ОТЧЕТ К ЛАБОРАТОРНОЙ РАБОТЕ № 1

Лабораторная работа № 1

Шифр Цезаря

Вариант №4

Ф. И. О. студента: Гюнтер Тимофей Вячеславович

Группа: ФИТ-221

Проверил: Дата:

Основные сведения

Прямое преобразование шифра Цезаря:

Обратное преобразование шифра Цезаря:

Таблица кодировки символов:

Результаты

ШИФР-ТЕКСТ (ШТ):
юмуумфгямнмфапфъувтеипущмутмкюмфчзйпушжувмрпщмуммймчфммкъипушчмлгхихтгшщпщмтгфвьшмщмр

РАСШИФРОВАННЫЙ ТЕКСТ (ОТ):
чемменьшеженщинумылюбимтемлегченравимсямыейитемеевернеегубимсредьобольстительныхсетей

КЛЮЧ: 7

АВТОР И ПРОИЗВЕДЕНИЕ (ОТ): пушкиневгенийонегин

ЗАШИФРОВАННЫЕ ФАМИЛИЯ И НАЗВАНИЕ (ШТ):
цъяспфмйкмфпрхфмкпф

Варианты расшифрования исходного ШТ при различных значениях ключа:

k = 1:

элттлувюлмлуяоущтбсдзотшлтслйэлуцжиотчетблпошлтллилцуллйщзотчцлквфзфсвчшошлсвубычлшлп

k = 2:

ькссктбэклктюнтшсаргжнсчкскркиьктхезнсцдсакончкскказкхткккишжнсцхкйбужурбцчнчкрбтаъцкчко

k = 3:

ыйррйсаьйкйсэмсчряпвемрцйрпйзыйсфджмрхгряйнмцйрййжйфсййзчемрхфйиатетпахцмцйпасящхйцйн

k = 4:

ъиппиряыийирьлрцпюобдлпхипоижъиругелпфвпюимлхипииеиуриижцдлпфуизясдсояфхлхиоярюшфихим

k = 5:

щзоозпюъзизпыкпхоэнагкофзонзещзптвдкоубоэзлкфзозздзтпззехгкоутзжюргрнюуфкфзнюпэчузфзл

k = 6:

шжннжоэщжзжоъйофньмявйнужнмждшжосбгйнтаньжкйужнжжгжсожждфвйнтсжеэпвпмэтуйужмэоьцтжужк

k = 7:

чемменьшеженщинумылюбимтемлегченравимсямыейитемеевернеегубимсредьобольстительныхсетей

k = 8:

цдллдмычдедмшзмтлъкэазлсдлкдвцдмпябзлрюлъдизсдлддбдпмддвтазлрпдгынанкырсзсдкымъфрдсди

k = 9:

хгккглъцгдглчжлскщйьяжкргкйгбхглоюажкпэкщгзжргкггаголггбсяжкпогвъмямйъпржргйълщупгргз

k = 10:

фвййвкщхвгвкцекрйшиыюейпвйивафвкнэяейоьйшвжепвйввявнквварюейонвбщлюлищопепвищкштовпвж

k =11:

убиибйшфбвбйхдйпичзъэдиобизбяубймьюдиныичбедобиббюбмйббяпэдинмбашкэкзшнодобзшйчснбобе

k = 12:

таззаичуабаифгиозцжщьгзназжаютаилыэгзмъзцадгназааэалиааюоьгзмлаячйьйжчмнгнажчицрманад

k = 13:

сяжжязцтяаязувзнжхешывжмяжеяэсязкъьвжлщжхягвмяжяяьякзяяэнывжлкяюциыиецлмвмяецзхплямяг

k = 14:
рюееюжхсюяюжтбжмефдчъбелюедюьрюжйщыбекшефювблюеююыюйжююь
мъбекйюэхзъздхклблюдхжфокюлюв

k = 15:
пэддэефрэюэесаелдугцщадкэдгэыпэеишъадйчдуэбакэдээъэиеээылщадйиэьфж
щжгфйкакэгфеунйэкэб

k = 16:
оыггьдупьэьдрядкгтвхшягйыгвьъоьдзчщягицгтьаяйьгььщьздььъкшягизьыуеше
вуийяйьвудтмиьйьа

k = 17:
ныввыгтоыьыгпюгйвсбфчювиыывбыщныгжцшювзхвсыяюиывыышыжгыыщйч
ювзжыътдчдбтзиюиыбтгслзыиыя

