Оглавление

[Обозначения 3](#_Toc57228119)

[Введение 4](#_Toc57228120)

[Раздел 1. 5](#_Toc57228121)

[Описание алгоритма RC6. 5](#_Toc57228122)

[Раздел 2. 8](#_Toc57228123)

[Проектирование программы. 8](#_Toc57228124)

[2.1. Проектирование графического интерфейса. 8](#_Toc57228125)

[2.2. Проектирование криптоалгоритма. 10](#_Toc57228126)

[2.3. Вывод ключа шифрования из парольной фразы. 10](#_Toc57228127)

[Раздел 3. 11](#_Toc57228128)

[Тестирование элементов программы. 11](#_Toc57228129)

[3.1 Тестирование криптоалгоритма. 11](#_Toc57228130)

[3.2 Тестирование самой программы. 12](#_Toc57228131)

[Раздел 4. 19](#_Toc57228132)

[Итоги. 19](#_Toc57228133)

[4.1 Заключение. 19](#_Toc57228134)

[4.2 Немного из истории криптоалгоритма. 19](#_Toc57228135)

# Обозначения

При описании алгоритмов используются следующие обозначения :

a+b – сложение

a⊕b – сложение по модулю 2

a<<<b – циклический сдвиг величины a на b бит влево

a>>>b – циклический сдвиг величины a на b бит вправо

# Введение

Цель работы – написание приложения, способное шифровать и расшифровывать указанный пользователем файл (в файловом режиме) или введенный пользователем текст (в текстовом режиме) по криптоалгоритму RC6 на ключе, выводимом из указываемой пользователем парольной фразы.

Для выполнения цели работы следует решить следующие задачи:

* Спроектировать графический интерфейс пользователя, представляющий функционал приложения
* Изучить и реализовать заданный криптоалгоритм RC6
* Протестировать и отладить реализованный криптоалгоритм
* Протестировать и отладить графический интерфейс в совокупности с криптоалгоритмом

# Раздел 1.

## Описание алгоритма RC6.

RC6 является симметричным блочным криптографическим алгоритмом шифрования. Поддерживает блоки длиной 128 бит и ключи длиной 128, 192 и 256 бит, однако алгоритм может быть сконфигурирован для поддержки более широкого диапазона длин как для блоков, так и ключей (от 0 до 2040).

Алгоритм является полностью параметризированным, для спецификации алгоритма с конкретными параметрами принято обозначение RC6-w/r/b, где

w-длина машинного слова в битах

r-число раундов

b-длина ключа в битах

**Таблица 1.1. Алгоритмы шифрования/дешифрования алгоритма RC6**

**Алгоритм дешифрования для RC6-w/r/b**

**Input**:

Зашифрованный текст, хранящийся в четырёх w-битных входных регистрах A, B, C, D.

Количество раундов r

w-битные раундовые ключи S[0,…, 2r + 3]

**Output**:

Расшифрованный текст, хранящийся в A,B,C,D.

**Алгоритм**:

C = C - S[2r + 3]

A = A - S[2r + 2]

for i = r downto 1 do

{ (A,B,C,D)=(D,A, B,C)

u = (D \* (2D + 1)) <<< lg w

t = (B \* (2B + 1)) <<< lg w

C = ((C - S[2i + 1]) >>> t) ⊕ u

A = ((A - S[2i]) >>> u) ⊕ t

}

D = D-S[1]

B = B- S[0]

**Алгоритм шифрования для RC6-w/r/b**

**Input**:

Открытый текст, хранящийся в четырёх w-битных входных регистрах A, B, C, D.

Количество раундов r

w-битные раундовые ключи S[0,…, 2r + 3]

**Output**:

Шифрованный текст, хранящийся в A,B,C,D.

**Алгоритм**:

B = B + S[0]

D = D + S[1]

for i = 1 to r do

{ t = (B\* (2B + 1)) <<< lg w

u = (D\*(2D + 1)) <<< lg w

A = ((A ⊕ t) <<< u) + S[2i]

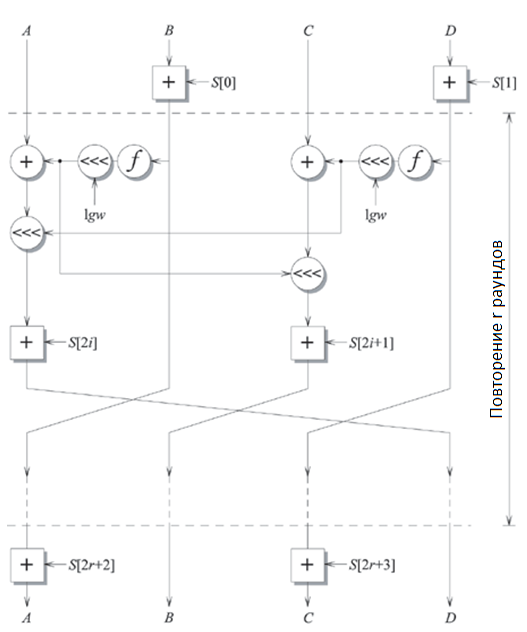
C = ((C ⊕ u) <<< t) + S[2i + 1]

(A, B, C, D)=(B, C, D, A)

}

A = A + S[2r + 2]

C = C + S[2r + 3]



**Изображение 1.1. Сеть Фейстеля алгоритма RC6**

Для криптоалгоритма RC-6 выполняется процедура расширения ключа, цель которой - получить из b-байтного исходного ключа, заданного пользователем, предварительно преобразованного в массив из k слов

L[0,…,c-1], w-битную таблицу ключей S[0,..,2r+3], требующуюся для алгоритма шифрования. Преобразование происходит с помощью двух псевдослучайных величин Qw=Odd((f-1)\*2^w), Pw=Odd((e-2)\*2^w), Odd- округление до ближайшего нечетного целого, f- золотое сечение, e-экспонента.

