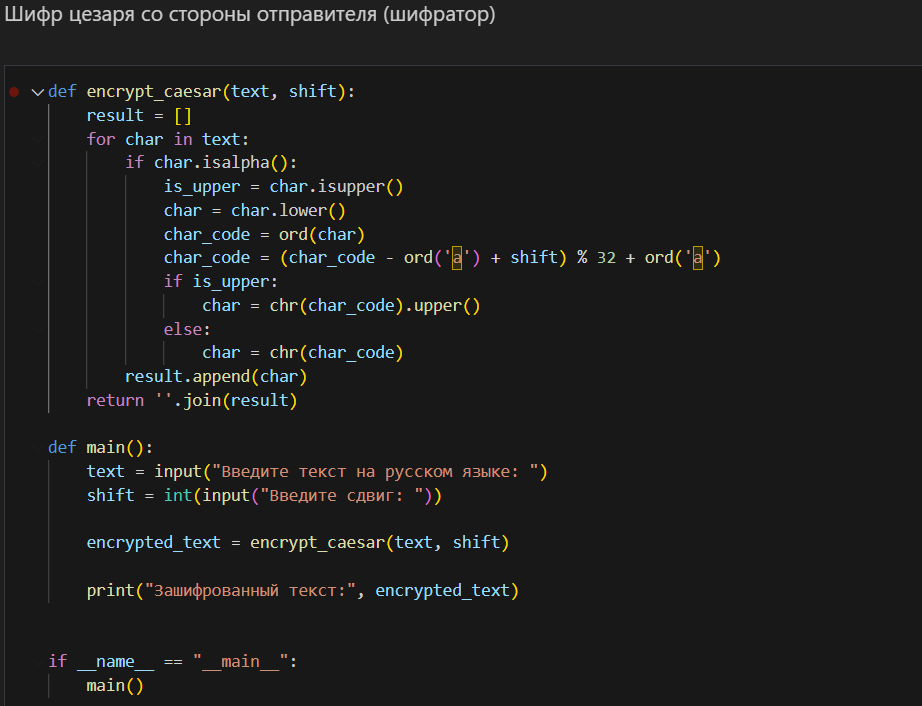
**Порядок выполнения работы:**

1. Реализовать в виде программы шифр Цезарь. Язык исходного текста русский или английский по выбору исполнителя.
2. Реализовать в виде программы атаку полным перебором ключа, используя для оценки правильности выбора ключа визуальный метод или исходный текст для автоматического сравнения результата дешифрования.
3. Предложить варианты усложнения шифра. Предложенные варианты оформить в виде алгоритма
4. Оценить криптографическую стойкость реализованного шифра.
   * + 1. Реализация шифра Цезарь

Отправитель отправляет открытый текст и сдвиг



Получатель вводит закрытый текст и сдвиг который был известен от пользователя лично

def decrypt\_caesar(ciphertext, shift):

    result = []

    for char in ciphertext:

        if char.isalpha():

            is\_upper = char.isupper()

            char = char.lower()

            char\_code = ord(char)

            char\_code = (char\_code - ord('а') - shift) % 32 + ord('а')

            if char\_code < ord('а'):

                char\_code += 32

            if is\_upper:

                char = chr(char\_code).upper()

            else:

                char = chr(char\_code)

        result.append(char)

    return ''.join(result)

def main():

    ciphertext = input("Введите зашифрованный текст на русском языке: ")

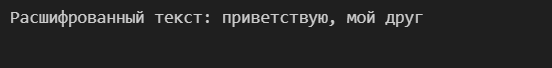
    shift = int(input("Введите сдвиг: "))

    decrypted\_text = decrypt\_caesar(ciphertext, shift)

    print("Расшифрованный текст:", decrypted\_text)

if \_\_name\_\_ == "\_\_main\_\_":

    main()



Описание кода:

Функция encrypt\_caesar выполняет шифрование Цезаря для каждой буквы в исходном тексте с использованием фиксированного сдвига.

Функция decrypt\_caesar представляет обратный процесс и выполняет расшифровку текста, используя тот же фиксированный сдвиг.

Пользователь вводит исходный текст и фиксированный сдвиг, который будет использоваться для шифрования и дешифрования.

Программа вызывает соответствующую функцию (encrypt\_caesar или decrypt\_caesar) в зависимости от выбора пользователя и выводит зашифрованный или расшифрованный текст.