k = 18:
мъббъвснъыъвоэвибрауцэбзъбаъшмъвехчэбжфбръюэзъбъъчъевъъшицэбжеъщ
сгцгасжзээзъасвркжъзъю

k = 19:
лщаащбрмщъщбньбзапятхьажщаящчлщбдфцьаеуапщэьжщащщцщдбщщчзхьа
едщшрвхвярежьжщярбпйещжщэ

k = 20:
кшяяшаплшщшамыажяоюсфыяешяюшцкшагухыядтяошьыешяшшхшгашшщж
фыядгшчпбфбюпдеыешюпаоидшешь

k = 21:
йчюючяокчшчялъяеюнэруъюдчюэчхйчявтфъюгсюнчыъдчюччфчвяччхеуъюгв
чцоауаэогдъдчэоянзгчдчы

k = 22:
ицээцюнйцчцюкщюдэмьптщэгцэьцфицюбсущэврэмцъщгцэццуцбюццфдтщэв
бцхнятяьнвгщгцьнюмжвцгцъ

k = 23:
зхььхэмихцхэйшэгьлыосшьвхьыхузхэартшьбпьлхщшвхьххтхаэххугсшьбахфм
юсюымбвшвхымэлебхвхщ

k = 24:
жфыыфьлзфхфьичьвыкънрчыбфыъфтжфьяпсчыаоыкфшчбфыффсфяьффтврч
ыаяфулэрэълабчбфълькдафбфш

k = 25:
еуъъуыкжуфуызцыбъйщмпцъауъщусеуыюорцъянъйучцауъууруюыуусбпцъяю
уткьпьщкяацаущкыйгяуауч

k = 26:
дтщщтъйетутъжхъащишлохщятщштрдтъэнпхщюмщитцхятщттптэъттраохщю
этсйыоышйюяхятшйъивютятц

k = 27:
гсшшсщидстсщефщяшзчкнфшюсшчспгсщьмофшэлшзсхфюсшссосьщсспянф
шэьсриънъчиэюфюсчищзбэсюсх

k = 28:
врччршзгрсршдушючжцймучэрчцровршылнучькчжрфуэрчррнрышрроюмучь
ырпзщмщцзьэуэрцзшжаьрэрф

k = 29:
бпццпчжвпрпчгтчэцехилтцьпцхпнбпчъкмтцыйцепутьпцппмпъчппнэлтцыъпо
жшлшхжыьтьпхжчеяыпьпу

k = 30:
аоххоцебопоцвсцьхдфзксхыохфомаоцщйлсхъихдотсыохоолощцоомьксхъщоне
чкчфеъысыофецдюъоыот

k = 31:
янффнхданонхбрхыфгужйрфънфунлянхшикрфщзфгнсрънфннкншхннлыйрф
щшнмдцйцудщърънудхгэщнънс

