Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЁТ
к лабораторной работе №2
на тему

ШИФРЫ ЦЕЗАРЯ И ВИЖЕНЕРА

Выполнил: студент гр.253504
Фроленко К.Ю.
Проверил: ассистент кафедры информатики
Герчик А.В.

Минск 2025

СОДЕРЖАНИЕ

1 ФОРМУЛИРОВКА ЗАДАЧИ

В данной работе необходимо разработать программное средство для шифрования и дешифрования текстовых файлов с использованием Шифра Цезаря и Шифра Виженера. Реализация должна обеспечивать возможность обработки текстовых данных в удобной форме, позволяя пользователю выбирать метод шифрования, указывать параметры кодирования и получать результат в наглядном виде.

Шифр Цезаря должен позволять задавать величину сдвига, которая определяет, на сколько позиций будет смещен каждый символ исходного текста в алфавите. Для Шифра Виженера необходимо реализовать механизм работы с ключом, который определяет последовательность сдвигов для шифрования и расшифрования текста. Оба метода должны поддерживать работу с русским и английским алфавитами, учитывать регистр символов и корректно обрабатывать неалфавитные символы, оставляя их без изменений.

Программа должна обеспечивать возможность загрузки текста из файлов и сохранения обработанных данных, что позволит избежать необходимости ввода текста вручную. Графический интерфейс должен предоставлять удобные средства выбора алгоритма, задания параметров и управления процессом обработки данных. Важно, чтобы пользователь мог интуитивно взаимодействовать с приложением, легко переключаясь между методами шифрования и дешифрования, а также выбирать необходимые параметры без сложных настроек.

Дополнительно необходимо предусмотреть обработку ошибок, связанных с некорректными входными данными, и реализовать систему уведомлений для информирования пользователя о ходе выполнения операций. Реализация должна быть удобной, надежной и обеспечивать корректную работу алгоритмов в различных сценариях использования.

2 ХОД РАБОТЫ

Для реализации программного средства шифрования и дешифрования текстовых файлов разработано графическое приложение, позволяющее пользователю выбирать метод шифрования, задавать параметры, загружать текстовые файлы и сохранять обработанные данные. Приложение поддерживает два алгоритма: Шифр Цезаря и Шифр Виженера, а также работу с текстами на русском и английском языках.

Графический интерфейс включает элементы для выбора метода шифрования, переключения между режимами шифрования и дешифрования, задания параметров, загрузки входного файла и сохранения результата. Визуальное представление интерфейса программы приведено на Рисунке 1.

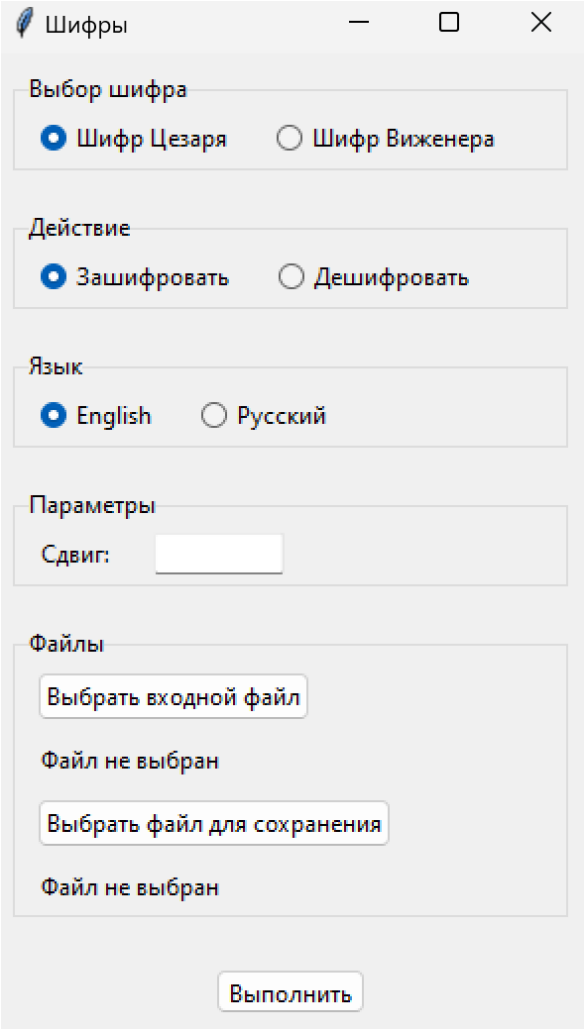

Рисунок 1 – Интерфейс программы для работы с шифрами

В основе работы Шифра Цезаря лежит сдвиг символов входного текста на указанное пользователем количество позиций в пределах выбранного алфавита. Для шифрования и дешифрования текста разработаны функции *caesar_encrypt* и *caesar_decrypt*, выполняющие преобразование символов. В

процессе обработки текста символы латинского или кириллического алфавита изменяются в соответствии с указанным сдвигом, а остальные символы, такие как цифры, пробелы и знаки препинания, остаются неизменными.

