Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

кафедра Информатики

Дисциплина: Программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

ТЕКСТОВЫЙ РЕДАКТОР (ТИПА WORDPAD)

БГУИР КП 1-53 01 07 10724 ПЗ

Студент: гр. 653505 Шимановская П.Н.

Руководитель: ассистент кафедры

Информатики Летохо А.С.

Минск 2017

**Содержание**

[**Введение** 2](#_Toc483346428)

[**Основная часть** 6](#_Toc483346429)

[1. Постановка задачи 6](#_Toc483346430)

[2. Требования к программе текстового редактора 7](#_Toc483346431)

[3. Разработка программы 9](#_Toc483346432)

[4. Принцип работы программы 16](#_Toc483346433)

[5. Пример работы с текстовым редактором 19](#_Toc483346434)

[**Заключение** 21](#_Toc483346435)

[**Список используемой литературы** 22](#_Toc483346436)

[**Приложение 1. Исходный код программы** 23](#_Toc483346437)

# **Введение**

Текстовый редактор — самостоятельная [компьютерная программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) или компонент программного комплекса (например, [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) [интегрированной среды разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) или окно ввода в браузере), предназначенная для создания и изменения [текстовых данных](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5) в общем и [текстовых файлов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) в частности.

Текстовые редакторы предназначены для работы с текстовыми файлами в интерактивном режиме. Текстовые редакторы могут обеспечивать выполнение разнообразных функций, а именно:

* редактирование строк текста;
* возможность использования различных шрифтов символов;
* копирование и перенос части текста с одного места на другое или из одного документа в другой;
* контекстный поиск и замена частей текста;
* задание произвольных межстрочных промежутков;
* автоматический перенос слов на новую строку;
* автоматическая нумерацию страниц;
* обработка и нумерация сносок;
* выравнивание краев абзаца;
* создание таблиц и построение диаграмм;
* проверка правописания слов и подбор синонимов;
* построение оглавлений и предметных указателей;
* распечатка подготовленного текста на принтере в нужном числе экземпляров и т.п.

Возможности текстовых редакторов различны — от программ, предназначенных для подготовки небольших документов простой структуры, до программ для набора, оформления и полной подготовки к типографскому изданию книг и журналов (издательские системы).

Наиболее известный текстовый редактор — Microsoft Word.

Полнофункциональные издательские системы — Microsoft Publisher, Corel Ventura и Adobe PageMaker. Издательские системы незаменимы для компьютерной верстки и графики. Значительно облегчают работу с многостраничными документами, имеют возможности автоматической разбивки текста на страницы, расстановки номеров страниц, создания заголовков и т.д. Создание макетов любых изданий — от рекламных листков до многостраничных книг и журналов — становится очень простым, даже для новичков.

Основным примером интерфейса разрабатываемого приложения были взяты текстовые редакторы Блокнот и WordPad.

Блокнот ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Notepad) — простой [текстовый редактор](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80), являющийся частью [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows), начиная с вышедшей в [1985 году](https://ru.wikipedia.org/wiki/1985_%D0%B3%D0%BE%D0%B4) [Windows 1.0](https://ru.wikipedia.org/wiki/Windows_1.0).

Блокнот использует [оконный класс](https://ru.wikipedia.org/wiki/Windows_API) EDIT. Вплоть до вышедшей в [2000 году](https://ru.wikipedia.org/wiki/2000_%D0%B3%D0%BE%D0%B4) [Windows ME](https://ru.wikipedia.org/wiki/Windows_ME) поддерживались только самые базовые функции, многие функции были доступны только из меню, а максимальный размер [файла](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB) составлял 64 [килобайта](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BB%D0%BE%D0%B1%D0%B0%D0%B9%D1%82) (предел класса EDIT). В настоящее время редактор поддерживает контекстную замену, [горячие клавиши](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D1%80%D1%8F%D1%87%D0%B0%D1%8F_%D0%BA%D0%BB%D0%B0%D0%B2%D0%B8%D1%88%D0%B0) (например, Ctrl+S для сохранения файла), снят предел в 64 Кбайт и добавлена поддержка [Юникода](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4).

Блокнот не способен корректно работать с файлами в [текстовом формате Unix](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82_Unix), где символом переноса является байт с кодом 10, в отличие от Windows и DOS, где используются байты 13, 10.

