Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Исследование методов**

**текстовой стеганографии**

Студент: Рубашек А. А.

ФИТ 3 курс 5 группа

Преподаватель:

Савельева Маргарита Геннадьевна

1. **Цель работы**

Изучение стеганографических методов встраивания/извлечения тайной информации с использованием электронного файла-контейнера текстового формата, приобретение практических навыков программной реализации методов

1. **Задания**

Разработать авторское приложение, реализующее один из методов текстовой стеганографии на основе модификации пространственно-геометрических параметров текста-контейнера. Варианты заданий приведены в таблице. Дополнительные параметры согласуются с преподавателем.

1. **Ход работы**

В соответствии с вариантом, необходимо было реализовать метод для модификации цвета. Каждый пиксель — это точка в пространстве цветов, включающем в себя множество оттенков. Используя метод стеганографии "модификация цвета сообщения", можно скрыть информацию в изображении, меняя цвет определенных пикселей таким образом, чтобы изменения были незаметны для человеческого глаза, но могли быть распознаны с помощью специальных инструментов.

Для реализации метода модификация цвета был разработан метод modify\_color, который изменяет цвет букв. Он представлен на рисунке 3.1.

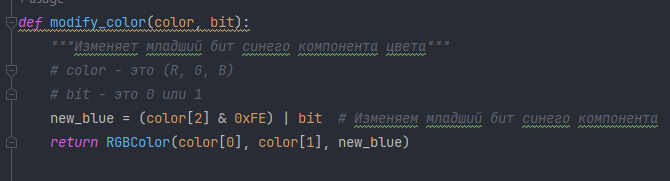


Рисунок 3.1 – Метод modify\_color

А метод, который использует modify\_color для сокрытия сообщения в тексте-контейнере, представлен в Приложении 1.

Для извлечения текста из контейнера есть обратный метод decoded\_message, представленный в Приложении 2.

Посмотрим на результаты работы всех этих методов. После модификации текста с помощью метода на модификации цвета, документ сохраняется в отдельном файле. Примеры исходного и изменённого файла представлены на рисунках 3.2 – 3.3.

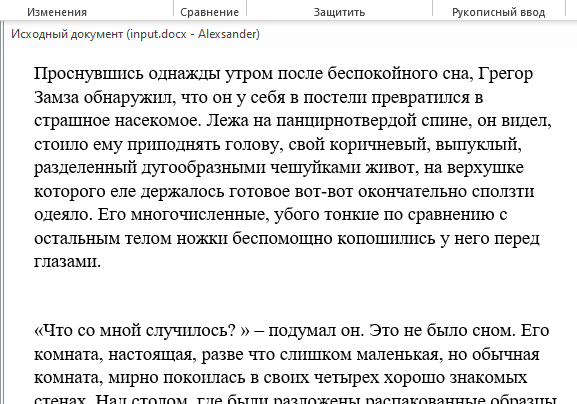


Рисунок 3.2 – Исходный документ

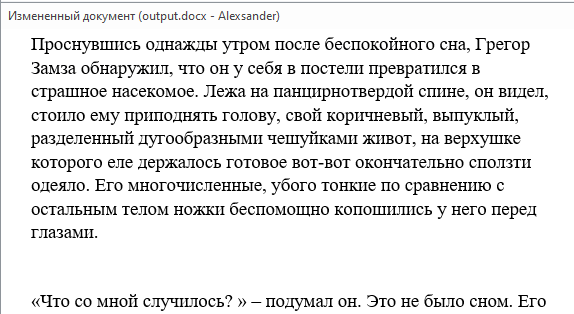


Рисунок 3.3 – Изменённый документ

Человеческим глазом отличия не увидеть, но все еще можно проанализировать их с помощью встроенного инструментария Word. Пример сравнения исходного и изменённого документа предоставлен на рисунке 3.4.

