Министерство образования и науки РФ

Псковский государственный университет

Колледж ПсковГУ

Утверждаю  
Заместитель директора   
по учебной работе  
\_\_\_\_\_\_\_\_ О.В. Ефимова  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г.

ЗАДАНИЕ

на разработку курсовой работы

*Специальность:* 09.02.03 «Программирование в компьютерных системах»

*Уровень:* Базовый  
*Предмет:* МДК 03.01 Технология разработки программного обеспечения

Тема: ***Разработка программ для решения задач защиты информации***

* изложение материала по теме «шифрование и дешифрование текста тремя разными методами»
* описание алгоритма программы
* составление блок-схемы, характеризующей работу программы
* составление и испытание программы, шифрующей и дешифрующей текст тремя различными методами.

*Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Курс \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Группа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Согласовано

протокол заседания цикловой комиссии   
информационных технологий

№ \_\_ от \_\_\_ \_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

Председатель цикловой комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания " "

Срок окончания комплексного   
курсового проектирования: " "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание выдал (руководитель) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc502223343)

[АКТУАЛЬНОСТЬ ТЕМЫ 4](#_Toc502223344)

[ЦЕЛИ РАБОТЫ 4](#_Toc502223345)

[ЗАДАЧИ РАБОТЫ 4](#_Toc502223346)

[РАЗРАБОТКА АЛГОРИТМА ПРОГРАММЫ 5](#_Toc502223347)

[ЭТАПЫ РАЗРАБОТКИ ПРОГРАММЫ 5](#_Toc502223348)

[ПЕРВЫЙ ЭТАП 5](#_Toc502223349)

[ВТОРОЙ ЭТАП 6](#_Toc502223350)

[ТРЕТИЙ ЭТАП. ТЕОРЕТИЧЕСКИЙ 7](#_Toc502223351)

[DES 7](#_Toc502223352)

[AES 8](#_Toc502223353)

[RSA 11](#_Toc502223354)

[ЧЕТВЕРТЫЙ ЭТАП 13](#_Toc502223355)

[ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ 14](#_Toc502223356)

[РЕЗУЛЬТАТЫ РАБОТЫ 16](#_Toc502223357)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 17](#_Toc502223358)

[ПРИЛОЖЕНИЯ 18](#_Toc502223359)

[КАСТОМИЗАЦИЯ UI. 18](#_Toc502223360)

[СПРАВОЧНАЯ СИСТЕМА 18](#_Toc502223361)

[ТЕКСТ МОДУЛЕЙ ПРОГРАММЫ 18](#_Toc502223362)

[СКРИНШОТЫ 31](#_Toc502223363)

[ДИСК DVD-R 36](#_Toc502223364)

# ВВЕДЕНИЕ

Шифрова́ние — обратимое преобразование информации в целях скрытия от неавторизованных лиц, с предоставлением, в это же время, авторизованным пользователям доступа к ней. Главным образом, шифрование служит задачей соблюдения конфиденциальности передаваемой информации. Важной особенностью любого алгоритма шифрования является использование ключа, который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма. Такое определение шифрования дает Википедия. Из него можно сделать вывод, что, используя шифрование можно с помощью ключа «скрыть» и «открыть» какую-либо информацию.

Изучением способов шифрования занимается наука криптография, зародившаяся около 4-х тысяч лет назад. Сейчас существует множество способов шифрования. В своей работе я рассмотрю 3 из них: AES, DES и RSA. Первые 2 способа – симметричные и отличаются длинной ключа и применяемыми методами шифрования информации. Третий способ ассиметричный.

# АКТУАЛЬНОСТЬ ТЕМЫ

В настоящее время, когда, передача сбор и использование различной информации стала доступен для всех (фото, видео, скрины переписок и т.п.) вопрос о сохранении её конфиденциальности встал особо остро. Моя программа поможет зашифровать необходимую информацию: от пароля до архива. И также безопасно передать ее получателю в зашифрованном виде.

# ЦЕЛИ РАБОТЫ

Целью данной работы являются:

* + 1. Углубление и систематизация знаний, совершенствование навыков по разработке программного обеспечения и его реализации.
    2. Создание программы D' Шифр, способной зашифровывать и расшифровывать файл тремя способами:
* AES
* DES
* RSA

# ЗАДАЧИ РАБОТЫ

Задачами данной работы являются:

1. Изложение теоретического материала по теме «шифрование и дешифрование файлов одним из трех способов»
2. Описание алгоритма программы
3. Описание программы
4. Составление блок-схемы, характеризующей работу программы
5. Написание и испытание программы D' Шифр, шифрующей и дешифрующей файлы одним из трех способов.

# РАЗРАБОТКА АЛГОРИТМА ПРОГРАММЫ

Исходя из поставленных задач было мной было принято решение, что алгоритм программы должен выглядеть следующим образом:

Пользователь выбирает вид шифрования (AES, DES или RSA) -> Пользователь выбирает файл -> Пользователь выбирает что хочет сделать с файлом (зашифровать или расшифровать) -> Программа производит необходимые действия и предлагает пользователю сохранить файл.

# ЭТАПЫ РАЗРАБОТКИ ПРОГРАММЫ

## ПЕРВЫЙ ЭТАП

Для реализации мной был выбран язык программирования java. По нескольким причинам:

* Java имеет множество библиотек, которые можно интегрировать в проект для решения основных задач (javafx для построения пользовательского интерфейса) или для оживления пользовательского интерфейса (javazoom для интегрирования музыки).
* Имеющийся положительный опыт написания программного обеспечения на java
* Наличие большого количества информационных ресурсов и литературы по данному языку программирования
* Java предоставляет готовые классы для работы с необходимыми мне алгоритмами шифрования

## ВТОРОЙ ЭТАП

На этом этапе была разработана схема пользовательского интерфейса и взаимодействия окон в нем.



Данная схема отражает только пользовательский интерфейс.

## ТРЕТИЙ ЭТАП. ТЕОРЕТИЧЕСКИЙ

На этом этапе мной были изучены механизмы и математический основания работы каждого из алгоритмов.

Существуют симметричные (для шифрования и дешифровки используется один ключ) и ассиметричные (используются открытый и закрытый ключи) алгоритмы шифрования. В свою очередь они делятся на блочные (файл разбивается на блоки фиксированной длины, затем они шифруются) и поточные (шифруется каждый символ).

DESData Encryption Standard (стандарт шифрования данных) является симметричным (ГОСТ 28147-89) блочным алгоритмом. Действовал в качестве государственного стандарта в области шифрования данных в США с 1977 по 2001 годы. Длина блока 64 бита, длина ключа 56 бит, количество раундов – 16. DES является классической сетью Фейштеля с двумя ветвями. Алгоритм преобразует за несколько раундов 64-битный входной блок данных в 64-битный выходной блок. Стандарт DES построен на комбинированном использовании перестановки, замены и гаммирования.

*Для всех блок-схем по умолчанию принять следующее: в блоке выбора да/нет – правая стрелка «да», левая «нет».*

Блок-схема алгоритма шифрования:

end = 16

round — переменная хранящая номер раунда

Процесс шифрования состоит из двух перестановок, которые называют начальной и финальной (конечной) перестановками, и 16 раундов Фейштеля. Каждый раунд использует различные сгенерированные 48 - битовые ключи (Ключ).

Начальная перестановка бит IP

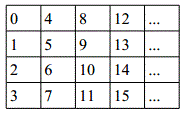
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 | 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 | 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 | 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 | 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

Конечная перестановка бит IP-1

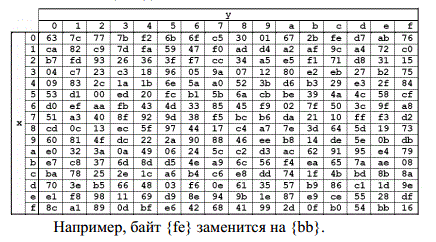
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 | 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 | 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 | 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 | 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

С ростом вычислительных мощностей DES стал менее надежным. Из-за небольшого числа возможных ключей (всего 256), появляется возможность их полного перебора за реальное время. В 1998 году Electronic Frontier Foundation, используя специальный компьютер DES-Cracker, удалось взломать DES за 3 дня. В связи с чем на смену ему пришел AES.

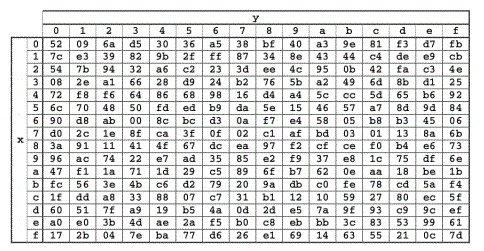
AESAdvanced Encryption Standard (Расширенный стандарт шифрования)относится к симметричный блочным алгоритмам. Пришел на смену DES в 2001 году. В основу стандарта положен шифр Rijndael (линейно-подстановочные преобразования с использованием табличных вычислений). Длина блока равна 128 бит, ключа 128/192/256 бит (количество раундов соответственно 10, 12, 14.

