Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и
информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Представление Графической Информации
Лабораторная работа №7

Выполнил:
Студент IV курса ИВТ,
группы ИП-713
Михеев Никита Алексеевич

Работу проверил:
доцент кафедры ПМиК
Перцев И.В.

Новосибирск 2020 г.

Задание

Стеганография. Записать в младшие биты True Color BMP файла (контейнера) текстовый файл. Размер текстового файла взять 10% от размера графического. Извлечь файл из контейнера. Сравнить файлы.

Результат работы

В результате работы была реализована программа на языке Python версии 3.9. Сначала загружаем исходное изображение и текстовый файл. В изображении заменяем последние 2 бита из 8 в каждом цвете. Оригинальное изображение дополняется. После всех изменений изображение сохраняется в другом файле и сразу этот же файл открывается для считывания текста. Производится считывание позиции (индекса) битов, которые были получены ранее. Массив битов записывается в массив байтов и затем объединяется в одну строку текста и выводится на экран. Полученный текст из изображения совпадает с исходным.

Текстовый файл был выбран размером 311kB. Размер изображения: 2360 kB. Информация, помещенная в изображение равна 13.1% от него. Теоретически, максимальный объем при использовании 6 бит от каждого пикселя будет равен: (1024 * 768 * 6 / 8 / 1024) / 2360 = 0.2440, что примерно = 25% от исходного файла.

Рис. 1 – демонстрация работы программы (слева направо: оригинальное изображение, изображение с добавленным текстом).

```
"C:\Users\Lolimpo\Google Drive\SibS
 "C:/Users/Lolimpo/Google Drive/Sib
Original text size: 2484672
Decoded text size: 2484672

Process finished with exit code 0
```

Рис.2 – информация после извлечения из изображения осталась без изменений.

Листинг

Lab7.py:

```
import io
from PIL import Image


def main() -> int:
    """Putting text to BMP"""
    image = Image.open('true_color.bmp')
    image_bytes = io.BytesIO()
    image.save(image_bytes, 'BMP')
    image_bytes = bytearray(image_bytes.getvalue())

    with open('text.txt', 'r') as f:
        text = f.read()
        # print('Original text:', text)
        text_to_bits = ''.join([bin(ord(i))[2:].zfill(8) for i in text])
        print('Original text size:', len(text_to_bits))
        text_to_bits_binary = bin(len(text_to_bits))[2:].zfill(64)

    for i in range(64):
        image_bytes[i + 64] &= ~1
        image_bytes[i + 64] |= int(text_to_bits_binary[i])

    free_bytes = len(image_bytes) - 128

    for i in range(len(text_to_bits)):
        byte_index = 128 + i * free_bytes // len(text_to_bits)
        image_bytes[byte_index] &= ~1
        image_bytes[byte_index] |= int(text_to_bits[i])

    new_image = Image.open(io.BytesIO(image_bytes))
    new_image.save('lab7.bmp', 'BMP')

    """Reading text from BMP"""
    image = Image.open('lab7.bmp')
    image_bytes = io.BytesIO()
    image.save(image_bytes, 'BMP')
    image_bytes = bytearray(image_bytes.getvalue())

    text_size_bits = ''.join(str(image_bytes[i + 64] & 1) for i in
range(64))
    text_size = int(text_size_bits, 2)
    print('Decoded text size:', text_size)

    free_bytes = len(image_bytes) - 128

    text_to_bits = ''
    for i in range(text_size):
        byte_index = 128 + i * free_bytes // text_size
        text_to_bits += str(image_bytes[byte_index] & 1)

    text = ''.join([chr(int(text_to_bits[i * 8: (i + 1) * 8], 2)) for i in
range(text_size // 8)])
    # print('Decoded text:', text)

    return 0
```

```
if __name__ == '__main__':
    exit(main())
```