Цель работы:

Приобретение навыков исследования свойств стегоконтейнеров, разработки стегосистем и их применения для встраивания информации и сокрытия данных с помощью текстовых файлов.

Задача работы:

Программно реализовать 3 метода встраивания слова в текстовый файл: метод прямой замены символов, метод с использованием дополнительных пробелов, метод добавления(замены) специальных символов.

Теоретическая часть

Стеганография, использующая текстовые контейнеры для скрытия данных, называется текстовой. При скрытии информации используются допущения при расположении и количестве символов в тексте, не учитываемые при прочтении человеком и компьютерном анализе текстового файла. Это может быть дополнительное количество пробелов и знаков табуляции в разных частях строки, чередование некоторых не учитываемых служебных символов, больших и маленьких букв, букв из разных алфавитов, но похоже выглядящих. К методам текстовой стеганографии относят: форматирование, изменение порядка следования маркеров конца строки, метод хвостовых пробелов, метод знаков одинакового начертания и изменения кода пробела. [1]

Суть метода форматирования состоит в увеличении строки путем увеличения числа пробелов между словами, когда один пробел соответствует, например, биту 0, два пробела – биту 1. Однако, прямое его применение хотя и возможно, но на практике основным недостатком данного метода является оформление текста, которое становится неряшливым, что позволяет легко заподозрить в нем наличие сокрытого сообщения.

Метод изменения порядка следования маркеров конца строки CR/LF использует индифферентность подавляющего числа средств отображения текстовой информации к порядку следования символов перевода строки (CR) и возврата каретки (LF), ограничивающих строку текста. Традиционный порядок следования CR/LF соответствует 0, а инвертированный LF/CR означает 1.

Метод хвостовых пробелов предполагает дописывание в конце коротких строк (менее 225 символов; значение 225 выбрано достаточно произвольно) от 0 до 15 пробелов, кодирующих значение полубайта.

Метод знаков одинакового начертания предполагает подмену (бит 1) или отказ от такой подмены (бит 0) русского символа латинским того же начертания. [2]

Практическая часть

Согласно задачам данной лабораторной работы, необходимо программно реализовать механизм сокрытия слова (кириллица) в текстовом файле (кириллица). Для выполнения этих задач был выбрал язык программирования Python 3.8.

Метод №1. Метод прямой замены символов.

Метод заключается в прямой замене кириллических букв на соответствующие им визуально латинские буквы. Программа допускает выбор двух букв из представленного списка, первая из которых кодирует «0» скрываемого слова в двоичном виде, вторая кодирует «1» скрываемого слова в двоичном представлении.

Преобразование слова, состоящего из кириллических букв в двоичный код, осуществляется следующим способом: из кода символа первой буквы слова в десятичном виде (Unicode) отнимается число 1024 для уменьшения размера двоичного слова, затем полученное десятичное число преобразуется в двоичный код со стандартизированной размерностью в 7 бит, затем аналогичные действия производятся с последующими буквами слова, в итоге получается строка, состоящая из «0» и «1».

Демонстрация работы программы:

```
Не следует, однако забывать, что постоянное информационно-пропагандистское обеспечение нашей
деятельности требуют от нас анализа систем массового участия. Задача организации, в особенности же
постоянное информационно-пропагандистское обеспечение нашей деятельности требуют от нас анализа
новых предложений. Идейные соображения высшего порядка, а также постоянное
информационно-пропагандистское обеспечение нашей деятельности в значительной степени обуславливает
создание существенных финансовых и административных условий.
```

Рисунок 1. Метод №1, исходный текст.

```
Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): 1
Зашифровать введите 1, расшифровать введите 2: 1
Введите слово для встраивания в текст: Стегано
Введите 2 символа (A,B,E,K,M,H,O,P,C,T,X,a,e,o,p,c,y,x) вместо "01" для шифрования: ao

Заменено символов:      49  из  18969  исходных символов
Изменение размера:      35549  ->  35500       | - 49  байт
```

Рисунок 2. Метод №1, демонстрация работы программы при шифровании.