State (форма) – матрица (двумерный массив) байтов, расположенных следующем образом:

Round (раунд) – итерация цикла преобразования над State (формой). Количество итераций зависит от длины ключа.

Round key (раундовый ключ) ― уникальный ключ, который применяется в каждом отдельном в раунде.

S-Box (таблица подстановок) –таблица, которая задает отображение одного байта в другой:



Обратная таблица подстановок ― аналогично S-Box, задает обратное отображение:

Блок-схема алгоритма шифрования:

ExpandKey — Функция для вычисления всех раундовых ключей

SubBytes — Функция для подстановки байтов, использующая таблицу подстановок

ShiftRows — Функция, обеспечивающая циклический сдвиг в форме на различные величины

MixColumns — Функция, которая смешивает данные внутри каждого столбца формы

AddRoundKey — Сложение ключа раунда с формой

round — переменная хранящая номер раунда

end — количество раундов

Процесс шифрования данных состоит из нескольких этапов:

* вычисление всех раундовых ключей; подстановка байтов с помощью основной таблицы S-Box
* сдвиг в форме с использованием различающихся величин
* смешивание данных внутри каждого столбца матрицы (формы); сложение формы и раундового ключа.

Дешифровка производится в обратном порядке, но вместо таблицы S-Box применяется обратная таблица постановок, которая была упомянута выше. Если привести пример, при наличии ключа длиной 4 бита для перебора потребуется всего 16 стадий (раундов), то есть необходимо проверить все возможные комбинации, начиная с 0000 и заканчивая 1111. Естественно, такая защита взламывается достаточно быстро. Но если взять ключи побольше, для 16 бит потребуется 65 536 стадий, а для 256 бит – 1,1 х 1077. И как было заявлено американскими специалистами, на подбор правильной комбинации (ключа) уйдет порядка 149 триллионов лет.

AES-шифрование на сегодняшний день является продвинутым и защищенным, независимо от того, какая применяется длина ключа. Неудивительно, что именно этот стандарт используется в большинстве криптосистем и имеет достаточно широкие перспективы на развитие и усовершенствование в обозримом будущем, тем более что очень вероятным может оказаться и объединение нескольких типов шифрования в одно целое (например, параллельное использование симметричного и асимметричного или блочного и потокового шифрования).

RSA(аббревиатура от фамилий Rivest, Shamir и Adleman) это ассиметричный (ГОСТ 34.10-2001) алгоритм шифрования основывающийся на вычислительной сложности задач факторизации произведения двух простых больших чисел. Для шифрования используется простая операция возведения в степень по модулю N. Для дешифрования необходимо вычислить функцию Эйлера от числа N, для этого необходимо знать разложение числа n на простые множители (В этом и состоит задача факторизации).

Генерация пары (открытый и закрытый) ключей в RSA осуществляется следующим образом:

1. Выбираются два простых числа p и q (такие что p неравно q).
2. Вычисляется модуль N = p∗q.
3. Вычисляется значение функции Эйлера от модуля N: ϕ(N)=(p−1) (q−1).
4. Выбирается число e, называемое открытой экспонентой, число e должно лежать в интервале 1 < e < ϕ (N), а также быть взаимно простым со значением функции ϕ (N).
5. Вычисляется число d, называемое секретной экспонентой, такое, что d∗ e = 1 (mod ϕ(N)), то есть является мультипликативно обратное к числу e по модулю ϕ (N).

Итак, мы получили пару ключей:

Пара (e, N) - открытый ключ.

Пара (d, N) - закрытый ключ.

Обычно шифрование RSA используется для передачи сообщений по следующему сценарию: Маша и Петя переписываются в интернете и хотят использовать шифрование. Маша заранее сгенерировала закрытый и открытый ключ, а затем отправила открытый ключ Пете. Чтобы послать зашифрованное сообщение Пете нужно зашифровать сообщение *m*, используя открытый ключ, который прислала ему Маша (*e*, *N*) : *C*=*E*(*M*) = *M e mod*(*N*) и отправляет. Маша принимает зашифрованное сообщение *c*. Используя закрытый ключ (*d*, *N*), расшифровывает его *M* = *D*(*C*) = *C d mod* (*N*).

Проиллюстрируем то, что описано выше:

Причины безопасности RSA. Представим себе, что к Маше и Пете присоединяется Катя, которая хочет узнать, что пишет Петя. У Кати есть открытый ключ Маши (e, N), для того, чтобы расшифровать сообщение c, необходимо знать закрытый ключ (d, N). Мы знаем, что d∗e=1(modϕ(N)), однако Катя не знает ϕ(N)=(p−1)∗(q−1), т.е задача сводится к нахождению простых чисел p и q (хотя это не всегда так), которые связаны с известным N следующим образом N=p∗q, что сложно выполнимо.

Благодаря вышеописанным свойствам криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. Алгоритм используется в большом числе криптографических приложений, включая PGP, S/MIME, TLS/SSL, IPSEC/IKE и других. Но есть и минусы: в связи с высокими временными затратами на шифрование алгоритм используется главным образом для шифрования небольших данных (пароли, короткие сообщения).

## ЧЕТВЕРТЫЙ ЭТАП

На данном этапе мной была разработана блок-схема программы:

# ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

В главном окне программы, сформированном средствами javafx пользователь выбирает алгоритм, которым хотел бы воспользоваться (рис. 1). Информация об этом передается в виде аргумента title в функцию display класса AlertWindow, отвечающего за формирование окна выбора файла и обработку действий в нем (рис. 2). Для реализации выбора файла мной был использованы методы класса FileChooser (рис. 3), возвращающие объект File. После подтверждения выбора файла путь открывается окно выбора действия с файлом windowChoose класса AlertWindow (рис. 4). Если был выбран алгоритм RSA, то помимо кнопок «Расшифровать» и «Зашифровать» появляется кнопка «Сгенерировать ключи» (рис. 5), ей можно воспользоваться для генерации пары ключей вне процесса шифрования файла. В этом указано имя файла, который был выбран (полученное с помощью file().getName()). Далее возможно несколько вариантов:

* Кнопка «Зашифровать». Если выбран алгоритм RSA, то проверяется размер файл, когда он превышает 116 б появляется сообщение об ошибке BadFile, в который параметром err функции display передается текст с сутью ошибки (рис. 6) (это связано с большим временем необходимым для шифрования файла). Переменной toDo присваивается значение «encrypt». После появляется окно с информацией о удачной зашифровке (зашифровка файла еще не произведена!) и предложением сохранить его CloseDialog (рис. 7). Подробнее об окне CloseDialog ниже.
* Кнопка «Расшифровать». Проверяется не был ли данный файл расшифрован ранее (содержится, ли в его имени «\_расшифрован»). Если да, то появляется окно ошибки (рис. 8). Возможно отказаться от расшифровки или продолжить её. Если нет, то производится проверка зашифрован ли файл (содержится, ли в его имени «\_зашифрован»). Если нет, то появляется аналогичное окно ошибки (рис. 9). Затем проверяется, каким алгоритмом был зашифрован файл (по части в названии файла), если не совпадает с выбранным, то появляется окно ошибки (рис. 10). В это случае произвести расшифровку нельзя. Если ошибок не найдено, то переменной toDo присваивается значение «decrypt» и появляется окно с информацией о удачной расшифровке (расшифровка файла еще не произведена!) и предложением сохранить его CloseDialog (рис.7).

При попытке закрыть окно CloseDialog появится предупреждение с просьбой сделать выбор (текст просьбы может быть разным и выбирается с помощью функции random из имеющихся в программе вариантов) (рис. 11 ,12, 13). При нажатии на кнопку «Не сохранять» закроются все окна, кроме главного. На кнопку «Сохранить» происходит зашифровка или расшифровка (в зависимости от значения переменной toDo) файла выбранным алгоритмом (считывается значение из заголовка окно, ранее переданный туда title). Процесс зашифровки и расшифровки вынесен именно сюда с целью уменьшения нагрузки при работе программы (файл обрабатывается только в том случае, если пользователь хочет его сохранить).

Зашифровка и расшифровка реализована в классе Controller. С помощью класса Cipher. Путь к необходимому файлу получается путем AlertWindow.file.getAbsolutePath(). Сохранение производится в ту же директорию, где расположен сам файл. К имени добавляется «\_расшифрован/зашифрован\_имя алгоритма». Расширение остается прежним. Например: file\_зашифрован\_DES.txt. После сохранения появляется информационное окно (рис. 14). После его закрытия пользователь возвращается в главное окно.

Подробно процедуры и их назначение описаны в комментариях к коду программы.