2. Атака полным перебором ключа (bruteforce)

def decrypt\_caesar(ciphertext, shift):

    result = []

    for char in ciphertext:

        if char.isalpha():

            is\_upper = char.isupper()

            char = char.lower()

            char\_code = ord(char)

            char\_code = (char\_code - ord('а') - shift) % 32 + ord('а')

            if char\_code < ord('а'):

                char\_code += 32

            if is\_upper:

                char = chr(char\_code).upper()

            else:

                char = chr(char\_code)

        result.append(char)

    return ''.join(result)

def brute\_force\_caesar(ciphertext):

    for shift in range(32):  # Шифр Цезаря имеет 32 возможных сдвига

        decrypted\_text = decrypt\_caesar(ciphertext, shift)

        print(f"Сдвиг {shift}: {decrypted\_text}")

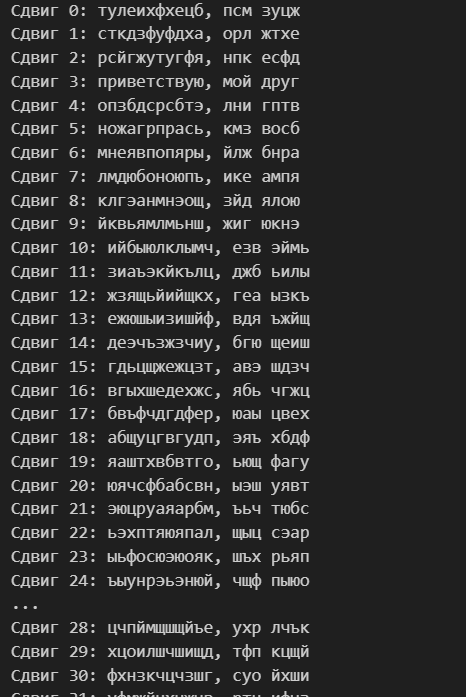
def main():

    ciphertext = input("Введите зашифрованный текст на русском языке: ")

    brute\_force\_caesar(ciphertext)

if \_\_name\_\_ == "\_\_main\_\_":

    main()



Описание кода:

Функция brute\_force\_decrypt представляет атаку полным перебором ключа для расшифровки зашифрованного текста. Она перебирает все возможные сдвиги в диапазоне от 1 до 32 (для русского алфавита) и выводит результаты для каждого из них.

Пользователь вводит зашифрованный текст, который нужно расшифровать.

Программа вызывает функцию brute\_force\_decrypt, которая перебирает все возможные сдвиги и выводит потенциальные варианты расшифрованного текста.

Пользователь может выбрать правильный результат или проанализировать текст, чтобы определить правильный сдвиг.

Программа выводит возможные варианты расшифрованного текста

3. Усложнение с помощью ключа

def encrypt\_caesar\_with\_keyphrase(text, keyphrase):

    result = []

    keyphrase = keyphrase.lower()

    key\_index = 0

    for char in text:

        if char.isalpha():

            is\_upper = char.isupper()

            char = char.lower()

            char\_code = ord(char)

            key\_char = keyphrase[key\_index % len(keyphrase)]

            key\_shift = ord(key\_char) - ord('а')

            char\_code = (char\_code - ord('а') + key\_shift) % 32 + ord('а')

            if is\_upper:

                char = chr(char\_code).upper()

            else:

                char = chr(char\_code)

            key\_index += 1

        result.append(char)

    return ''.join(result)

def decrypt\_caesar\_with\_keyphrase(ciphertext, keyphrase):

    result = []

    keyphrase = keyphrase.lower()

    key\_index = 0

    for char in ciphertext:

        if char.isalpha():

            is\_upper = char.isupper()

            char = char.lower()

            char\_code = ord(char)

            key\_char = keyphrase[key\_index % len(keyphrase)]

            key\_shift = ord(key\_char) - ord('а')

            char\_code = (char\_code - ord('а') - key\_shift) % 32 + ord('а')

            if char\_code < ord('а'):

                char\_code += 32

            if is\_upper:

                char = chr(char\_code).upper()

            else:

                char = chr(char\_code)

            key\_index += 1

        result.append(char)

    return ''.join(result)

def decrypt\_caesar\_with\_keyphrase(ciphertext, keyphrase):

    result = []

    keyphrase = keyphrase.lower()

    key\_index = 0

    for char in ciphertext:

        if char.isalpha():

            is\_upper = char.isupper()

            char = char.lower()

            char\_code = ord(char)

            key\_char = keyphrase[key\_index % len(keyphrase)]

            key\_shift = ord(key\_char) - ord('а')

            char\_code = (char\_code - ord('а') - key\_shift) % 32 + ord('а')

            if char\_code < ord('а'):

                char\_code += 32

            if is\_upper:

                char = chr(char\_code).upper()

            else:

                char = chr(char\_code)

            key\_index += 1

        result.append(char)

    return ''.join(result)

def main():

    text = input("Введите текст: ")

    keyphrase = input("Введите ключевую фразу: ")

    encrypted\_text = encrypt\_caesar\_with\_keyphrase(text, keyphrase)