Слово было встроено в исходный текст, визуально различий нет.

```
Не следует, однако забывать, что постоянное информационно-пропагандистское обеспечение нашей
деятельности требуют от нас анализа систем массового участия. Задача организации, в особенности же
постоянное информационно-пропагандистское обеспечение нашей деятельности требуют от нас анализа
новых предложений. Идейные соображения высшего порядка, а также постоянное
информационно-пропагандистское обеспечение нашей деятельности в значительной степени обуславливает
создание существенных финансовых и административных условий.
```

Рисунок 3. Метод №1, текст после встраивания слова.

Произведем обратную операцию, т.е. дешифрование, для того, чтобы получить встроенное слово из зашифрованного текста.

```
Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): 1
Зашифровать введите 1, расшифровать введите 2: 2
Введите 2 символа (A,B,E,K,M,H,O,P,C,T,X,a,e,o,p,c,y,x) вместо "01" для дешифрования: ao
Расшифрованное слово: Стегано
```

Рисунок 4. Метод №1, демонстрация работы программы при дешифровании

Метод №2. Метод с использованием дополнительных пробелов.

Метод заключается в прямой замене пробела, стоящего за символом окончания предложения (.?!), на два пробела для кодирования «1» скрываемого слова в двоичном виде, либо пропуск замены для кодирования «0» скрываемого слова.

Преобразование слова, состоящего из кириллических букв в двоичный код, осуществляется следующим способом: из кода символа первой буквы слова в десятичном виде (Unicode) отнимается число 1024 для уменьшения размера двоичного слова, затем полученное десятичное число преобразуется в двоичный код со стандартизированной размерностью в 7 бит, затем аналогичные действия производятся с последующими буквами слова, в итоге получается строка, состоящая из «0» и «1».

Демонстрация работы программы:

```
Равным образом новая модель организационной деятельности представляет собой интересный эксперимент
проверки новых предложений. Товарищи! дальнейшее развитие различных форм деятельности влечет за
собой процесс внедрения и модернизации соответствующий условий активизации. Значимость этих
проблем настолько очевидна, что постоянное информационно-пропагандистское обеспечение нашей
деятельности способствует подготовки и реализации новых предложений. Задача организации, в
особенности же рамки и место обучения кадров играет важную роль в формировании позиций, занимаемых
участниками в отношении поставленных задач.
```

Рисунок 5. Метод №2, исходный текст.

```
Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): 2
Зашифровать введите 1, расшифровать введите 2: 1
Введите слово для встраивания в текст: Крипто

Пробелов добавлено:      20  к  18971  исходным символам
Изменение размера:       35553  ->  35573       | + 20  байт
```

Рисунок 6. Метод №2, демонстрация работы программы при шифровании.

Слово было встроено в исходный текст, визуальные различия присутствуют.

```
Равным образом новая модель организационной деятельности представляет собой интересный эксперимент
проверки новых предложений. Товарищи! дальнейшее развитие различных форм деятельности влечет за
собой процесс внедрения и модернизации соответствующий условий активизации.  Значимость этих
проблем настолько очевидна, что постоянное информационно-пропагандистское обеспечение нашей
деятельности способствует подготовки и реализации новых предложений.  Задача организации, в
особенности же рамки и место обучения кадров играет важную роль в формировании позиций, занимаемых
участниками в отношении поставленных задач.
```

Рисунок 7. Метод №2, текст после встраивания слова.

Произведем обратную операцию, т.е. дешифрование, для того, чтобы получить встроенное слово из зашифрованного текста.

```
Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): 2
Зашифровать введите 1, расшифровать введите 2: 2
Расшифрованное слово: Крипто
```

Рисунок 8. Метод №1, демонстрация работы программы при дешифровании

Метод №3. Метод добавления(замены) специальных символов.

Метод заключается в замене символа точки (U+002E) символом, визуально похожим на символ простой точки (U+0323) для кодирования «1» скрываемого слова в двоичном виде, либо пропуск замены для кодирования «0» скрываемого слова.

Преобразование слова, состоящего из кириллических букв в двоичный код, осуществляется следующим способом: из кода символа первой буквы слова в десятичном виде (Unicode) отнимается число 1024 для уменьшения размера двоичного слова, затем полученное десятичное число преобразуется в двоичный код со стандартизированной размерностью в 7 бит, затем аналогичные действия производятся с последующими буквами слова, в итоге получается строка, состоящая из «0» и «1».

