МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Чувашский государственный университет имени И.Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

Лабораторная работа №9

По дисциплине “Защита информации” на тему:

«Метод изменения количества пробелов в конце текстовых строк»

Выполнил: ст. гр. ИВТ-11-21

Еремеев С.А.

Проверил: канд. техн. наук. Ковалев С.В.

Чебоксары 2025

**Теоретическая часть**

# Цель работы

Целью работы является знакомство со стеганографическим методом скрытия информации путем изменения количества пробелов в конце текстовых строк.

# Задание на работу

В данной лабораторной работе необходимо программно реализовать алгоритмы прямого и обратного стеганографического преобразования.

Пустой и заполненный текстовые контейнеры должны задаваться пользователем в виде файлов либо через графический интерфейс – стандартный диалог выбора файла и/или поле ввода имени файла, либо через параметры командной строки. Выбор вида интерфейса (графический или командной строки) – на усмотрение студента, выполняющего работу. Аналогичным образом задается скрываемое сообщение. Кроме того, интерфейс пользователя должен задавать режим работы алгоритма: с одним или двумя пробелами, или с «обычным» и «неразрывным» пробелом. Во втором случае должно также задаваться количество кодируемых бит на строку.

Перед прямым стеганографическим преобразованием должна выполняться предварительная обработка пустого контейнера для удаления пробелов перед окончаниями строк. До начала стеганографического преобразования предобработчик должен выводить пользователю информацию о максимальной емкости контейнера и возможности размещения заданного сообщения в выбранном контейнере при выбранном режиме работы алгоритма.

Для простоты реализации можно принять, что сообщение начинает размещаться в контейнере с первой возможной позиции (то есть, после первой же строки) и размещается последовательно; при этом стеганографический ключ отсутствует.

# Основные сведения

Основные понятия стеганографии и обобщенная структурная схема стеганографической системы приведены в тексте лекций.

Изучаемый в данной лабораторной работе метод основан на добавлении пробелов в конец каждой текстовой строки. Конец строки индентифицируется, в зависимости от операционной системы, парой символов CR/LF или символом CR.

Аналогично рассмотренному в предыдущей лабораторной работе методу изменения интервала между предложениями, сообщение представляется в виде битовой последовательности; каждый бит сообщения определяет количество пробелов, дописываемых в конец каждой строки контейнера (один пробел для единичного бита, два пробела – для нулевого бита).

Приведем пример контейнера, аналогичного рассмотренному ранее, но с небольшой модификацией – в новом контейнере каждая строка завершается символом CR:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 1 | . |  | С | л | о | в | а | **CR** |
| ф | р | а | з | ы |  | 2 | . |  | С | л | о | в | а |  | ф | р | а | з | ы | **CR** |
| 3 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 4 | . | **CR** |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 5 | . |  | С | л | о | в | а | **CR** |
| ф | р | а | з | ы |  | 6 | . |  | С | л | о | в | а |  | ф | р | а | з | ы | **CR** |
| 7 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 8 | . | **CR** |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 9 | . |  | С | л | о | в | а | **CR** |
| ф | р | а | з | ы |  | А | . |  | С | л | о | в | а |  | ф | р | а | з | ы | **CR** |

Пусть в данном примере, как и в ранее рассмотренном, символ сообщения имеет значение **"М"**, в двоичной форме код символа **"М"** будет представлен битами 10001100b. Тогда заполненный контейнер будет выглядеть следующим образом:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 1 | . |  | С | л | о | в | а |  | **CR** | |
| ф | р | а | з | ы |  | 2 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  |  | | **CR** |
| 3 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 4 | . |  |  | **CR** |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 5 | . |  | С | л | о | в | а |  |  | | **CR** |
| ф | р | а | з | ы |  | 6 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | **CR** | |
| 7 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 8 | . |  | **CR** |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 9 | . |  | С | л | о | в | а |  | |  | **CR** |
| ф | р | а | з | ы |  | А | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | |  | **CR** |

Затемнением выделены пробелы после окончания строк, кодирующие биты скрываемого сообщения.

Очевидно, что данный метод по сравнению с ранее рассмотренным будет характеризоваться более эффективным использованием контейнера в случае коротких строк и длинных предложений, и менее эффективным – в случае длинных строк и коротких предложений. В любом случае, данный метод обладает большей скрытностью по сравнению с предыдущим, поскольку добавляемые невидимые пробелы находятся в стороне от основного текста и не влияют на расстояние между элементами текста. Недостатком этого метода (как, впрочем, и предыдущего), является то, что некоторые редакторы текста могут автоматически убирать концевые пробелы.