**Таблица 1.2. Алгоритм процедуры расширения ключа криптоалгоритма RC6**

**Алгоритм расширения ключа для RC6-w/r/b**

**Input**:

b-байтный ключ, заданный пользователем, предварительно преобразованный в массив из k слов L[0,..,k-1]

r- количество раундов

**Output**:

w- битная таблица ключей S[0,..,2r + 3]

**Алгоритм**:

S[0]=Pw

for i=1 to 2r+3 do

S[i]=S[i-1]+Qw

A=B=i=j=0

v=3\*max{k, 2r+4}

for s=1 to v do

{

A=S[i]=(S[i]+A+B) <<< 3

B=L[j]=(L[j]+A+B) <<< (A+B)

i=(i+1) mod (2r+4)

j=(j+1) mod k

}

Положим для данной реализации RC-6:

Количество раундов r=20

Длина машинного слова в битах w=32

Длина ключа – по усмотрению пользователя один из вариантов: 128,192,256 бит

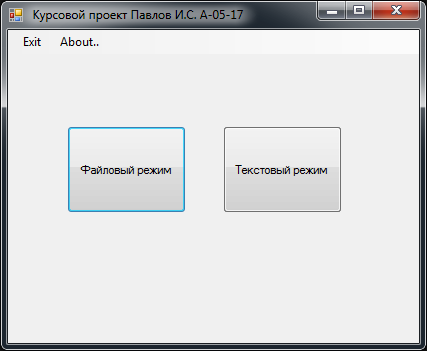
Соответственно, Qw=Q32= 9E3779B9, Pw=P32= B7E15163.

В качестве режима блочного шифрования будет использоваться режим электронной кодовой книги.

# Раздел 2.

## Проектирование программы.

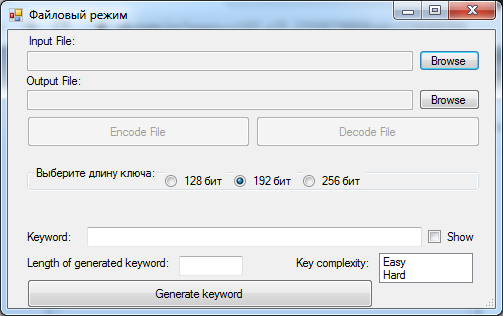
### 2.1. Проектирование графического интерфейса.

При запуске программы наблюдаем следующее окно (Изображение 2.1):  


**Изображение 2.1. Стартовое окно программы.**

В данном окне имеем разделение на два различных режима работы программы: Файловый режим и Текстовый режим.

Рассмотрим окно файлового режима (Изображение 2.2):



**Изображение 2.2. Окно файлового режима.**

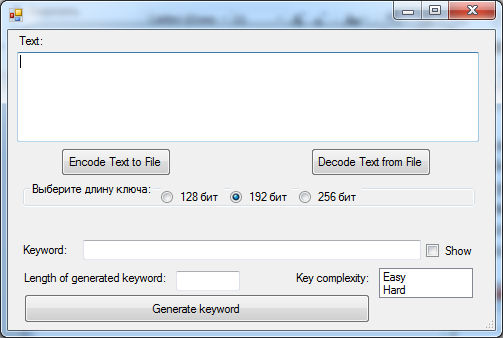
Окно позволяет выбрать два файла: файл для кодировки и файл, в который будет происходить кодировка (кодировка невозможна, если пользователь не укажет пути файлов).

Кодировка происходит на основе ключа, выводимого из парольной фразы, введенной пользователем в поле Keyword.

Кроме того, пользователь может выбрать длину ключа для алгоритма шифрования (по умолчанию стоит длина в 192 бит).

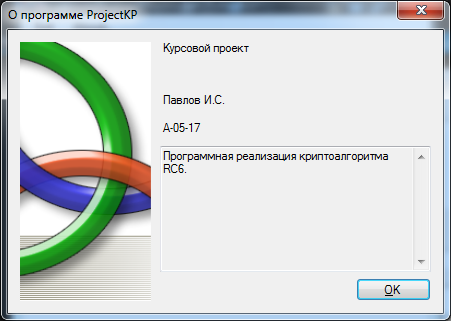
Для пользователя доступна функция генерации парольной фразы, если пользователь введет нужную длину парольной фразы и выберет сложность генерации.

Похожий функционал присутствует в текстовом режиме работы, однако сама суть текстового режима отличается. Пользователь имеет возможность ввести текст и закодировать его в файл по ключу шифрования, выведенному из парольной фразы, а так же расшифровать текст из закодированного файла (Изображение 2.3) :



**Изображение 2.3. Окно текстового режима.**

Кроме того, в программе присутствует окно «About», в которое можно перейти из окна старта программы(Изображение 2.4):



**Изображение 2.4. Окно “About”.**

### 2.2. Проектирование криптоалгоритма.

Для проектирования части программы связанной с криптоалгоритмом, была взята и отредактирована уже готовая реализация RC6 из открытого git-репозитория(источник предоставлен в конце отчёта, отредактированный код криптоалгоритма находится в приложении 1).

Ниже представлено краткое описание полей и методов реализации.

**Класс RC6 содержит:**

**Поля:**

*private const int R -количество раундов*

*private static uint[] RoundKey -массив раундовых ключей*

*private const int W –длина машинного слова в битах*

*private static byte[] MainKey – ключ, на основе которого происходит процедура расширения ключей*

*private const uint P32 = 0xB7E15163 –константа экспоненты*

*private const uint Q32 = 0x9E3779B9- константа золотого сечения*

**Методы:**

*private static uint RightShift(uint value, int shift)-сдвиг вправо без потери*

*private static uint LeftShift(uint value, int shift)-сдвиг влево без потери*

*private static void GenerateKey(int Long, byte[] keyCheck)- генерация массива раундовых ключе на основании MainKey и длины ключа.*

*private static byte[] ToArrayBytes(uint[] uints, int Long)-преобразование из набора машинных слов в массив байтов*

*public byte[] EncodeRc6(byte[] byteText)- кодировка по криптоалгоритму RC6 заданной последовательности байтов.*

*public byte[] EncodeRc6(string plaintext) – перегрузка предыдущего метода для входной строки.*

*public byte[] DecodeRc6(byte[] cipherText)- декодировка по криптоалгоритму RC6 заданной последовательности байтов.*

### 2.3. Вывод ключа шифрования из парольной фразы.

Для вывода ключа шифрования из парольной фразы используется встроенный в C# класс PasswordDeriveBytes, основанный на расширении алгоритма формирования ключа на основе пароля PBKDF1.

Сначала парольная фраза преобразуется в массив байт. Затем этот массив передается в класс PasswordDeriveBytes, и с помощью метода GetBytes(int long) мы получаем ключ на основе парольной фразы длиной long.

