Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Филиал

«Минский радиотехнический колледж»

РЕАЛИЗАЦИЯ АЛГОРИТМА ЦИФРОВОЙ ПОДПИСИ

(АЛГОРИТМ RSA)

Пояснительная записка

к курсовому проекту по дисциплине

« Основы алгоритмизации и программирования»

**КП 52493.000000.000 ПЗ**

Руководитель (С.А Апанасевич)

Учащийся гр. 52493 (Д. С. Ермакович)

2017**Содержание**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

КП 52493.000000.000

Разраб.

Ермакович

Провер.

Апанасевич

Реценз.

Н. Контр.

Утверд.

Реализация алгоритма цифровой подписи (алгоритм RSA)

Лит.

Листов

44

МРК

**Введение** 3

**1 Постановка** 4

**1.1 Описание предметной области** 4

**1.2 Обзор существующих решений** 6

**1.3 Входные и выходные данные** 8

**2 Проектирование задачи** 9

**2.1 Диаграмма вариантов использования** 9

**2.2 Разработка алгоритма работы программы** 10

**2.3 Разработка алгоритмов решения задачи** 12

**3 Программная реализация** 14

**3.1 Выбор и обоснование инструментов разработки** 14

**3.2 Структура программы** 16

**3.3 Описание разработанных процедур и функций** 17

**3.4 Назначение и условие применения программы** 19

**4 Системные требования** 20

**4.1 Руководство пользователя** 21

**4.2 Тестирование** 29

**Заключение** 32

**Список использованной литературы** 33

**Приложение А** 34

# **Введение**

Электронно-цифровая подпись (*ЭЦП*) — электронный аналог собственноручной подписи — используемый в системах электронного документооборота для придания электронному документу юридической силы, равной бумажному документу, подписанного собственноручной подписью правомочного лица и/или скрепленного печатью. Документ (файл), подписанный ЭЦП, гарантированно защищен от изменений — проверка подписи мгновенно выявит расхождение. ЭЦП обеспечивает проверку целостности документов, конфиденциальность, установление лица, отправившего документ. Это позволяет усовершенствовать процедуру подготовки, доставки, учета и хранения документов, гарантировать их достоверность. Главное преимущество использование ЭЦП — значительное сокращение временных и финансовых затрат на оформление и обмен документацией. Таким образом, по функциональности ЭЦП даже превосходит обычную подпись. Предположим, что две стороны (назовем их условно «А» и «Б») решили организовать между собой обмен документами на машинных носителях. Как должен выстраиваться документооборот между этими сторонами? В первую очередь, стороны должны договориться об использовании средств ЭЦП. Лучше, если это будет программный или программно-аппаратный комплекс, сертифицированный в нашей стране. После того, как средство ЭЦП выбрано, стороны должны выполнить генерирование ключей — по открытому и секретному (личному) ключу ЭЦП для каждой стороны.

Из исходного текста документа и его ЭЦП формируется электронный документ. Здесь нужно понимать, что электронный документ — это не просто файл на магнитном носителе с текстом документа, а файл, состоящий из двух частей: общей (в которой содержится текст) и особенной, содержащей все необходимые ЭЦП (рис. 3). Текст документа без ЭЦП — это не более чем обычный текст, который не имеет юридической силы. Его можно распечатать, передать по электронной почте, отредактировать, но нельзя установить его подлинность. ЭЦП без текста документа вообще представляет собой непереводимую игру букв. Восстановить документ по ЭЦП невозможно точно так же, как невозможно восстановить дворец по найденному кирпичу. ЭЦП сама по себе не имеет ни ценности, ни смысла.

(http://www.bestreferat.ru/referat-205649.html)

# **1 Постановка**

## **1.1 Описание предметной области**

Тема проекта – “Алгоритм цифровой подписи RSA”. Используя средства Borland Delphi 7 необходимо разработать программу, которая будет добавлять цифровую подпись к txt документу и создавать новый файл с результатом, а также проверять передаваемый файл с цифровой подписью, используя вводимый пользователем открытый ключ.

Суть электронной подписи заключается в том, чтобы получатель с помощью открытого ключа и цифровой подписи мог проверить, изменен ли документ кем - либо еще в процессе доставки, кроме создателя электронной подписи.

Если изменить документ, не изменяя электронную подпись, то проверка покажет, что документ либо подпись неверны для данного документа. Для подделки электронной подписи, нужно подобрать такой же секретный ключ, с помощью которого отправитель создавал электронную подпись. А далее с помощью этого секретного ключа повторно сгенерировать электронную подпись.

Поскольку подписываемые документы — переменного (и как правило достаточно большого) объёма, в схемах ЭП зачастую подпись ставится не на сам документ, а на его хэш. Для вычисления хэша используются криптографические хэш-функции, что гарантирует выявление изменений документа при проверке подписи. Хэш-функции не являются частью алгоритма ЭП, поэтому в схеме может быть использована любая надёжная хэш-функция. (https://ru.wikipedia.org/wiki/Электронная\_подпись)

## **1.2 Обзор существующих решений**

В настоящее время существует несколько схем построения цифровой подписи:

* На основе алгоритмов симметричного шифрования. Данная схема предусматривает наличие в системе третьего лица — арбитра, пользующегося доверием обеих сторон. Авторизацией документа является сам факт зашифрования его секретным ключом и передача его арбитру.
* На основе алгоритмов асимметричного шифрования. На данный момент такие схемы ЭП наиболее распространены и находят широкое применение.

Кроме этого, существуют другие разновидности цифровых подписей (групповая подпись, неоспоримая подпись, доверенная подпись), которые являются модификациями описанных выше схем. Их появление обусловлено разнообразием задач, решаемых с помощью ЭП.

*Симметричные схемы ЭП.* Они менее распространены, чем асимметричные, так как после появления концепции цифровой подписи не удалось реализовать эффективные алгоритмы подписи, основанные на известных в то время симметричных шифрах. Первыми, кто обратил внимание на возможность симметричной схемы цифровой подписи, были основоположники самого понятия ЭП Диффи и Хеллман, которые опубликовали описание алгоритма подписи одного бита с помощью блочного шифра.

Симметричные схемы основаны на хорошо изученных блочных шифрах. В связи с этим симметричные схемы имеют следующие преимущества:

* Стойкость симметричных схем ЭП вытекает из стойкости используемых блочных шифров, надежность которых также хорошо изучена.
* Если стойкость шифра окажется недостаточной, его легко можно будет заменить на более стойкий с минимальными изменениями в реализации.

Однако у симметричных ЭП есть и ряд недостатков:

* Нужно подписывать отдельно каждый бит передаваемой информации, что приводит к значительному увеличению подписи. Подпись может превосходить сообщение по размеру на два порядка.
* Сгенерированные для подписи ключи могут быть использованы только один раз, так как после подписывания раскрывается половина секретного ключа.

*Асимметричные схемы ЭП.* Они относятся к криптосистемам с открытым ключом. В отличие от асимметричных алгоритмов шифрования, в которых шифрование производится с помощью открытого ключа, а расшифровка — с помощью закрытого, в асимметричных схемах цифровой подписи подписание производится с применением закрытого ключа, а проверка подписи — с применением открытого.

