МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 5

Конфиденциальность информации. Простейшие методы шифрования данных.

Выполнил студент 595 гр.

_____ А.В. Лаптев

Проверил:

_____ П.С. Ладыгин

Лабораторная работа защищена

«___» _____2023 г.

Оценка _____

Барнаул 2023

Цель работы: рассмотрение методов обеспечения конфиденциальности информации на примере простейшей криптографической защиты.

Задачи:

1. Создать копию «Модели угроз», разработанной в предыдущей лабораторной работе.
2. Согласно Варианту написать программу на любом языке программирования, которая зашифрует текст в вашем файле.
3. С использованием любого языка программирования написать программу, подбирающую ключ к шифротексту методом подбора или с помощью частотного анализа.

Описание алгоритма:

1. Задаем алфавит допустимых символов.
2. Выбираем режим работы программы. Если выбрано шифрование, то переход к п.18.

Расшифровка

3. Открываем файл для расшифровки.
4. Считываем содержимое файла посимвольно.
5. Если текущий символ принадлежит алфавиту, то переход к п.7.
6. Переходим к следующему символу.
7. Добавляем текущий символ в массив символов.
8. Задаем ключ для дешифровки равным 0.

Цикл

9. Если текст расшифрован, то переход к п.17.
10. Увеличиваем значение ключа на 1.

Внутренний цикл

11. Берем i-ый член массива символов и сравниваем с j-ым членом алфавита. Если символы не совпали, то увеличиваем счетчик символов на 1 и переход к п.11.
12. Вычисляем разность между номером символа и ключом.
13. Смещаем каждый символ в массиве на эту разность.
14. Заносим смещенный символ в новый массив символов.

Конец внутреннего цикла

Конец цикла

15. Выводим полученный массив

16. Спрашиваем пользователя о правильности расшифровки. Если неправильно, то переход к п.10.
17. Записываем полученный массив в файл.

Шифрование

18. Просим пользователя ввести ключ для шифрования.
19. Открываем файл, который нужно зашифровать.
20. Считываем содержимое файла посимвольно.

Цикл

21. Перебираем все члены алфавита. Если пройден последний член, то переход к п.26.
22. Если член алфавита не совпал с символом, то переход к п.21.
23. Смещаем символ на значение ключа в алфавите.
24. Присваиваем переменной смещенный символ.
25. Заносим этот символ в массив символов зашифрованного текста.

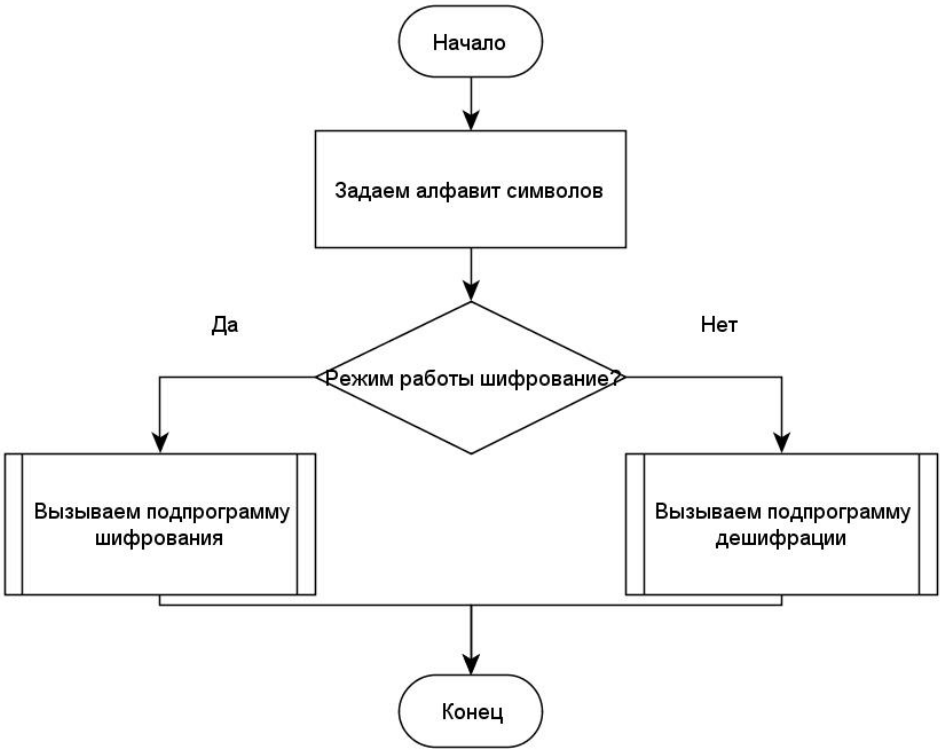
Конец цикла

26. Записываем полученный массив в файл.