Шифр Виженера использует заданное пользователем ключевое слово, определяющее последовательность сдвигов для каждого символа текста. Реализованы функции *vigenere_encrypt* и *vigenere_decrypt*, которые выполняют кодирование и декодирование с учетом ключа. Данный метод позволяет использовать более сложный способ шифрования, так как каждый символ текста заменяется другим символом с учетом соответствующего сдвига, вычисляемого на основе ключевого слова.

Для удобства работы с текстовыми файлами в программу добавлены функции загрузки входного текста из файла и сохранения обработанного текста в новый файл. Это позволяет пользователю работать с шифрованием и дешифрованием без необходимости ввода текста вручную.

На Рисунке 2 представлен пример работы программы. Исходный текст был зашифрован с помощью Шифра Виженера с заданным ключевым словом, а затем успешно расшифрован обратно, что подтверждает корректность работы алгоритмов.

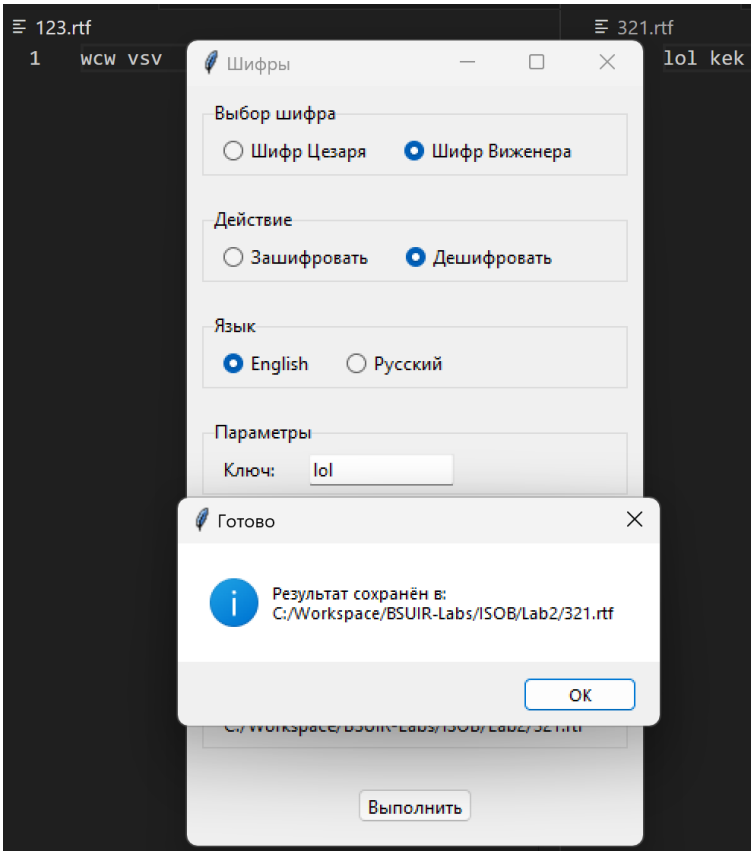

Рисунок 2 – Пример работы программы

ЗАКЛЮЧЕНИЕ

В ходе данной работы было разработано программное средство для шифрования и дешифрования текстовых файлов с использованием Шифра Цезаря и Шифра Виженера. Реализация обеспечила возможность выбора метода шифрования, задания параметров, загрузки и сохранения текстовых данных, а также интуитивно понятное управление процессом обработки.

На первом этапе была определена архитектура программы, выбраны основные алгоритмы и реализован графический интерфейс, позволяющий пользователю взаимодействовать с программой без необходимости работы с командной строкой. Далее был разработан алгоритм Шифра Цезаря, который выполняет сдвиг символов в пределах алфавита на заданное пользователем число позиций, а затем добавлена поддержка Шифра Виженера, использующего ключевое слово для формирования последовательности сдвигов. Оба метода корректно работают с русским и английским языками, учитывают регистр символов и сохраняют неизменными неалфавитные символы.

Для удобства работы реализована возможность загрузки входного текста из файла и сохранения результата в отдельный файл. В процессе тестирования проверена работоспособность обоих методов шифрования и дешифрования, корректность обработки различных языков и символов, а также работоспособность программы при вводе различных параметров. Дополнительно была добавлена система уведомлений, позволяющая информировать пользователя об ошибках и ходе выполнения операций.