Как и Блокнот, WordPad присутствует во всех версиях Windows, начиная с Windows 95. Программа WordPad имеет более широкие возможности, чем Блокнот, но в то же время значительно уступает многофункциональным пакетам обработки текста, подобным Microsoft Word. WordPad обладает большим набором инструментов, чем Блокнот (Notepad), и предназначен для подготовки документов с простым форматированием.

WordPad представляет собой эволюционировавшую версию программы [Windows Write](https://ru.wikipedia.org/wiki/Windows_Write) из [Windows 1.0](https://ru.wikipedia.org/wiki/Microsoft_Windows_1.0). Поддерживает форматирование и печать текста, но не имеет ряда таких важных инструментов, как таблицы, и средств проверки орфографии.

Предшественник WordPad, редактор Write, сохранял файлы в собственном формате \*[.WRI](https://ru.wikipedia.org/wiki/Windows_Write). Ранние версии WordPad также позволяли открывать файлы в этом формате, позже поддержка \*.WRI была удалена.

Собственного формата файлов WordPad не имеет. Фактически основным форматом, используемым этим редактором, является формат \*.[RTF](https://ru.wikipedia.org/wiki/Rich_Text_Format). Кроме того, вплоть до Windows XP (включительно) WordPad поддерживал также формат [\*.DOC](https://ru.wikipedia.org/wiki/.doc) (Word 6.0 — 2003), однако лишь в той степени, в какой позволяли возможности этого редактора. В версии для [Windows XP](https://ru.wikipedia.org/wiki/Windows_XP) возможность сохранения файлов в формате \*.DOC отсутствует (только [\*.TXT](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) и \*.RTF).

Позиция Файл (работа с файлами) главного меню имеет подменю со следующими командами:

* Создать (Ctrl+N) - очистка окна редактирования и подготовка к вводу нового текста. При этом его файл получает имя Документ.
* Открыть (Ctrl+O) - открытие и загрузка файла с его поиском в диалоговом окне.
* Сохранить (Ctrl+S) - запись файла на диск с текущим именем в текущую директорию.
* Сохранить как - запись файла на диск с заданным именем и заданной директорией (они задаются в появившемся диалоговом окне).
* Печать (Ctrl+P) - печать документа с установкой опций печати (выбор принтера и др.) в появившемся диалоговом окне.
* Предварительный просмотр - просмотр страницы перед печатью (полезно для оценки расположения абзацев и встроенных рисунков).
* Параметры страницы - установка параметров страницы, который позволяет задать размер и ориентацию страницы, величину полей.
* Отправить по электронной почте - установка связи с электронной почтой для получения по ней документа или пересылки редактируемого документа по электронной почте.
* О программе – содержит информацию о названии, версии и разработчике программы.
* Выход - завершение работы с редактором.

Имеется панель быстрого доступа, разделённая на несколько подменю.

Буфер обмена:

* Отменить (Ctrl+Z) - отмена последней операции редактирования.
* Вернуть (Ctrl+Y) – повторное выполнение последнего действия.
* Вырезать (Ctrl+X) - перенос выделенного фрагмента документа в буфер обмена. При этом данный фрагмент исключается из документа.
* Копировать (Ctrl+C) - копирование выделенного фрагмента в буфер обмена с сохранением этого фрагмента в документе.
* Вставить (Ctrl+V) - перенос фрагмента документа из буфера обмена в текст редактируемого документа с помещением фрагмента по месту расположения курсора.
* Специальная вставка (Alt+Ctrl+V) - перенос фрагмента документа из буфера с установлением специальной связи (используется при реализации механизма OLE).

Правка:

* Выделить все (Ctrl+A) - выделение всего текста (например, это удобно при его переформатировании).
* Поиск (Ctrl+F) - поиск заданной строки символов (задается в диалоговом окне). В случае успеха поиска курсор устанавливается на обнаруженной строке.
* Замена (Ctrl+H) - замена заданной строки другой строкой.

Вставка:

* Дата и время - вставка по месту расположения курсора строки с указанием текущего времени и даты.
* Объект - открыть диалоговое окно Вставка объекта для включения в документ объекта, созданного другим приложением (рисунка, проигрываемого звукового файла и т.д.).
* Рисунок Paint (Ctrl+D) – позволяет вставить рисунок по месту расположения курсора, созданный в приложении Microsoft Paint.
* Изображения – вставка изображения из файла.