Инструкция для пользователя доступна в разделе Мануал раздела Помощь.

# РЕЗУЛЬТАТЫ РАБОТЫ

1. Изложен теоретический материала по теме «шифрование и дешифрование файлов одним из трех способов»
2. Описан алгоритма программы
3. Составлено описание работы программы
4. Составлена блок-схема, характеризующая работу программы
5. Написана и испытана программа D' Шифр, шифрующая и дешифрующая файлы одним из трех способов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования
2. ГОСТ Р 34.10-2001 Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи
3. <https://ru.wikipedia.org/wiki/RSA>
4. <https://ru.wikipedia.org/wiki/AES>
5. <https://ru.wikipedia.org/wiki/DES>
6. <http://teh-box.ru/programming/algoritm-shifrovaniya-aes-dlya-samyx-malenkix.html>
7. <http://fb.ru/article/279219/aes-shifrovanie-dannyih>
8. <http://teh-box.ru/informationsecurity/algoritm-shifrovaniya-rsa-na-palcax.html>
9. «Методы дискретной математики в криптологии» - В.М. Фомичев. «ДИАЛОГ-МИФИ» 2003 г.
10. «Основы криптографии» - А.П. Алферов, А.Ю. Зубов, А.С. Кузьмин, А.В. Черемушкин М.: Гелиос АРВ 2002 г.
11. «Просто криптография» Виктор Де Касто Страта 2016 г.

# ПРИЛОЖЕНИЯ

## КАСТОМИЗАЦИЯ UI.

Для придания программе индивидуальности мной было принято решение кастомизировать пользовательский интерфейс. Он выполнен в стеле 8-bit игры Mario. Для этого интегрирована музыка средствами библиотеки javazoom (класс Player). В главном окне программы использован характерный фон. В остальных окнах присутствуют изображения, также выполненные в едином стиле. Многие изменены мной вручную, с помощью программы Photoshop CS5 (рис. 15). Использован шрифт Courier New. В названиях окон использованы иконки, соответствующие информационному наполнению.

## СПРАВОЧНАЯ СИСТЕМА

В главном окне имеется кнопка помощь. Нажав на нее можно выбрать раздел «О программе», который содержит информацию о версии и авторе (рис. 16), или «Мануал» – внутреннюю справочную систему программы (рис. 17). Файл справки создан в программе Help & Manual 6 (рис. 18) . Для подключения к программе проект был опубликован в формате HTML. Для открытия используется WebView.

## ТЕКСТ МОДУЛЕЙ ПРОГРАММЫ

Main.java

package javas;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.geometry.Side;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.MenuButton;

import javafx.scene.control.MenuItem;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.layout.\*;

import javafx.stage.Stage;

import javazoom.jl.decoder.JavaLayerException;

import javax.sound.sampled.LineUnavailableException;

import javax.sound.sampled.UnsupportedAudioFileException;

import java.io.IOException;

public class Main extends Application{

public Main() throws IOException, JavaLayerException, UnsupportedAudioFileException, LineUnavailableException {

}

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) throws Exception {

primaryStage.setTitle("D' Шифр");

primaryStage.getIcons().add(new Image("icon\_mini.jpg"));

primaryStage.setOnCloseRequest(e -> Player.stopMusic());

//создание и наполнение меню

MenuButton menuHelp = new MenuButton("Помощь");

menuHelp.setPrefWidth(86);

menuHelp.setPopupSide(Side.RIGHT);

//пункт О программе и открытие при нажатии

MenuItem helpItem = new MenuItem("О программе");

// задаем Id для управления стилем

helpItem.setId("labelMenu");

helpItem.setOnAction(e-> About.display());

MenuItem manualItem = new MenuItem("Мануал");

manualItem.setId("labelMenu");

manualItem.setOnAction(e -> Manual.display());

menuHelp.getItems().addAll(helpItem, manualItem);

//задаем ориентацию меню и добавляем в бокс элементы

VBox menuBox = new VBox();

menuBox.getChildren().addAll(menuHelp);

ImageView imageView = new ImageView("icon\_.jpg");

Label labelMain = new Label("Выберите алгоритм шифрования:");

Label labelProgName = new Label("D' ШИФР");

labelProgName.setId("labelProgName");

// создание кнопок и привязка действий

Button buttonRsa = new Button("RSA");

buttonRsa.setId("buttonAlg");

buttonRsa.setOnAction(e -> AlertWindow.display("RSA"));

Button buttonDes = new Button("DES");

buttonDes.setId("buttonAlg");

buttonDes.setOnAction(e -> AlertWindow.display("DES"));

Button buttonAes = new Button("AES");

buttonAes.setId("buttonAlg");

buttonAes.setOnAction(e -> AlertWindow.display("AES"));

Button buttonStopMario = new Button("Я не люблю Марио!");

buttonStopMario.setOnAction(e -> stop());

//созданеи боксов для разметки

HBox buttonBox = new HBox(40);

buttonBox.setPadding(new Insets(10, 0, 0, 0));

buttonBox.setAlignment(Pos.CENTER);

buttonBox.getChildren().addAll(buttonAes, buttonDes, buttonRsa);

VBox leftBox = new VBox(2);

leftBox.setPadding(new Insets(0, 15, 0, 0));

leftBox.getChildren().addAll(imageView, labelProgName );

leftBox.setAlignment(Pos.TOP\_CENTER);

VBox centerBox = new VBox(15);

centerBox.setAlignment(Pos.TOP\_CENTER);

centerBox.getChildren().addAll(labelMain, buttonBox);

HBox bottomBox = new HBox(200);

bottomBox.setAlignment(Pos.BOTTOM\_LEFT);

bottomBox.getChildren().addAll(menuBox, buttonStopMario);

//разметка главного окна

BorderPane layoutMain = new BorderPane();

layoutMain.setPadding(new Insets(20,20,5,20));

//определение места отображения Боксов

layoutMain.setLeft(leftBox);

layoutMain.setCenter(centerBox);

layoutMain.setBottom(bottomBox);

//применяем разметку к первому окну

Scene sceneMain = new Scene(layoutMain, 500, 180);

//задания первого окна при открытии приложения

primaryStage.setScene(sceneMain);

//задаем стиль

sceneMain.getStylesheets().add("style.css");

primaryStage.show();

// фон

BackgroundImage myBI = new BackgroundImage(new Image("fon1.jpg", 450, 180, true, false), BackgroundRepeat.REPEAT, BackgroundRepeat.REPEAT, BackgroundPosition.DEFAULT, BackgroundSize.DEFAULT);

layoutMain.setBackground(new Background(myBI));

play ();

}

private void play() {

Player.playMusic();

}

public void stop() {

Player.stopMusic();

}

}

Выдержка из Controller.java

package javas;

import javax.crypto.\*;

import java.io.\*;

import java.security.\*;

class Controller {

private static Key pubKeyRSA;

private static Key privKeyRSA;

static void DES(int n) throws Exception {

Cipher cipher = Cipher.getInstance("DES");

SecretKey secKeyDES;

String fileName = AlertWindow.file.getName();

int indexEncrypt = fileName.lastIndexOf(".");

int indexDecrypt = fileName.lastIndexOf("з");

//разделение на зашифровку и расшифровку

// сперва зашифровка

if (n == 1) {

// System.out.println("зашифровка DES");

//генерация ключа

KeyGenerator keyGenerator = KeyGenerator.getInstance("DES");

secKeyDES = keyGenerator.generateKey();

//сериализация ключа

FileOutputStream fileOutputStream = new FileOutputStream("secKeyDES");

ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);

objectOutputStream.writeObject(secKeyDES);

objectOutputStream.close();

//указание, что хотим зашифровать

cipher.init(Cipher.ENCRYPT\_MODE, secKeyDES);

//получаем путь к файлу, который будем шифровать

String cleartextFile = AlertWindow.file.getAbsolutePath();

//формируем имя файла, в который будем сохранять расшифроанный докумнет

String cipherFile = AlertWindow.file.getParent() + "\\" + fileName.substring(0, indexEncrypt) + "\_зашифрован\_DES." + fileName.substring(fileName.lastIndexOf(".") + 1);

// объявляем потоки

FileInputStream fis = new FileInputStream(cleartextFile);

FileOutputStream fos = new FileOutputStream(cipherFile);

// System.out.println("Зашифровали " + cleartextFile);

// System.out.println("Вот сюда " + cipherFile);

// сохраняем полученный файл

CipherOutputStream cos = new CipherOutputStream(fos, cipher);

byte[] block = new byte[8];

int i;

while ((i = fis.read(block)) != -1) {

cos.write(block, 0, i);

}

// закрываем поток

cos.close();

// расшифровка

} else if (n == 2) {

// System.out.println("расшифровываем DES");