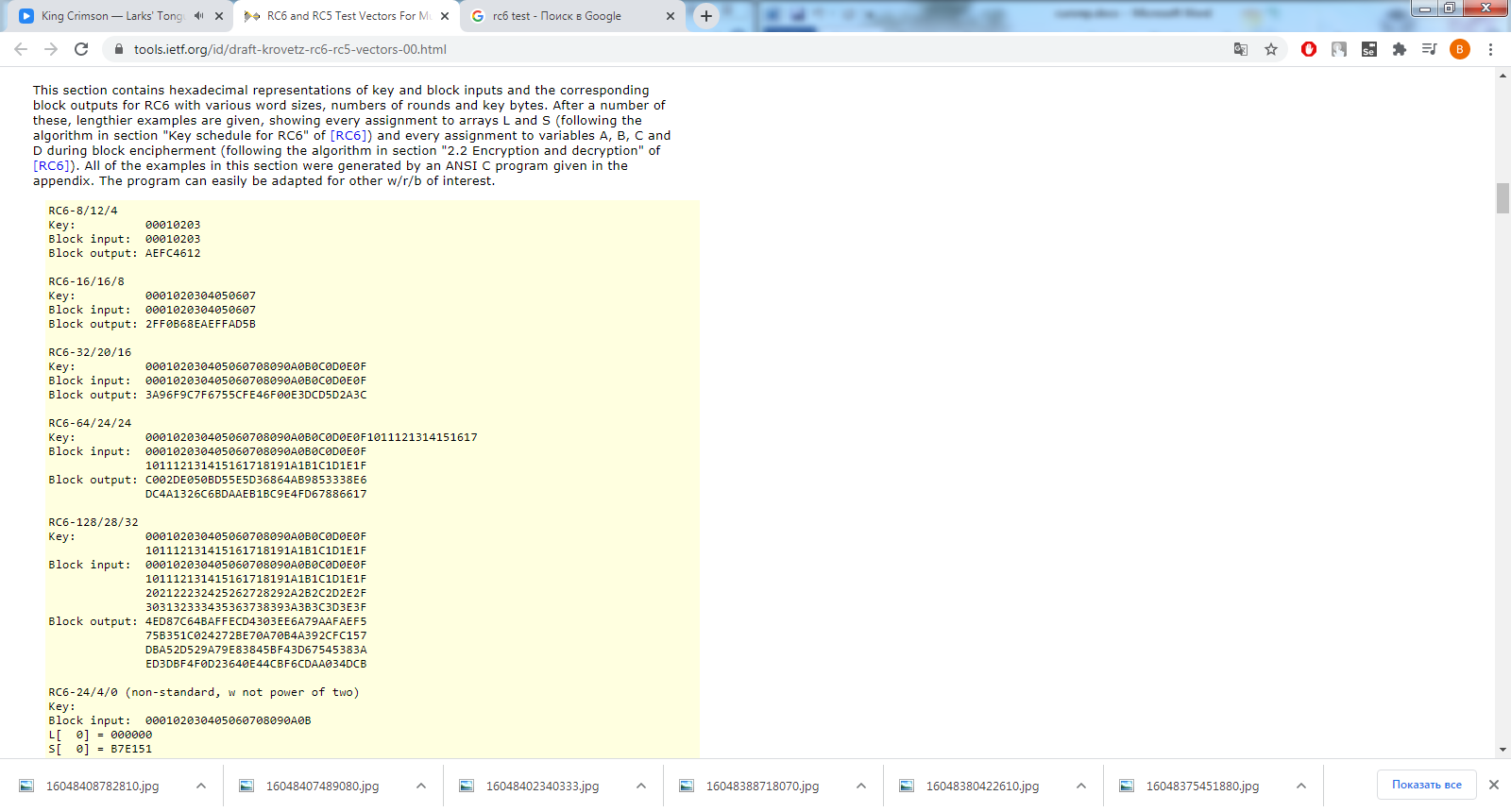
# Раздел 3.

## Тестирование элементов программы.

### 3.1 Тестирование криптоалгоритма.

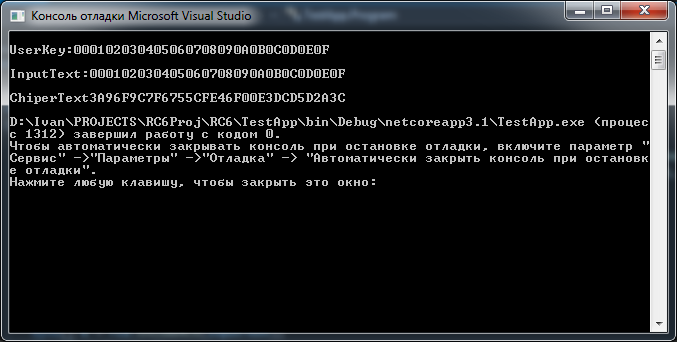
В качестве тестирования имеющейся реализации криптоалгоритма было произведено сравнение результатов конкретного теста из открытого источника и результатов собственного тестирования (код программы присутствует в приложении 3, источник присутствует в списке источников).

Тест из открытого источника для 16-байтового ключа, с 20 раундами и с длиной машинного слова в 32 бита (Изображение 3.1):



**Изображение 3.1. Тест из открытого источника.**

Собственный тест для аналогичных наборов данных (Изображение 3.2):



**Изображение 3.2. Результат тестирования на аналогичных данных.**

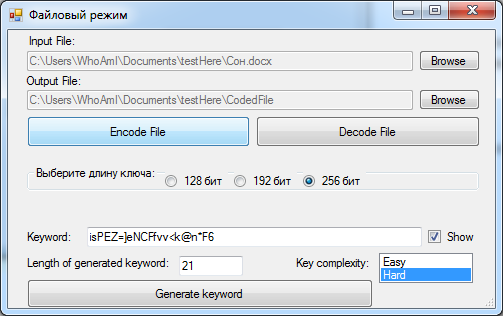
Шифротекст нашего теста совпал с тестом из открытого источника.

