Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Институт информатики и вычислительной техники

Кафедра прикладной математики и кибернетики

# Лабораторная работа №8 по дисциплине

**Прикладная стеганография**

Выполнил:

студент гр.МГ-411 Шевельков П.С.

ФИО студента

«15» мая 2025 г.

Новосибирск 2025 г.

# Задание на лабораторную работу:

Разработать приложение с графическим интерфейсом для встраивания произвольной информации в заданный текст. Текстовые данные взять из открытой базы произведений: GuttenbergProject https://dev.gutenberg.org/

Метод встраивания взять на выбор. Подготовить доклад и рассказать о выбранном методе.

Отчет по работе должен содержать описание метода и основных функций программы, полученные результаты.

# Результаты работы программы:

# 

рисунок 1. Интерфейс программы.

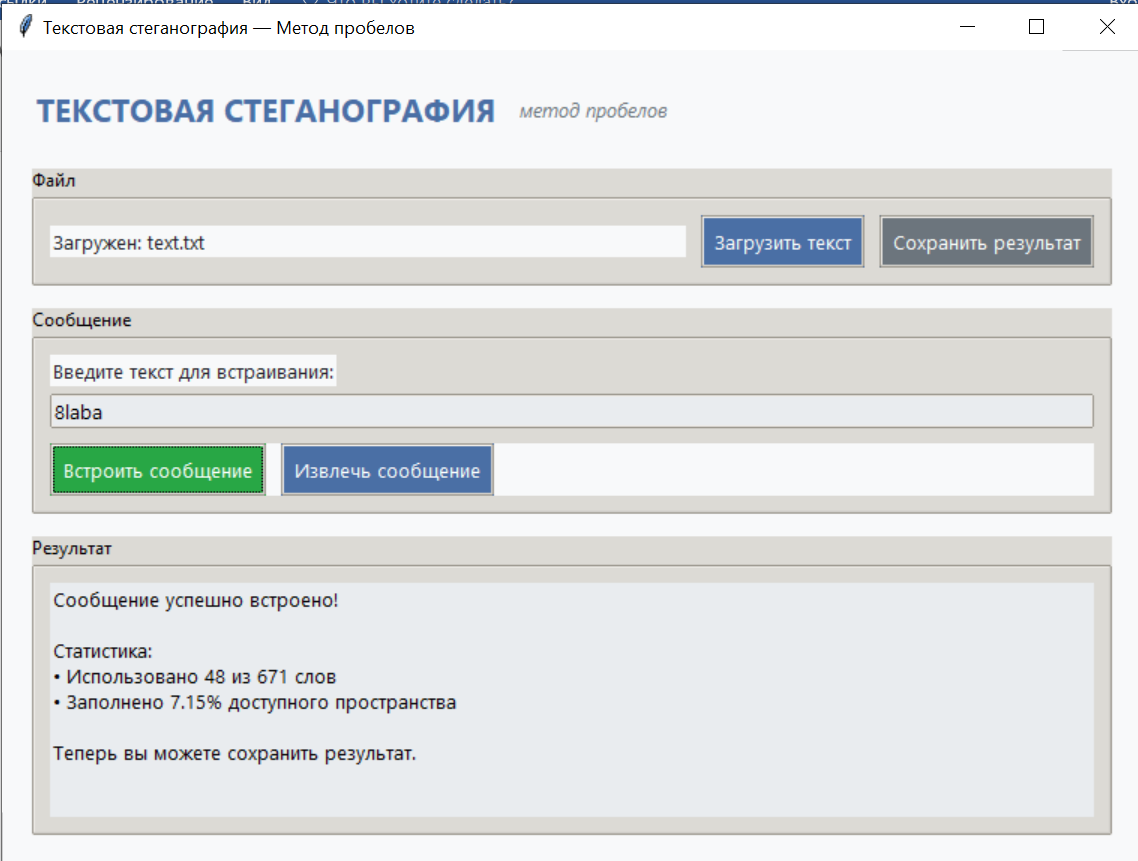


рисунок 2 Встраивание сообщения.

