МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО».

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

ОСНОВЫ ШИФРОВАНИЯ ДАННЫХ

ВАРИАНТ 11

|  |  |  |
| --- | --- | --- |
| выполнил  преподаватель |  | Студент 4 курса  факультета Программной инженерии и компьютерной техники  Группы P34121  Кустарев Иван Павлович  Старший преподаватель (квалификационная категория "старший преподаватель"), факультета программной инженерии и компьютерной техники  Маркина Татьяна Анатольевна |

Санкт-Петербург

2023 г.

Название и цель работы

Название: Основы шифрования данных

Цель работы: изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

Вариант задания

Вариант = 11 %10 = 1. Реализовать в программе шифрование и дешифрацию содержимого файла по методу Цезаря с ключевым словом.

Листинг разработанной программы с комментариями

import string

# язык задаётся путём ввода русского/английского keyword и body

# загрузка данных (секретное слово,

# индекс смещения, режим (шифрование/дешифрование),

# текст для шифровки/дешифровки) из файла

def read\_file(file\_name):

f = open(file\_name, encoding='utf-8', mode='r')

try:

file\_content = f.readlines()

for i in range(0, len(file\_content)):

line = file\_content[i].strip()

# file\_content[i] = line.split("=")[1]

# чтение ключевого слова

key\_word = file\_content[0].strip()

# чтение индекса смещения

key\_index = file\_content[1].strip()

# чтение режима шифровака (расшифровка)

mood = file\_content[2].strip()

# чтение текста, который нужно зашифровать (расшифровать)

body = ''

for i in range(3, len(file\_content)):

body += (file\_content[i] + '\r')

try:

key\_index = int(key\_index)

if mood.upper() == "TRUE":

mood = True

else:

mood = False

# валидация данных из файла на приведение типов и тп

except TypeError:

raise Exception("Invalid typing")

if isinstance(key\_word, str) and isinstance(body, str):

return key\_word, key\_index, mood, body

else:

raise Exception("Invalid typing")

except FileNotFoundError:

raise Exception("File not found")

except IndexError:

raise Exception("Invalid file structure")

finally:

f.close()

# проверка на то, что символ принадлежит алфавиту (рус/англ).

# знаки препинания не являются частями алфавитов (https://clck.ru/35sqmi)

def is\_alpha(word: str, canonical\_alpha: str):

word = word.upper()

for char in word:

if not (char in canonical\_alpha):

return False

return True

# валидация ключевого слова

def is\_correct\_key\_word(key\_word: str, canonical\_alpha: str):

if not is\_alpha(key\_word, canonical\_alpha):

raise Exception("Keyword has not alpha symbols")

if not key\_word.isupper():

raise Exception("Keyword is not in uppercase")

for char in key\_word:

if key\_word.count(char) > 1:

raise Exception("The keyword contains repeated characters")

if len(key\_word) > len(canonical\_alpha) - 2 or len(key\_word) < 0:

raise Exception("Keyword len should be in 0..." + str(len(canonical\_alpha) - 2))

# валидация индекса сдвига

def is\_correct\_key\_index(key\_index: int, canonical\_alpha: str):

if key\_index > len(canonical\_alpha) - 2 or key\_index < 0:

raise Exception("Key\_index len should be in 0..." + str(len(canonical\_alpha) - 2))

# создание алфавита сдвигов

def create\_shifts\_alpha(key\_word: str, canonical\_alpha: str):

alpha\_list = canonical\_alpha

for i in key\_word:

alpha\_list = alpha\_list.replace(i, "")

return key\_word + alpha\_list

# получение зашифрованного символа на основе входящего символа в верхнем регистре

def get\_encrypted\_char(alpha\_list, upper\_char, key\_index: int, canonical\_alpha: str):

return alpha\_list[(ord(upper\_char) - ord(canonical\_alpha[0]) - key\_index)]

# получение расшифрованного символа на основе входящего символа в верхнем регистре

def get\_decrypted\_char(alpha\_list: str, upper\_char, key\_index: int, canonical\_alpha: str):

return canonical\_alpha[(alpha\_list.rfind(upper\_char) + key\_index) % len(canonical\_alpha)]

# функция для шифровки/дешифровки

def encryption\_decryption(encryption: bool, key\_word: str,

key\_index: int, body: str, canonical\_alpha: str):

# валидация секретного слова и индекса смещения

is\_correct\_key\_word(key\_word, canonical\_alpha)

is\_correct\_key\_index(key\_index, canonical\_alpha)

alpha\_list = create\_shifts\_alpha(key\_word, canonical\_alpha)

result = ""

for char in body:

# итерация по символам текста и

# шифровка/дешифровка алфавитных символов + пропуск неалфавитных символов

if is\_alpha(char, canonical\_alpha):

if char.islower():

# если символ - в lowercase

char = char.upper()

if encryption:

char = get\_encrypted\_char(alpha\_list, char, key\_index, canonical\_alpha)

else:

char = get\_decrypted\_char(alpha\_list, char, key\_index, canonical\_alpha)

char = char.lower()

else:

# если символ - в uppercase

if encryption:

char = get\_encrypted\_char(alpha\_list, char, key\_index, canonical\_alpha)

else:

char = get\_decrypted\_char(alpha\_list, char, key\_index, canonical\_alpha)

result += char

else:

result += char

return result

def cesar\_encryption(key\_word: str, key\_index: int, text\_body: str, canonical\_alpha: str):

return encryption\_decryption(True, key\_word, key\_index, text\_body, canonical\_alpha)

def cesar\_decryption(key\_word: str, key\_index: int, encrypted\_body: str, canonical\_alpha: str):

return encryption\_decryption(False, key\_word, key\_index, encrypted\_body, canonical\_alpha)