### 3.2 Тестирование самой программы.

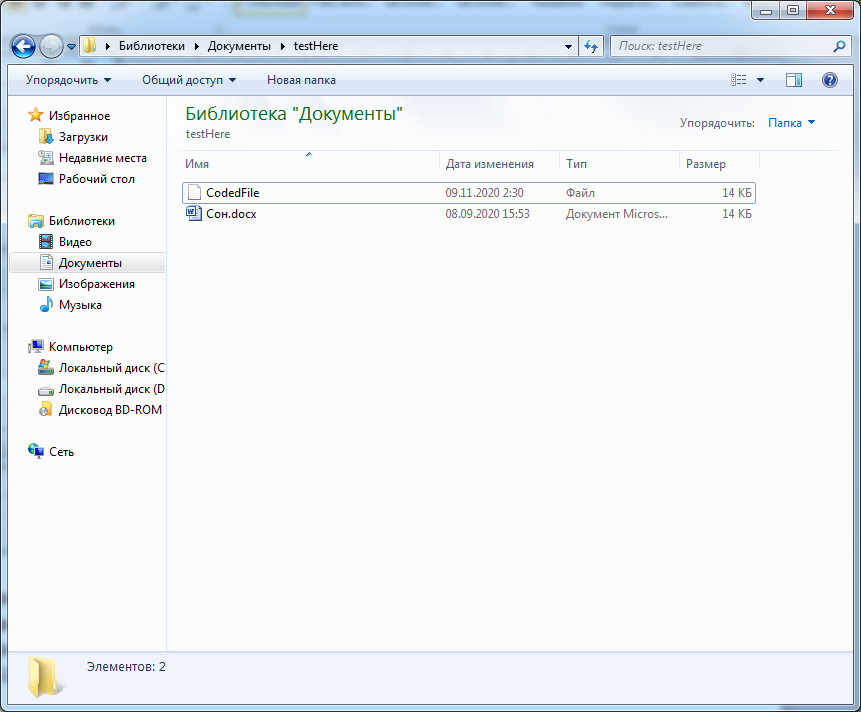
Для тестирования программы построим несколько тест-кейсов и убедимся в том, что программа работает корректно:

**Тест №1:**

Используем файловый режим программы, выбираем путь файла для шифрования и путь для зашифрованного файла, генерируем парольную фразу размером 21 с высокой сложностью и запоминаем ее. Выбираем длину ключа в 256 бит. Жмёт кнопку «Encode File» и закрываем программу. (Изображения 3.1.1-3.1.2.)

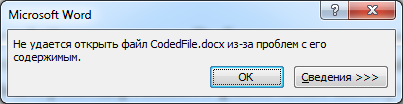


**Изображение 3.1.1.Тест №1**



**Изображение 3.1.2. Тест №1**

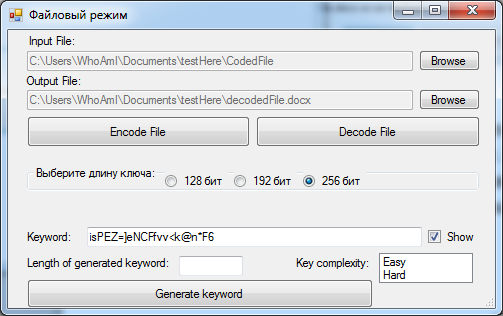
Обратим внимание, что созданный файл не удастся открыть в соответствующей программе т.к. он закодирован(Изображение 3.1.3)



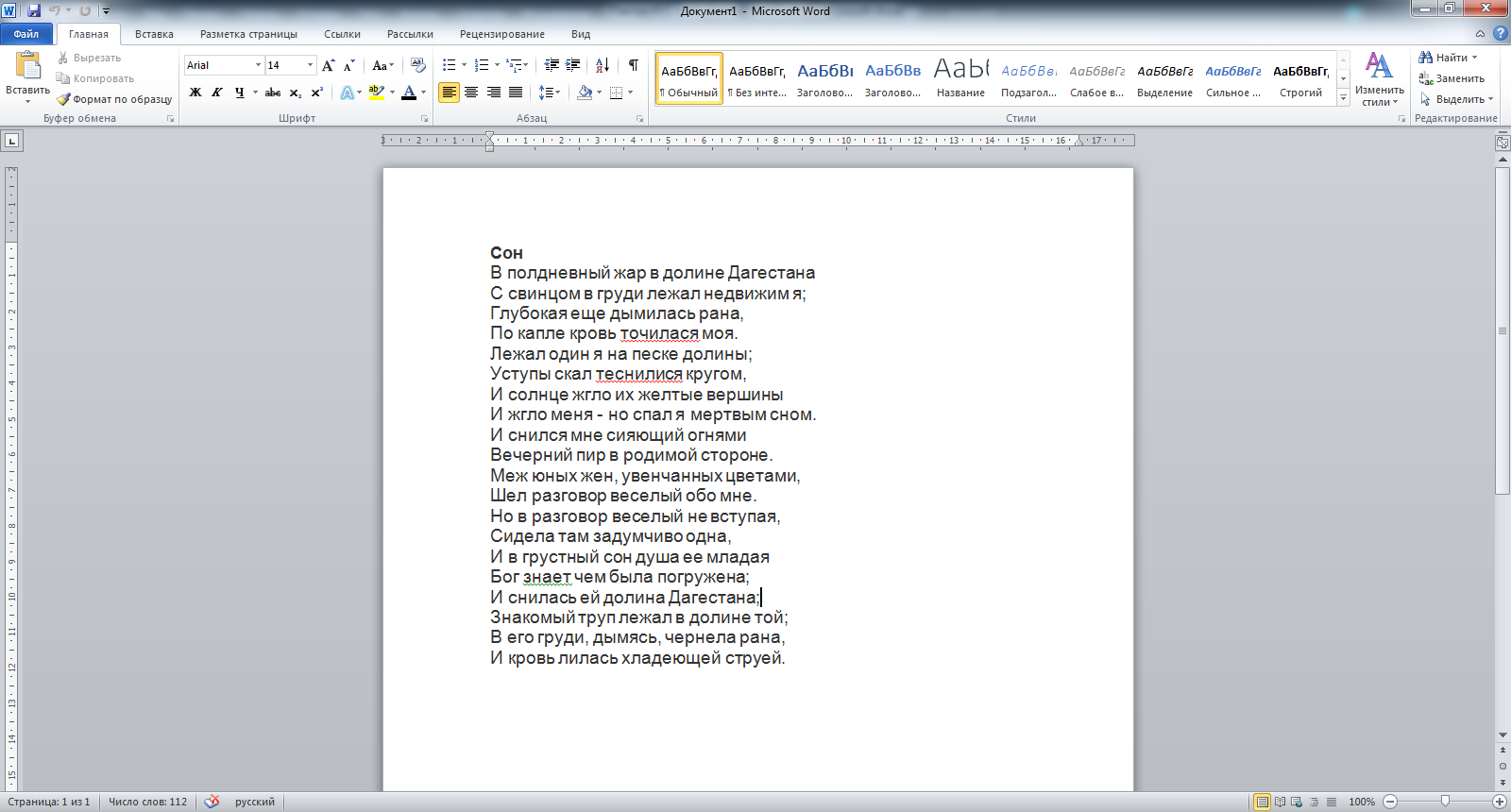
**Изображение 3.1.3. Тест №1**

Запускаем программу в файловом режиме. Выбираем длину ключа в 256 бит, вводим запомненную парольную фразу. Выбираем закодированный раннее файл и выбираем путь для раскодированного файла. Жмёт кнопку «Decode File». Наблюдаем, что файл раскодировался верно и его можно запустить.