Существует более совершенная модификация этого метода, основанная на применении двух разных символов пробелов. Как известно, в таблице кодов ASCII существует так называемый «обычный» пробел с десятичным кодом 32, а также «неразрывный» пробел с десятичным кодом «160». Визуально неразрывный пробел выглядит точно так же, как и обычный – то есть, пустое место. Однако, многие текстовые редакторы обрабатывают факт наличия неразрывного пробела таким образом, что по такому пробелу запрещен перенос предложения на другую строку. Другими словами, если между двумя альфанумерическими символами находится неразрывный пробел, то *оба* эти символа, вместе с пробелом между ними, должны быть расположены на одной строке.

Таким образом, биты скрываемого сообщения можно кодировать не одним или двумя (обычными) пробелами, а, например, нулевой бит – обычным пробелом, и единичный бит – неразрывным пробелом. Тогда сторонам стеганографического обмена необходимо заранее условиться о том, сколько бит сообщения будет кодироваться на одной строке контейнера.

Вернемся к ранее рассмотренному примеру, но теперь будем использовать обычный и неразрывный пробелы, и условимся, что мы будем кодировать по два бита скрываемого сообщения на строку. Тогда заполненный контейнер будет выглядеть следующим образом (напомним, что М = 10001100b):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 1 | . |  | С | л | о | в | а | 160 | 32 | **CR** |
| ф | р | а | з | ы |  | 2 | . |  | С | л | о | в | а |  | ф | р | а | з | ы | 32 | 32 | **CR** |
| 3 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 4 | . | 160 | 160 | **CR** |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 5 | . |  | С | л | о | в | а | 32 | 32 | **CR** |
| ф | р | а | з | ы |  | 6 | . |  | С | л | о | в | а |  | ф | р | а | з | ы | **CR** |
| 7 | . |  | С | л | о | в | а |  | ф | р | а | з | ы |  | 8 | . | **CR** |
| С | л | о | в | а |  | ф | р | а | з | ы |  | 9 | . |  | С | л | о | в | а | **CR** |
| ф | р | а | з | ы |  | А | . |  | С | л | о | в | а |  | ф | р | а | з | ы | **CR** |

Такой вариант метода позволяет существенно более эффективно использовать контейнер и скрывать в нем сравнительно больший объем конфиденциальных данных. Разумеется, необходимо руководствоваться здравым смыслом и чувством меры: если на одну строку контейнера в 80 символов будет приходиться по 1 Кб пробелов, то большой размер файла контейнера, явно не соответствующий видимому тексту, будет вызывать подозрения.

Напомним, что текстовый контейнер должен быть подвергнут предобработке, удаляющей «лишние» пробелы на концах строк. В отличие от метода изменения интервала между предложениями, использование маркера окончания сообщения в случае одиночного сообщения в контейнере не будет обязательным.