Код программы:

```
import re

import warnings

warnings.filterwarnings("ignore")


# ascii_rus_codes = [i for i in range(1040, 1103+1, 1)]

# ascii_rus_codes += [1105, 1025] # is created to check whether the letter is RUS

alphabet = ['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', 'ё']
```

```
special_symbols_remover = lambda string: re.split('; |!| |, |\.|\?|-', string)

# if 1077 1105
# print(ord('а') - 1072)
# print(ord('я') - 1072)

class Caesar():
    def __init__(self) -> None:
        self.m = 32
        self.encrypted = []
        self.preprocessed_text = ''
        self.bool_table = []

    def preprocessing(text):
        if len(text.split()) > 1: # for cases when we put preprocessed data
            preprocessed_text = ''.join(special_symbols_remover(text)).lower()
        else: preprocessed_text = text

        return preprocessed_text

    def auxilliary(self, preprocessed_text):
        code = [char for char in preprocessed_text]
        for index in range(len(preprocessed_text)):
            # bool_alpha_table = [char in alphabet for char in self.preprocessed_text]
            if self.bool_alpha_table[index]:   # проверка на русскую букву
                ascii_number = ord(preprocessed_text[index])
```

```
            if ascii_number in {1077, 1105}:
                code[index] = 5
            else: code[index] = ord(preprocessed_text[index]) + ord("a")
            # self.encrypted[index] = (code[index] + key) % self.m
        return code

    def encryption(self):
        given_text = input("Текст, который будем шифровать: ")

        self.preprocessed_text = self.preprocessing(given_text)

        encryption_key = int(input("Ключ шифрования: "))

        encoded = [char for char in self.preprocessed_text]
        self.encrypted = encoded
        encrypted_text = [char for char in self.encrypted]

        self.bool_alpha_table = [char in alphabet for char in self.preprocessed_text]

        for index in range(len(self.preprocessed_text)):
            if self.bool_alpha_table[index]:   # проверка на русскую букву
                ascii_number = ord(self.preprocessed_text[index])
                if ascii_number in {1077, 1105}: # e/ё
                    encoded[index] = 5
                else: encoded[index] = ord(self.preprocessed_text[index]) - ord("a")

                self.encrypted[index] = (encoded[index] + encryption_key) % self.m
```

```
            encrypted_text[index] = chr(self.encrypted[index] + ord("a")) # только с индексами, где есть русские буквы

        with open("encrypted_text.txt", "w") as f:
            f.writelines(f'Зашифрованный текст: {encrypted_text}\n')
            f.writelines(f'Ключ: {encryption_key}')

    def decryption(self):
        open("decrypted.txt", "w").close()
        # 5 times if not guessing
        iterations = 0
        given_text = input("Текст на расшифровку: ")

        preprocessed_text = self.preprocessing(given_text)
        if preprocessed_text == '':
            decoded = [char for char in self.encrypted]
        else:
            decoded = [char for char in preprocessed_text]

        self.bool_alpha_table = [char in alphabet for char in preprocessed_text]

        decrypted = [str(char) for char in decoded]
        # self.auxilliary(preprocessed_text)
        code = self.auxilliary(preprocessed_text)

        decryption_key = int(input('Ключ расшифровки: '))
        print()
        for index in range(len(decoded)):
```

```
            if self.bool_alpha_table[index]:
                decoded[index] = (code[index] - decryption_key) % self.m
                decrypted[index] = chr(decoded[index] + ord('a'))
        with open("decrypted.txt", "a") as f:
            f.write(f'key: {decryption_key},\ndecrypted text:{decrypted} \n')
            f.write('--------------------------------------------------\n')
        print(f"Потенциально исходный текст: {''.join(decrypted)}")
        print()

class Encoding:
    def __init__(self, text):
        self._bool_table_completion(text)
    def _bool_table_completion(self, text):
        alphabet = ['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', 'ё']

        self.bool_alpha_table = [char in alphabet for char in text]

    def letters_to_code(self, text):
        # alphabet = ['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', 'ё']

        code = [char for char in text]
        for index in range(len(text)):
            if self.bool_alpha_table[index]:   # проверка на русскую букву
```

```
            ascii_number = ord(text[index])
            if ascii_number in {1077, 1105}:
                code[index] = 5
            else: code[index] = ord(text[index]) - ord("a")
            # self.encrypted[index] = (code[index] + key) % self.m
        return code


    def code_to_letters(self, index, code):
        if self.bool_alpha_table[index]:
            return chr(code + ord('a'))
        return code
                # return [chr(code[index] + ord('a')) for index in code if
self.bool_alpha_table[index]]


# print(letters_to_code('мама'))


def given_var():
    m = 32
                                                    encrypted =
'юмуумфгямнмфапфъувтеипущмутмкюмфчзйпушжувмрпщмуммймчфммкъи
пушчмлгхихтгшщпщмтгфвьшмщмр'
    decoded = [char for char in range(len(encrypted))]
    decrypted = [" for char in decoded]
    with open("given_text_decryption.txt", "w") as f:
        f.write(f'Исходный текст: {encrypted}\n')
        f.write('----------------------------------------------------------------------------------
---\n')
        for key in range(1, m+1, 1):
```

```
for index in range(len(encrypted)):
    ascii_number = ord(encrypted[index])
    if ascii_number in {1077, 1105}:
        decoded[index] = 5
    else: decoded[index] = ord(encrypted[index]) - ord("a") # to number
    # for index in range(len(decoded)):
    decoded[index] = (decoded[index] - key) % m
    decrypted[index] = chr(decoded[index] + ord('a'))  # back to ascii
    if index == len(encrypted)-1:
        print(f'{key}, {"".join(decrypted)}\n')
        f.write(f'{key}, {"".join(decrypted)}\n')
```