(Изображения 3.1.4-3.1.5)



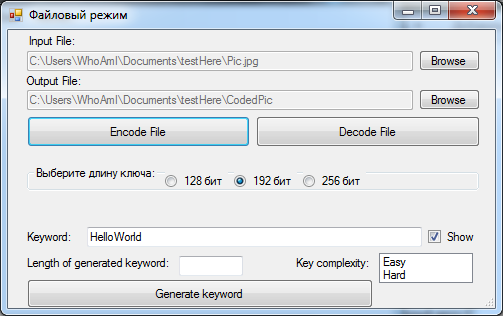
**Изображение 3.1.4. Тест №1**



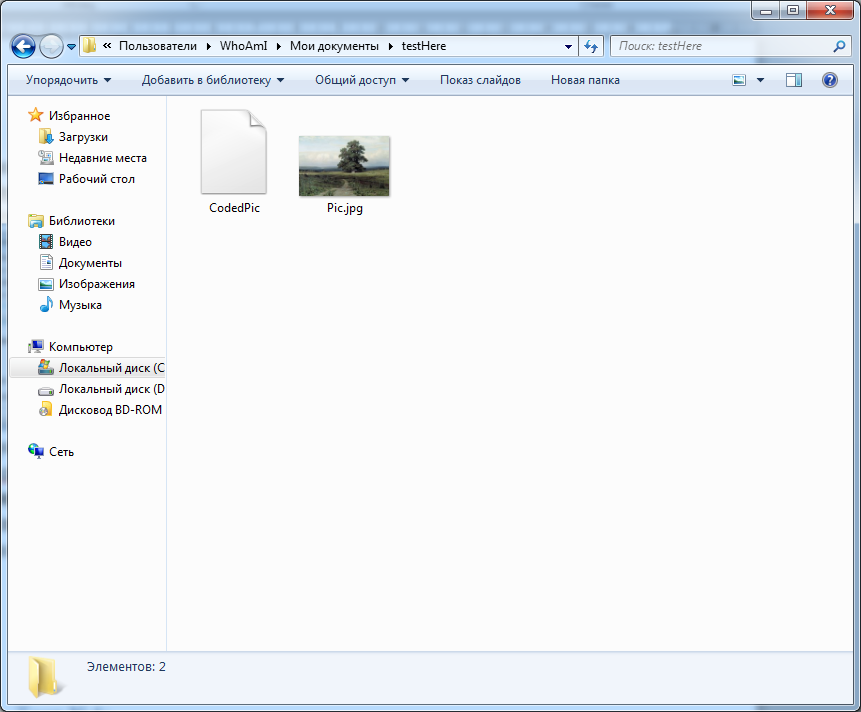
**Изображение 3.1.5. Тест №1**

**Тест №2:**

Используем файловый режим программы, выбираем путь файла для шифрования и путь для зашифрованного файла, вводим парольную фразу. Жмёт кнопку «Encode File» и закрываем программу. (Изображения 3.1.6-3.1.7)