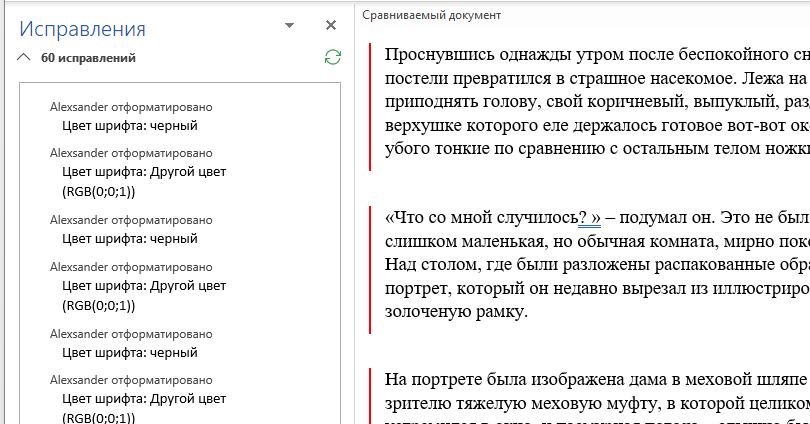


Рисунок 3.4 – Сравнение двух документов

Как видно из сравнения, стандартный черный цвет менялся на RGB(0,0,1), человеческий глаз никогда не заметит такое различие. На рисунке 3.5 изображен код вызова основных методов, и результат выполнения

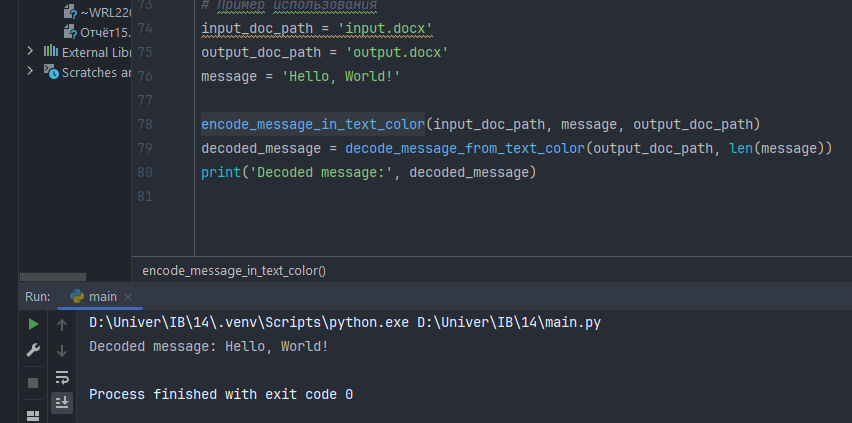


Рисунок 3.5 – Вызов основных методов и результат выполнения

Рассмотрим программное средство Sword. Оно представлено на рисунке 3.6. Рассмотрим процесс осаждения выбранного нами текста в файл.