Демонстрация работы программы:

```
Задача организации, в особенности же сложившаяся структура организации обеспечивает широкому кругу
(специалистов) участие в формировании систем массового участия. Идейные соображения высшего
порядка, а также постоянное информационно-пропагандистское обеспечение нашей деятельности
способствует подготовки и реализации новых предложений. С другой стороны консультация с широким
активом играет важную роль в формировании позиций, занимаемых участниками в отношении поставленных
задач. Повседневная практика показывает, что реализация намеченных плановых заданий в значительной
степени обуславливает создание системы обучения кадров, соответствует насущным потребностям. Таким
образом консультация с широким активом требуют определения и уточнения соответствующий условий
активизации.
```

Рисунок 9. Метод №3, исходный текст.

```
Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): 3
Зашифровать введите 1, расшифровать введите 2: 1
Введите слово для встраивания в текст: Установка

Замена спец. символов:   28  из  18443  исходных символов
Изменение размера:       34561  ->  34589       | + 28  байт
```

Рисунок 10. Метод №2, демонстрация работы программы при шифровании.

Слово было встроено в исходный текст, визуальные различия заметны при детальном рассмотрении.

```
Задача организации, в особенности же сложившаяся структура организации обеспечивает широкому кругу
(специалистов) участие в формировании систем массового участия. Идейные соображения высшего
порядка, а также постоянное информационно-пропагандистское обеспечение нашей деятельности
способствует подготовки и реализации новых предложений. С другой стороны консультация с широким
активом играет важную роль в формировании позиций, занимаемых участниками в отношении поставленных
задач. Повседневная практика показывает, что реализация намеченных плановых заданий в значительной
степени обуславливает создание системы обучения кадров, соответствует насущным потребностям. Таким
образом консультация с широким активом требуют определения и уточнения соответствующий условий
активизации.
```

Рисунок 11. Метод №3, текст после встраивания слова.

Произведем обратную операцию, т.е. дешифрование, для того, чтобы получить встроенное слово из зашифрованного текста.

```
Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): 3
Зашифровать введите 1, расшифровать введите 2: 2
Расшифрованное слово: Установка
```

Рисунок 12. Метод №3, демонстрация работы программы при дешифровании

Метод №1. Метод прямой замены символов.

```
Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): 1
Зашифровать введите 1, расшифровать введите 2: 1
Введите слово для встраивания в текст: Степано
Введите 2 символа (A,B,E,K,M,H,O,P,C,T,X,a,e,o,p,c,y,x) вместо "01" для шифрования: ao

Заменено символов:      49  из  18969  исходных символов
Изменение размера:      35549  ->  35500      | - 49  байт
```

Рисунок 2. Метод №1, демонстрация работы программы при шифровании.

Объем встраивания составил 49 символов, что соответствует 49 битам (7 символов по 7 бит) скрываемого слова, записанного в двоичном виде.
Однако, так как символы латинского алфавита имеют вес 1 байт, а символы кириллического алфавита 2 байта. То в данном случае размер текстового файла уменьшился на 49 байт, с 35549 байт до 35500 байт.
Визуальных различий с исходным текстом нет. О присутствии встроенного слова можно узнать лишь при подсчете количества знаков препинания и подсчете количества букв, размер файла будет не соответствовать тому, что в нем находятся только кириллические символы.

Метод №2. Метод с использованием дополнительных пробелов.

```
Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): 2
Зашифровать введите 1, расшифровать введите 2: 1
Введите слово для встраивания в текст: Крипто

Пробелов добавлено:     20  к  18971  исходным символам
Изменение размера:      35553  ->  35573      | + 20  байт
```

Рисунок 6. Метод №2, демонстрация работы программы при шифровании.

Объем встраивания составил 20 символов (пробелов), что соответствует 20 единичным («1») битам скрываемого слова, записанного в двоичном виде.
Так как символы пробела добавлялись, то размер текстового файла увеличился на 20 байт, с 35553 байт до 35573 байт.
Визуальные отличия видны из-за неравномерности распределения двойных пробелов. О присутствии встроенного слова можно догадаться даже визуально, без рассчетов.