Абзац содержит кнопки:

* Маркер - форматирование текста в виде маркированного списка за счет добавления в начало строки маркера.
* Выравнивание – по левому краю (Ctrl+L), по центру (Ctrl+E), по правому краю (Ctrl+R) и по ширине (Ctrl+J).
* Уменьшить/Увеличить отступ – уменьшает/увеличивает отступ абзаца.
* Междустрочный интервал – изменение междустрочного интервала. Добавление или удаление интервалов после абзаца.
* Абзац - вызов диалогового окна с опциями форматирования параграфа (величина отступа слева, справа, отступа первой строки с равнением по левому краю, по центру, по правому краю или по ширине).

Шрифт:

* Позволяет выбрать и установить шрифт, стиль, цвет и размер текста.

Меню Вид порождает подменю со следующими позициями:

* Увеличить/Уменьшить – увеличение/уменьшение масштаба документа.
* Линейка – отображение или скрытие линеек. Линейка предназначена для определения размеров и выравнивания текста и объектов в документе.
* Строка состояния - включение/выключение строки состояния.

# **Основная часть**

1. Постановка задачи

В задачу курсового проекта входит:

* Разработать программное средство на языке С++ способное выполнять работу с текстом и текстовыми файлами.
* Программное средство должно выполнять следующие функции:

1. Форматирование текста.

1. Редактирование абзаца и отступов.
2. Возможность поиска и замены текста.
3. Возможность открытия текстовых файлов.
4. Возможность сохранения текста в файл.
5. Проверка орфографии.

* Реализовать понятный и удобный для пользователя интерфейс программного средства.

Таким образом, задача данного курсового проекта сводится к разработке программного средства, способного осуществлять работу с текстовыми файлами и сохранять отформатированный текст в файл.

1. Требования к программе текстового редактора

Необходимо написать программу на С++, которая смогла бы работать с текстом и предоставлять возможность дальнейшей работы с данным текстом, учитывая сохранение и открытие текстовых файлов. В связи с этим, к программе предъявляются следующие требования:

1. Понятный интерфейс.

Интерфейс программного средства будет состоять из двух основных областей. Первая область (рабочая) представляет из себя поле, в котором пользователь может непосредственно вводить и редактировать текст. Вторая область – это меню, содержащее все необходимые для работы с текстом и управления программой команды.

1. Функциональность.

Данная программа должна удовлетворять следующим требованиям:

* Предоставлять пользователю возможность выбора шрифта, стиля и размера текста.
* Делать проверку орфографии.
* Давать возможность отмены последнего действия пользователя, копирования, вставки и вырезания фрагментов текста или целого текста.
* Поддерживать работу с файлом (сохранение и открытие).

1. Оптимальность используемых алгоритмов. Разработанные алгоритмы протестированы и максимально оптимизированы под поставленные задачи.
2. Совместимость. Разрабатываемый текстовый редактор должен быть совместим с семейством ОС Windows (XP,Vista,7,10).
3. Среда разработки. Средой разработки данного программного средства выбран C++ BuiderXE от Borland. Выбор остановился на этой среде разработки потому, как в ней реализовано множество полезных компонент, как для работы с текстом, так и для работы с формой и меню. В данном проекте оказались востребованными такие не визуальные компоненты (значок компонента не отображается при запуске приложения, виден только в режиме конструктора), как:

* TMainMenu – позволяет конструировать и создавать на форме полосу главного меню, а также сопутствующие выпадающие меню.
* TPopupMenu – этот компонент придаёт другому компоненту, с которым он связывается, дополнительные функциональные возможности, он может быть связан с любым другим компонентом (формой, кнопкой и т.д.), у которого есть свойство PopupMenu. Это обычное меню, где пользователь определяет порядок действий при активизации компонента, с которым данное меню связано.
* TOpenDialog – вызывает стандартный диалог Windows открытия файлов.
* TSaveDialog – вызывает стандартный диалог Windows сохранения файлов.
* TActionList – обеспечивает диспетчеризацию действий разработчика и, соответственно, событий компонентов. Компонент представляет собой интерфейс разработчика, позволяющий ему упорядочить свою работу с действиями в процессе проектирования путём создания списков действий.
* TImageList – список изображений, помещаемых на кнопки.
* TFontDialog – компонент вызова стандартного диалога Windows для выбора шрифта.
* TColorDialog – компонент вызова стандартного диалога Windows для работы с цветом (в данном случае фона окна).