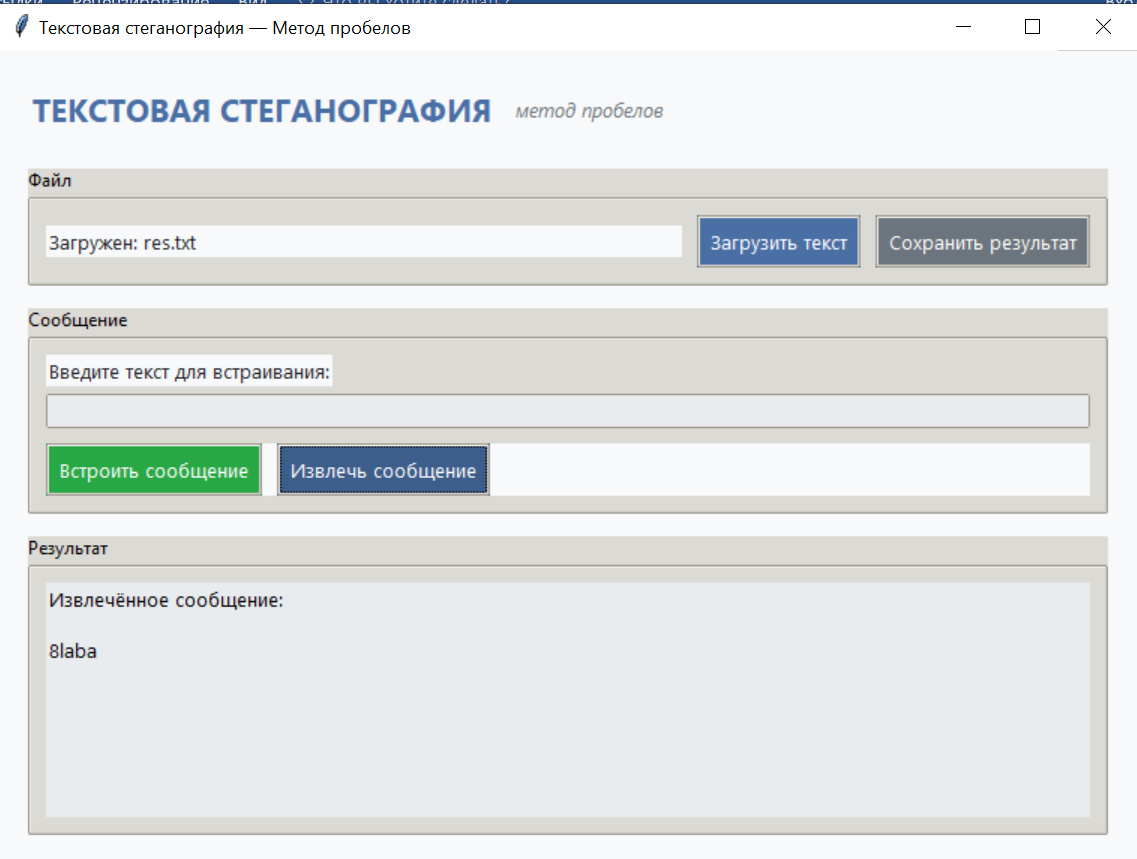


рисунок 3 Извлечение сообщения из сохранённого файла.

**Описание работы**

Данная программа реализует текстовую стеганографию с использованием метода пробелов. Она позволяет пользователю скрывать сообщения внутри обычного текста, изменяя количество пробелов между словами. Программа написана на Python с использованием библиотеки Tkinter для создания графического интерфейса.