Метод №3. Метод добавления(замены) специальных символов.

```
Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): 3
Зашифровать введите 1, расшифровать введите 2: 1
Введите слово для встраивания в текст: Установка

Замена спец. символов:   28  из  18443  исходных символов
Изменение размера:       34561  ->  34589       | + 28  байт
```

Рисунок 10. Метод №2, демонстрация работы программы при шифровании.

Объем встраивания(замены) составил 28 символов (замена 28 точек на визуально аналогичные), что соответствует 28 единичным («1») битам скрываемого слова, записанного в двоичном виде.
Так как символ аналогичной точки имеет размер 2 байта, а обычная точка 1 байт, то размер текстового файла увеличился на 28 байт, с 34561 байт до 34589 байт.
Визуально встроенное слово не заметно, однако есть различия в размере исходного и конечного файлов. Данной проблемы можно было избежать, заменив кодировку файла, либо подобрав такие 2 визуально похожие символа, которые имели бы одинаковый вес: например, кириллическая буква «о» (U+043E) и греческая буква омикрон «о» (U+03BF).

Полученные данные являются лишь оценочными и зависят не только от свойств контейнера, но и от свойств помещаемого в него информации, хотя и в меньшей степени.

Вывод:

В результате работы над программной реализацией 3х методов встраивания слова в текстовый файл были приобретены навыки исследования свойств стегоконтейнеров, разработки стегосистем и их применения для встраивания информации и сокрытия данных с помощью текстовых файлов.

Наиболее эффективным в плане сохранения размера файла и визуальной неотличимости показал себя метод №3, с заменой спец. символов. Подобрав такие 2 визуально похожие символа, которые имели бы одинаковый вес: например, кириллическая буква «о» (U+043E) и греческая буква омикрон «ο» (U+03BF), можно было бы добиться практически идеальной скрытности встраиваемого слова в текстовом файле.

Список использованной литературы:

1. Текстовая стеганография // Интернет-ресурс. Открытый доступ. URL: https://ru.wikipedia.org/wiki/Текстовая_стеганография (дата обращения: 9.04.2020).
2. Основы текстовой стеганографии на практике // Интернет-ресурс. Открытый доступ. URL: http://www.iso27000.ru/chitalnyi-zai/steganografiya/tekstovaya-steganografiya (дата обращения 10.04.2020).