// достаем ключ из файла

FileInputStream fileInputStream = new FileInputStream("secKeyDES");

ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);

secKeyDES = (SecretKey) objectInputStream.readObject();

//указание, что хотим расшифровать

cipher.init(Cipher.DECRYPT\_MODE, secKeyDES);

//получаем путь к файлу, который будем расшифровывать

String cipherFile = AlertWindow.file.getAbsolutePath();

//формируем имя файла, в который будем сохранять расшифроанный докумнет

String cleartextAgainFile = AlertWindow.file.getParent() + "\\" + fileName.substring(0, indexDecrypt) + "расшифрован\_DES." + fileName.substring(fileName.lastIndexOf(".") + 1);

// объявляем потоки

FileOutputStream fos = new FileOutputStream(cleartextAgainFile);

FileInputStream fis = new FileInputStream(cipherFile);

// System.out.println("Расшифровываем сюда" + cleartextAgainFile);

// System.out.println("Вот это " + cipherFile);

// сохраняем полученный файл

CipherInputStream cis = new CipherInputStream(fis, cipher);

byte[] block = new byte[8];

int i;

while ((i = cis.read(block)) != -1) {

fos.write(block, 0, i);

}

fos.close();

}

}

// часть вырезана//

static void RsaKayGen() throws Exception {

KeyPairGenerator genPair = KeyPairGenerator.getInstance("RSA");

KeyPair keys = genPair.generateKeyPair();

pubKeyRSA = keys.getPublic();

privKeyRSA = keys.getPrivate();

// System.out.println(privKeyRSA);

// System.out.println(pubKeyRSA);

//записываем закрытый ключи в файл

FileOutputStream fileOutputStreamPriv = new FileOutputStream("privKeyRSA");

ObjectOutputStream objectOutputStreamPriv = new ObjectOutputStream(fileOutputStreamPriv);

objectOutputStreamPriv.writeObject(privKeyRSA);

objectOutputStreamPriv.close();

FileOutputStream fileOutputStreamPub = new FileOutputStream("pubKeyRSA");

ObjectOutputStream objectOutputStreamPub = new ObjectOutputStream(fileOutputStreamPub);

objectOutputStreamPub.writeObject(pubKeyRSA);

objectOutputStreamPub.close();

}

static void RSA(int n) throws Exception {

Cipher cipher = Cipher.getInstance("RSA");

String fileName = AlertWindow.file.getName();

int indexEncrypt = fileName.lastIndexOf(".");

int indexDecrypt = fileName.lastIndexOf("з");

if (n==1) {

// System.out.println("Зашифровка RSA");

//проверяем есть ли на ПК публичный ключ для шифроания файлов

if (!new File("pubKeyRSA").exists()) {

//если нет, генерим пару ключей и сохраняем

//генерируем пару ключей (открытый и закрытый)

try {

RsaKayGen();

} catch (Exception e) {

e.printStackTrace();

}

} else {

//если есть, берем из файла

// достаем ключ из файла

// System.out.println("есть");

FileInputStream fileInputStream = new FileInputStream("pubKeyRSA");

ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);

pubKeyRSA = (Key) objectInputStream.readObject();

}

//указание, что хотим зашифровать

cipher.init(Cipher.ENCRYPT\_MODE, pubKeyRSA);

String cleartextFile = AlertWindow.file.getAbsolutePath();

String cipherFile = AlertWindow.file.getParent() + "\\" + fileName.substring(0, indexEncrypt) + "\_зашифрован\_RSA." + fileName.substring(fileName.lastIndexOf(".") + 1);

FileInputStream fis = new FileInputStream(cleartextFile);

FileOutputStream fos = new FileOutputStream(cipherFile);

System.out.println("Зашифровали " + cleartextFile);

System.out.println("Вот сюда " + cipherFile);

CipherOutputStream cos = new CipherOutputStream(fos, cipher);

byte[] block = new byte[8];

int i;

while ((i = fis.read(block)) != -1) {

cos.write(block, 0, i);

}

cos.close();

} else if (n==2) {

// достаем ключ из файла

FileInputStream fileInputStream = new FileInputStream("privKeyRSA");

ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);

privKeyRSA = (Key) objectInputStream.readObject();

// System.out.println(privKeyRSA);

//указание, что хотим расшифровать

cipher.init(Cipher.DECRYPT\_MODE, privKeyRSA);

String cipherFile = AlertWindow.file.getAbsolutePath();

String cleartextAgainFile = AlertWindow.file.getParent() + "\\" + fileName.substring(0, indexDecrypt) + "расшифрован\_RSA." + fileName.substring(fileName.lastIndexOf(".")+1);

FileOutputStream fos = new FileOutputStream(cleartextAgainFile);

FileInputStream fis = new FileInputStream(cipherFile);

System.out.println("Расшифровываем сюда" + cleartextAgainFile);

System.out.println("Вот это " + cipherFile);

CipherInputStream cis = new CipherInputStream(fis, cipher);

byte[] block = new byte[8];

int i;

while ((i = cis.read(block)) != -1) {

fos.write(block, 0, I;}

fos.close();

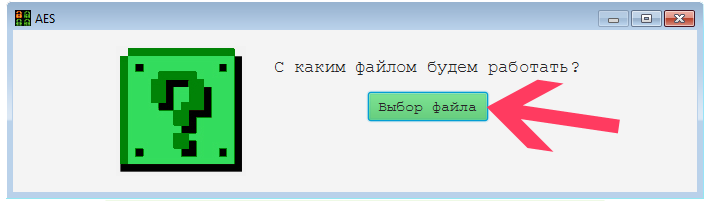
}}}

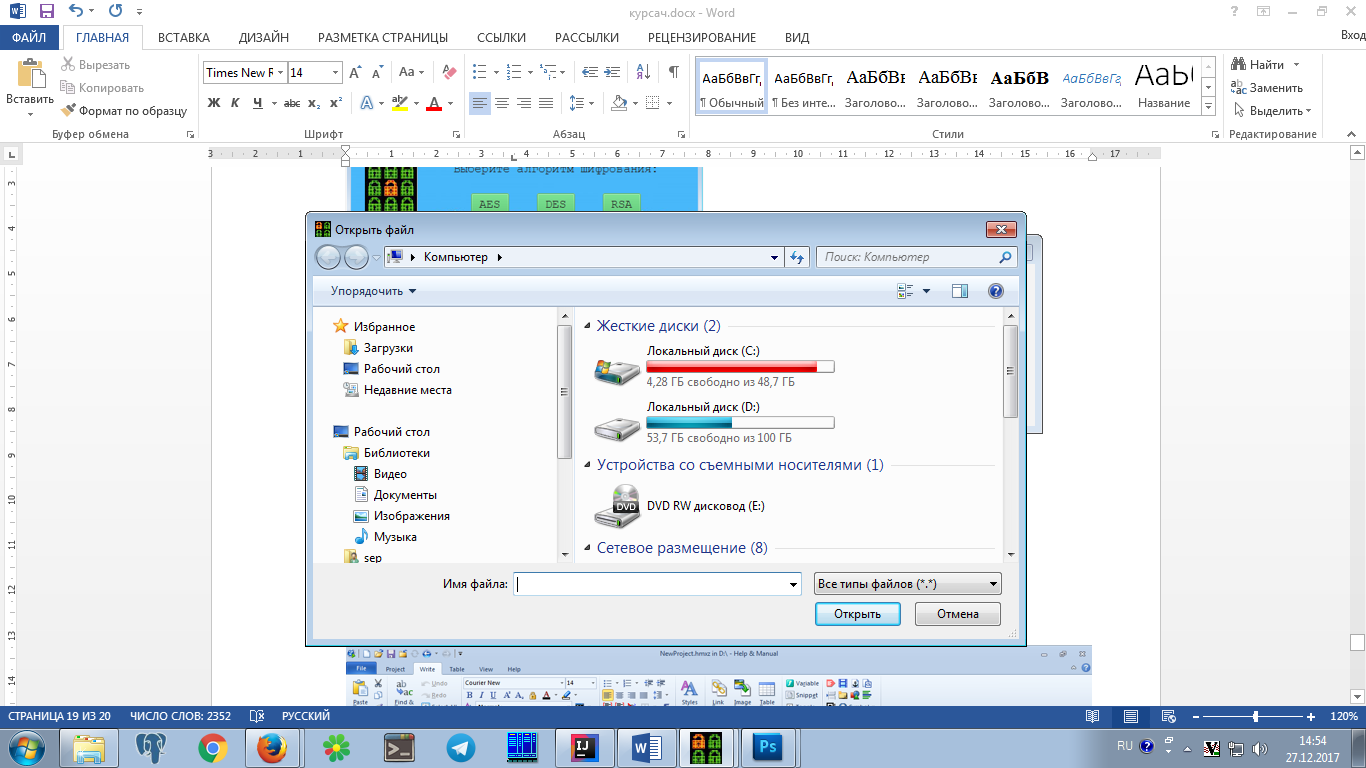
Представлены не все модули. Остальные можно посмотреть на приложенном диске.