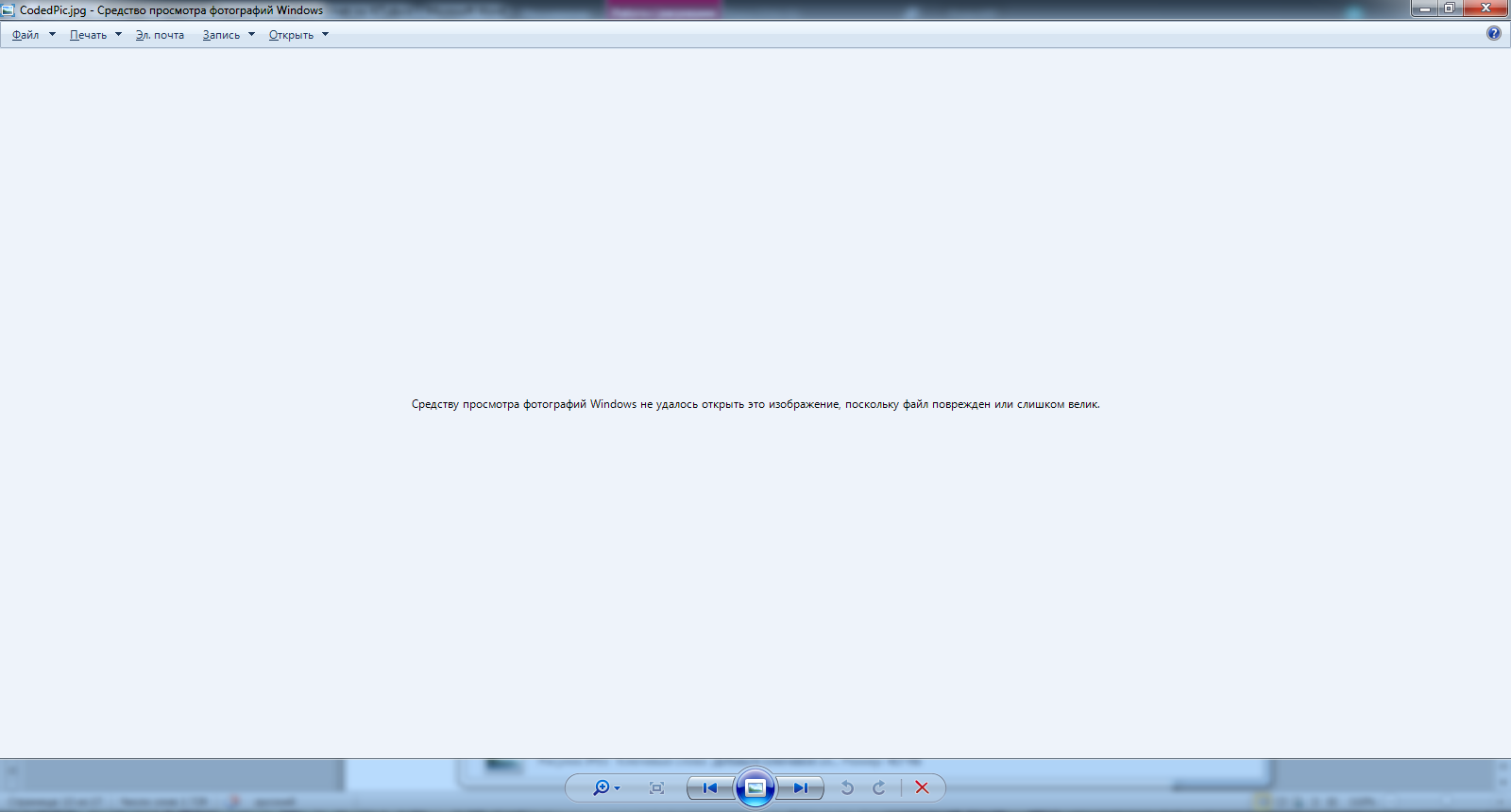


**Изображение 3.1.6. Тест №2**



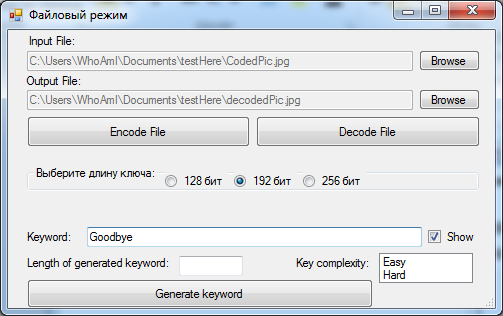
**Изображение 3.1.7. Тест №2**

Обратим внимание, что созданный файл не удастся открыть в соответствующей программе т.к. он закодирован.(Изображение 3.1.8)

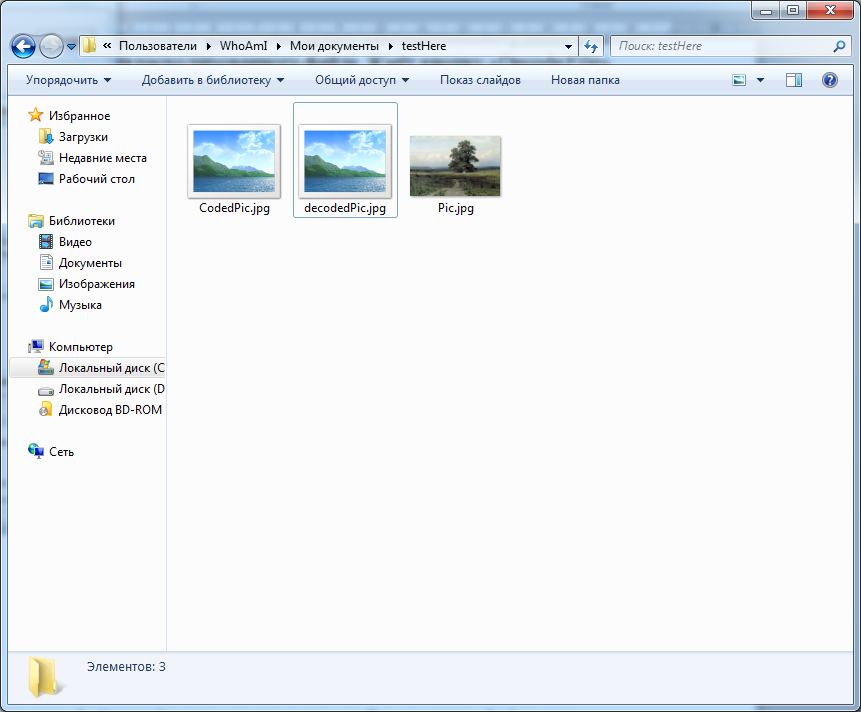


**Изображение 3.1.8. Тест №2**

Запускаем программу в файловом режиме. Вводим отличную от вводимой для закодирования парольную фразу. Выбираем закодированный раннее файл и выбираем путь для раскодированного файла. Жмёт кнопку «Decode File» (Изображения 3.1.9-3.1.10)