Приложение 1. Листинг программного кода.

```
file_in = file_out = 0
file_in_path = 'text_plain.txt'
file_out_path = 'text_encrypted.txt'
size_plain = size_delta = count_plain = 0

# Проверка, хватит ли нужных символов для шифрования
def check_symbols():
        zero_count_sym = one_count_sym = zero_count_code = one_count_code = 0
        a = file_in.read(1)
        while len(a) > 0:
                if method == '1':
                        if a == zero:       zero_count_sym += 1
                        elif a == one:      one_count_sym += 1
                elif method == '3':
                        if a == '.':        one_count_sym += 1
                a = file_in.read(1)
        for b in cypher_word:
                if b == '0':                zero_count_code += 1
                elif b == '1':              one_count_code += 1
        if method == '1':
                if one_count_code > one_count_sym:                          return -1
                elif zero_count_code > zero_count_sym:                      return -2
        elif method == '3':
                if one_count_code + zero_count_code > one_count_sym:        return -1
        else:   return 0

# Проверка, правильно ли указаны символы для шифрования
def check_dictionary():
        if option == 1:
                if zero not in dictionary.keys():
                        print('Неверный символ: ', zero)
                        return -1
                if one not in dictionary.keys():
                        print('Неверный символ: ', one)
                        return -1
        if option == 2:
                if zero not in dictionary.values():
                        print('Неверный символ: ', zero)
                        return -1
                if one not in dictionary.values():
                        print('Неверный символ: ', one)
                        return -1
        return 0

# Проверка, хватит ли предложений для шифрования
def check_sentence():
        count = 0
        pa = file_in.read(1)
        a = file_in.read(1)
        while len(a) > 0:
                if pa in '.?!' and a != '.':
                        count += 1
                pa = a
                a = file_in.read(1)
        if len(cypher_word) > count: return -3
        else:   return 0

def size_file(file, file_path):
        if file != 0:       file.close()
        file = open(file_path, 'r', encoding='utf-8')
        i = a = 0
        c = file.read(1)
        while len(c) > 0:
                a += 1
                if ord(c) > 127 or ord(c) == 10:        i += 2
                else:                                   i += 1
                c = file.read(1)
        file.close()
        return a, i

# Завершение работы при ошибке
def exit1():
        if file_in != 0:    file_in.close()
        if file_out != 0:   file_out.close()
        print ('exit(-1)')
        exit(-1)
```

```
# Выбор метода шифрования/дешифрования
method = input('Введите номер метода (1 - латиница, 2 - пробелы, 3 - спец. символы): ')
if method not in '123':
          print ('Неверный символ: ', method)
          exit1()

# Выбор шифрования/дешифрования
option = input('Зашифровать введите 1, расшифровать введите 2: ')
if option not in '12':
          print ('Неверный символ: ', option)
          exit1()


#--------ШИФРОВАНИЕ--------
if option == '1':

          file_in = open(file_in_path, 'r', encoding='utf-8')
          file_out = open(file_out_path, 'w', encoding='utf-8')

          plain_word = input('Введите слово для встраивания в текст: ')
          cypher_word = ''
          pc = '$'

          if method == '1':
                    dictionary = dict(zip('A B E K M H O P C T X a e o p c y x'.split(),'A B E K M
                    H O P C T X a e o p c y x'.split()))
                    zeroone = input('Введите 2 символа (A,B,E,K,M,H,O,P,C,T,X,a,e,o,p,c,y,x) вместо
                    "01" для шифрования: ')
                    zero = zeroone[0]
                    one = zeroone[1]
                    if check_dictionary() == -1: exit1
          elif method == '3':
                    zero = '.'
                    one = ','

          # Преобразование слова в двоичный код
          for i in plain_word: cypher_word += format(ord(i)-1024, '07b')
          # print ('Слово в двоичном коде:      ', cypher_word)
          print('')

          # Проверка на нехватку символов
          if method == '1' or method == '3':              res = check_symbols()
          elif method == '2':                             res = check_sentence()
          if res == -1:
                    print ('В исходном тексте не хватает символов: ', one)
                    exit1()
          elif res == -2:
                    print ('В исходном тексте не хватает символов: ', zero)
                    exit1()
          elif res == -3:
                    print('В исходном тексте недостаточно предложений для зашифровки слова: ',
                    plain_word)
                    exit1()
          else:
                    file_in.close()
                    file_in = open(file_in_path, 'r', encoding='utf-8')


          #----МЕТОД ЛАТИНИЦА----
          if method == '1':

                    # Разбор двоичного кода
                    for word in cypher_word:
                        # Разбор исходного текста
                              c = file_in.read(1)
                              while len(c) > 0:
                                  # Замена при нуле
                                        if word == '0' and c == zero:
                                                  file_out.write(dictionary[c])
                                                  break
                                        # Замена при единице
                                        elif word == '1' and c == one:
                                                  file_out.write(dictionary[c])
                                                  break
                                        # Пропуск
                                        else:
                                                  file_out.write(c)
                                                  c = file_in.read(1)

                    # Вставка оставшейся части текста
                    file_out.write(file_in.read())