## СКРИНШОТЫ

Рис. 1 – главное окно программы

Рис. 2 – окно выбора файла



Рис. 3 – выбор файла

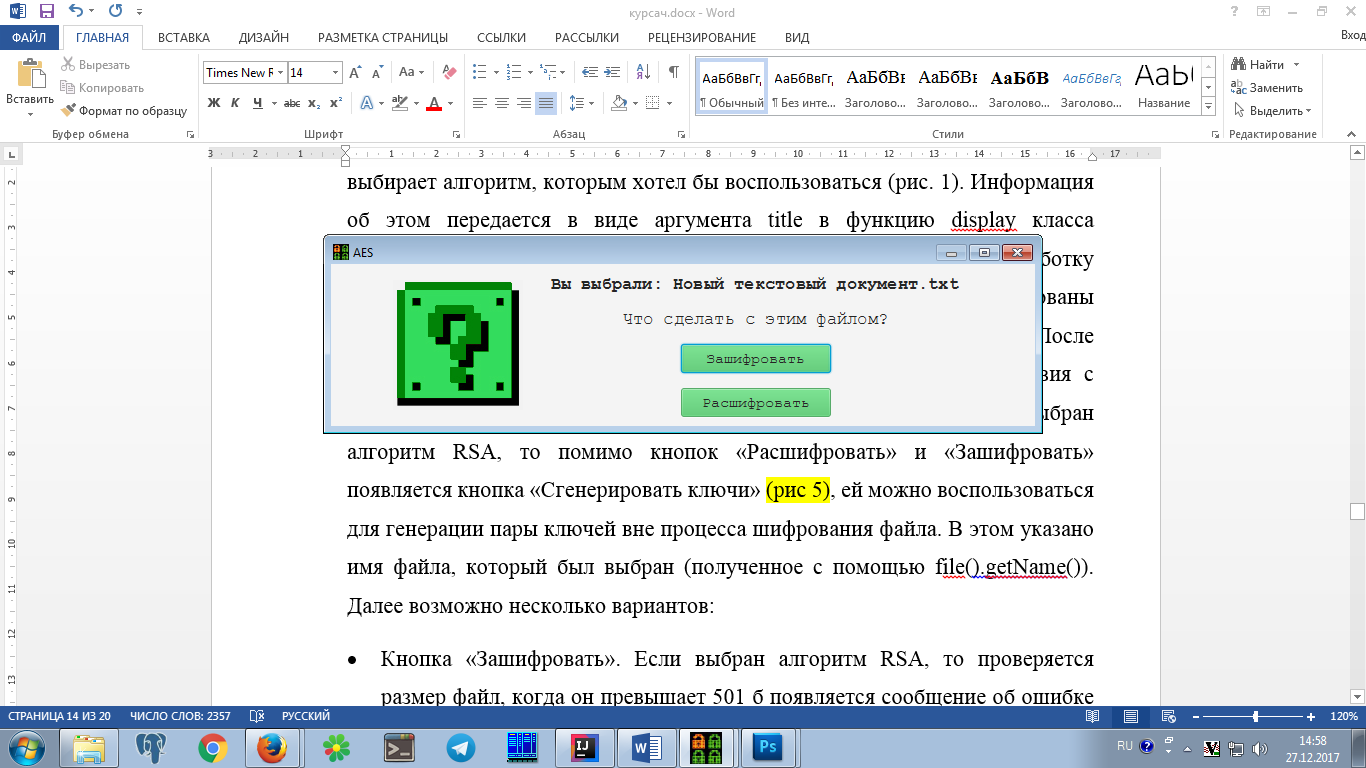
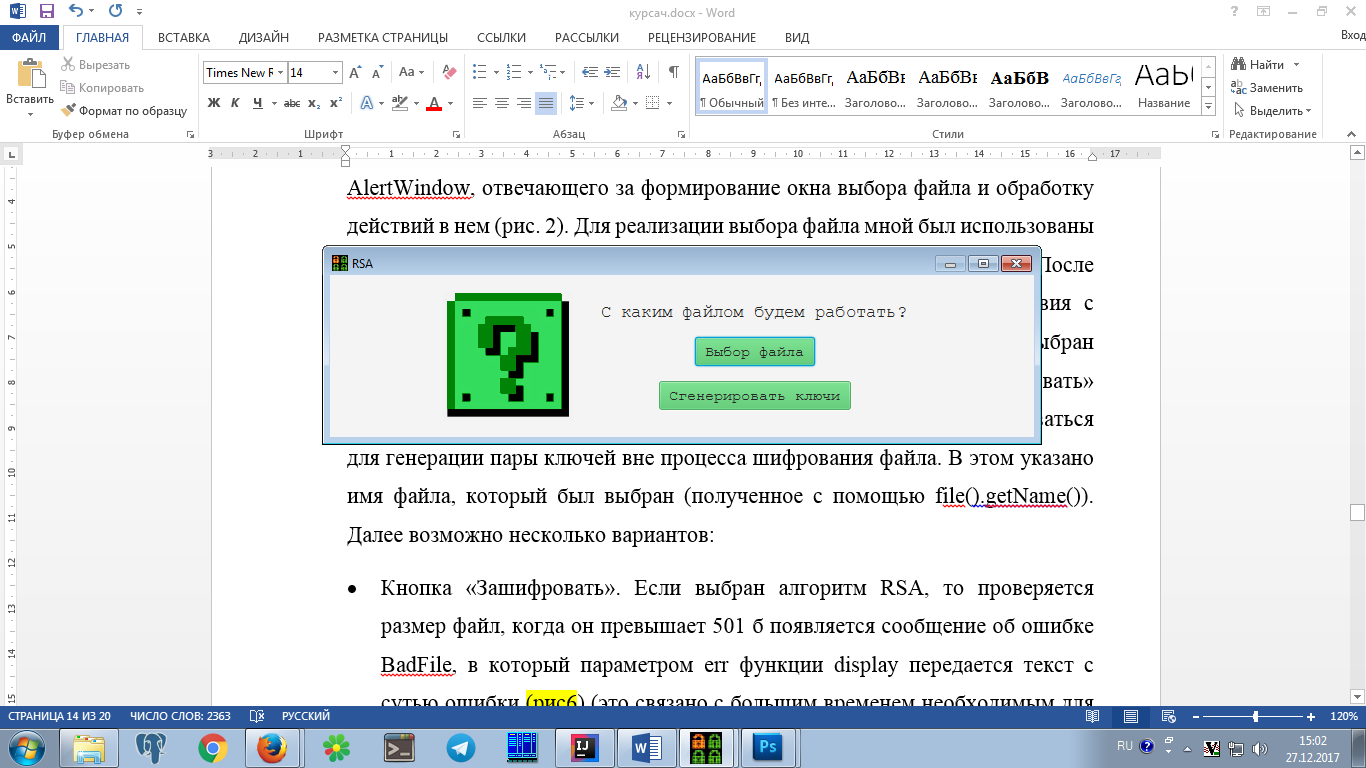
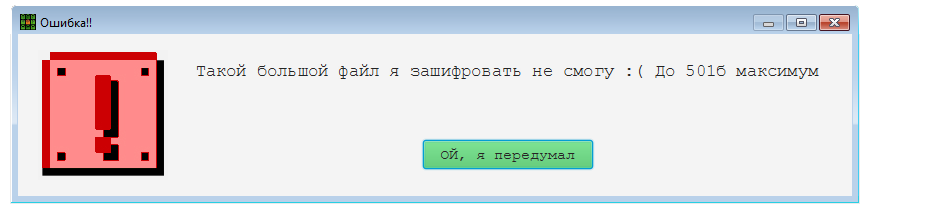
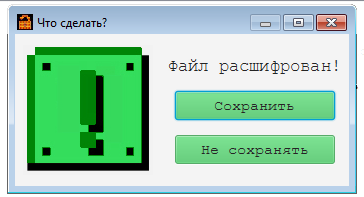
Рис. 4 – окно действия с файлом