Общепризнанная схема цифровой подписи охватывает три процесса:

* Генерация ключевой пары. При помощи алгоритма генерации ключа равновероятным образом из набора возможных закрытых ключей выбирается закрытый ключ, вычисляется соответствующий ему открытый ключ.
* Формирование подписи. Для заданного электронного документа с помощью закрытого ключа вычисляется подпись.
* Проверка (верификация) подписи. Для данных документа и подписи с помощью открытого ключа определяется действительность подписи.

Для того, чтобы использование цифровой подписи имело смысл, необходимо выполнение двух условий:

* Верификация подписи должна производиться открытым ключом, соответствующим именно тому закрытому ключу, который использовался при подписании.
* Без обладания закрытым ключом должно быть вычислительно сложно создать легитимную цифровую подпись.

Схемы электронной подписи могут быть одноразовыми и многоразовыми. В одноразовых схемах после проверки подлинности подписи необходимо провести замену ключей, в многоразовых схемах это делать не требуется.

Также алгоритмы ЭП делятся на детерминированные и вероятностные[[9]](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C#cite_note-.D0.A2.D0.B8.D0.BF.D1.8B_.D0.AD.D0.9F-9). Детерминированные ЭП при одинаковых входных данных вычисляют одинаковую подпись. Реализация вероятностных алгоритмов более сложна, так как требует надежный источник энтропии, но при одинаковых входных данных подписи могут быть различны, что увеличивает криптостойкость. В настоящее время многие детерминированные схемы модифицированы в вероятностные.

Перечень алгоритмов ЭП. Асимметричные схемы:

* FDH (Full Domain Hash), вероятностная схема RSA-PSS (Probabilistic Signature Scheme), схемы стандарта PKCS#1 и другие схемы, основанные на алгоритме RSA
* Схема Эль-Гамаля
* Американские стандарты электронной цифровой подписи: DSA, ECDSA (DSA на основе аппарата эллиптических кривых)
* Российские стандарты электронной цифровой подписи: ГОСТ Р 34.10-94 (в настоящее время не действует), ГОСТ Р 34.10-2001 (не рекомендован к использованию после 31 декабря 2017 года), ГОСТ Р 34.10-2012
* Евразийский союз: ГОСТ 34.310-2004[[10]](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C#cite_note-10) полностью идентичен российскому стандарту ГОСТ Р 34.10-2001
* Украинский стандарт электронной цифровой подписи ДСТУ 4145-2002
* Белорусский стандарт электронной цифровой подписи СТБ 1176.2-99 (в настоящее время не действует), СТБ 34.101.45-2013
* Схема Шнорра
* Pointcheval-Stern signature algorithm
* Вероятностная схема подписи Рабина
* Схема BLS (Boneh-Lynn-Shacham)
* Схема GMR (Goldwasser-Micali-Rivest)

(здесь все из вики)

## **1.3 Входные и выходные данные**

В качестве входных данных может поступать только файл либо файл и открытый ключ. В первом случае создается копия входного файла с записанной электронной подписью в конце. Во втором случае на вход поступает файл с уже имеющейся электронной подписью в конце и открытый ключ для проверки.

В первом случае (создание подписи) выходным данным будет являться новый файл с копией документа и электронной подписью в конце, файл с открытым ключом и файл с закрытым ключом.

Во втором случае выходных данных не будет, за исключением вывода на экран информации о подлинности файла.

# **2 Проектирование задачи**

## **2.1 Диаграмма вариантов использования**

Оператор

Открыть файл

Создание ЭЦП

Вывод результатов работы

Рисунок 2.1 – Диаграмма вариантов использования программы.

Проверка документа

Сначала пользователь в любом случае открывает файл, с которым он хочет далее взаимодействовать. Затем пользователь либо применяет создание ЭЦП для файла, с созданием файлов с ключами и парой документ – подпись, либо вводит Открытый ключ и проверяет подлинность документа. В конце цепочки действий выводится сообщение с результатом работы программы.

## **2.2 Разработка алгоритма работы программы**

Первым этапом является открытие пользователем в меню MainMenu необходимого ему функционала: «Создать подпись»(1) либо «Проверить подпись»(2). После этого на экране отобразятся кнопки в зависимости от решения:

при выборе 1 варианта, отобразится кнопка «Открыть файл», реализованная через MainMenu, а также кнопка «Создать эл. подпись»;

при выборе 2 варианта, отобразится кнопка «Открыть файл», поле для ввода Открытого ключа, а также кнопка «Проверить эл. подпись»;

Далее пользователь в любом случае открывает нужный файл (оригинал либо копия с ЭЦП), используя диалоговое окно OpenDialog. После этого в первом случае пользователь нажимает кнопку «Создать эл. подпись». Во втором случае пользователю нужно ввести открытый ключ в поле для ввода, сформированный и записанный в файл заранее, при создании ЭЦП. Далее пользователь должен нажать кнопку «Проверить эл. подпись».

В любом случае на экран выводится результат работы программы. В первом случае это сообщение об успехе и вывод пути до нового файла, а во втором это информация о подлинности документа.

## **2.3 Разработка алгоритмов решения задачи**

Алгоритм работы программы состоит из событий, которые активируются по действию пользователя.

В проект входит 1 форма – Form1 и 1 модуль – Unit1.

Форма Form1 состоит из следующих элементов:

* Кнопка Button1 – «Создать эл. подпись»;
* Кнопка Button2 – «Проверить эл. подпись»;
* Кнопка Button3 – «Открыть файл»;
* Поле для ввода Edit2 – ввод Открытого ключа;
* Метки Label1 и Label3 – отображение пути выбранного файла;
* MainMenu – для определения начального курса выполнения программы;

Кнопка Button3 напрямую связана с вызовом OpenDialog.

Содержание MainMenu:

* N2 – «Создать подпись»;
* N3 – «Проверить подпись»;
* N4 – «Выход»;

выбор N2 отображает на экране компоненты: Button1, и Button3;

выбор N3 отображает на экране компоненты: Button2, и Button3;

выбор N4 завершает выполнение программы – вызывает процедуру Close();

# **3 Программная реализация**

## **3.1 Выбор и обоснование инструментов разработки**

КП написан на языке программирования Delphi, который поддерживает объектно-ориентированные конструкции. Среда разработки – BorlandDelphi 7.

BorlandDelphi 7, выпущенная в августе 2002 года, стала стандартом для многих разработчиков Delphi.

Используя среду разработки BorlandDelphi 7 можно быстро и эффективно создавать, как простые, так и сложные программы. BorlandDelphi 7 - среда предназначена для быстрой (RAD) разработки прикладного ПО для операционных систем Windows. Благодаря уникальной совокупности простоты языка и генерации машинного кода, позволяет непосредственно, и, при желании, достаточно низкоуровневого взаимодействия с операционной системой, а также с библиотеками, написанными на C/C++. Выделение и освобождение памяти контролируется в основном пользовательским кодом, что, с одной стороны, ужесточает требования к качеству кода, а с другой — делает возможным создание сложных приложений с высокими требованиями к отзывчивости (работа в реальном времени).

Система программирования Delphi версии 7 фирмы Enterprise (Borland) предоставляет наиболее широкие возможности для программирования приложений ОС Windows.