Основные функции программы:

1. **Загрузка текста**: Пользователь может загрузить текстовый файл, который будет использоваться для встраивания сообщения. Программа отображает статус загрузки и позволяет пользователю видеть, какой файл был загружен.
2. **Встраивание сообщения**: Пользователь вводит сообщение, которое он хочет скрыть. Программа преобразует это сообщение в двоичный код и встраивает его в текст, добавляя пробелы между словами в зависимости от значений битов (один пробел для '0' и два пробела для '1').
3. **Извлечение сообщения**: Программа может извлекать скрытое сообщение из текста, анализируя количество пробелов между словами. Если сообщение успешно извлечено, оно отображается в текстовом поле.
4. **Сохранение результата**: Пользователь может сохранить изменённый текст с встроенным сообщением в новый файл.
5. **Интерфейс**: Программа имеет современный и удобный интерфейс, который включает в себя кнопки для загрузки, сохранения, встраивания и извлечения сообщений, а также текстовые поля для отображения статуса и результатов.

**Вывод**

Программа "Текстовая стеганография — Метод пробелов" демонстрирует эффективный способ скрытия информации в текстах, используя простые методы работы с пробелами. Она предоставляет пользователю интуитивно понятный интерфейс для выполнения операций, связанных с загрузкой текста, встраиванием и извлечением сообщений. Данная реализация может быть полезна для тех, кто интересуется стеганографией и хочет изучить основы скрытия информации в текстовых данных. Программа также может быть расширена для поддержки других методов стеганографии и улучшения пользовательского опыта.

## **Ссылка на программу:**

<https://github.com/bothyD/steganograf>

**Листинг**

import tkinter as tk

from tkinter import filedialog, messagebox

from tkinter import ttk

import os

class TextStegoApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Текстовая стеганография — Метод пробелов")

        self.text\_data = ""

        self.setup\_theme()

        self.setup\_ui()

    def setup\_theme(self):

        # Современная цветовая схема

        self.colors = {

            "primary": "#4a6fa5",       # Основной цвет

            "primary\_dark": "#3d5d8a",  # Темный вариант основного

            "secondary": "#6c757d",     # Вторичный цвет

            "bg": "#f8f9fa",            # Фон

            "text": "#212529",          # Текст

            "light\_accent": "#e9ecef",  # Светлый акцент

            "success": "#28a745",       # Успешное действие

            "warning": "#ffc107"        # Предупреждение

        }

        # Настройка стиля

        self.style = ttk.Style()

        self.style.theme\_use('clam')

        # Кнопки

        self.style.configure('TButton',

                            background=self.colors["primary"],

                            foreground='white',

                            font=('Segoe UI', 10),

                            padding=6)

        self.style.map('TButton',

                     background=[('active', self.colors["primary\_dark"])])

        # Метки

        self.style.configure('TLabel',

                            background=self.colors["bg"],

                            foreground=self.colors["text"],

                            font=('Segoe UI', 10))

        # Поля ввода

        self.style.configure('TEntry',

                            fieldbackground=self.colors["light\_accent"],

                            font=('Segoe UI', 10))

        # Рамки

        self.style.configure('TFrame', background=self.colors["bg"])

        # Специальные стили

        self.style.configure('Primary.TButton',

                            background=self.colors["primary"],

                            foreground='white')

        self.style.configure('Success.TButton',

                            background=self.colors["success"],

                            foreground='white')

        self.style.map('Success.TButton',

                     background=[('active', '#218838')])

        self.style.configure('Secondary.TButton',

                            background=self.colors["secondary"],

                            foreground='white')

        self.style.map('Secondary.TButton',

                     background=[('active', '#5a6268')])

    def setup\_ui(self):

        self.root.geometry("700x500")

        self.root.configure(bg=self.colors["bg"])

        # Создаем основную структуру

        main\_frame = ttk.Frame(self.root, padding="20 20 20 20", style='TFrame')

        main\_frame.pack(fill=tk.BOTH, expand=True)

        # Заголовок

        header\_frame = ttk.Frame(main\_frame, style='TFrame')

        header\_frame.pack(fill=tk.X, pady=(0, 20))

        title\_label = tk.Label(header\_frame,

                              text="ТЕКСТОВАЯ СТЕГАНОГРАФИЯ",

                              font=('Segoe UI', 16, 'bold'),

                              bg=self.colors["bg"],

                              fg=self.colors["primary"])

        title\_label.pack(side=tk.LEFT)

        subtitle\_label = tk.Label(header\_frame,

                                text="метод пробелов",

                                font=('Segoe UI', 10, 'italic'),

                                bg=self.colors["bg"],

                                fg=self.colors["secondary"])

        subtitle\_label.pack(side=tk.LEFT, padx=(10, 0), pady=8)