                    # Подсчет объема встраивания
                    count_plain = size_file(file_in, file_in_path)[0]
                    size_plain = size_file(file_in, file_in_path)[1]
                    size_delta = len(cypher_word)
                    print('Заменено символов:    ', size_delta, ' из ', count_plain,
                    ' исходных символов')
                    print('Изменение размера:    ', size_plain, ' -> ', size_plain-size_delta, ' | -
```

```
', size_delta, ' байт')


#----МЕТОД ПРОБЕЛЫ----
elif method == '2':

        # Разбор двоичного кода
        for word in cypher_word:
                # Разбор исходного текста
                c = file_in.read(1)
                while len(c) > 0:
                        if word == '1' and pc in '.?!' and c != '.':
                                file_out.write(' ')
                                file_out.write(c)
                                pc = c
                                break
                        elif word == '0' and pc in '.?!' and c != '.':
                                file_out.write(c)
                                pc = c
                                break
                        else:
                                file_out.write(c)
                                pc = c
                                c = file_in.read(1)

        # Вставка оставшейся части текста
        file_out.write(file_in.read())

        # Подсчет объема встраивания
        count_plain = size_file(file_in, file_in_path)[0]
        size_plain = size_file(file_in, file_in_path)[1]
        size_delta = cypher_word.count('1')
        print('Пробелов добавлено:   ', size_delta, ' к ', count_plain, ' исходным
        символам')
        print('Изменение размера:    ', size_plain, ' -> ', size_plain+size_delta, ' |
        +', size_delta, ' байт')



#----МЕТОД СПЕЦ. СИМВОЛЫ----
elif method == '3':

        # Разбор двоичного кода
        for word in cypher_word:
            # Разбор исходного текста
                c = file_in.read(1)
                while len(c) > 0:
                # Замена при единице
                        if word == '1' and c == '.':
                                file_out.write(chr(803))
                                # file_out.write(' ')
                                break
                        # Замена при нуле
                        elif word == '0' and c == '.':
                                file_out.write('.')
                                break
                        # Пропуск
                        else:
                                file_out.write(c)
                                c = file_in.read(1)

        # Вставка оставшейся части текста
        file_out.write(file_in.read())

        # Подсчет объема встраивания
        count_plain = size_file(file_in, file_in_path)[0]
        size_plain = size_file(file_in, file_in_path)[1]
        size_delta = cypher_word.count('1')
        print('Замена спец. символов:          ', size_delta, ' из ', count_plain, '
        исходных символов')
        print('Изменение размера:    ', size_plain, ' -> ', size_plain+size_delta, ' |
        +', size_delta, ' байт')
```

```
#--------ДЕШИФРОВАНИЕ--------
elif option == '2':

        file_out = open(file_in_path, 'w', encoding='utf-8')
        file_in = open(file_out_path, 'r', encoding='utf-8')

        cypher_word = string = plain_word = ''


        #----МЕТОД ЛАТИНИЦА----
        if method == '1':

                dictionary = dict(zip('A B E K M H O P C T X a e o p c y x'.split(),'А В Е К М
                Н О Р С Т Х а е о р с у х'.split()))
                zeroone = input('Введите 2 символа (A,B,E,K,M,H,O,P,C,T,X,a,e,o,p,c,y,x) вместо
                "01" для дешифрования: ')
                zero = zeroone[0]
                one = zeroone[1]
                if check_dictionary() == -1: exit1

                # Дешифрование двочиного кода
                c = file_in.read(1)
                while len(c) > 0:
                        if c in dictionary.keys():
                                if dictionary[c] == zero:
                                        cypher_word += '0'
                                elif dictionary[c] == one:
                                        cypher_word += '1'
                                file_out.write(dictionary[c])
                        else:
                                file_out.write(c)
                        c = file_in.read(1)


        #----МЕТОД ПРОБЕЛЫ----
        elif method == '2':

                # Дешифрование двочиного кода
                ppc = pc = ''
                ppc = file_in.read(1)
                pc = file_in.read(1)
                c = file_in.read(1)
                while len(c) > 0:
                        file_out.write(pc)
                        if ppc in '.?!' and pc == ' ' and c == ' ':
                                cypher_word += '1'
                                c = file_in.read(1)
                        elif ppc in '.?!' and (pc == ' ' or pc == chr(10)) and c != ' ':
                                cypher_word += '0'
                        ppc = pc
                        pc = c
                        c = file_in.read(1)


        #----МЕТОД СПЕЦ. СИМВОЛЫ----
        elif method == '3':

                # Дешифрование двочиного кода
                pc = file_in.read(1)
                c = file_in.read(1)
                while len(c) > 0:
                        if pc == chr(803) and c == ' ':
                                cypher_word += '1'
                                file_out.write('.')
                                c = file_in.read(1)
                        elif pc == '.':
                                cypher_word += '0'
                                file_out.write(pc)
                        else:
                                file_out.write(pc)
                        pc = c
                        c = file_in.read(1)

        # Преобразование двоичного кода
        for i in range(len(cypher_word)):
                string += cypher_word[i]
                if len(string) == 7 and string != '0000000':
                        plain_word += chr(int(string, 2)+1024)
                        string = ''

        print ('Расшифрованное слово:', plain_word)

file_in.close()
file_out.close()
exit(0)
```