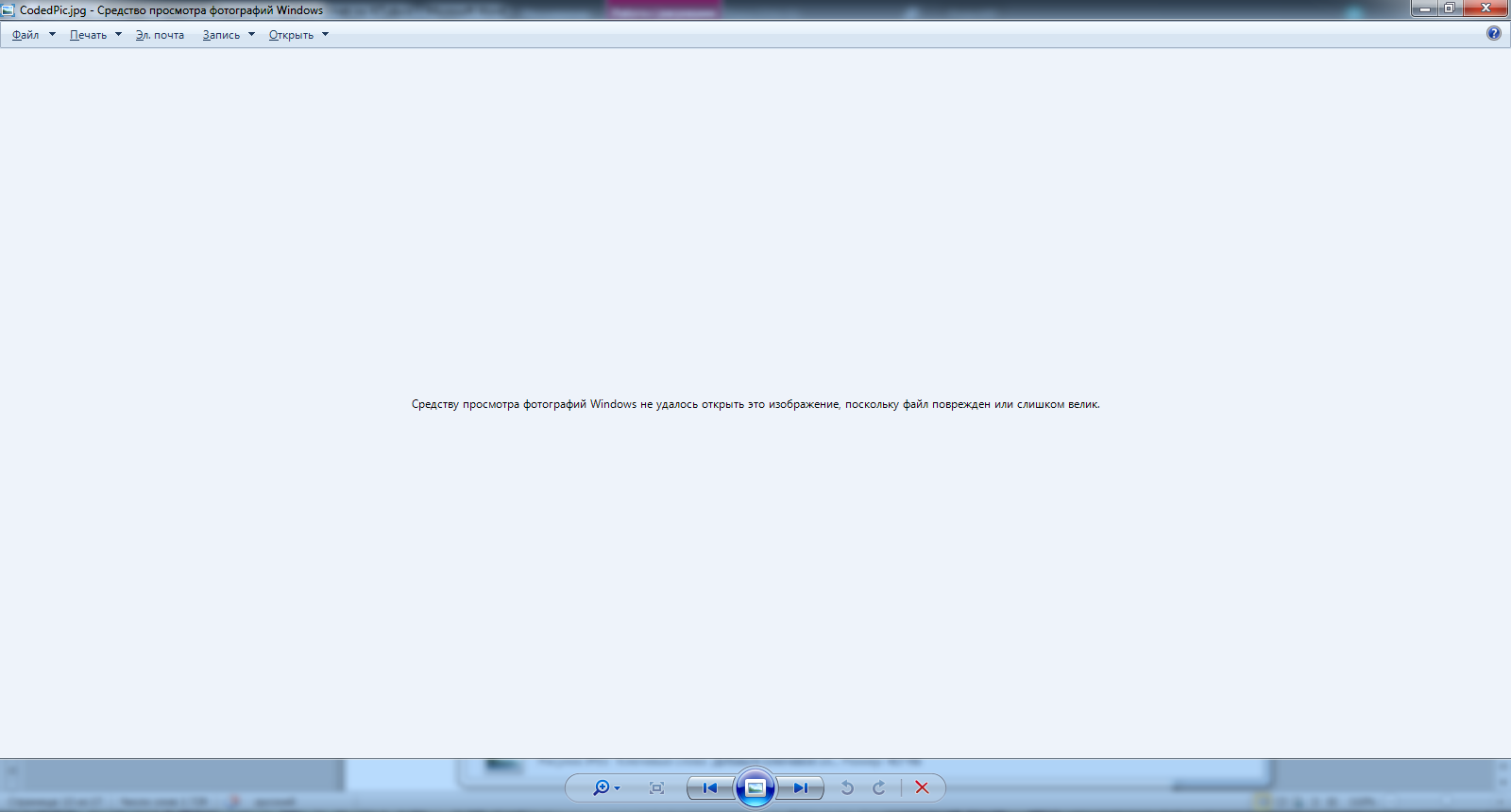


**Изображение 3.1.9.Тест №2**



**Изображение 3.1.10. Тест №2**

Обратим внимание, что созданный раскодированный файл не удастся открыть в соответствующей программе т.к. была введена неверная парольная фраза.( Изображения 3.1.11)

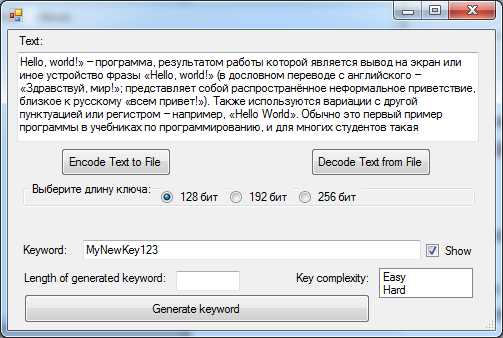


**Изображение 3.1.11. Тест №2**

Заметим, что такое же поведение будет демонстрироваться, если кодировать файл, выбирая одну длину ключа, а при декодировке файла выбирать другую.

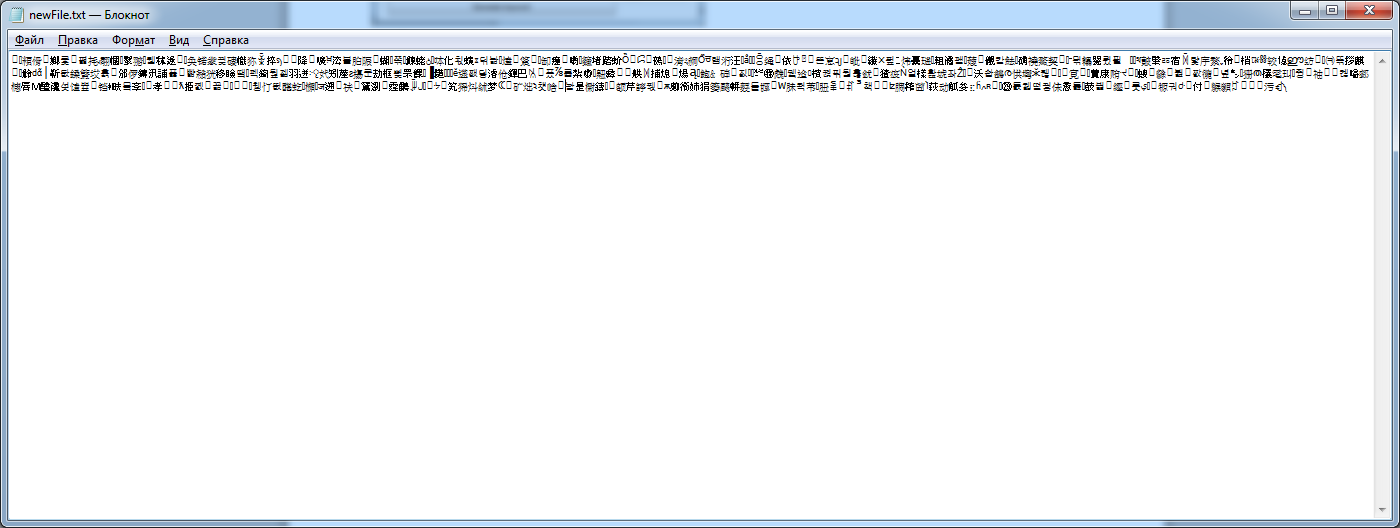
**Тест №3:**

Используем текстовый режим программы, вводим текст, вводим парольную фразу. Выбираем длину ключа в 128 бит. Жмёт кнопку «Encode Text to File» и выбираем путь. Закрываем программу. (Изображение 3.1.12)



**Изображение 3.1.12. Тест №3**

Запустим текстовый файл и посмотрим, что в нем содержится. (Изображение 3.1.13)

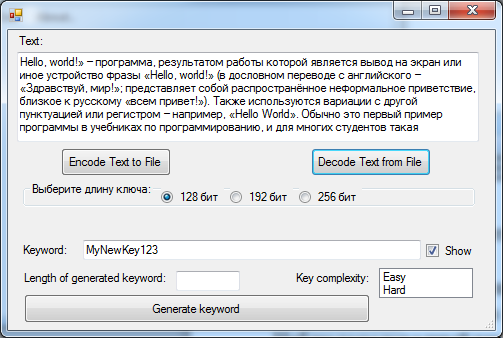


**Изображение 3.1.13. Тест №3**

Файл состоит из «непонятных символов», т.к. он закодирован. Раскодируем его.

Запускаем программу в текстовом режиме. Вводим парольную фразу. Выбираем длину ключа в 128 бит . Выбираем закодированный раннее файл.