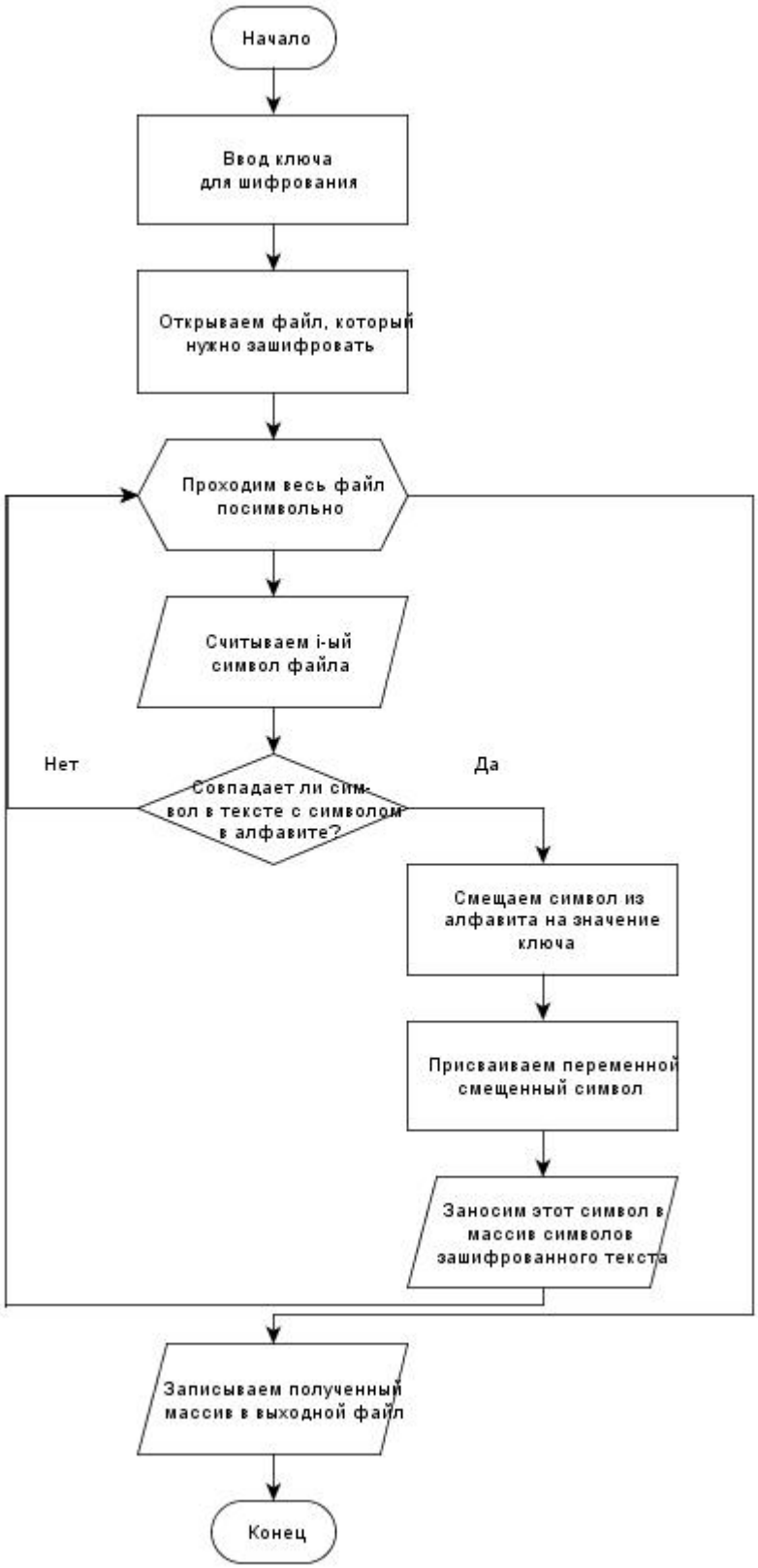
Конец

Блок-схема алгоритма:

