101101110011010010111011101110100011100100111001011010
11011110110100111101100010011100110111100101100101011010110111010
0
01101111011011010111101101101001011100110111001101101100011001010
1
10010001101111011101100110000101101110011010010111011101111011011
1
01110111011001101100011101110110010101110100011100110111011011110
11011010010111001101110100011011110111001001101001011100100111100
1
01100101011100110110101101101001011110010111101101110000001110010
01
10111101100011011001010111001101110011011110110111010001101001011
1
00100110010101111011011100000110111101110011011011000110010101100
1
00011011110111011001100000101110100011001010110110001100110011110
01
01101110011110000011110010111100101111011011100000011100100110111
101
10001101100101011100110111001101111011011100100110000101111010011
1
01100110100101110100011010010111011101111011011100000011100100110
10
01011100100110111101100100011110000111100101111011011010010111101
1
01101111011000100110010001111001011001010111001101110100011101100
1
10000101111010011100000011101000111101101110010011110010110010101
11
00100110010101100100011000010111101101110011011011010110010101101
1
10011101110111000101100100011110010110100101110000001111001011110
11
01100100011100100111010101100111011110110110010001110010011101010
1
10011101100000101111011011100110110111101100010011110000011110010
111
01000110100101111001011110100111000000111010001111011011101100111
10
11011010110110111101110100011011110111001001111000001111001011100
00
01111001011110110111000000111001001101111011101110111011001101100
01
11011101100101011101000111001101110111011110110110010001100101011
1
01110111010001100101011011000110011001111001011011100110111101110
0
11011101000110011001111001011110110110101101110011011110110011
1
01101001011100000111100101111011011100000110111101101011011011110
1
10110001100101011011100110100101111001011110110110110001110001011
0
01000110010101111001011101000111001001111001011010110111101101010
0
11011110110110101101100001011101000111100101100100011110000111100
1
01101101011110110110101100111011011100110111101110110011001010
1
10111001101001011001010110110101111011011101100111001001100101011
0
11010110010101101110011010010111101001110000001110100011110110111
01
01011011000111100101100101011001000110110001111001011010010110110
1
01111011011101100111101101110000001110010011011110110110001110010
1
10110001101111011001010111101101101001011100110111010001101111011
1
00100110100101110010011110010110010101110011011010110110100101111
0
01011110110111000000111001001101111011000110110010101110011011100
11
01111011011001000110111101110000001101111011011000110111001110111
01
10010101110100011100110111011101111011011100110110111101110110011
0
11110110101101110101011100000011011100110111101110011011101000110
01
10011110010111000101111011011011100110111101110110011110000011110
01
01110000011110010111101101110011011011110110001001111000001111001
01

1101000110100101111001011110100111000001110100011110110111011101111
0110011011000110010101101110011010010111100101111010011100000111101
0001111011011001100110000101101011011101000110111101110110011110111
0110100101111011011001100110000101101011011101000110111101110011001
1011110111011001111011011101100111101101110100011110010110100101111
1010011011100110100101111011011100100111100101100101011011000110111
1101110110011001010110101101100001011110100111000001110100011110111
0111001101100101011010101100110011110010110100101111010011100000011
1101000111101101101010011101000110111001101111011100110110000101111
1010011100000111010001111011011001110110111101110011011101010110011
0001100000101110010011100110111010001110110011000010111101001110000
0111010001111011011100100111100101100101011011000110111101110110011
1001010111001001111001011001010111001101110100011101100110000101111
0100011100100111100101101011011110110101000001101111011001000110111
1101100010011011100110111101111011011101000110111101101101011101011
0111101001110000011101000111101101101011011000010110101101111011011
1011010110000101110100011001010111001001101001011000010110110001101
0110011110010110111001111000011110010111100101111011011011010110101
0101110010011110110111000000110010101110010011001010111010001111001
0110100101110110011000010110010101110100011110110111000000111001001
1011110110001101100101011100110111001101111000011110010111101101101
1110011001010110011101101010011011100111010001110010011011110111001
0001101001011010010111101101101001011110110110101001101110011101001
0111001001101111011100000110100101101001011110100111000000111010001
1110110111000000111001001101001011100100110111101100100011000010111
1011011101100110111101101011011100100111010101100111011110110110111
1001100000101110011011110110111010001101001011100100110010101111011
0111000000111001001101111011000110110010101110011011100110111100001
1110010111101101101010011101100110111101101100011100010110001101101
1001011010010111101101101001011110110110100101101110011101100110111
1101101100011100010110001101101001011010010111101001110000001110100
0111101101110010011110010110010101101100011011110111011001100101011
1100100111100101100101011100110110101101101111011001010111101101101
1111011000100110010001111001011001010111001101110100011101100110111
1101111011011100000011110010110000101110010011000010110101101110100
0110010101110010011010010111101001110101011100010111010001111011011
1100000111001001101111011000110110010101110011011100110111100001111
1001011110110111001101101111011000110110100101100001011011000110011
1001111001011011100110111101100111011011110111101101110000001110010
0110111101100111011100100110010101110011011100110110000101111011011
1010010111101101110010011001010110011101110010011001010111001101111
0011011000010111010001110010011110010110101101111011010100000111001
1001100101011001000110110101100101011101000110111101101101011110111
0110100101111010011101010111001001111001011001010110111001101001011
1101110111101101101001011100110111010001101111011100100110100101101