Delphi – это продукт Borland International для быстрого создания приложений. Процесс создания интерфейса будущей программы напоминает забаву с игровым компьютерным конструктором. Поэтому RAD-среды еще называют визуальными средами разработки: какими мы видим рабочие и диалоговые окна программы при проектировании, такими они и будут, когда программа заработает.

Высокопроизводительный инструмент визуального построения приложений включает в себя настоящий компилятор кода и предоставляет средства визуального программирования, несколько похожие на те, что можно обнаружить в MicrosoftVisualBasic (она не является RAD-системой) или в других инструментах визуального проектирования. В основе Delphi лежит язык Delphi, который является расширением объектно-ориентированного языка Pascal. В Delphi также входят локальный SQL-сервер, генераторы отчетов, библиотеки визуальных компонентов, и прочее, необходимое для того, чтобы чувствовать себя совершенно уверенным при профессиональной разработке информационных систем или просто программ для Windows-среды.

Прежде всего Delphi предназначен для профессиональных разработчиков, желающих очень быстро разрабатывать приложения в архитектуре клиент-сервер. Delphi производит небольшие по размерам высокоэффективные исполняемые модули (.exe и .dll), поэтому в Delphi должны быть, прежде всего, заинтересованы те, кто разрабатывает продукты на продажу. С другой стороны, небольшие по размерам и быстро исполняемые модули означают, что требования к клиентским рабочим местам существенно снижаются – это имеет немаловажное значение и для конечных пользователей.

Преимущества Delphi по сравнению с аналогичными программными продуктами:

* Быстрота разработки приложения (RAD);
* Высокая производительность разработанного приложения;
* Низкие требования разработанного приложения к ресурсам компьютера;
* Наращиваемость за счет встраивания новых компонент и инструментов в среду Delphi;
* Возможность разработки новых компонентов и инструментов собственными средствами Delphi (существующие компоненты и инструменты доступны в исходных кодах);

– удачная проработка иерархии объектов.

Система программирования Delphi рассчитана на программирование различных приложений и предоставляет большое количество компонентов для этого. К тому же работодателей интересует, прежде всего, скорость и качество создания программ, а эти характеристики может обеспечить только среда визуального проектирования, способная взять на себя значительные объемы рутинной работы по подготовке приложений, а также согласовать деятельность группы постановщиков, кодировщиков, тестеров и технических писателей. Возможности Delphi полностью отвечают подобным требованиям и подходят для создания систем любой сложности.

## 

## **3.2 Структура программы**

Программа состоит из трех главных частей: панели Файл, панели Главная, поле ввода.

Панель Файл содержит элементы управления работы с документом. Можно создавать, открывать и сохранять файлы. Также можно производить печать файла или его экспорт.

Панель Главная содержит инструменты форматирования содержимого документа. Используя данные инструменты, пользователь может форматировать текст разными способами.

Информация, которая вводится в документ вводится через компонент RxRichEdit. Компонент RxRichEdit входит в библиотеку компонентов RxLib.

Структура программы также содержит большое количество дополнительных форм и окон. К примеру, если при нажатии на закрытие формы пользователь не сохранил документ, то будет высвечено диалоговое окно, которое предложит пользователю сохранить документ перед выходом из программы.

Используя горячие клавиши можно быстро форматировать документ. Также в документ можно вставлять объекты: формулы, таблицы, презентации рисунки и другие.

При подготовке текстовых документов на компьютере используются три Группы операций:

* Операции ввода позволяют перевести исходный текст из его внешней формы в электронный вид, т.е., в файл. Ввод может быть осуществлен как с помощью клавиатуры, так и методом сканирования.
* Операции редактирования позволяют изменить уже существующий документ путем добавления или удаления его фрагментов, перестановки частей документа, слияние нескольких файлов в один или, наоборот, разбиение единого документа.
* Операции форматирования позволяют оформить документ: отформатировать, ввести шрифтовое оформление, т.е. точно определить, как будет выглядеть текст на бумаге после печати на принтере.
* Операции сохранения и распечатки позволяют сохранить текст в папке на диске и распечатать страницы текста на принтере.

## **3.3 Описание разработанных процедур и функций**

Процедуры и функции UnitTextPoy:

functionTTextForm.SelAtrText :TRxTextAttributes;

ФункцияSelAtrTextвозвращаетRxRichEdit.SelAttributes, еслидлинавыделенноготекстанеравна нулю, иначевозвращаетRxRichEdit.WordAttributes.

procedure SetIndent(Left,Right,Top,Bottom:Integer; RichEdit:TRxRichEdit);

Процедура устанавливает поля страницы. Входными параметрами являются четыре значения поля: Left, Right, Top, Bottom, а также RxRichEdit, к которому необходимо применить размеры полей.

procedureCallAsk(REdit:TRxRichEdit);

ПроцедурапроверкиRxRichEditна изменение.

procedureTTextForm.UpdateCursorPos;

Процедура, которая вызывается по событию OnKeyDownи устанавливает положение скрола документа в положение каретки RxRichEdit.

procedureTTextForm.CheckFileSave;

Процедура проверяет сохранен ли файл.

procedure TTextForm.FileNewExecute(Sender: TObject);

Процедурасозданияновогофайла.

procedure TTextForm.PerformFileOpen(constAFileName: string);

Процедура загрузки файла при запуске программы.

procedure TTextForm.FormShow(Sender: TObject);

Процедура проверяет надо ли нам загружать файл при открытии программы.

procedure TTextForm.RulerResize(Sender: TObject);

Процедура адаптации линейки по форме.

function TTextForm.RulerToIndent(RulerPos: Integer; IsRight: Boolean): Integer;

Функция перемещения ползунка линейки к установленному отступу. Процедура принимает позицию, куда необходимо переставить ползунок, а также проверяет ползунок правого отступа ли это. При этом применяет соответствующие действия.

procedurePanelShift(Panel:TPanel; BufPosition:Boolean; LeftEnd:Integer);

Процедура сдвига панели буфера обмена. Входными параметрами является панель, переменная, которая проверяет, в какую сторону двигать панель и позиция выдвижения.

procedure TTextForm.StringGridBufMouseDown(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

Процедура нажатия на ячейку таблицы буфера обмена.

procedure TTextForm.MusicOnOffExecute(Sender: TObject);

Процедура включения и выключения звука нажатия клавиш.

ПроцедурыифункцииUnitUnitFindAndReplaceForm:

procedureEndSearch(Text:string);

Процедура выводит текст, который не был найдет и открывает диалоговое окно, которое предлагает пользователю выполнить следующие действия: продолжить поиск или замену текста, закрыть окно поиска.

procedure TFormFindAndReplaceText.ButtonContinueClick(Sender: TObject);

ПроцедуразакрываетформуUnitFindAndReplaceForm.

procedure TFormFindAndReplaceText.ButtonCloseClick(Sender: TObject);

ПроцедуразакрываетформуUnitFindAndReplaceForm и форму поиска или замены текста.

ПроцедурыифункцииUnitUnitSaveDoc:

procedure TFormSaveDoc.FormActivate(Sender: TObject);

Процедура показывает пользователю диалоговое окно, в котором предлагается выполнить следующие действия: сохранить файл, не сохранять файл, отменить и закрыть форму.