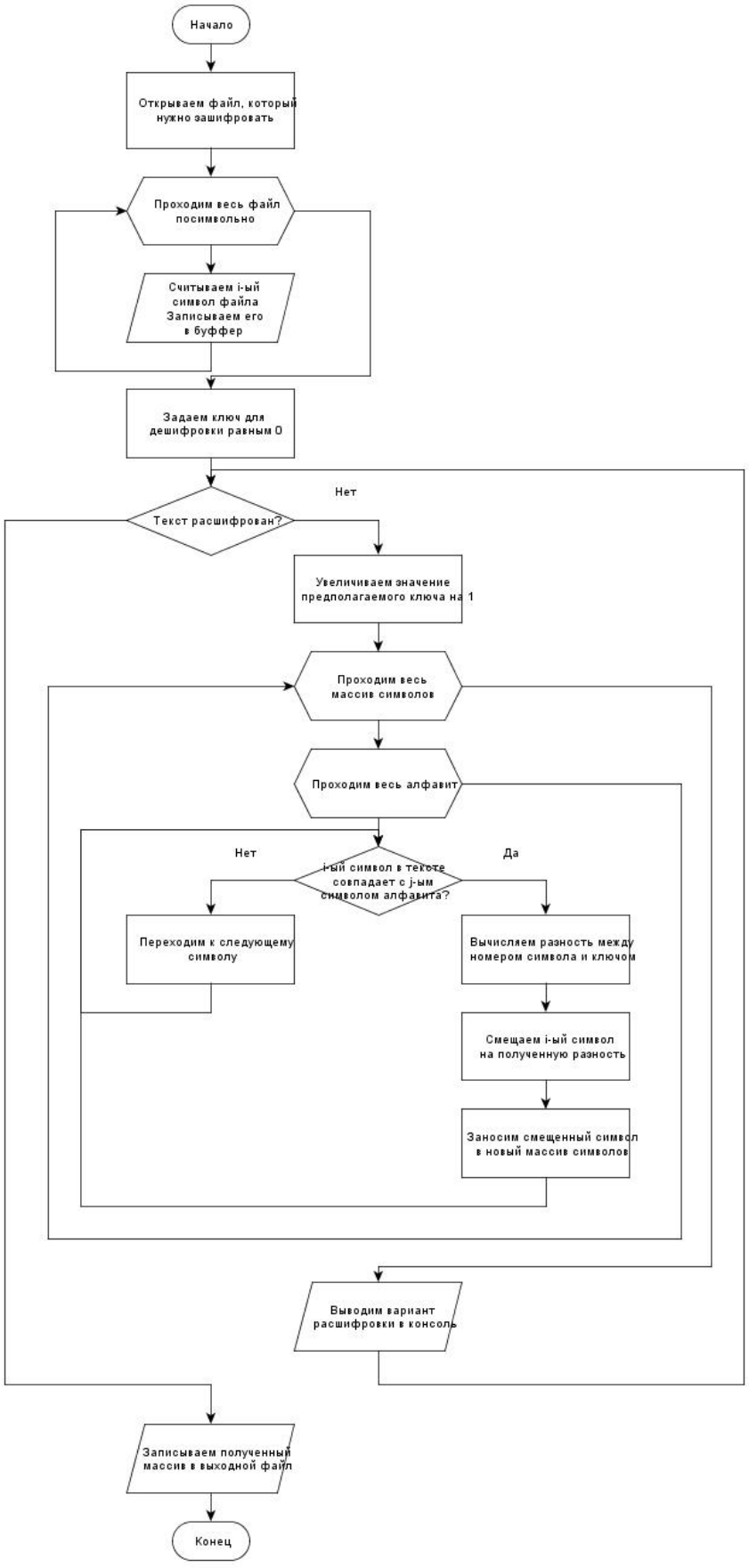
Основная программа:



Шифрование:


Начало
Ввод ключа для шифрования
Открываем файл, который нужно зашифровать
Проходим весь файл посимвольно
Считываем i-ый символ файла
Нет
Совпадает ли символ в тексте с символом в алфавите?
Да
Смещаем символ из алфавита на значение ключа
Присваиваем переменной смещенный символ
Заносим этот символ в массив символов зашифрованного текста
Записываем полученный массив в выходной файл
Конец