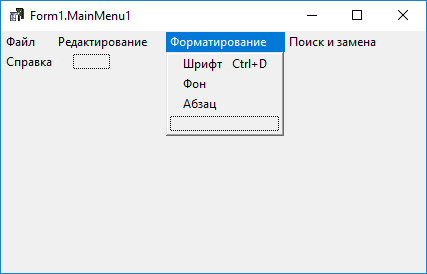
Для непосредственной работы с текстом был выбран компонент TRichEdit – многострочное окно редактирования текстов в обогащённом формате \*.RTF, позволяющее производить выбор цвета, шрифта, стиля для разных фрагментов текста, поиск и замену текста, редактирование абзаца и т.п. В окне редактирования предусмотрены типичные комбинации горячих клавиш: Ctrl+C – копирование выделенного текста в буфер обмена Clipboard, Ctrl+X – вырезание выделенного текста в буфер обмена Clipboard, Ctrl+V – вставка текста из буфера обмена Clipboard, Ctrl+Z – отмена последней команды редактирования.

Также для большей простоты в использовании были использованы инструментальная панель быстрых кнопок TToolBar, дублирующая основные разделы меню и строка состояния TStatusBar.

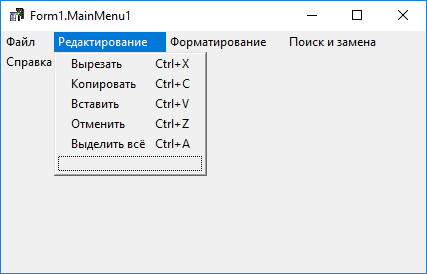
1. Разработка программы

В программе планируется реализовать следующий функционал (на скриншотах приведена организация главного (MainMenu) и контекстного (PopupMenu) меню приложения):

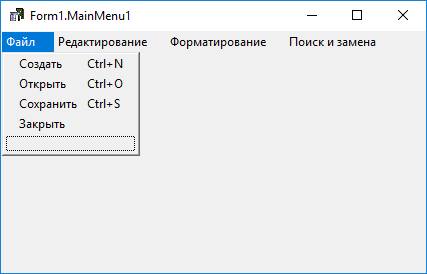
* Возможность форматирования текста: выбор различных шрифтов, стилей написания символов, регулировка размера и цвета текста. Также работа с абзацем (установка определённых отступов и выравнивания текста) и фоном окна редактирования.



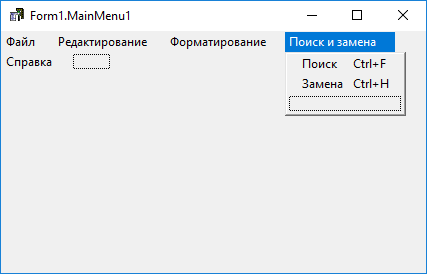
* Возможность редактирования текста: работа с буфером обмена, а именно копирование, вставка, вырезание и удаления выделенного пользователем текста, а также функции выделения всего текста и отмены последнего действия.



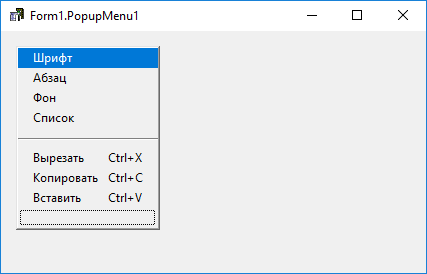
* Возможность работы с текстовыми файлами: открытие текстовых файлов, сохранение отформатированного текста в файл при помощи стандартного диалога Windows. Также создание нового пустого окна редактирования с предварительным диалогом с пользователем о надобности сохранения текущего документа.



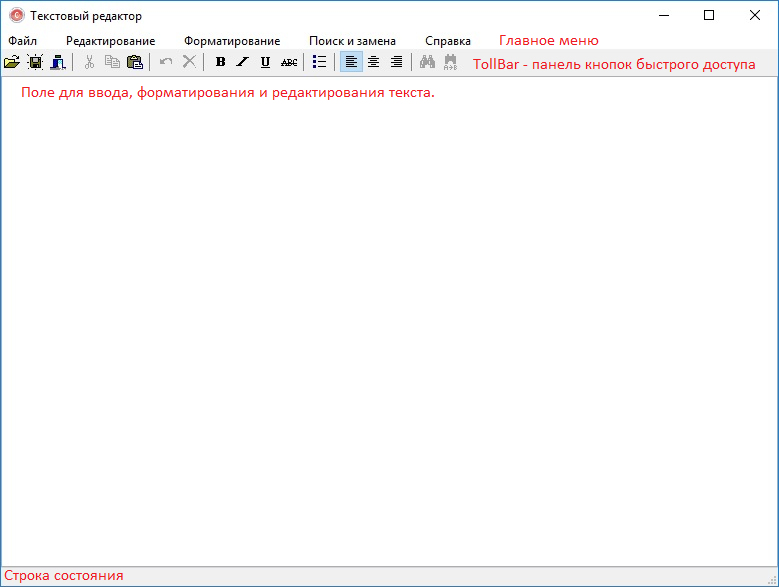
* Возможность поиска и замены слов, фрагментов текста.