ПроцедурыифункцииUnitUnitBufText:

function GetRTFSelection( REdit: TRxRichEdit):string;

Функция, которая вызывается при копировании текста. На входе имеет RxRichEdit, и возвращает изнего выделенный текст формата RTF в переменную типа string.

procedure NextBufText(REdit:TRxRichEdit; StrGrd:TStringGrid);

Процедура, которая вызывается при копировании текста. На входе имеет RxRichEditи таблицу буфера обмена. Вызывается функция GetRTFSelection и текст помещается в динамический массив со смещением таблицы буфера обмена.

procedureClearBuf (StrGrd:TStringGrid);

Процедураочистки таблицы буфера обмена.

procedure PasteFromBuf(RichEdit:TRxRichEdit; ARow:Integer);

Процедура нажатия на ячейку таблицы и извлечение текста из ячейки динамического массива. Текст вставляется в RxRichEdit.

ПроцедурыифункцииUnitParaFmt:

functionFormatParagraph(Paragraph: TRxParaAttributes): Boolean;

Функция, которая проверяет каким образом открыто окно, а также присваивает значения RxParaAttributes.

procedure TParaFormatDlg.SetAttr(Paragraph: TRxParaAttributes);

Процедураприсвоенияустановленныхотступов.

## **3.4 Назначение и условие применения программы**

Практически в любой фирме, организации необходимо создавать текстовые документы. Процедура подготовки текстовых документов занимает достаточно много времени.

Текстовой редактор – это программа, позволяющая автоматизировать операции с текстовыми документами.

При подготовке текстовых документов на компьютере используются три Группы операций:

* Операции ввода позволяют перевести исходный текст из его внешней формы в электронный вид, т.е., в файл. Ввод может быть осуществлен как с помощью клавиатуры, так и методом сканирования.
* Операции редактирования позволяют изменить уже существующий документ путем добавления или удаления его фрагментов, перестановки частей документа, слияние нескольких файлов в один или, наоборот, разбиение единого документа.
* Операции форматирования позволяют оформить документ: отформатировать, ввести шрифтовое оформление, т.е. точно определить, как будет выглядеть текст на бумаге после печати на принтере.
* Операции сохранения и распечатки позволяют сохранить текст в папке на диске и распечатать страницы текста на принтере.

Текстовые редакторы прошли быстрый путь развития от простых программ, выполнявших несколько операций форматирования, до систем, имеющих множество разнообразных средств для работы с текстом, рисунками, таблицами, позволяющих подготовить не только любые документы учрежденческой деятельности, но и оригинал – макет печатных изданий.

# **4 Системные требования**

Таблица 4.1 – Системные требования программы “Текстовый редактор с дополнительным функционалом”

|  |  |
| --- | --- |
| Компонент | Требование |
| ПК и процессор | Процессор с тактовой частотой 500 МГц или выше; |
| Память | Не менее 256 МБ оперативной памяти; для работы с графикой и использования мгновенного поиска, |
| Жесткий диск | 10МБ свободного дискового пространства. |
| Монитор | Монитор с разрешением 1024 х 768 или больше.  Требования к видеоадаптеру: построитель текстуры 20 и вершинный построитель текстуры 2.0. Выпуск драйверов не ранее 1 ноября 2004 г. |
| Операционная система | Windows XP ивыше. |
| Дополнительно | Требования и функциональные возможности продуктов могут различаться в зависимости от конфигурации системы и используемой операционной системы. |

## **4.1 Руководство пользователя**

Практически вся текстовая информация, редактируемая в системе, редактируется в визуальном текстовом редакторе, или WYSIWYG (WhatYouSeeIsWhatYouGet) редакторе, который позволяет работать с текстом, таблицами, изображениями, гиперссылками и т.п. Интерфейс редактора во многом схож с интерфейсом MicrosoftWord, но обладает некоторыми особенностями и гораздо проще в использовании.

На рисунке 4.1 представлен общий вид редактора (в зависимости от реализации могут отсутствовать некоторые функции).

В верхней части программы располагается главная панель с двумя вкладками: Файл, Главная. На панели Главная располагаются инструменты редактирования файла. На панели Файл располагаются инструменты работы с файлом.

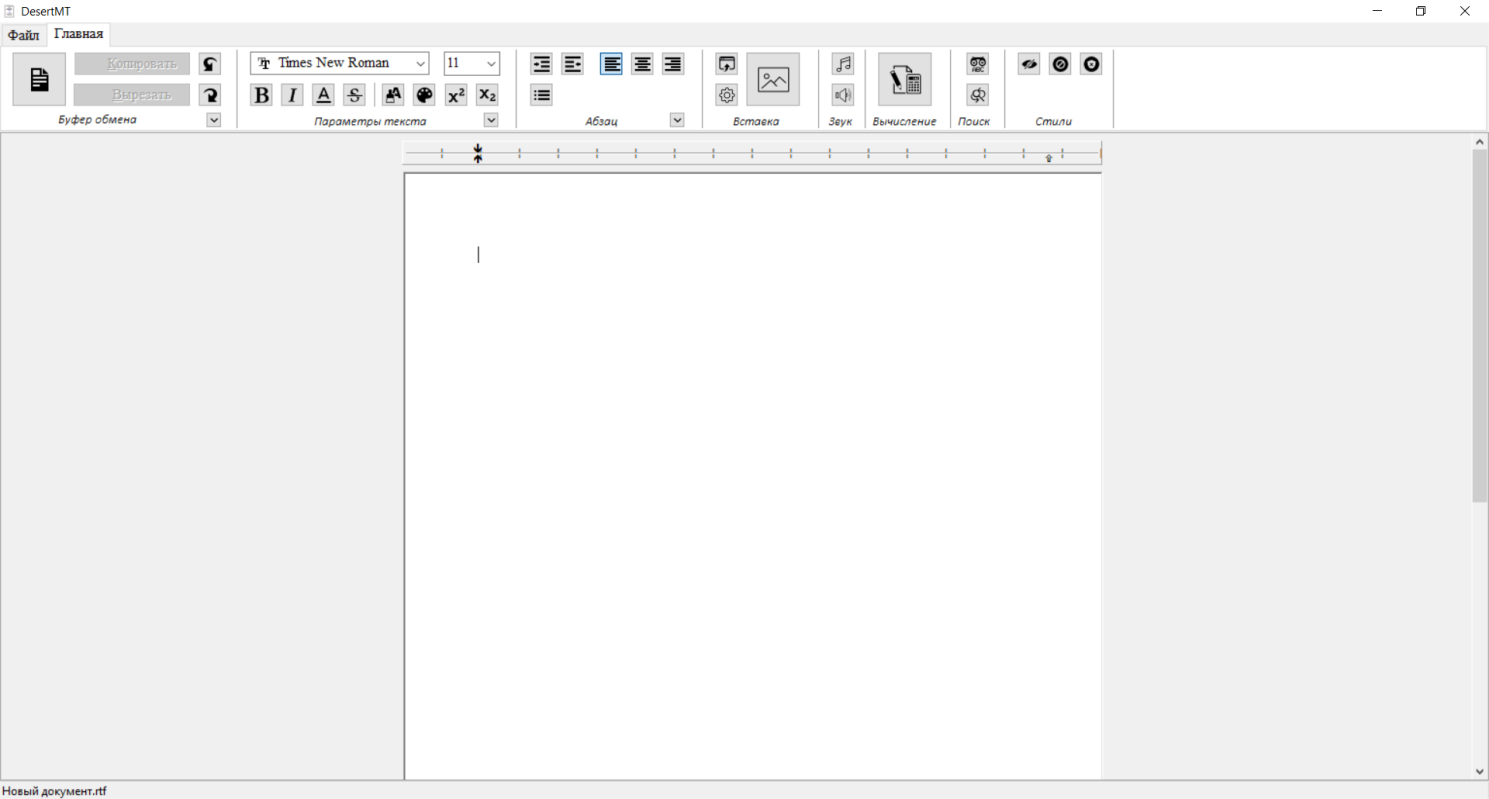


Рисунок 4.1 – Главная форма программы

В программе реализовано две панели: панель инструментов работы с файлами и панель инструментов редактирования текста.