Дешифрование:



Листинг программы:

```
def coder():
    buffer_char = []
    key = int(input("Введите ключ: "))
    while key < 0:
        key = int(input("Введите ключ: "))

    with open("input.txt", "r") as input_file:
        while True:
            char = input_file.read(1)
            if not char:
                break
            for i in range(len(alphabet)):
                if char == alphabet[i]:
                    bias = i + key
                    if 0 <= i <= 25:
                        if bias <= 25:
                            coder_char = alphabet[bias]
                        else:
                            coder_char = alphabet[bias - 26]
                    elif 26 <= i <= 51:
                        if bias <= 51:
                            coder_char = alphabet[bias]
                        else:
                            coder_char = alphabet[bias - 26]
                    elif 52 <= i <= 84:
                        if bias <= 84:
                            coder_char = alphabet[bias]
                        else:
                            coder_char = alphabet[bias - 33]
                    else:
                        if bias <= len(alphabet) - 1:
                            coder_char = alphabet[bias]
                        else:
```

```
                coder_char = alphabet[bias - 33]
            buffer_char.append(coder_char)
            break

    with open("output.txt", "w") as output_file:
        for i in range(len(buffer_char)):
            output_file.write(buffer_char[i])



def decoder():
    buffer_char = []
    with open("output.txt", "r") as input_file:
        while True:
            char = input_file.read(1)
            if not char:
                break
            buffer_char.append(char)

    right_text = False
    key = 0
    while right_text == False:
        key += 1
        new_buffer_char = []
        for j in range(len(buffer_char)):
            for i in range(len(alphabet)):
                if buffer_char[j] == alphabet[i]:
                    subb = i - key
                    if 0 <= i <= 25:
                        if subb >= 0:
                            decoder_char = alphabet[subb]
                        else:
                            bias = 0 - subb
                            decoder_char = alphabet[26 - bias]
                    elif 26 <= i <= 51:
```

```
                if subb >= 26:
                    decoder_char = alphabet[subb]
                else:
                    bias = 26 - subb
                    decoder_char = alphabet[52 - bias]
            elif 52 <= i <= 84:
                if subb >= 52:
                    decoder_char = alphabet[subb]
                else:
                    bias = 52 - subb
                    decoder_char = alphabet[85 - bias]
            else:
                if subb >= 85:
                    decoder_char = alphabet[subb]
                else:
                    bias = 85 - subb
                    decoder_char = alphabet[len(alphabet) - bias]
            new_buffer_char.append(decoder_char)
            break
    print(new_buffer_char)

    check = str(input("Расшифрован ли текст в файле? Y/N\t"))
    if check == "Y" or check == "y" or check == "Yes" or check == "yes" or check
== "YES":
        right_text == True
        break
    elif check == "N" or check == "n" or check == "No" or check == "no" or check
== "NO":
        right_text == False
    # else:
    #     check = str(input("Расшифрован ли текст в файле? Y/N\t"))
    #     key -= 1

with open("input_new.txt", "w") as output_file:
```

```
        for i in range(len(new_buffer_char)):
            output_file.write(new_buffer_char[i])


if __name__ == '__main__':
    alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',
                'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
                'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'а', 'б', 'в', 'г', 'д', 'е', 'ё', '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', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', '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']
    print("1. Кодирование шифром Цезаря.")
    print("2. Декодирование шифра Цезаря.")
    variant = int(input("Введите вариант работы программы: "))
    while variant < 1 or variant > 2:
        variant = int(input("Введите вариант работы программы: "))

    if variant == 1:
        coder()
    elif variant == 2:
        decoder()
```

Вывод: в ходе лабораторной работы были рассмотрены методы обеспечения конфиденциальности информации на примере простейшей криптографической защиты.

Ответы на контрольные вопросы:

1. Какой метод шифрования наиболее эффективен на сегодняшний день?

Ответ: Из представленных в методичке - шифр Виженера, а вообще - хеширование очень распространено, цифровые подписи, алгоритмы симметричного шифрования - AES, ассиметричные - ECC.

2. В каких сферах деятельности обычного пользователя встречается потребность в шифровании данных?

Ответ: Банковские сети, аккаунты в мессенджерах, пароли от аккаунтов в социальных сетях,