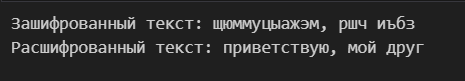
    decrypted\_text = decrypt\_caesar\_with\_keyphrase(encrypted\_text, keyphrase)

    print("Зашифрованный текст:", encrypted\_text)

    print("Расшифрованный текст:", decrypted\_text)

if \_\_name\_\_ == "\_\_main\_\_":

    main()



**Описание кода:**

Пользователь вводит исходный текст и ключевую фразу на русском языке.

Функция encrypt\_caesar\_with\_keyphrase берет каждый символ исходного текста и применяет к нему шифр Цезаря с сдвигом, определенным на основе соответствующей буквы ключевой фразы.

Зашифрованный текст сохраняется в переменной encrypted\_text.

Затем вызывается функция decrypt\_caesar\_with\_keyphrase, которая производит обратную операцию, используя ключевую фразу для расшифровки зашифрованного текста.

Расшифрованный текст сохраняется в переменной decrypted\_text.

Программа выводит зашифрованный и расшифрованный текст на экран

**Оценка системы Цезарь.**

Шифр Цезаря с фиксированным сдвигом является очень простым и не обладает высокой криптографической стойкостью. Он уязвим для атак полным перебором, какой бы длинной не была ключевая фраза. Криптографическая стойкость шифра Цезаря определяется количеством возможных ключей, которые можно перебрать, и в данном случае у нас всего 32 возможных ключа (сдвига).

Кроме того, шифр Цезаря уязвим к частотному анализу, так как он не меняет частоты букв в тексте. Если у атакующего есть достаточно большой зашифрованный текст, он может анализировать частоту букв и использовать ее для определения сдвига.

Для повышения криптографической стойкости шифра Цезаря можно использовать следующие методы:

Ключевая фраза: Вместо фиксированного сдвига можно использовать ключевую фразу. Это преобразует шифр Цезаря в шифр Виженера, который более стойкий к атакам.

Использование других алфавитов: Вместо стандартного алфавита можно использовать алфавиты с большим числом символов, что увеличит количество возможных сдвигов.

Использование более сложных шифров: Для более высокой криптографической стойкости лучше использовать более сложные алгоритмы шифрования, такие как AES или RSA.

Современные криптографические библиотеки: В реальных приложениях следует использовать современные криптографические библиотеки и алгоритмы, которые обеспечивают более высокую стойкость и безопасность.

Конспект:

1. \*\*Шифр Цезаря:\*\*

- Одна из самых простых форм шифрования.

- Основан на сдвиге букв в алфавите.

- Буквы сдвигаются на фиксированное количество позиций.

- Пример: Если сдвиг равен 3, "A" становится "D", "B" становится "E", и так далее.

- Применяется как шифр подстановки.

2. \*\*Шифр Виженера:\*\*

- Более сложный, чем Шифр Цезаря.

- Основан на использовании ключевого слова (или фразы).

- Ключ повторяется до длины открытого текста.

- Каждая буква ключа используется для сдвига соответствующей буквы открытого текста.

- Обеспечивает более сильное шифрование.

3. \*\*Шифр Скитала:\*\*

- Также известен как шифр цилиндра.

- Основан на обертывании текста вокруг цилиндра.

- Текст перематывается вдоль спирали, а затем считывается в другом направлении.

- Требует знания количества оборотов и направления чтения для расшифровки.

4. \*\*Шифр перестановки с использованием простых таблиц:\*\*

- Также называется шифром столбцов.

- Основан на перестановке символов в тексте.

- Текст записывается в таблицу, а затем символы считываются в определенном порядке.

- Ключ определяет порядок чтения.

- Пример: Если ключ - "2314", то текст будет читаться в порядке столбцов 2, 3, 1, 4.