Рисунок 4.2 – Панель Файл.

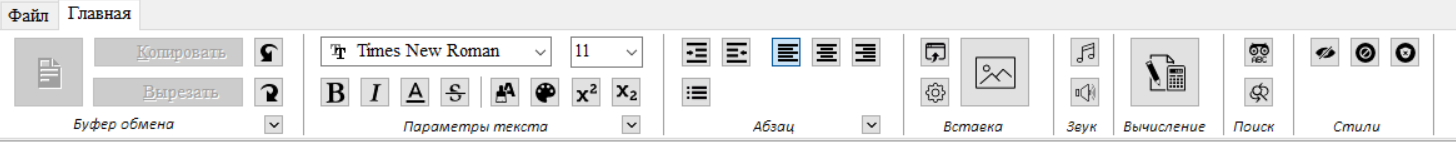


Рисунок 4.3 – Панель Главная

В редакторе, так же, как и везде в Windows-приложениях, работают горячие клавиши работы с буфером обмена клавиатуры, т.е. <Shift+Insert>, <Shift-Delete>, <Ctrl+Insert> или <Ctrl+V>, <Ctrl+X>, <Ctrl+C>. При открытии буфера обмена в левой части программы появится панель, на которой располагается таблица скопированных пользователем данных.

Для удаления части текста необходимо выделить его и нажать на клавишу **<**Del**>** или **<**BackSpace**>**.

Действие Отменить**.** Данная кнопка отменяет последнее выполненное действие. Нажатие на пиктограмму соответствует комбинации клавиш Ctrl+Z.

Действие Вернуть**.** Нажатие на кнопку приводит к возврату действия, отменённого предыдущей операцией. Нажатие на пиктограмму отмены соответствует комбинации клавиш Ctrl+Y.

Таблица 4.2 – Назначение кнопок в разделе Буфер обмена.

|  |  |
| --- | --- |
| Картинка | Назначение |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\icon_104.png | Вставить текст. |
|  | Копировать текст |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\cut.png | Вырезать текст |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1487186768_170_CurvedArrowRightDown.png | Отмена. |
|  | Возврат. |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1488324148_ic_keyboard_arrow_down_48px.png | Открытие/закрытие Буфера обмена. |

Для открытия Буфера обмена необходимо нажать на сноску в разделе Буфер обмена.

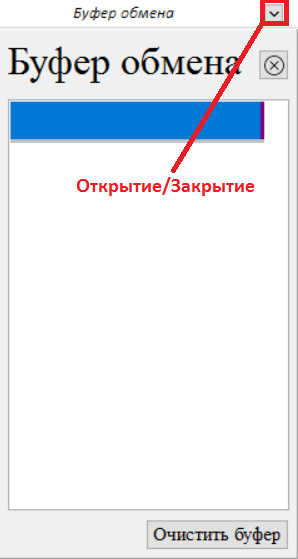


Рисунок 4.2 – Панель Буфера обмена.

Полужирный шрифт.Нажатие на кнопку изменяет шрифт выделенного фрагмента текста на полужирный. Нажатие на пиктограмму соответствует комбинации клавиш Ctrl+B.

Курсив.Нажатие на кнопку изменяет шрифт выделенного фрагмента текста на курсив. Нажатие на пиктограмму соответствует комбинации клавиш Ctrl+I.

Подчёркнутый.Нажатие на кнопку изменяет шрифт выделенного фрагмента текста на подчёркнутый. Нажатие на пиктограмму соответствует комбинации клавиш Ctrl+U.

Изменение цвета текста и цвета фона текста. Применяет цвет текста/фона к выделенному тексту.

Таблица 4.3 - Назначение кнопок в разделе Параметры текста.

|  |  |
| --- | --- |
| Картинка | Назначение |
|  | Текст жирного стиля. |
|  | Наклонный текст. |
|  | Подчеркнутый текст. |
|  | Зачеркнутый текст. |
|  | Цвет текста. |
|  | Цвет фона текста. |
|  | Верхний индекс. |
|  | Нижний индекс. |
|  | Открытие/закрытие панели Параметров текста. |

Выравнивание текста.С помощью кнопок текст выравнивается соответственно: по левому краю /по правому краю /по центру текстового редактора.

Список.После нажатия на выделенные абзацы объединятся в ненумерованный список.

Отступ.Кнопка увеличивает отступ. Для этого надо поставить курсор на нужную строку и нажать мышью на пиктограмму. Чтобы уменьшить отступ надо нажать на пиктограмму.

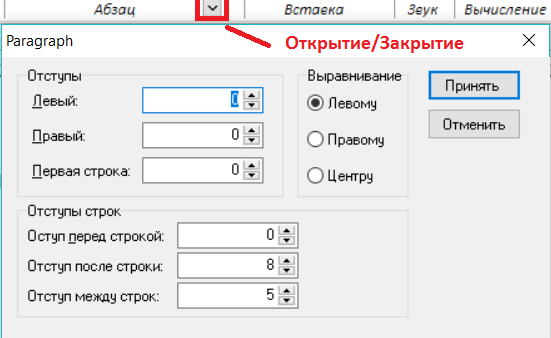


Рисунок 4.3 – Панель параграф.

Таблица 4.3 - Назначение кнопок в разделе Абзац/Отступы.

|  |  |
| --- | --- |
| Картинка | Назначение |
|  | Увеличить отступ слева. |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\left-indent.png | Уменьшить отступ слева. |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\left-side-alignment.png | Выравнивание по левому боку. |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\center-paragraph.png | Выравнивание по центру. |
|  | Выравнивание по правому боку. |
|  | Установка списка. |
|  | Открытие/закрытие панели Параметры отступов. |

Вставка объекта. По нажатию на кнопку откроется форма выбора объекта. Присутствует вставка программ MSOffice, Paint, Adobe и другие.