* Организация контекстного меню, связанного с окном редактирования для большего удобства в использовании приложения.

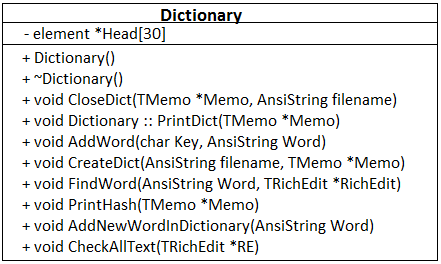


Интерфейс разрабатываемого приложения:



Работа программы основана на событиях и их обработчиках. Любое действие пользователя обработано в коде и не вызывает ошибок, вылетов и зависания программы.

Рассмотрим алгоритм проверки орфографии вводимого пользователем текста. Для реализации данной функции в программе был разработан класс Dictionary. Построим диаграмму этого класса.



Для более быстрой работы со словарём было решено использовать алгоритм наподобие алгоритма работы с хэш-таблицей. В качестве ключа к хэш-таблице была выбрана первая буква каждого слова в словаре. Т.к. на буквы «ь», «ы» и «ъ» слов не существует, в классе было создано поле типа структуры element (структура содержит информативное поле типа AnsiString, где хранится само слово, и указатель типа самой структуры element\* на следующий элемент в стеке) – массив из 30 указателей на голову каждого стека, содержащего слова в алфавитном порядке на определённую букву - хэш.

При запуске программы все слова из текстового документа Dictionary.txt – словаря – загружаются в программу (при помощи вызова метода CreateDict (AnsiString filename, TMemo \*Memo) у объекта класса Dictionary dict), и формируется массив стеков. Далее при каждом нажатии клавиши символа пользователем срабатывает обработчик события:

void \_\_fastcall TForm1::RichEdit1KeyPress (TObject \*Sender, wchar\_t &Key),   
и набираемое им в окне редактирования текста слово считывается и записывается в переменную типа AnsiString. Происходит проверка на вводимый символ. Если пользователь ввёл какой-либо знак препинания или пробел, считывание слова прекращается и для объекта класса Dictionary dict вызывается метод:

void FindWord (AnsiString Word, TRichEdit \*RichEdit),

при помощи которого введённое пользователем слово ищется в массиве стеков, а именно в том стеке, в котором хранятся слова на первую букву введённого пользователем слова (в данном приложении проверяются на корректность только слова, записанные при помощи строчных русских букв, т.к. в исходном словаре находятся только слова, удовлетворяющие этому условию). Если такового слова в словаре найдено не было, то оно выделяется красным цветом.

Имеется возможность добавлять слова в словарь. Для этого нужно выделить слово, которое хотите добавить в словарь и нажать соответствующую кнопку на панели быстрого доступа или в подменю меню «Словарь» кнопку «Добавить слово в словарь». После этого действия требуемой для занесения в словарь слово будет присоединено к одному из стеков в алфавитном порядке, а в текстовый файл это слово запишется при закрытии программы вызовом метода CloseDict (TMemo \*Memo, AnsiString filename) у объекта класса Dictionary dict.

Начало  
void \_\_fastcall TForm1::RichEdit1KeyPress(TObject \*Sender, wchar\_t &Key)

(int)Key == 8

(Key == ' ' || Key == ',' || Key == '.' || Key == ';' || Key == '?' || Key == '!'т|| Key == '\r' || Key == ')' || Key == '(') && separator

(int)Key == 8

A

B

Конец

А

AnsiString Word = "";

bool b = true;

j = 0;  
j <= i

words[j] >= 1072 && words[j] <= 1103

Word = Word + (char)(words[j] - 1104)

words[j] == 1105

Word = Word + char(words[j] - 1177)

b = false

j++

b

dict.FindWord(Word, RichEdit1, FindDialog1)

i = 0;

separator = false;

В

Key != ' ' && Key != ',' && Key != '.' && Key != ';' && Key != '?' && Key != '!' && Key != '\r' && Key != ')' && Key != '('

words[i] = (int)Key;

i++;