        # Секция загрузки файла

        file\_frame = ttk.LabelFrame(main\_frame, text="Файл", padding="10 10 10 10")

        file\_frame.pack(fill=tk.X, pady=(0, 15))

        self.file\_status = tk.StringVar(value="Файл не загружен")

        file\_status\_label = ttk.Label(file\_frame, textvariable=self.file\_status)

        file\_status\_label.pack(side=tk.LEFT, fill=tk.X, expand=True)

        load\_btn = ttk.Button(file\_frame, text="Загрузить текст", command=self.load\_text, style='TButton')

        load\_btn.pack(side=tk.LEFT, padx=(10, 0))

        save\_btn = ttk.Button(file\_frame, text="Сохранить результат", command=self.save\_text, style='Secondary.TButton')

        save\_btn.pack(side=tk.LEFT, padx=(10, 0))

        # Секция сообщения

        message\_frame = ttk.LabelFrame(main\_frame, text="Сообщение", padding="10 10 10 10")

        message\_frame.pack(fill=tk.X, pady=(0, 15))

        ttk.Label(message\_frame, text="Введите текст для встраивания:").pack(anchor=tk.W, pady=(0, 5))

        self.message\_entry = ttk.Entry(message\_frame, width=80, font=('Segoe UI', 10))

        self.message\_entry.pack(fill=tk.X, pady=(0, 10))

        btn\_frame = ttk.Frame(message\_frame, style='TFrame')

        btn\_frame.pack(fill=tk.X)

        embed\_btn = ttk.Button(btn\_frame, text="Встроить сообщение",

                             command=self.embed\_message,

                             style='Success.TButton')

        embed\_btn.pack(side=tk.LEFT)

        extract\_btn = ttk.Button(btn\_frame, text="Извлечь сообщение",

                               command=self.extract\_message,

                               style='Primary.TButton')

        extract\_btn.pack(side=tk.LEFT, padx=(10, 0))

        # Секция результата

        result\_frame = ttk.LabelFrame(main\_frame, text="Результат", padding="10 10 10 10")

        result\_frame.pack(fill=tk.BOTH, expand=True)

        # Создаем текстовый виджет вместо метки для лучшего отображения

        self.output\_text = tk.Text(result\_frame,

                                 wrap=tk.WORD,

                                 bg=self.colors["light\_accent"],

                                 relief=tk.FLAT,

                                 font=('Segoe UI', 10),

                                 height=10)

        self.output\_text.pack(fill=tk.BOTH, expand=True)

        self.output\_text.config(state=tk.DISABLED)

        # Статус бар

        status\_frame = ttk.Frame(main\_frame, style='TFrame')

        status\_frame.pack(fill=tk.X, pady=(10, 0))

        self.status\_var = tk.StringVar(value="Готов к работе")

        status\_label = ttk.Label(status\_frame, textvariable=self.status\_var, font=('Segoe UI', 9, 'italic'))

        status\_label.pack(side=tk.LEFT)

    def update\_output(self, text):

        self.output\_text.config(state=tk.NORMAL)

        self.output\_text.delete(1.0, tk.END)

        self.output\_text.insert(tk.END, text)

        self.output\_text.config(state=tk.DISABLED)

    def load\_text(self):

        filepath = filedialog.askopenfilename(filetypes=[("Text files", "\*.txt"), ("All files", "\*.\*")])

        if filepath:

            try:

                with open(filepath, "r", encoding="utf-8") as f:

                    self.text\_data = f.read()

                filename = os.path.basename(filepath)

                self.file\_status.set(f"Загружен: {filename}")

                self.status\_var.set(f"Файл '{filename}' успешно загружен")

                self.update\_output("Текст загружен и готов к обработке.")