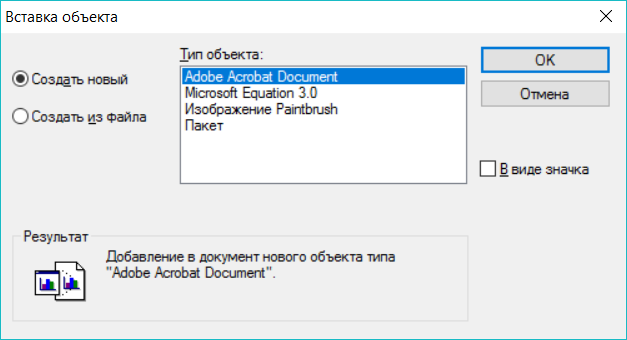


Рисунок 4.4 – Панель Объект

Вставка картинки. По нажатию. На кнопку откроется проводник, в котором необходимо выбрать картинку формата .png, .jpg, .bmpи другие.

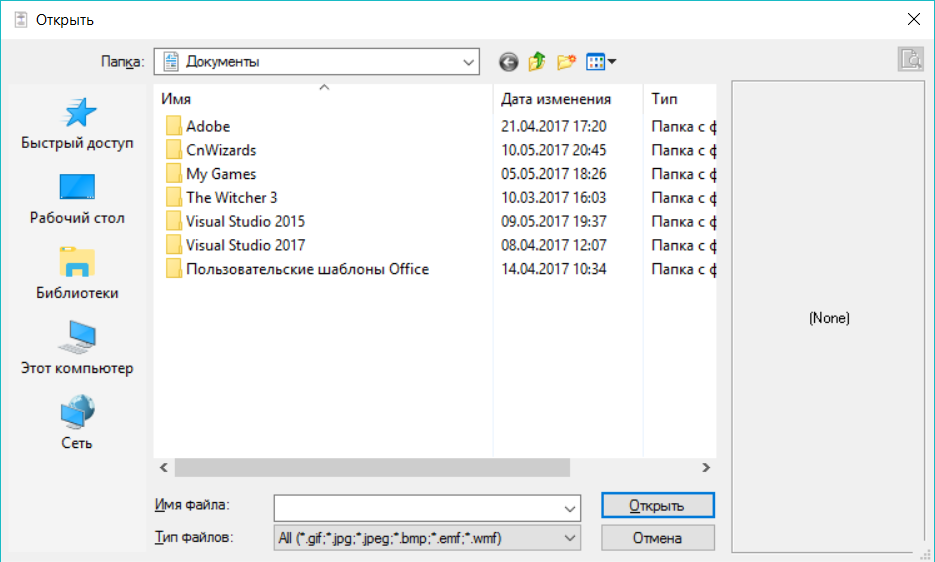


Рисунок 4.5 – Вставка картинки.

Параметры объекта. Параметры объекта применяются к уже вставленному в документ объекту. Пользователю необходимо выделить нужный ему объект и нажать кнопку Параметры объекта. В параметрах объекта можно поменять то, как будет отображаться объект в документе: он может быть загружен в документ, а также просто иметь ссылку на объект.

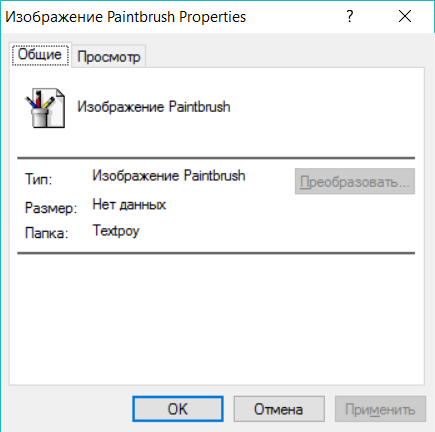


Рисунок 4.6 – Параметры объекта.

Таблица 4.4 - Назначение кнопок в разделе Вставка.

|  |  |
| --- | --- |
| Картинка | Назначение |
|  | Вставить объект. |
|  | Вставить картинку. |
|  | Параметры объекта. |

Выбрать звук. По нажатию на кнопку откроется проводник, в котором пользователь выберет звуковой файл.

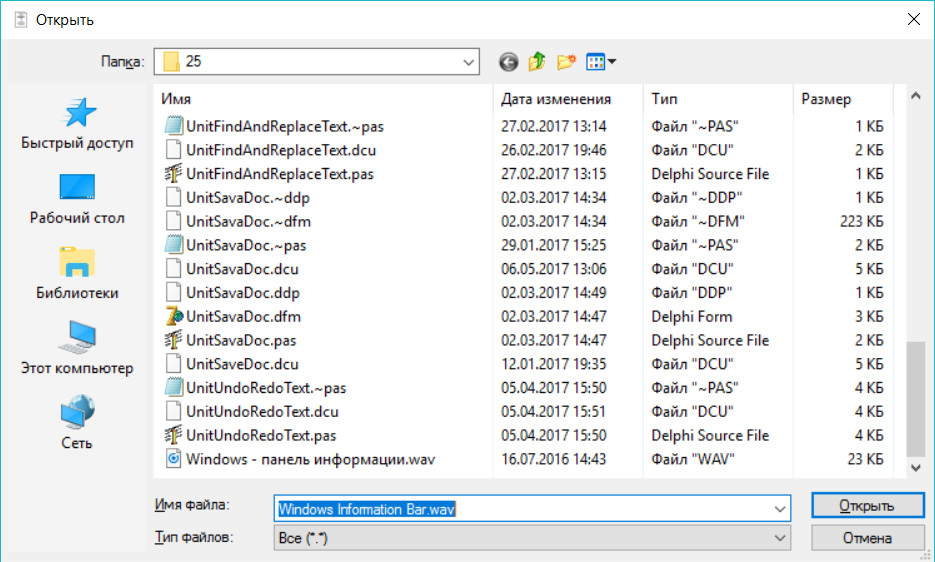


Рисунок 4.7 – Выбор звукового файла.

Калькулятор. По нажатию на кнопку откроется форма калькулятора, в котором пользователь может выполнять различные подсчеты.

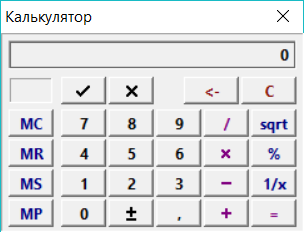


Рисунок 4.8 - Калькулятор

Найти текст. По нажатию на кнопку происходит открытие формы поиска текста. Пользователь вводит в текстовое поле текст, который необходимо найти.

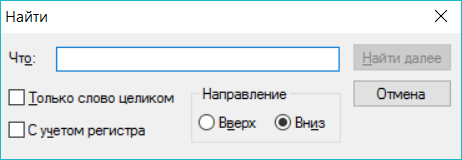


Рисунок 4.9 – Форма нахождения текста.

Заменить текст. По нажатию на кнопку происходит открытие формы поиска и замены текста. Пользователь вводит в текстовое поле текст, который необходимо найти и заменить. Текст можно заменять пошагово, но также можно заменить текст нажатием на кнопку “Заменить все”.

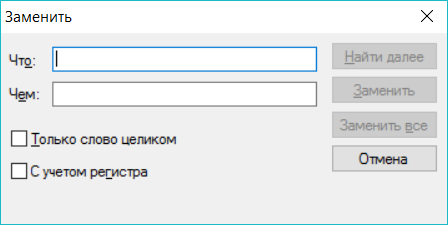


Рисунок 4.10 – Форма замены текста

Скрытый текст. По нажатию на кнопку выделенный текст исчезнет, но действие можно отменить сочетанием клавиш Ctrl + Z. Скрытый текст можно выделить в и еще раз применить к нему данный стиль текста.