Рис. 5 – окно RSA



Рис. 6 – ошибка RSA

Рис. 7 – окно сохранения файла

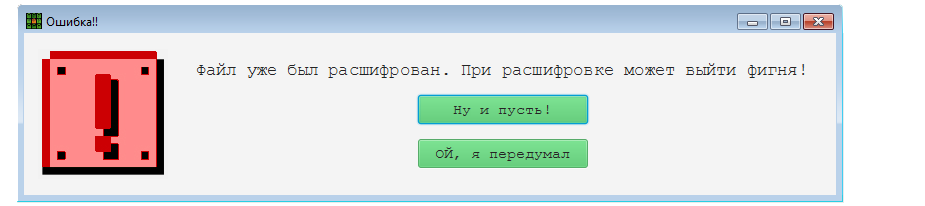
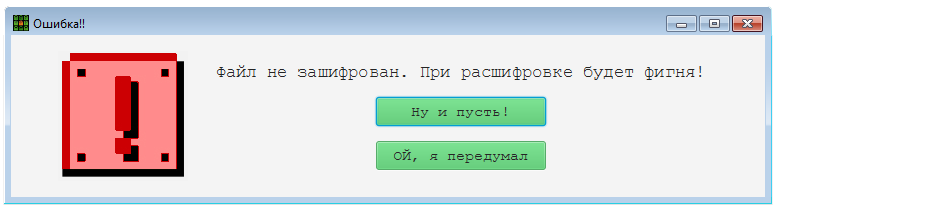
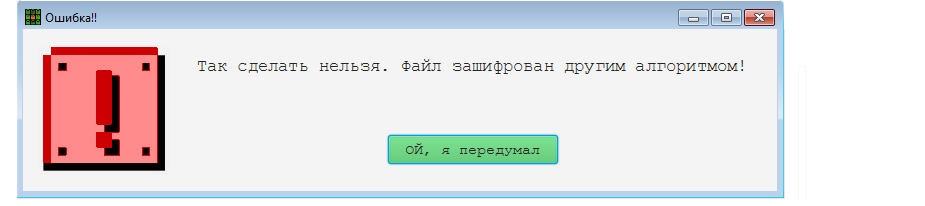
Рис. 8 – ошибка «Файл уже расшифрован»

Рис. 9 – ошибка «Файл не зашифрован»



Рис. 10 – ошибка «Файл зашифрован другим алгоритмом»

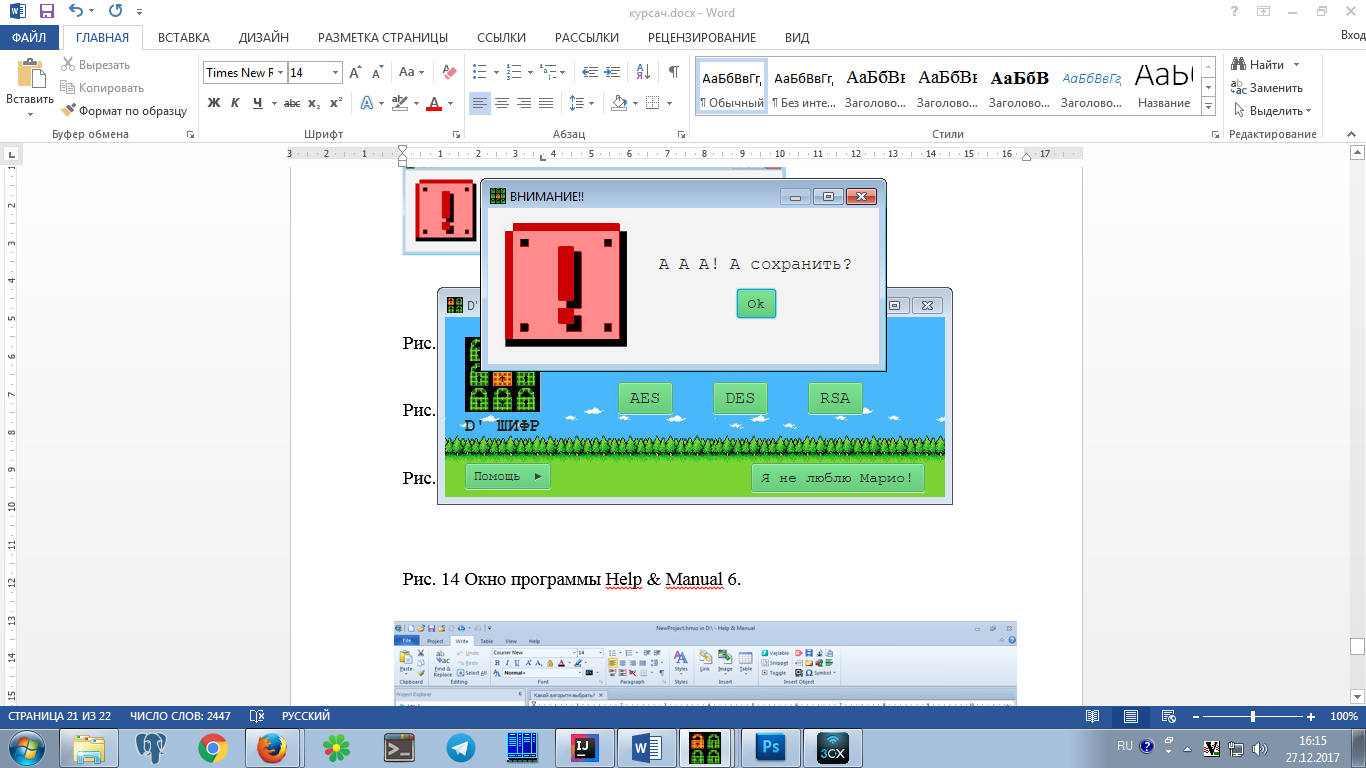
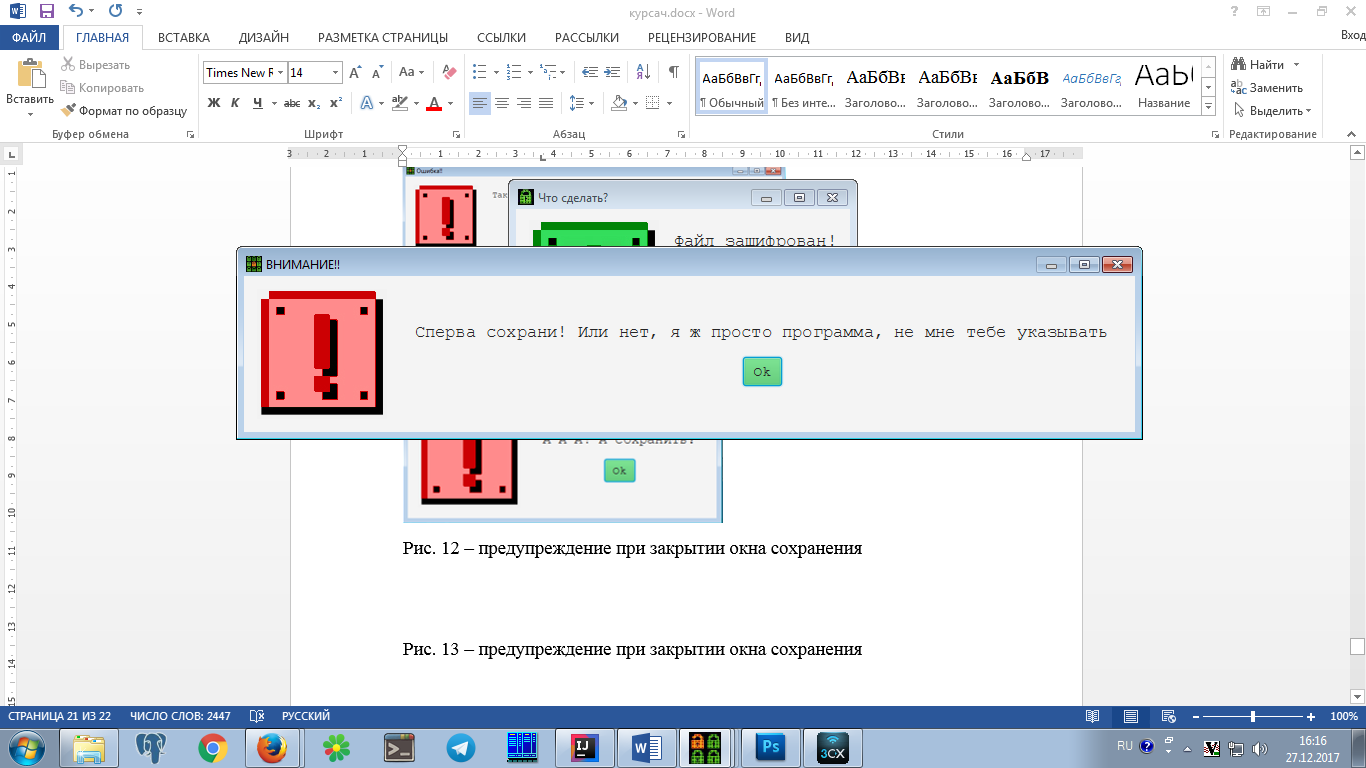
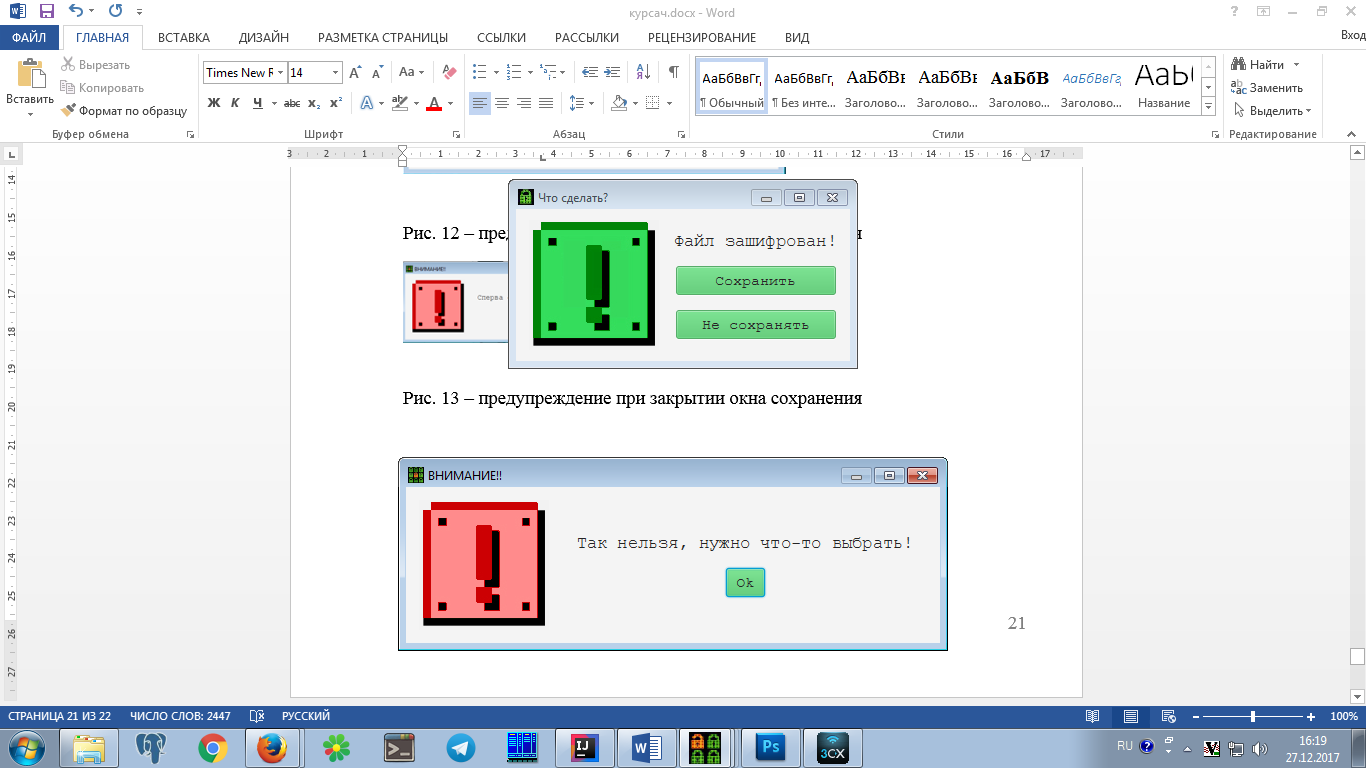
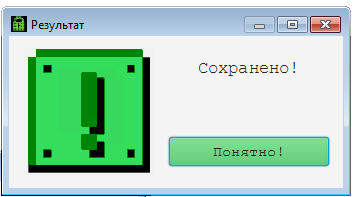
Рис. 11 – предупреждение при закрытии окна сохранения