            except Exception as e:

                messagebox.showerror("Ошибка", f"Не удалось загрузить файл: {str(e)}")

                self.status\_var.set("Ошибка при загрузке файла")

    def save\_text(self):

        if not self.text\_data:

            messagebox.showwarning("Ошибка", "Нет текста для сохранения.")

            self.status\_var.set("Ошибка: нет текста для сохранения")

            return

        filepath = filedialog.asksaveasfilename(

            defaultextension=".txt",

            filetypes=[("Text files", "\*.txt"), ("All files", "\*.\*")]

        )

        if filepath:

            try:

                with open(filepath, "w", encoding="utf-8") as f:

                    f.write(self.text\_data)

                filename = os.path.basename(filepath)

                self.status\_var.set(f"Файл '{filename}' успешно сохранён")

            except Exception as e:

                messagebox.showerror("Ошибка", f"Не удалось сохранить файл: {str(e)}")

                self.status\_var.set("Ошибка при сохранении файла")

    def embed\_message(self):

        if not self.text\_data:

            messagebox.showwarning("Ошибка", "Сначала загрузите текст.")

            self.status\_var.set("Ошибка: текст не загружен")

            return

        message = self.message\_entry.get()

        if not message:

            messagebox.showwarning("Ошибка", "Введите сообщение.")

            self.status\_var.set("Ошибка: сообщение не введено")

            return

        message += '\0'  # стоп-символ

        binary = ''.join(f"{ord(c):08b}" for c in message)

        # Разделение по словам вручную

        words = []

        word = ''

        for ch in self.text\_data:

            if ch.isspace():

                if word:

                    words.append(word)

                    word = ''

            else:

                word += ch

        if word:

            words.append(word)

        if len(words) < len(binary):

            messagebox.showerror("Недостаточно текста", "Недостаточно слов для скрытия сообщения.")

            self.status\_var.set("Ошибка: недостаточно слов в тексте")

            return

        embedded\_text = ''

        for i, word in enumerate(words):

            embedded\_text += word

            if i < len(binary):

                embedded\_text += '  ' if binary[i] == '1' else ' '

            else:

                embedded\_text += ' '

        self.text\_data = embedded\_text

        capacity = len(words)

        used = len(binary)

        percent = used / capacity \* 100

        info\_text = (f"Сообщение успешно встроено!\n\n"

                   f"Статистика:\n"

                   f"• Использовано {used} из {capacity} слов\n"

                   f"• Заполнено {percent:.2f}% доступного пространства\n\n"

                   f"Теперь вы можете сохранить результат.")

        self.update\_output(info\_text)

        self.status\_var.set(f"Сообщение встроено ({percent:.2f}% заполнения)")

    def extract\_message(self):

        if not self.text\_data:

            messagebox.showwarning("Ошибка", "Нет текста для анализа.")

            self.status\_var.set("Ошибка: текст не загружен")

            return

        bits = ''

        i = 0

        while i < len(self.text\_data):

            if self.text\_data[i].isspace():

                space\_count = 0

                while i < len(self.text\_data) and self.text\_data[i] == ' ':

                    space\_count += 1

                    i += 1

                if space\_count == 1:

                    bits += '0'

                elif space\_count == 2:

                    bits += '1'

            else:

                i += 1

        chars = []

        for j in range(0, len(bits), 8):

            byte = bits[j:j+8]

            if len(byte) < 8:

                break

            ch = chr(int(byte, 2))

            if ch == '\0':

                break

            chars.append(ch)

        message = ''.join(chars)

        if message:

            self.update\_output(f"Извлечённое сообщение:\n\n{message}")

            self.status\_var.set(f"Сообщение успешно извлечено")

        else:

            self.update\_output("Сообщение не найдено или повреждено.")

            self.status\_var.set("Сообщение не обнаружено")

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

    root = tk.Tk()

    app = TextStegoApp(root)

    root.mainloop()