Таким образом, поставленная задача была успешно выполнена. Программа позволяет эффективно выполнять шифрование и расшифровку текстовых данных, корректно обрабатывает входные параметры и обеспечивает удобство работы за счет графического интерфейса. Полученные результаты подтверждают надежность реализации и соответствие требованиям, предъявленным к функционалу программного средства.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программного кода

```
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter import ttk

EN_UPPER = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
EN_LOWER = EN_UPPER.lower()

RU_UPPER = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"
RU_LOWER = RU_UPPER.lower()

def get_alphabets(language):
    if language == "English":
        return EN_UPPER, EN_LOWER
    else:
        return RU_UPPER, RU_LOWER

def caesar_encrypt(text, shift, language):
    alph_upper, alph_lower = get_alphabets(language)
    len_alpha = len(alph_upper)
    result = ""
    for char in text:
        if char in alph_upper:
            index = alph_upper.find(char)
            new_index = (index + shift) % len_alpha
            result += alph_upper[new_index]
        elif char in alph_lower:
            index = alph_lower.find(char)
            new_index = (index + shift) % len_alpha
            result += alph_lower[new_index]
        else:
            result += char
    return result

def caesar_decrypt(text, shift, language):
    return caesar_encrypt(text, -shift, language)

def generate_key(text, key, language):
    alph_upper, alph_lower = get_alphabets(language)
    key_filtered = "".join([ch for ch in key if ch in alph_upper or ch in
alph_lower])
    if not key_filtered:
        return ""
    expanded_key = ""
    key_index = 0
    for char in text:
        if char in alph_upper or char in alph_lower:
            expanded_key += key_filtered[key_index % len(key_filtered)]
            key_index += 1
        else:
            expanded_key += char
    return expanded_key

def vigenere_encrypt(text, key, language):
    alph_upper, alph_lower = get_alphabets(language)
    len_alpha = len(alph_upper)
    expanded_key = generate_key(text, key, language)
    if not expanded_key:
```

```
        return text
    cipher_text = ""
    for t_char, k_char in zip(text, expanded_key):
        if t_char in alph_upper:
            shift = alph_upper.find(k_char.upper())
            new_index = (alph_upper.find(t_char) + shift) % len_alpha
            cipher_text += alph_upper[new_index]
        elif t_char in alph_lower:
            shift = alph_upper.find(k_char.upper())
            new_index = (alph_lower.find(t_char) + shift) % len_alpha
            cipher_text += alph_lower[new_index]
        else:
            cipher_text += t_char
    return cipher_text

def vigenere_decrypt(text, key, language):
    alph_upper, alph_lower = get_alphabets(language)
    len_alpha = len(alph_upper)
    expanded_key = generate_key(text, key, language)
    if not expanded_key:
        return text
    original_text = ""
    for t_char, k_char in zip(text, expanded_key):
        if t_char in alph_upper:
            shift = alph_upper.find(k_char.upper())
            new_index = (alph_upper.find(t_char) - shift + len_alpha) %
len_alpha
            original_text += alph_upper[new_index]
        elif t_char in alph_lower:
            shift = alph_upper.find(k_char.upper())
            new_index = (alph_lower.find(t_char) - shift + len_alpha) %
len_alpha
            original_text += alph_lower[new_index]
        else:
            original_text += t_char
    return original_text

def read_file(filename):
    try:
        with open(filename, 'r', encoding="utf-8") as f:
            return f.read()
    except Exception as e:
        messagebox.showerror("Ошибка чтения файла", str(e))
        return None

def write_file(filename, text):
    try:
        with open(filename, 'w', encoding="utf-8") as f:
            f.write(text)
    except Exception as e:
        messagebox.showerror("Ошибка записи файла", str(e))

class CipherApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Шифры")
        self.geometry("300x500")
        self.minsize(300, 500)
        self.cipher_type = tk.StringVar(value="Цезаря")
        self.action = tk.StringVar(value="Зашифровать")
        self.language = tk.StringVar(value="English")
        self.shift_value = tk.StringVar()
        self.vigenere_key = tk.StringVar()
```

```
        self.input_filename = tk.StringVar(value="Файл не выбран")
        self.output_filename = tk.StringVar(value="Файл не выбран")
        self.create_widgets()

    def create_widgets(self):
        frame_cipher = ttk.LabelFrame(self, text="Выбор шифра")
        frame_cipher.pack(padx=10, pady=10, fill="x")
        rb_caesar = ttk.Radiobutton(frame_cipher, text="Шифр Цезаря",
variable=self.cipher_type, value="Цезаря", command=self.toggle_params)
        rb_vigenere = ttk.Radiobutton(frame_cipher, text="Шифр Виженера",
variable=self.cipher_type, value="Виженера", command=self.toggle_params)