Таблица 4.5 - Назначение кнопок в разделе Звук, Вычисления, Поиск, Стили.

|  |  |
| --- | --- |
| Картинка | Назначение |
|  | Выбрать звук. |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\speaker-high-volume.png | Вкл/Выкл звук. |
|  | Калькулятор. |
|  | Найти текст. |
|  | Замена текста. |
|  | Скрытый текст. |
|  | Стиль выключенного текста. |
|  | Заблокированный текст. |

Новый документ. Создает новый документ в окне программы.

Открыть документ. Открывает проводник, в котором пользователь выбирает, какой файл необходимо открыть.

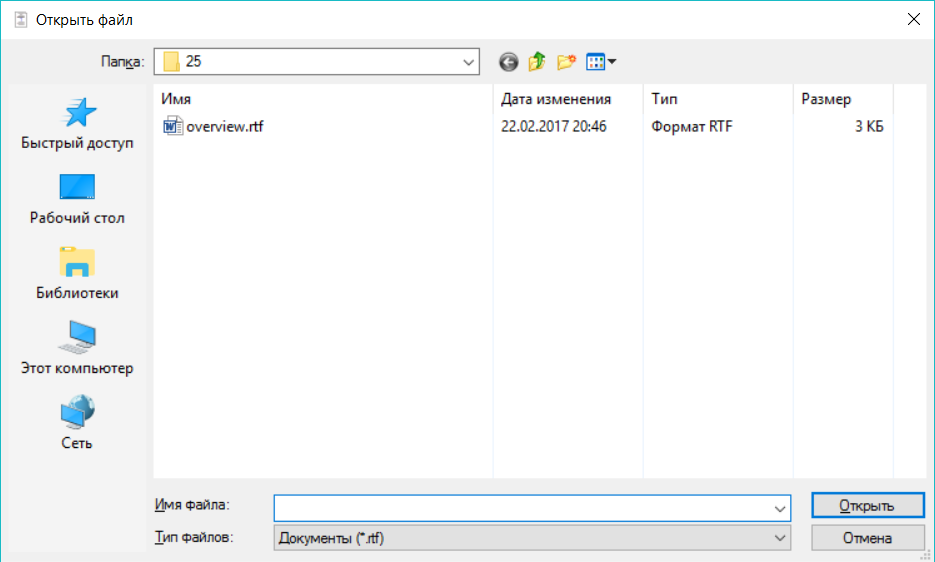


Рисунок 4.11 – Открытие файла

Сохранить документ. Открывает проводник, если документа нет на диске, если документ есть, то он перезапишется.

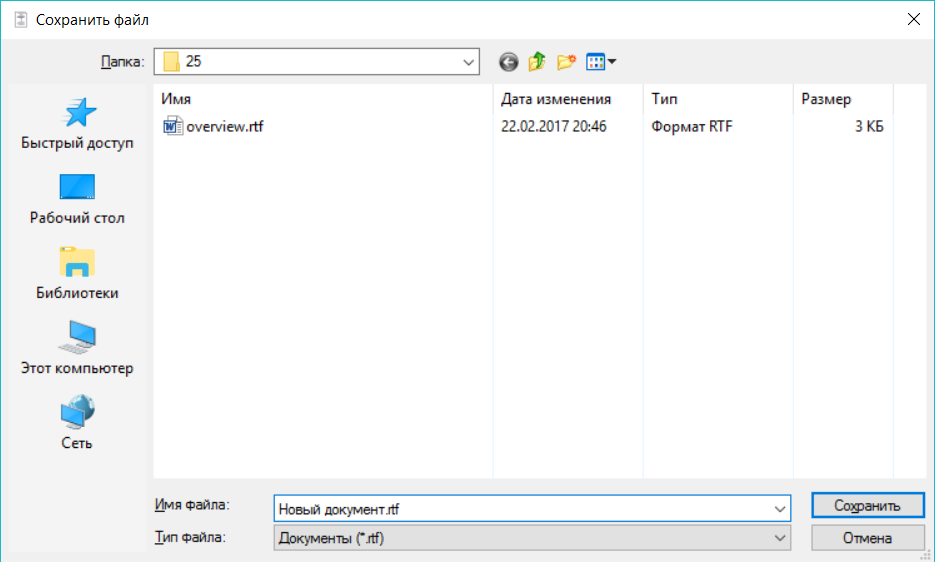


Рисунок 4.12 – Сохранение файла.

Печать документа. Открывает диалоговое окно для печати/экспорта документа.

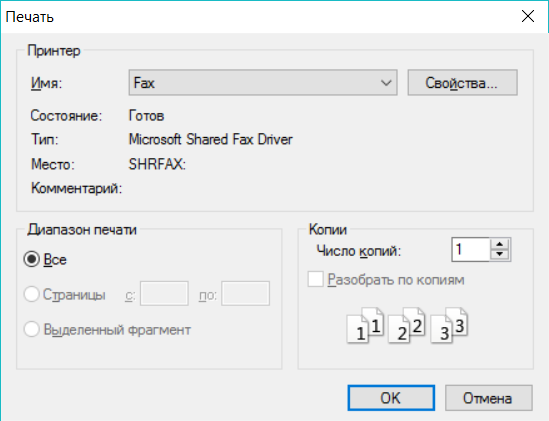


Рисунок 4.13 – Печать файла.

Таблица 4.6 - Назначение кнопок на панели Файл.

|  |  |
| --- | --- |
| Картинка | Назначение |
|  | Новый документ. |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\icon_092.png | Открыть документ. |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\icon_093.png | Сохранить документ. |
|  | Сохранить документ как. |
|  | Печать документа. |
|  | Справка. |
| C:\Users\Lenovo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\exit.png | Выход. |

Также есть форма справки. При нажатии на кнопку Справка открывается форма со сведениями ОС и версией программы.

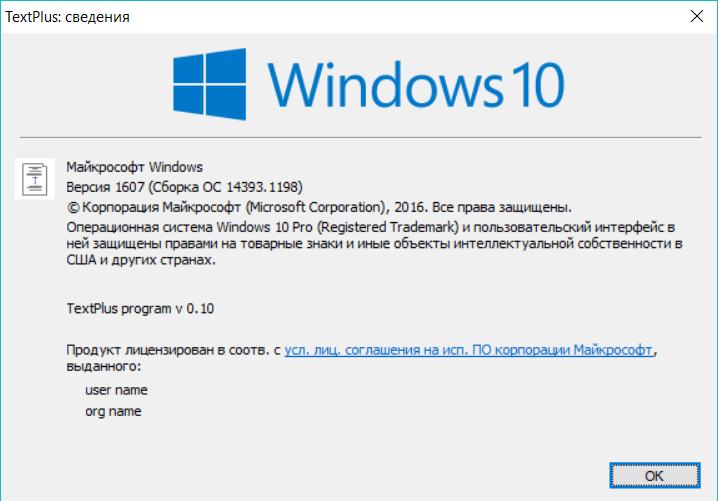


Рисунок 4.14 – Справка о программе.

## **4.2 Тестирование**

В процессе разработки программы возникали следующие ошибки: поиск и замена текста, реализация отмены и возврата.