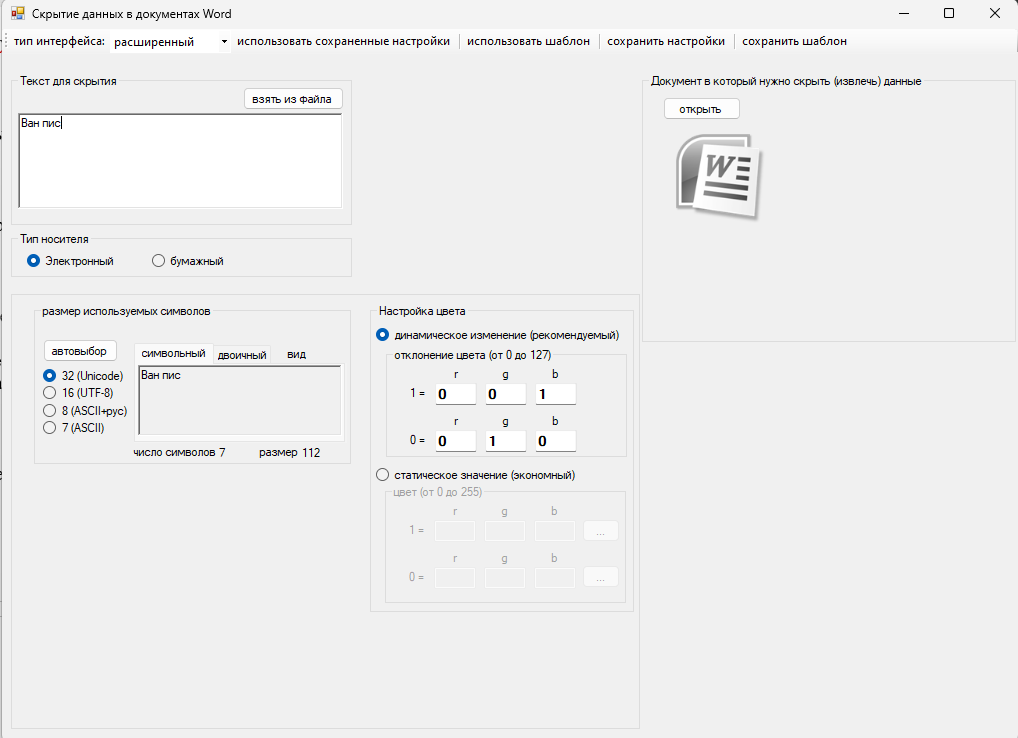


Рис. 3.6 – Приложение Sword, выбираем документ-контейнер

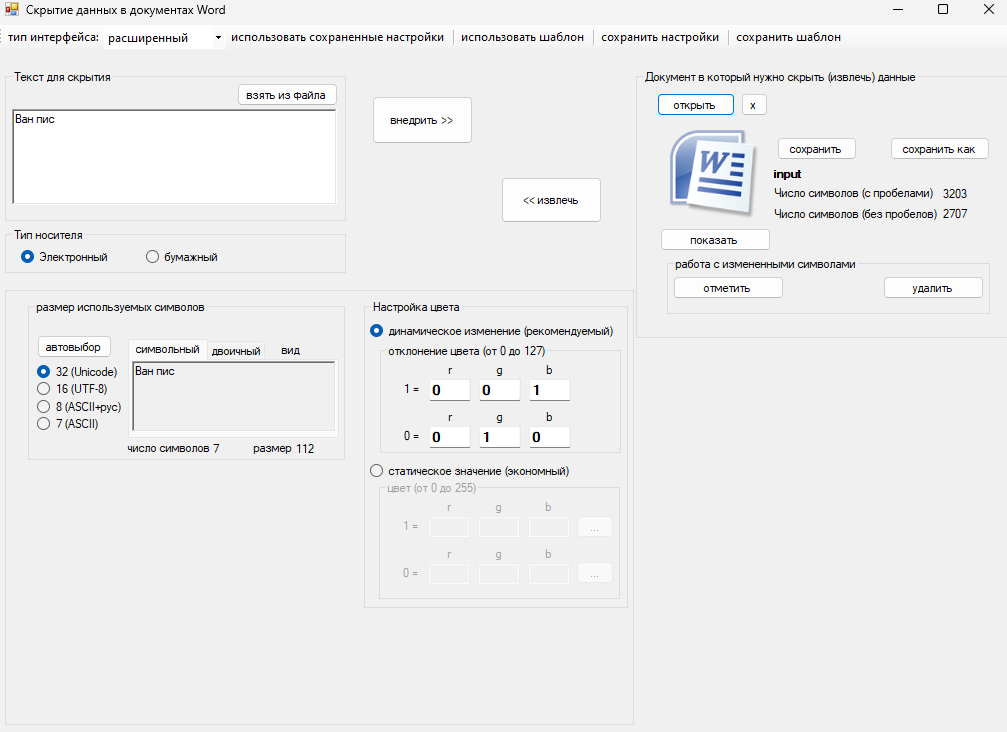


Рис. 3.7 – Выбираем текст-контейнер