1001011110110111011101110110011011000111011101100101011101000111100
1101110111011110110110010001100101011101110111010001100101010110110
0110011001111001011011100110111101110011011101000110011001111001010
1110110110011101101100011000010111011001101110011011110110011101101
1111011110110110010101100111011011110111101101110011011101010110000
1001110011011110010110010101101011011101000110000101111011011101000
0110100101110010011001010111101101110010011110010110010101101100010
1011110111011001100101011010110110000101111011011101000110100101110
0010011001010111101101110110011110110111000001110010011011110110110
0001111001011011000110111101101101011101000111001001111001011010110
0111101101010110011110110111010101111010011010110110111101101101010
1110110111001101101101011110000111100101110011011011000110010101110
1011011010010111001101110100011011110111001001101001011101110111101
1101110000011100100110010101100100011100110111010001100001011101101
0110110001110111011001010111010001111011011100110110111101100010010
1011110111100101111011011100110110111101110110011011110110101101110
0101011100000110111001101111011100110111010001100110011110010111101
1101110011011011110110001101101001011000010110110001100110011110010
0110111001111000011110010111000001111001011110110110011001100001010
1010110111010001101111011101100111101001110000011101000111101101110
1010011011100110000101101110011010010111100101111010011100000111010
0001111011011100000111001001100101011001000111001101110100011000010
0111011001101100011001010110111001101001011110010111101101110101010
1100100111100101100101011011100111100001111001011100000111100101110
1011011011110111101101110100011011110110110101111010011100000111010
0001111011011010110110000101101011011110110110101001110100011010010
0111101101110000011100100110111101100011011001010111001101110011010
1110000111100101111011011011110111001101110101011001000111100101101
0101011100110111010001110110011011000111011101101100011010010111000
1101100110011110010111101101101001011110110110111101110011011101010
0110010001111001011001010111001101110100011101100110110001110111010
1100010111010001110011011101110111010001110010011110010110101101110
1011010001000110111101101100011001110110111101100101011110110111010
1001110010011001010110110101110111011110110110100101110011011101000
0110111101110010011010010111011101111011011100110111010101100100010
1110010110010101110011011101000111011001101111011101100110000101101
1100011000010111101101110011011010110110111101110010011001010110010
0101111010011100000111010001111011011010110110000101101011011110110
0110110001101001011101000110010101110010011000010111010001110101010
1100100110000101111011011010010111101101101001011100110110101101110
0101011100110111001101110100011101100110111101111010011100000111010
0001111011011011110110001001110011011011000111010101110100011110010
0110100101110110011000010111000101100100011110010110100101100101010
1110110110100101101110011101000110010101110010011001010111001101110
1000011110010111101101110110011011000110000101110011011101000110101

0101111010011100000111010001111011011011100110010101110100011110001
0110010101101100011010010111101101101110011000010111010101101011101
1000010111010001110010011110010110101101111011010011100110010101111
1011011100110110110001110101011100100111100101100001011110010110111
1001101111011110110111011001111011011001110111001001100101011100101
0111100101100101011100110110101101101111011110010111101101101101011
1010010110011001101111011011000110111101100111011010010110100101111
1011011101100111101101110011011101100110100101110100011001010111101
1101100001011011100111010001101001011100100111100101101110011011111
0110011101101111011110110110001001101111011001110110000101111011011
0000010111000001101111011011000110110001101111011011100110000101111
1010011100000111010001111011011100000110111101101011011100100110111
1101110110011010010111010001100101011011000111011101111011011010011
0111001101101011011101010111001101110011011101000111011001111011011
1010010111101101101110011000010111010101101011011110100111000001111
0100011110110111001101110010011001010110010001101001011110110110011
0001110010011101010110011101101001011100000111100101111011011011011
0111010101111010011110110110001001111000011110010110110001100001011
1110110100101101101100011010010110111101111011011101000110100101111
0010011001010111101101101011101010111101001100001011110110110101
0101110011011101000110111101110010011010010110100101110100011100101
0111100101101011

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”, “shifr.txt”, “rasshifr.txt”.