Рис. 12 – предупреждение при закрытии окна сохранения



Рис. 13 – предупреждение при закрытии окна сохранения

Рис. 14 – окно удачного сохранения файла

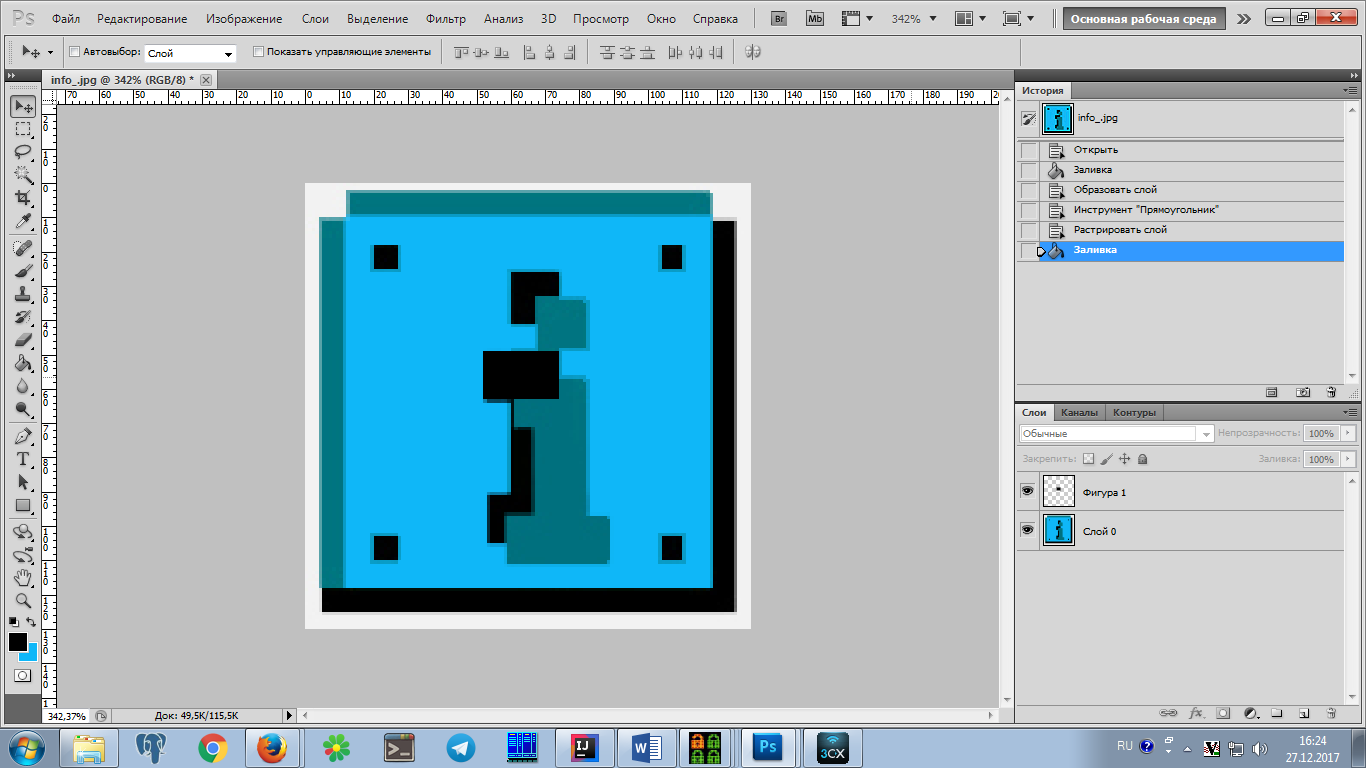
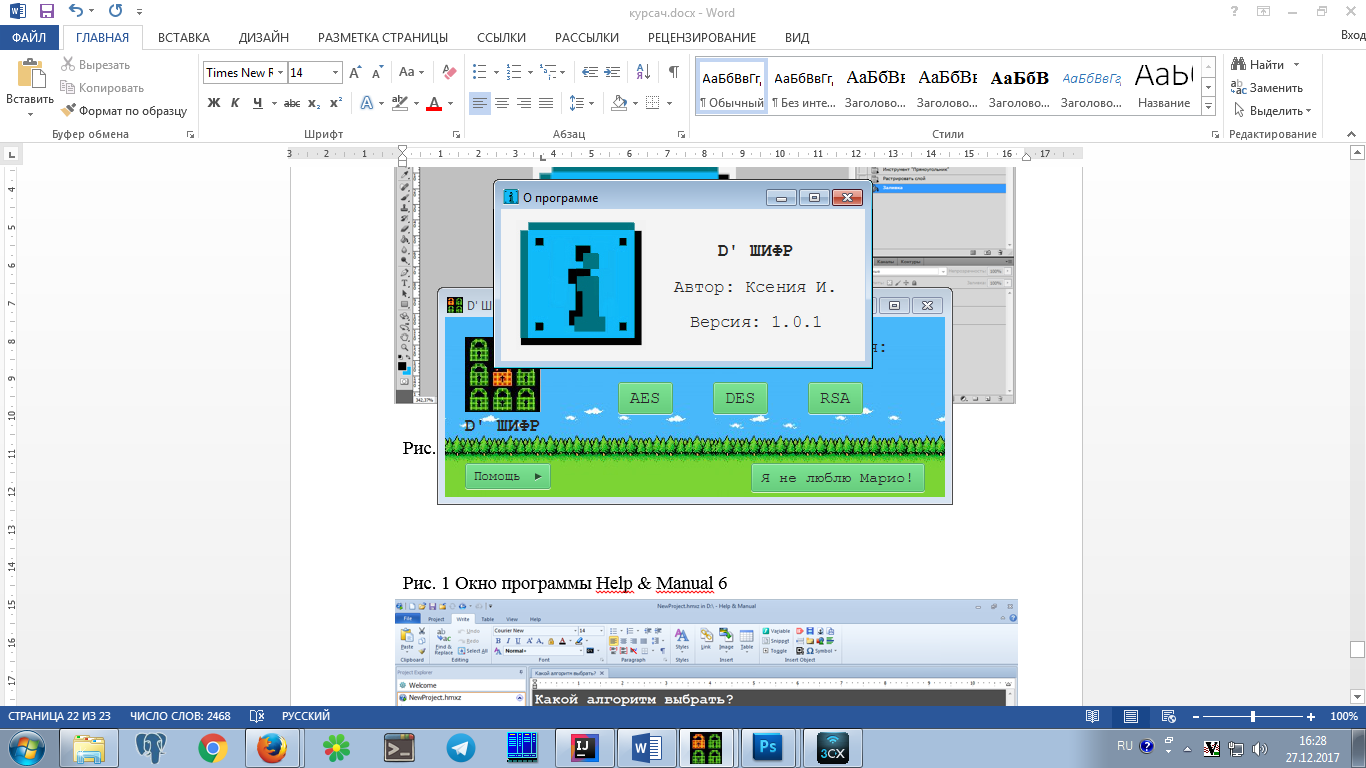