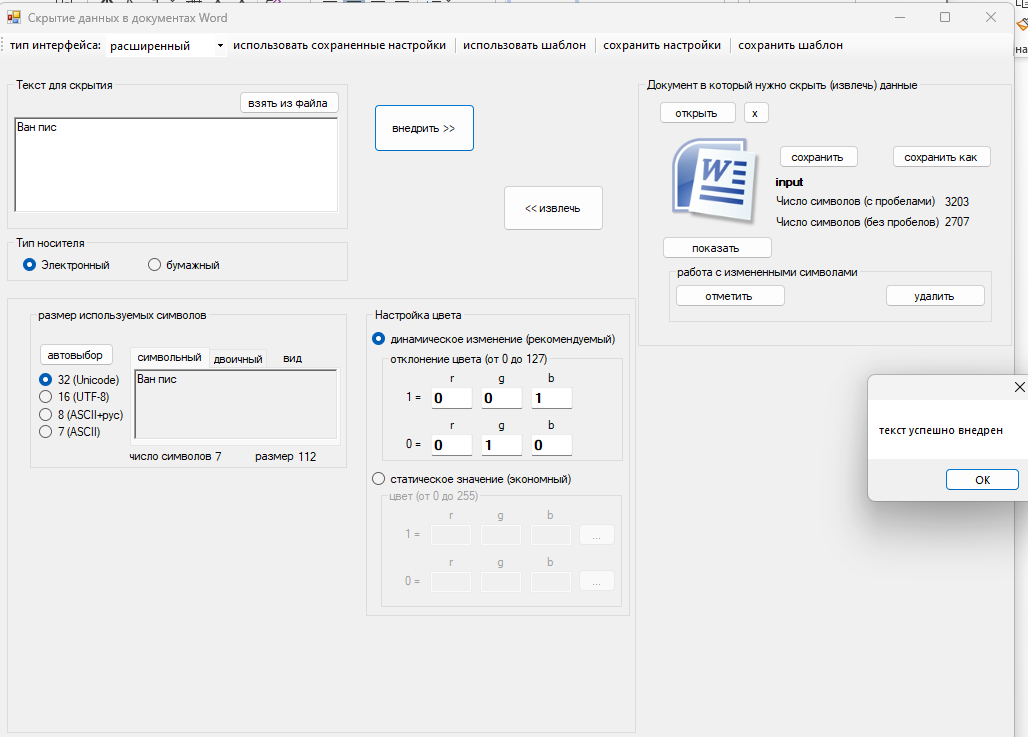


Рис. 3.8 – Внедряем сообщение в контейнер

Сообщение было осаждено. Далее, извлечём скрытое сообщение из текста-контейнера. Результаты представлены на рисунках 3.9 – 3.10.