**Практическая часть**

# Текст программы

using System**;**using System**.**IO**;**using System**.**Text**;**class Program  
**{** static void Main**(**string**[]** args**)  
 {** if **(**args**.**Length < 1**)  
 {** PrintUsage**();** return**;  
 }** string command = args**[**0**].**ToLower**();** try  
 **{** switch **(**command**)  
 {** case "embed":  
 if **(**args**.**Length < 5**)  
 {** PrintUsage**();** return**;  
 }** Embed**(**args**,** args**[**1**],** args**[**2**],** args**[**3**],** args**[**4**]);** break**;** case "extract":  
 if **(**args**.**Length < 3**)  
 {** PrintUsage**();** return**;  
 }** Extract**(**args**[**1**],** args**[**2**],** args**.**Length > 3 ? int**.**Parse**(**args**[**3**])** : 1**);** break**;** default:  
 PrintUsage**();** break**;  
 }  
 }** catch **(**Exception ex**)  
 {** Console**.**WriteLine**(**$"Ошибка: {ex**.**Message}"**);  
 }  
 }** static void PrintUsage**()  
 {** Console**.**WriteLine**(**"Использование:"**);** Console**.**WriteLine**(**"Lab9.exe embed <empty\_file> <message\_file> <output\_file> <mode> [bits]"**);** Console**.**WriteLine**(**"Lab9.exe extract <container\_file> <space\_mode> [bits\_per\_line]"**);** Console**.**WriteLine**(**"Режимы: basic - 1/2 пробела, advanced - обычный/неразрывный пробел"**);  
 }** static void Embed**(**string**[]** args**,** string emptyFile**,** string messageFile**,** string outputFile**,** string mode**)  
 {** string**[]** containerLines = File**.**ReadAllLines**(**emptyFile**);** string message = File**.**ReadAllText**(**messageFile**);** byte**[]** messageBytes = Encoding**.**ASCII**.**GetBytes**(**message**);** string bitString = ""**;** foreach **(**byte b in messageBytes**)  
 {** bitString += Convert**.**ToString**(**b**,** 2**).**PadLeft**(**8**,** '0'**);  
 }** for **(**int i = 0**;** i < containerLines**.**Length**;** i++**)  
 {** containerLines**[**i**]** = containerLines**[**i**].**TrimEnd**();  
 }** int capacity = mode**.**ToLower**()** == "basic" ? containerLines**.**Length :   
 containerLines**.**Length \* **(**args**.**Length > 5 ? int**.**Parse**(**args**[**5**])** : 1**);** Console**.**WriteLine**(**$"Максимальная емкость контейнера: {capacity} бит"**);** Console**.**WriteLine**(**$"Размер сообщения: {bitString**.**Length} бит"**);** if **(**bitString**.**Length > capacity**)  
 {** Console**.**WriteLine**(**"Ошибка: Сообщение слишком большое для контейнера"**);** return**;  
 }** string**[]** outputLines = new string**[**containerLines**.**Length**];** int bitIndex = 0**;** if **(**mode**.**ToLower**()** == "basic"**)  
 {** *// Метод 1: 1 или 2 пробела* for **(**int i = 0**;** i < containerLines**.**Length**;** i++**)  
 {** if **(**bitIndex < bitString**.**Length**)  
 {** outputLines**[**i**]** = containerLines**[**i**]** + **(**bitString**[**bitIndex**]** == '1' ? " " : " "**);** bitIndex++**;  
 }** else  
 **{** outputLines**[**i**]** = containerLines**[**i**];  
 }  
 }  
 }** else if **(**mode**.**ToLower**()** == "advanced"**)  
 {** *// Метод 2: обычный и неразрывный пробел* int bitsPerLine = args**.**Length > 5 ? int**.**Parse**(**args**[**5**])** : 1**;** for **(**int i = 0**;** i < containerLines**.**Length && bitIndex < bitString**.**Length**;** i++**)  
 {** string spaces = ""**;** for **(**int j = 0**;** j < bitsPerLine && bitIndex < bitString**.**Length**;** j++**)  
 {** spaces += bitString**[**bitIndex**]** == '1' ? "\u00A0" : " "**;** bitIndex++**;  
 }** outputLines**[**i**]** = containerLines**[**i**]** + spaces**;  
 }** *// Копируем оставшиеся строки* for **(**int i = bitIndex/bitsPerLine**;** i < containerLines**.**Length**;** i++**)  
 {** outputLines**[**i**]** = containerLines**[**i**];  
 }  
 }** File**.**WriteAllLines**(**outputFile**,** outputLines**);** Console**.**WriteLine**(**$"Сообщение успешно встроено в {outputFile}"**);  
 }** static void Extract**(**string containerFile**,** string mode**,** int bitsPerLine**)  
 {** string**[]** lines = File**.**ReadAllLines**(**containerFile**);** string bitString = ""**;** if **(**mode**.**ToLower**()** == "basic"**)  
 {** *// Извлечение для метода 1* foreach **(**string line in lines**)  
 {** int spaceCount = line**.**Length - line**.**TrimEnd**().**Length**;** if **(**spaceCount == 1**)** bitString += "1"**;** else if **(**spaceCount == 2**)** bitString += "0"**;  
 }  
 }** else if **(**mode**.**ToLower**()** == "advanced"**)  
 {** *// Извлечение для метода 2* foreach **(**string line in lines**)  
 {** string trimmed = line**.**TrimEnd**();** string spaces = line**.**Substring**(**trimmed**.**Length**);** foreach **(**char c in spaces**)  
 {** bitString += **(**c == '\u00A0'**)** ? "1" : "0"**;  
 }  
 }  
 }** string result = ""**;** for **(**int i = 0**;** i < bitString**.**Length - 7**;** i += 8**)  
 {** string byteStr = bitString**.**Substring**(**i**,** 8**);** result += **(**char**)**Convert**.**ToInt32**(**byteStr**,** 2**);  
 }** Console**.**WriteLine**(**$"Извлеченное сообщение: {result}"**);  
 }  
}**

**Прямое стеганографическое преобразование**

**Метод "basic" (1/2 пробела):**

if (mode.ToLower() == "basic")

{

for (int i = 0; i < containerLines.Length; i++)