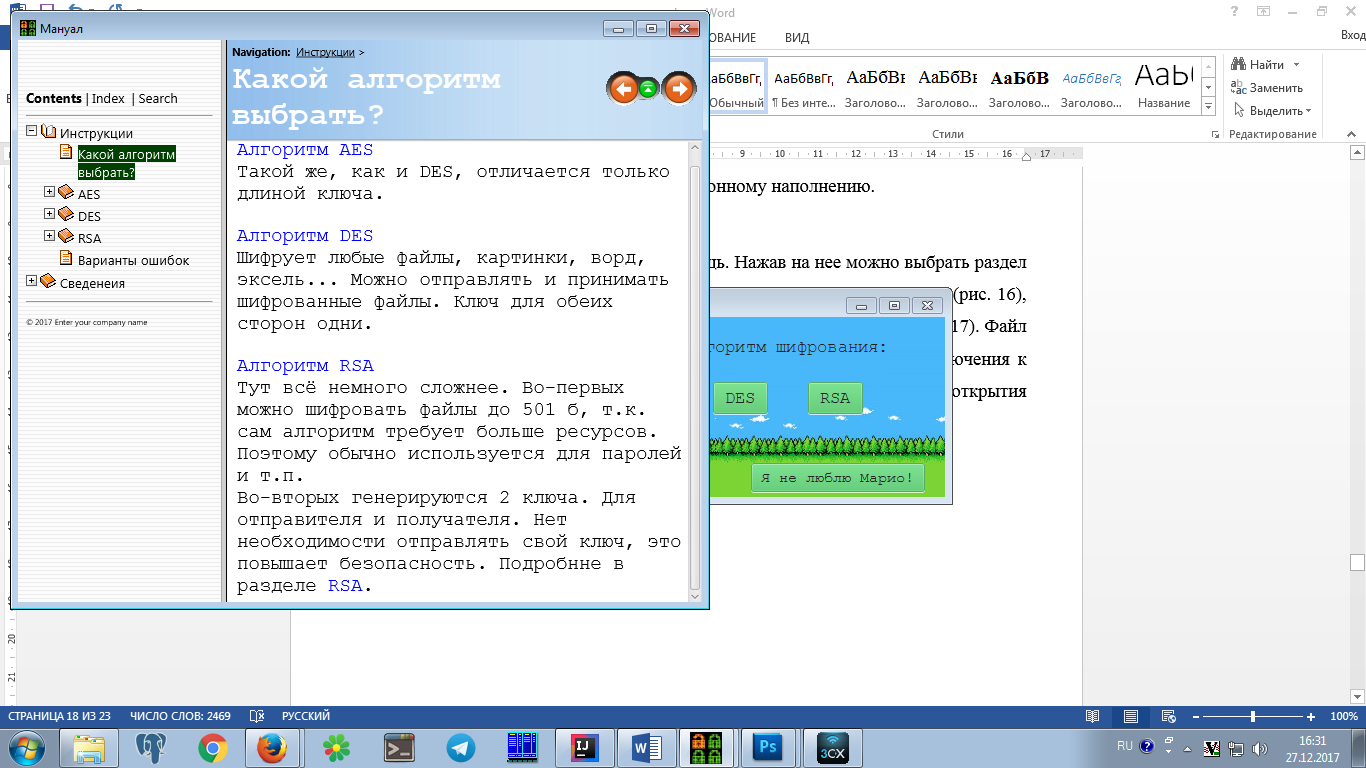
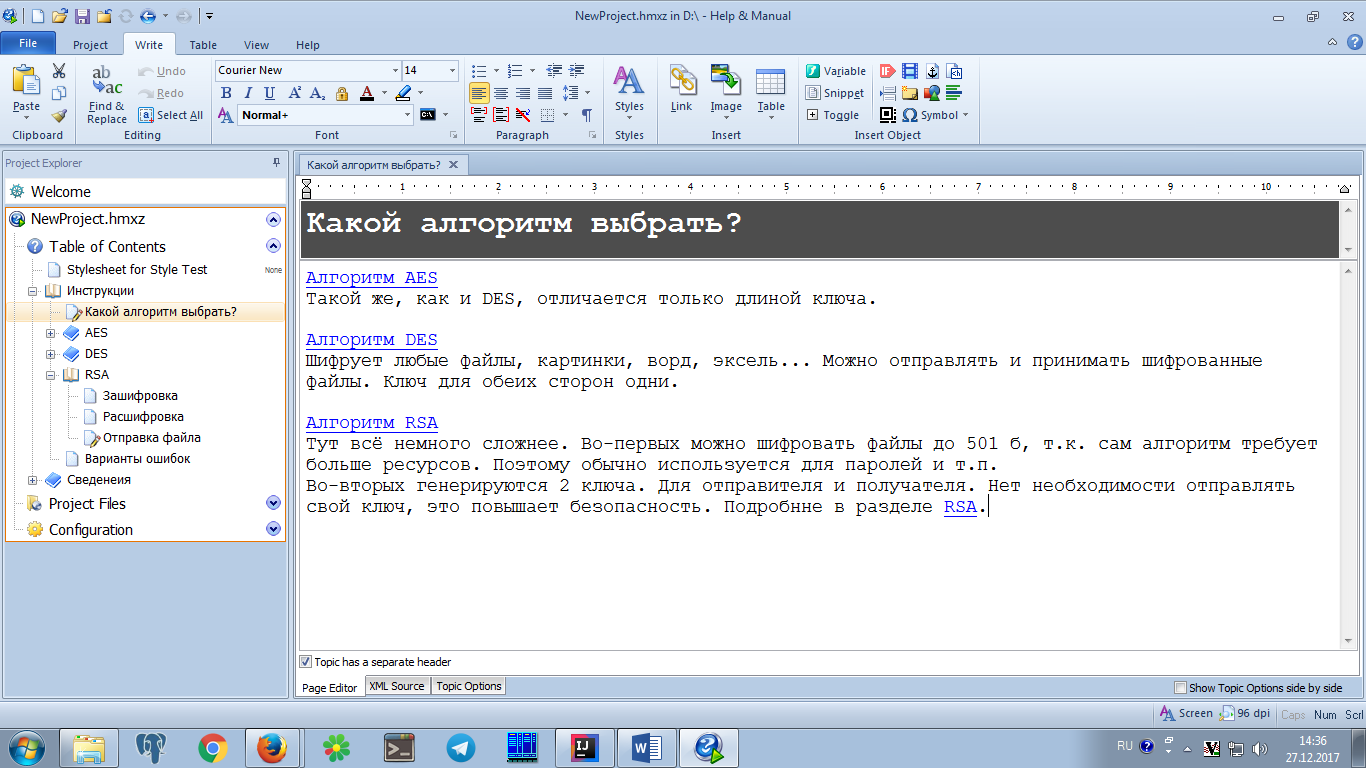
Рис. 15 – процесс создания картинки для окна «О программе»

Рис. 16 окно «О программе»



Рис. 17 окно «Мануал»

Рис. 18 Окно программы Help & Manual 6

## ДИСК DVD-R