**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:

ькссктбэклктюнтшсаргжнсчксркиьктхезнсцдсакончксккзкхтккишжнсцхкйбужурбцчнчкрбтаъцкчко

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: рюееюжхсюяюжтбжмефдчъбелюедюьрюжйщыбекшефювблюеююыюйжююьмъбекйюэхзъздхклблюдхжфокюлюв

k = 15: пэддэефрэюэесаелдугцщадкэдгэыпэеишъадйчдуэбакэдээъэиеээылщадйиэьфжщжгфйкакэгфеунйэкэб

k = 16: оьггьдупьэьдрядкгтвхшягйьгвьъоьдзчщягицгтьаяйьгььщьздььъкшягизьыуешевуийяйьвудтмиьйьа

k = 17: ныввыгтоыьыгпюгйвсбфчювиывбыщныгжцшювзхвсыяюиывыышыжгыыщйчювзжыътдчдбтзиюиыбтгслзыиыя

k = 18: мъббъвснъыъвоэвибрауцэбзъбаъшмъвехчэбжфбръюэзъбъъчъевъъшицэбжеъщсгцгасжзэзъасвркжъзъю

k = 19: лщаащбрмщъщбньбзапятхьажщаящчлщбдфцьаеуапщэьжщащщцщдбщщчзхьаедщшрвхвярежьжщярбпйещжщэ

k = 20: кшяяшаплшщшамыажяоюсфыяешяюшцкшагухыядтяошьыешяшшхшгашшцжфыядгшчпбфбюпдеыешюпаоидшешь

k = 21: йчюючяокчшчялъяеюнэруъюдчюэчхйчявтфъюгсюнчыъдчюччфчвяччхеуъюгвчцоауаэогдъдчэоянзгчдчы

k = 22: ицээцюнйцчцюкщюдэмьптщэгцэьцфицюбсущэврэмцъщгцэццуцбюццфдтщэвбцхнятяьнвгщгцьнюмжвцгцъ

k = 23: зхььхэмихцхэйшэгьлыосшьвхьыхузхэартшьбпьлхщшвхьххтхаэххугсшьбахфмюсюымбвшвхымэлебхвхщ

k = 24: жфыыфьлзфхфьичьвыкънрчыбфыъфтжфьяпсчыаоыкфшчбфыффсфяьффтврчыаяфулэрэълабчбфълькдафбфш

k = 25: еуъъуыкжуфуызцыбъйщмпцъауъщусеуыюорцъянъйучцауъууруюыуусбпцъяюуткьпьщкяацаущкыйгяуауч

k = 26: дтщщтъйетутъжхъащишлохщятщштрдтъэнпхщюмщитцхятщттптэъттраохщюэтсйыоышйюяхятшйъивютятц

k = 27: гсшшсщидстсщефщяшзчкнфшюсшчспгсщьмофшэлшзсхфюсшссосьщсспянфшэьсриънъчиэюфюсчищзбэсюсх

k = 28: врччршзгрсршдушючжцймучэрчцровршылнучькчжрфуэрчррнрышрроюмучьырпзщмщцзьэуэрцзшжаьрэрф

k = 29: бпццпчжвпрпчгтчэцехилтцьпцхпнбпчъкмтцыйцепутьпцппмпъчппнэлтцыъпожшлшхжыьтьпхжчеяыпьпу

k = 30: аоххоцебопоцвсцьхдфзксхыохфомаоцщйлсхъихдотсыохоолощцоомьксхъщонечкчфеъысыофецдюъоыот

k = 31: янффнхданонхбрхыфгужйрфънфунлянхшикрфщзфгнсрънфннкншхннлыйрфщшнмдцйцудщърънудхгэщнънс

**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("а")

                # 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}: # е/ё

                    encoded[index] = 5

                else: encoded[index] = ord(self.preprocessed\_text[index]) - ord("а")

                self.encrypted[index] = (encoded[index] + encryption\_key) % self.m

                encrypted\_text[index] = chr(self.encrypted[index] + ord("а")) # 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('а'))

        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("а")

                # 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('а'))

        return code

        # return [chr(code[index] + ord('а')) for index in code if self.bool\_alpha\_table[index]]

# print(letters\_to\_code('мама'))

def given\_var():

    m = 32

    encrypted = 'юмуумфгямнмфапфъувтеипущмутмкюмфчзйпушжувмрпщмуммймчфммкъипушчмлгхихтгшщпщмтгфвьшмщмр'

    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("а") # to number

                # for index in range(len(decoded)):

                decoded[index] = (decoded[index] - key) % m

                decrypted[index] = chr(decoded[index] + ord('а'))  # back to ascii

                if index == len(encrypted)-1:

                    print(f'{key}, {"".join(decrypted)}\n')

                    f.write(f'{key}, {"".join(decrypted)}\n')