{

if (bitIndex < bitString.Length)

{

outputLines[i] = containerLines[i] + (bitString[bitIndex] == '1' ? " " : " ");

bitIndex++;

}

else

{

outputLines[i] = containerLines[i];

}

}

}

**Метод "advanced" (обычный/неразрывный пробел):**

else if (mode.ToLower() == "advanced")

{

int bitsPerLine = args.Length > 5 ? int.Parse(args[5]) : 1;

for (int i = 0; i < containerLines.Length && bitIndex < bitString.Length; i++)

{

string spaces = "";

for (int j = 0; j < bitsPerLine && bitIndex < bitString.Length; j++)

{

spaces += bitString[bitIndex] == '1' ? "\u00A0" : " ";

bitIndex++;

}

outputLines[i] = containerLines[i] + spaces;

}

for (int i = bitIndex/bitsPerLine; i < containerLines.Length; i++)

{

outputLines[i] = containerLines[i];

}

}

**Обратное стеганографическое преобразование**

**Метод "basic" (1/2 пробела):**

if (mode.ToLower() == "basic")

{

foreach (string line in lines)

{

int spaceCount = line.Length - line.TrimEnd().Length;

if (spaceCount == 1) bitString += "1";

else if (spaceCount == 2) bitString += "0";

}

}

**Метод "advanced" (обычный/неразрывный пробел):**

else if (mode.ToLower() == "advanced")

{

foreach (string line in lines)

{

string trimmed = line.TrimEnd();

string spaces = line.Substring(trimmed.Length);

foreach (char c in spaces)

{

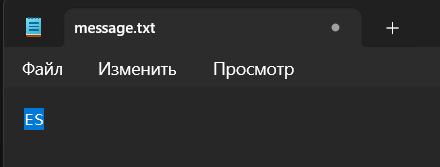
bitString += (c == '\u00A0') ? "1" : "0";

}

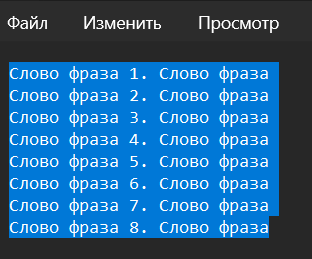
}

}

Текст скрываемого сообщения message.txt:

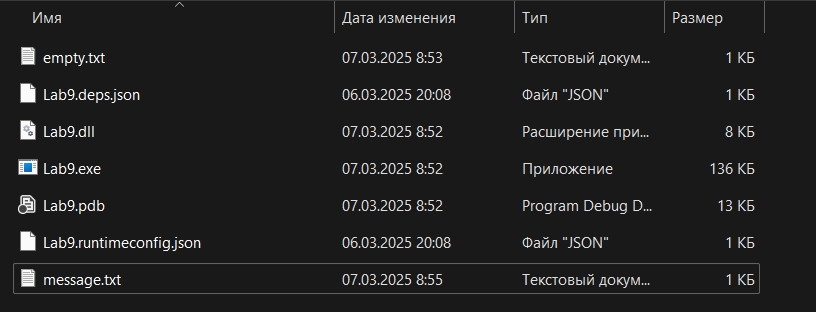


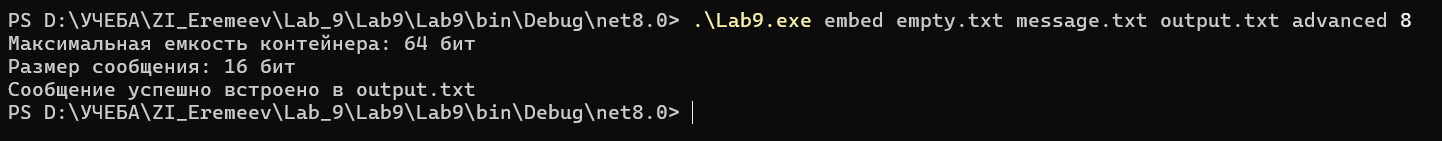
Текст заполняемого сообщения empty.txt



**Пример выполнения программы**

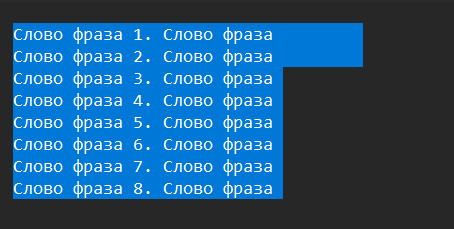
**Метод advanced**



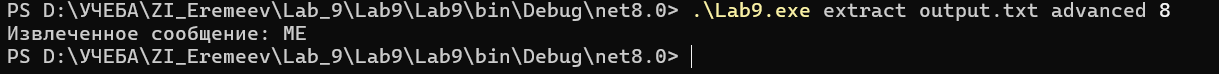




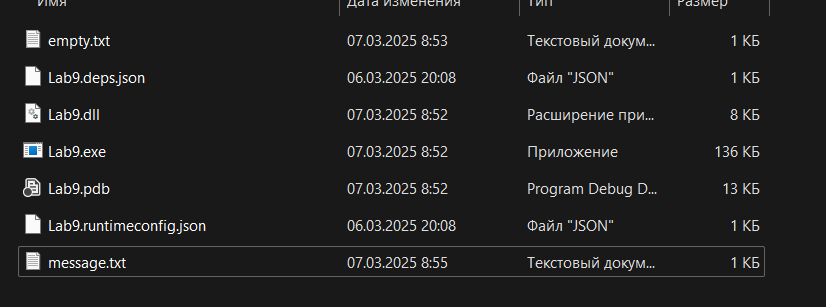
Содержимое output.txt:



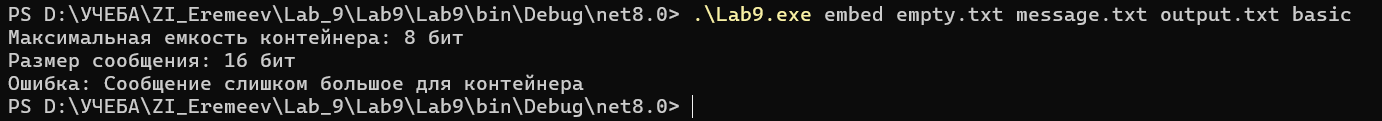
Вывод сообщения:



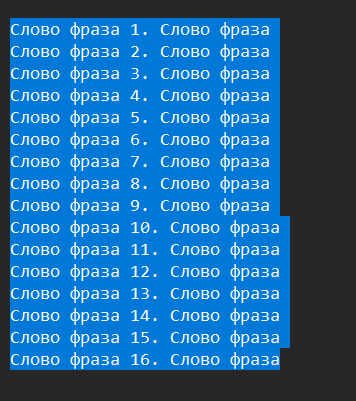
**Метод basic**

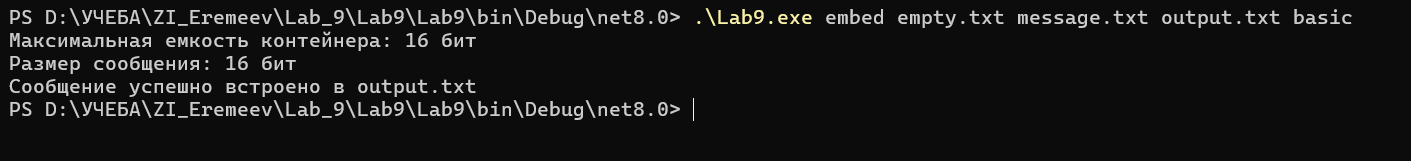
****

Обработка ошибок:

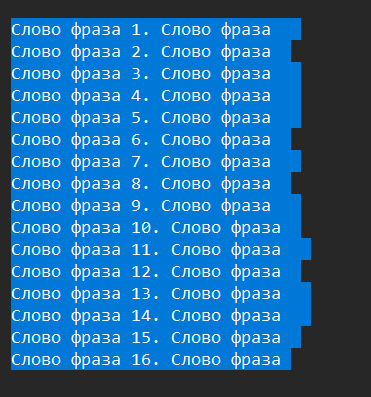


Содержимое empty.txt:

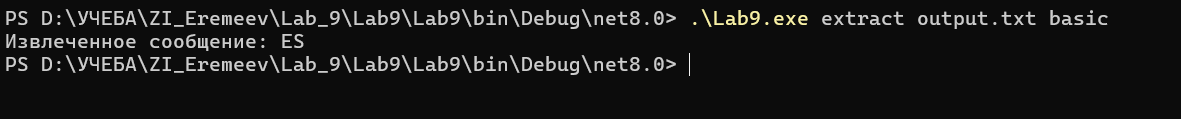




Содержимое output.txt:



Вывод сообщения:



# Вывод

В ходе лабораторной работы были реализованы два метода стеганографии: "basic" (1 или 2 пробела) и "advanced" (обычный и неразрывный пробелы). Первый метод проще, но вмещает меньше данных (1 бит на строку), второй — эффективнее и скрытнее (до 2 бит на строку в примере).