# основной процесс

def main\_fun(path: str):

# загрузка данных (секретное слово, индекс смещения, режим (шифрование/дешифрование),

# текст для шифровки/дешифровки) из файла

key\_word, k, mood, body = read\_file(path)

# выбор алфавита (англ/русский)

if body[0] in string.ascii\_uppercase:

canonical\_alpha = string.ascii\_uppercase

else:

a = ord('а')

canonical\_alpha = ''.join([chr(i).upper() for i in range(a, a + 32)])

if key\_word and k and body:

# запуск шифровки/дешифровки

if mood is True:

response = cesar\_encryption(key\_word, k, body, canonical\_alpha)

else:

response = cesar\_decryption(key\_word, k, body, canonical\_alpha)

print(response)

if \_\_name\_\_ == '\_\_main\_\_':

file\_path = input("Введите путь к файлу или Enter чтобы задать file.txt")

if file\_path == "":

file\_path = "file.txt"

try:

main\_fun(file\_path)

except Exception as e:

print(e)

Результаты работы программы

1. Шифрование (Английский):
   1. Входные данные:

keyword=ADZKIH

key\_index=14

mod=true

body=Interdum dictum. Sed ex. Morbi faucibus. Nisi in vitae vitae mattis nec justo mattis sed leo, luctus et ultricies. Nulla sapien in molestie sed aene-an tempus ultricies. Ultricies. Sit habitasse ut. Adipiscing dictum quis, arcu cursus nulla pellentesque cursus libero, amet, cras orci, sit eleifend qu

* 1. Выходные данные:

Tyhpkobx otnhbx. Ipo pf. Xakmt qlbntmbi. Ytit ty cthlp cthlp xlhhti ypn ubiha xlhhti ipo wpa, wbnhbi ph bwhktntpi. Ybwwl ildtpy ty xawpihtp ipo lpyp-ly hpxdbi bwhktntpi. Bwhktntpi. Ith slmthliip bh. Lotdtintyr otnhbx zbti, lknb nbkibi ybwwl dpwwpyhpizbp nbkibi wtmpka, lxph, nkli aknt, ith pwptqpyo zb

1. Расшифровка (Английский):

2.1 Входные данные:

keyword=ADZKIH

key\_index=14

mod=false

body=Tyhpkobx otnhbx. Ipo pf. Xakmt qlbntmbi. Ytit ty cthlp cthlp xlhhti ypn ubiha xlhhti ipo wpa, wbnhbi ph bwhktntpi. Ybwwl ildtpy ty xawpihtp ipo lpyp-ly hpxdbi bwhktntpi. Bwhktntpi. Ith slmthliip bh. Lotdtintyr otnhbx zbti, lknb nbkibi ybwwl dpwwpyhpizbp nbkibi wtmpka, lxph, nkli aknt, ith pwptqpyo zb

2.2 Выходные данные:

Interdum dictum. Sed ex. Morbi faucibus. Nisi in vitae vitae mattis nec justo mattis sed leo, luctus et ultricies. Nulla sapien in molestie sed aene-an tempus ultricies. Ultricies. Sit habitasse ut. Adipiscing dictum quis, arcu cursus nulla pellentesque cursus libero, amet, cras orci, sit eleifend qu

1. Шифрование (Русский):
   1. Входные данные:

keyword=АБВЯГ

key\_index=14

mod=true

body=Поставленных эксперимент что интересный образом играет соображения анализа занимаемых идейные и развития. Организационной рамки же важные по важную структура сфера анализа эксперимент модель способствует сфера в соображения разработке место особенности способствует задач. Количественный также же же

* 1. Выходные данные:

Баягсуьцююмж оыябцвщэцюг ига щюгцвцяюмъ атвсшаэ щфвсцг яаатвсчцющр сюсьщшс шсющэсцэмж щхцъюмц щ всшущгщр. Авфсющшсзщаююаъ всэыщ чц усчюмц ба усчюдп ягвдыгдвс яецвс сюсьщшс оыябцвщэцюг эахцьн ябаяатягудцг яецвс у яаатвсчцющр всшвстагыц эцяга аяатцююаягщ ябаяатягудцг шсхси. Ыаьщицягуцююмъ гсычц чц чц

1. Расшифровка (Русский):
   1. Входные данные:

keyword=АБВЯГ

key\_index=14

mod=false

body=Баягсуьцююмж оыябцвщэцюг ига щюгцвцяюмъ атвсшаэ щфвсцг яаатвсчцющр сюсьщшс шсющэсцэмж щхцъюмц щ всшущгщр. Авфсющшсзщаююаъ всэыщ чц усчюмц ба усчюдп ягвдыгдвс яецвс сюсьщшс оыябцвщэцюг эахцьн ябаяатягудцг яецвс у яаатвсчцющр всшвстагыц эцяга аяатцююаягщ ябаяатягудцг шсхси. Ыаьщицягуцююмъ гсычц чц чц

* 1. Выходные данные:

Поставленных эксперимент что интересный образом играет соображения анализа занимаемых идейные и развития. Организационной рамки же важные по важную структура сфера анализа эксперимент модель способствует сфера в соображения разработке место особенности способствует задач. Количественный также же же

**Вывод**

Во время выполнения данной лабораторной работы я:

1. Получил практические навыки написания программ для шифрования/расшифровки текста методом Цезаря с ключевым словом
2. Ознакомился с различными видами реализации данного метода шифрования
3. Выяснил какие ограничения налагаются на контрольное слово и индекс сдвига при реализации алгоритма