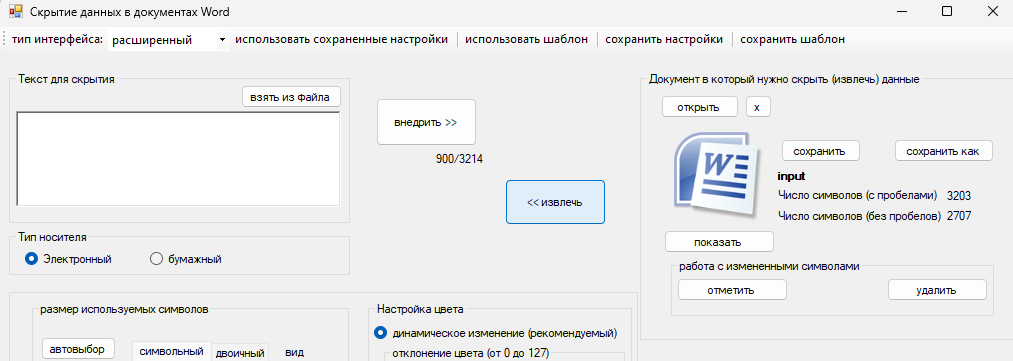


Рис. 3.9 – Извлечение сообщения из контейнера

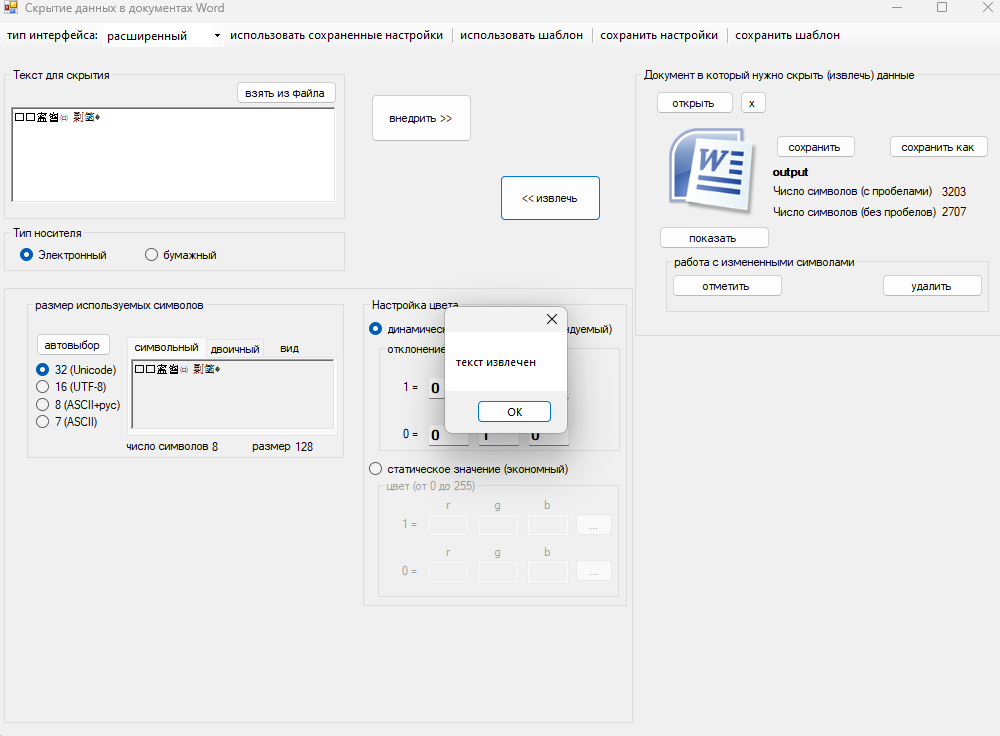


Рис. 3.10 – Извлечённое сообщение

**Вывод**

В ходе лабораторной работы были приобретены практические навыки программной реализации стеганографического метода осаждения/извлечения тайной информации с использованием текстовой стеганографии.

Также было разработано авторское приложение в соответствии с целью лабораторной работы.

**Приложение 1**

Метод encode\_message\_in\_text\_color

*def* encode\_message\_in\_text\_color(doc\_path, message, output\_path):  
 *"""Кодирует сообщение в цвет отдельных символов текста"""* doc = Document(doc\_path)  
 binary\_message = text\_to\_binary(message)  
 message\_length = len(binary\_message)  
 message\_index = 0  
  
 *for* para *in* doc.paragraphs:  
 *for* run *in* para.runs:  
 new\_text = ''  
 *for* char *in* run.text:  
 *if* message\_index < message\_length:  
 color = run.font.color.rgb *if* run.font.color *else* RGBColor(0, 0, 0)  
 new\_color = modify\_color(color, int(binary\_message[message\_index]))  
 new\_run = para.add\_run(char)  
 new\_run.font.color.rgb = new\_color  
 *# Сохраняем остальные атрибуты шрифта* new\_run.font.bold = run.font.bold  
 new\_run.font.italic = run.font.italic  
 new\_run.font.underline = run.font.underline  
 new\_run.font.size = run.font.size  
 new\_run.font.name = run.font.name  
 message\_index += 1  
 *else*:  
 new\_run = para.add\_run(char)  
 new\_run.font.color.rgb = run.font.color.rgb *if* run.font.color *else* RGBColor(0, 0, 0)  
 *# Сохраняем остальные атрибуты шрифта* new\_run.font.bold = run.font.bold  
 new\_run.font.italic = run.font.italic  
 new\_run.font.underline = run.font.underline  
 new\_run.font.size = run.font.size  
 new\_run.font.name = run.font.name  
  
 run.clear() *# Очищаем старый run* doc.save(output\_path)

**Приложение 2**

Метод decode\_message\_from\_text\_color

*def* decode\_message\_from\_text\_color(doc\_path, message\_length):  
 *"""Декодирует сообщение из цвета отдельных символов текста"""* doc = Document(doc\_path)  
 binary\_message = []  
  
 *for* para *in* doc.paragraphs:  
 *for* run *in* para.runs:  
 *for* char *in* run.text:  
 *if* len(binary\_message) < message\_length \* 8:  
 *if* run.font.color:  
 color = run.font.color.rgb  
 binary\_message.append(str(color[2] & 1))  
  
 binary\_message = ''.join(binary\_message)  
 *return* binary\_to\_text(binary\_message[:message\_length \* 8])