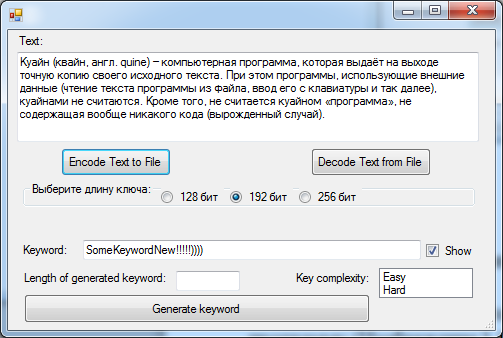
Наблюдаем исходный текст в окне программы. (Изображение 3.1.14)



**Изображение 3.1.14. Тест №3**

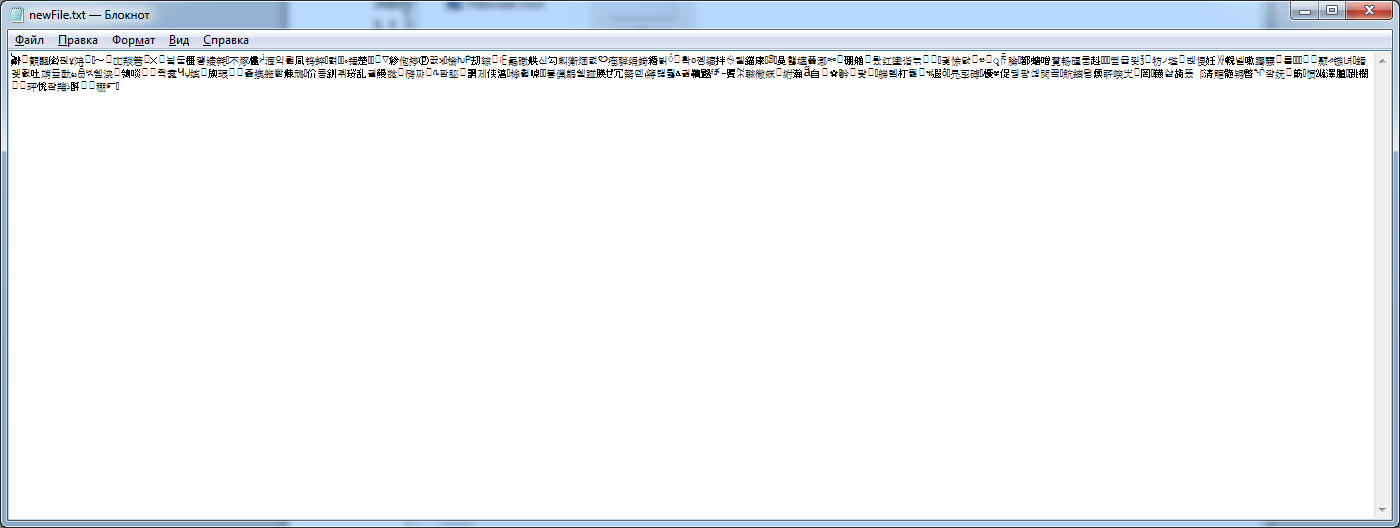
**Тест №4:**

Используем текстовый режим программы, вводим текст, вводим парольную фразу. Жмёт кнопку «Encode Text to File» и выбираем путь. Закрываем программу. (Изображение 3.1.15)



**Изображение 3.1.15. Тест №4**

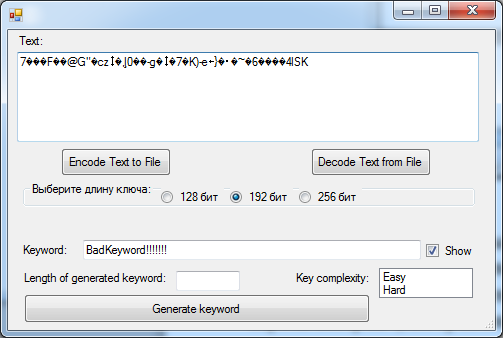
Запустим текстовый файл и посмотрим что в нем содержится (Изображение 3.1.16)



**Изображение 3.1.16. Тест №4**

Файл состоит из «непонятных символов», т.к. он закодирован. Попробуем раскодировать его, введя неверную парольную фразу.

Наблюдаем набор «непонятных» символов, что является критерием того, что пароль неверен. (Изображение 3.1.17)



**Изображение 3.1.17. Тест №4**

Заметим, что такое же поведение будет демонстрироваться, если кодировать текст, выбирая одну длину ключа, а при декодировке файла с текстом выбирать другую.

В результате тестирования были исключены ситуации, в которых пользователь мог выполнять некорректные с точки зрения логики работы программы действия.

# Раздел 4.

# Итоги.

## 4.1 Заключение.

В результате выполнения курсового проекта был изучен криптоалгоритм RC-6 и особенности его конкретной свободно распространяемой реализации.

Данный алгоритм был протестирован и результаты теста сравнивались с результатами теста из открытого источника. (Результаты совпали)

Была написана программа, способная зашифровывать и расшифровывать заданную пользователем информацию по криптоалгоритму RC-6 на ключе, выводимом из введенной пользователем парольной фразы. Кроме того, в данную программу был внедрен генератор случайных паролей, с параметризируемой сложностью и длиной генерируемой парольной фразы.

Данная программа была протестирована на различные способы использования.

## 4.2 Немного из истории криптоалгоритма.

Данный алгоритм является производным от криптоалгоритма RC-5 и был создан для удовлетворения требований конкурса AES, организованного Национальным Институтом Стандартов и Технологий США в 1997 году, для выбора нового криптографического стандарта, который должен был стать преемником криптоалгоритма DES.

В итоге, алгоритм RC-6 стал финалистом данного конкурса, однако не победил в нем, уступив криптоалгоритму Rijndael, ввиду того, что одна из примитивных операций – операция умножения медленно выполняется на некоторых видах оборудования. Из-за этого алгоритм теряет одно из своих ключевых преимуществ – высокую скорость выполнения, что и стало причиной для критики алгоритма и одной из преград для победы в конкурсе AES.

# Список использованных источников.

* Готовая реализация криптоалгоритма RC-6 с открытого источника: *https://github.com/Lybot/RC6*
* Описание алгоритма RC-6 в Википедии:

*https://ru.wikipedia.org/wiki/RC6*

* Наиболее полное описание криптоалгоритма RC-6 (сайт на английском): [*https://web.archive.org/web/20070101014400/http://theory.lcs.mit.edu/~rivest/rc6.pdf*](https://web.archive.org/web/20070101014400/http://theory.lcs.mit.edu/~rivest/rc6.pdf)
* *Описание класса PasswordDeriveBytes:*

*https://docs.microsoft.com/ru-ru/dotnet/api/system.security.cryptography.passwordderivebytes?view=net-5.0*

* Источник, из которого был взят тестовый набор(третья глава, 3ий тестовый набор) (сайт на английском):

*https://tools.ietf.org/id/draft-krovetz-rc6-rc5-vectors-00.html*