

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

«15» мая 2025 г.

Шевельков П.С.
ФИО студента

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: GutenbergProject <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:

Текстовая стеганография — Метод пробелов

ТЕКСТОВАЯ СТЕГАНОГРАФИЯ метод пробелов

Файл

Файл не загружен

Загрузить текст Сохранить результат

Сообщение

Введите текст для встраивания:

Встроить сообщение Извлечь сообщение

Результат

рисунок 1. Интерфейс программы.

Текстовая стеганография — Метод пробелов

ТЕКСТОВАЯ СТЕГАНОГРАФИЯ метод пробелов

Файл

Загружен: text.txt

Загрузить текст Сохранить результат

Сообщение

Введите текст для встраивания:

8laba

Встроить сообщение Извлечь сообщение

Результат

Сообщение успешно встроено!

Статистика:

- Использовано 48 из 671 слов
- Заполнено 7.15% доступного пространства

Теперь вы можете сохранить результат.

рисунок 2 Встраивание сообщения.

Текстовая стеганография — Метод пробелов

ТЕКСТОВАЯ СТЕГАНОГРАФИЯ метод пробелов

Файл

Загружен: res.txt Загрузить текст Сохранить результат

Сообщение

Введите текст для встраивания:

Встроить сообщение Извлечь сообщение

Результат

Извлечённое сообщение:

8laba

рисунк 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 встроением сообщения 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 скрывтия 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()
```