        rb_caesar.pack(side="left", padx=10, pady=5)
        rb_vigenere.pack(side="left", padx=10, pady=5)
        frame_action = ttk.LabelFrame(self, text="Действие")
        frame_action.pack(padx=10, pady=10, fill="x")
        rb_encrypt = ttk.Radiobutton(frame_action, text="Зашифровать",
variable=self.action, value="Зашифровать")
        rb_decrypt = ttk.Radiobutton(frame_action, text="Дешифровать",
variable=self.action, value="Дешифровать")
        rb_encrypt.pack(side="left", padx=10, pady=5)
        rb_decrypt.pack(side="left", padx=10, pady=5)
        frame_language = ttk.LabelFrame(self, text="Язык")
        frame_language.pack(padx=10, pady=10, fill="x")
        rb_en = ttk.Radiobutton(frame_language, text="English",
variable=self.language, value="English")
        rb_ru = ttk.Radiobutton(frame_language, text="Русский",
variable=self.language, value="Русский")
        rb_en.pack(side="left", padx=10, pady=5)
        rb_ru.pack(side="left", padx=10, pady=5)
        self.frame_params = ttk.LabelFrame(self, text="Параметры")
        self.frame_params.pack(padx=10, pady=10, fill="x")
        self.label_shift = ttk.Label(self.frame_params, text="Сдвиг:")
        self.entry_shift = ttk.Entry(self.frame_params,
textvariable=self.shift_value, width=10)
        self.label_key = ttk.Label(self.frame_params, text="Ключ:")
        self.entry_key = ttk.Entry(self.frame_params,
textvariable=self.vigenere_key, width=15)
        self.toggle_params()
        frame_files = ttk.LabelFrame(self, text="Файлы")
        frame_files.pack(padx=10, pady=10, fill="x")
        btn_input = ttk.Button(frame_files, text="Выбрать входной файл",
command=self.select_input_file)
        btn_input.pack(padx=10, pady=5, anchor="w")
        self.label_input = ttk.Label(frame_files,
textvariable=self.input_filename)
        self.label_input.pack(padx=10, pady=5, anchor="w")
        btn_output = ttk.Button(frame_files, text="Выбрать файл для
сохранения", command=self.select_output_file)
        btn_output.pack(padx=10, pady=5, anchor="w")
        self.label_output = ttk.Label(frame_files,
textvariable=self.output_filename)
        self.label_output.pack(padx=10, pady=5, anchor="w")
        btn_process = ttk.Button(self, text="Выполнить",
command=self.process_file)
        btn_process.pack(padx=10, pady=15)

    def toggle_params(self, *args):
        for widget in self.frame_params.winfo_children():
            widget.pack_forget()
        cipher = self.cipher_type.get()
        if cipher == "Цезаря":
            self.label_shift.pack(side="left", padx=10, pady=5)
            self.entry_shift.pack(side="left", padx=10, pady=5)
```

```
        else:
            self.label_key.pack(side="left", padx=10, pady=5)
            self.entry_key.pack(side="left", padx=10, pady=5)

    def select_input_file(self):
        filename = filedialog.askopenfilename(title="Выберите входной файл",
filetypes=[("Все файлы", "*.*")])
        if filename:
            self.input_filename.set(filename)

    def select_output_file(self):
        filename = filedialog.asksaveasfilename(title="Выберите файл для
сохранения", defaultextension=".txt", filetypes=[("Все файлы", "*.*")])
        if filename:
            self.output_filename.set(filename)

    def process_file(self):
        in_file = self.input_filename.get()
        out_file = self.output_filename.get()
        if in_file == "Файл не выбран" or out_file == "Файл не выбран":
            messagebox.showwarning("Внимание", "Пожалуйста, выберите входной и
выходной файлы.")
            return
        text = read_file(in_file)
        if text is None:
            return
        cipher = self.cipher_type.get()
        action = self.action.get()
        lang = self.language.get()
        result = ""
        if cipher == "Цезаря":
            try:
                shift = int(self.shift_value.get())
            except ValueError:
                messagebox.showerror("Ошибка", "Введите корректное числовое
значение для сдвига.")
                return
            if action == "Зашифровать":
                result = caesar_encrypt(text, shift, lang)
            else:
                result = caesar_decrypt(text, shift, lang)
        else:
            key = self.vigenere_key.get().strip()
            if not key:
                messagebox.showerror("Ошибка", "Введите ключ для шифра
Виженера.")
                return
            alph_upper, alph_lower = get_alphabets(lang)
            if any(ch not in alph_upper and ch not in alph_lower for ch in key):
                messagebox.showerror("Ошибка", "Ключ должен состоять только из
букв выбранного языка.")
                return
            if action == "Зашифровать":
                result = vigenere_encrypt(text, key, lang)
            else:
                result = vigenere_decrypt(text, key, lang)
        write_file(out_file, result)
        messagebox.showinfo("Готово", f"Результат сохранён в:\n{out_file}")

if __name__ == "__main__":
    app = CipherApp()
    app.mainloop()
```