Поиск и замена текста.

В реализации следующей ошибки приведу код, разработанный в процессе реализации. В дальнейшем этот код не понадобился, так как в компоненте RxRichEditэтот процесс реализован автоматически, ниже будет приведен пример работы и вызова.

Следующий код представляет решение до использования компонента RxRichEdit.

//методпоискатекста

procedure FindTextRichEdit(varPosCursor,ReplaceNumber:Cardinal; RichEdit: TRichEdit; FindDialog:TFindDialog);

varIsWholeWord:Boolean; CursorStart:Cardinal;

begin

with FindDialog do

begin

with RichEdit do

begin

//проверяем свойство Заглавных букв

//Copy(Lines.Text, PosCursor + 1,Length(Lines.Text)) берем только тот текст который за курсором

if frMatchCase in Options then

CursorStart:=Pos(FindDialog.FindText,Copy(Text, PosCursor + 1,Length(Text)))

else

CursorStart:=Pos(AnsiLowerCase(FindDialog.FindText), AnsiLowerCase(Copy(Text, PosCursor + 1,Length(Text))));

if CursorStart<>0 then

begin

SelStart:=CursorStart + PosCursor - 1;

//проверяем стоит ли опция слово целиком

if (frWholeWord in Options) then

begin

//делаем проверку слова от 0 до n+1 символов слов

IsWholeWord:= WholeWord(Copy(Text,SelStart,Length(FindDialog.FindText)+2),SelStart);

if not IsWholeWord then

begin

PosCursor:=SelStart+Length(FindDialog.FindText);

//рекурсия

FindTextRichEdit(PosCursor,ReplaceNumber,RichEdit,FindDialog);

end;

end;

//проверяемпозициюкурсора

if (SelStart>=PosCursor) then

begin

//выделяемтекст

SelLength := Length(FindDialog.FindText);

//запоминаем позицию курсора для следующего поиска

PosCursor := SelStart + SelLength;

Inc(ReplaceNumber);

end;

end

else

EndSearch(FindDialog,ReplaceNumber);

end

end;

//переводим фокус на ричэдит

RichEdit.SetFocus;

end;

Данный алгоритм реализует рекурсивный поиск текста по всем параметрам диалогового окна FindDialog.

Но решение в компоненте RxRichEdit более компактнее, так как все уже реализовано, то необходимо просто открыть форму. Для поиска и для замены текста это выглядит следующим образом:

//поисктекста

Dialog := RichEdit.FindDialog(RichEdit.SelText);

//заменатекста

Dialog := RichEdit.ReplaceDialog(RichEdit.SelText,'');

Реализация отмены и возврата.

В реализации отмены и возврата был использован динамический массив, последовательно в него заносились данные, которые были удалены. Однако в компоненте RxRichEditэти функции уже реализованы автоматически.

Следующий код представляет решение до использования компонента RxRichEdit.

// удалениепервойотмены

procedure ReplaceItem(varArrayU\_R:DinArray);

vari:Byte;

begin

for i:=1 to (Length(ArrayU\_R)-1) do

ArrayU\_R[i-1]:=ArrayU\_R[i];

end;

// Извлечение RTFтекста из RichEdit

function GetRtfText(RichEdit : TRichEdit) : String;

varStringS :TStringStream;

begin

StringS := TStringStream.Create('');

RichEdit.Lines.SaveToStream(StringS);

Result := StringS.DataString;

FreeAndNil(StringS);

end;

//добавлениеВозврата

procedure AddRedo(varRedo:DinArray; RichEdit:TRichEdit);

begin

SetLength(Redo,Length(Redo)+1);

Redo[Length(Redo)-1].Text:=GetRtfText(RichEdit);

end;

//добавлениеОтмены

procedure AddUndo(RichEdit:TRichEdit; Act:ActionText);

begin

if Length(UndoArray)<50 then

SetLength(UndoArray,Length(UndoArray)+1)

else

ReplaceItem(UndoArray);

UndoArray[Length(UndoArray)-1].Text:=GetRtfText(RichEdit);

UndoArray[Length(UndoArray)-1].PosText:=RichEdit.SelStart + 1;

UndoArray[Length(UndoArray)-1].Act:=Act;

end;

//Вставкаотмены

procedure UndoText(RichEdit:TRichEdit);

begin

if Length(UndoArray)=0 then

Exit;

if Length(UndoArray)<>1 then

AddRedo(RedoArray,RichEdit);

RichEdit.Text:=UndoArray[Length(UndoArray)-1].Text;

RichEdit.SelStart:=UndoArray[Length(UndoArray)-1].PosText;

if Length(UndoArray)<>1 then

SetLength(UndoArray,Length(UndoArray)-1);

end;

//вставкаВозрата

procedure RedoText(RichEdit:TRichEdit);

begin

if Length(RedoArray)=0 then

Exit;

AddUndo(RichEdit,RedoArray[Length(RedoArray)-1].Act);

RichEdit.Text:=RedoArray[Length(RedoArray)-1].Text;

RichEdit.SelStart:=UndoArray[Length(RedoArray)-1].PosText;

SetLength(RedoArray,Length(RedoArray)-1);

end;

После использования компонента RxRichEditнеобходимо было только переопределить метод отмены и возврата на кнопки (вызвать их).

//возврат

RichEdit.Redo;

RichEditSelectionChange(Self);

//заносимвбуферкопирование

RichEdit.CopyToClipboard;

//записьбуффер

NextBufText(RichEdit,StringGridBuf);

# **Заключение**

В ходе выполнения курсовой работы предоставленная программа была выполнена и работала. Конечно это простой текстовый редактор и у него мало функций по сравнению с MS WORD. Он чем-то похож на простой блокнот.

Единственное его отличие что он был написан на языке Delphi. Язык Delphi работал без наладок, был прост в обращении, возникающие ошибке при программировании легко исправлялись. Delphi зарекомендовал себя с лучшей стороны, и я бы порекомендовал его как для начинающих программистов, так и для профессионалов.

В последнее время компьютерные технологии продвигаются очень интенсивно, иэто способствует бурному развитию программного обеспечения. Каждые полгода выходят продукты с множеством нововведений. Так и текстовые редакторы не стоят на месте. С каждым разом все больше и больше функций заключают в себе данные программы. Но их развитие поставлено таким образом, что с каждой новой версией программа сохраняет предыдущий набор возможностей и пользователь может использовать как старые, так и новые функции, последние введены лишь для облегчения работы с программой.

В современных условиях все больше в нашу жизнь входят электронные средства документооборота. Программы, для работы с такими документами тоже быстро развиваются. Появляются все новые мощные и удобные программы, которые могут использоваться как дома, так и на работе.

# **Список использованной литературы**

1. Delphi в примерах, играх и программах. Дополнительные материалы – В.Рубанцев
2. Delphi. Готовые алгоритмы (Стивенс)
3. Delphi. Программирование для Windows, OS X, iOS и Android (Осипов Д.)
4. Delphi. Трюки и эффекты (Чиртик и до)
5. Кнут.Исскуство программирования.т.1
6. Кнут.Исскуство программирования.т.2
7. Кнут.Исскуство программирования.т.3

# **Приложение А**

unit TextPoy;