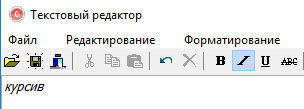
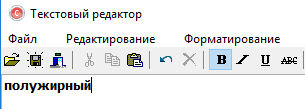
separator = true;

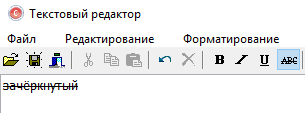
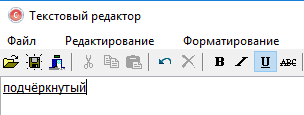
1. Принцип работы программы

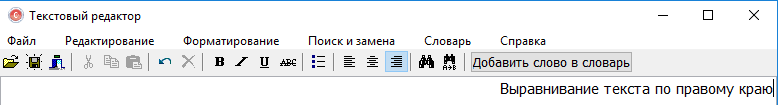
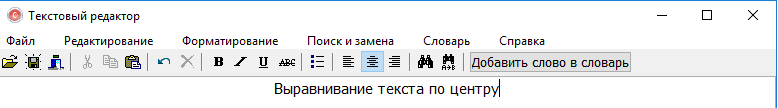
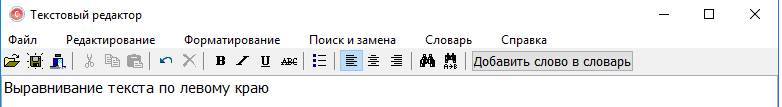
Изначально, пока поле редактирования текста пусто, пользователю доступны все команды кроме:

* Вырезать
* Копировать
* Удалить
* Отменить
* Найти
* Заменить

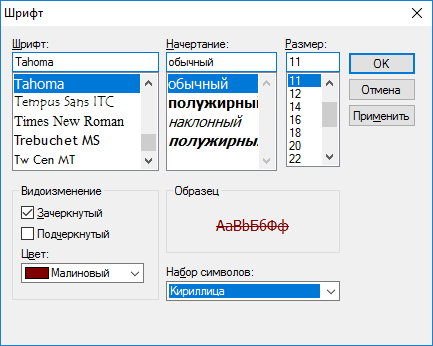
Пользователь может начать вводить текст в поле ввода или же открыть какой-нибудь существующий (в формате \*.TXT или \*.RTF) файл. Далее он может работать с текстом при помощи функционала программы.

Стили текста:   
****

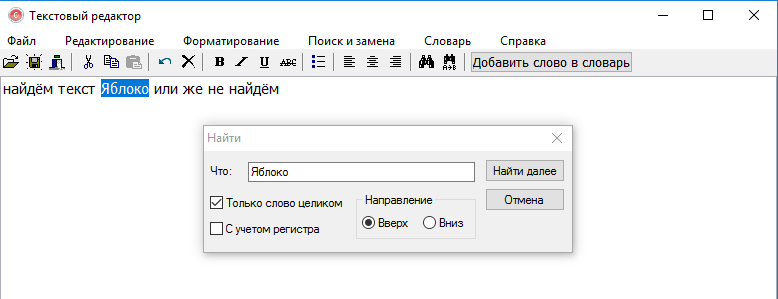
****

Выравнивание текста:

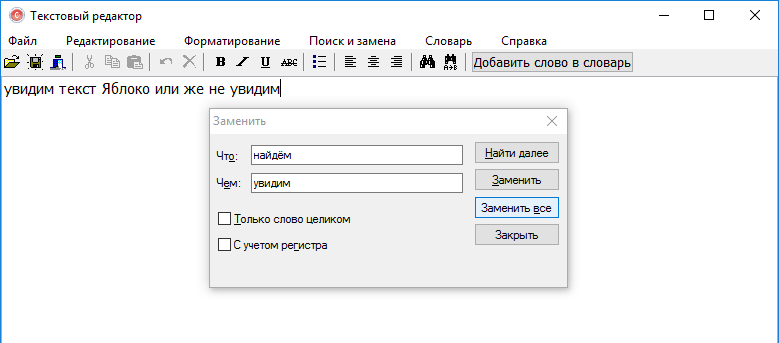
Диалоговое окно для выбора шрифта текста. (При нажатии на кнопку «Применить» можно, не закрывая диалогового окна, посмотреть, как будет выглядеть текст после заданных изменений. Если будет нажата кнопка «Отмена», то диалоговое окно закроется. Если же будет нажата кнопка «ОК», то диалоговое окно закроется и будет применены атрибуты, выбранные в данной форме.):



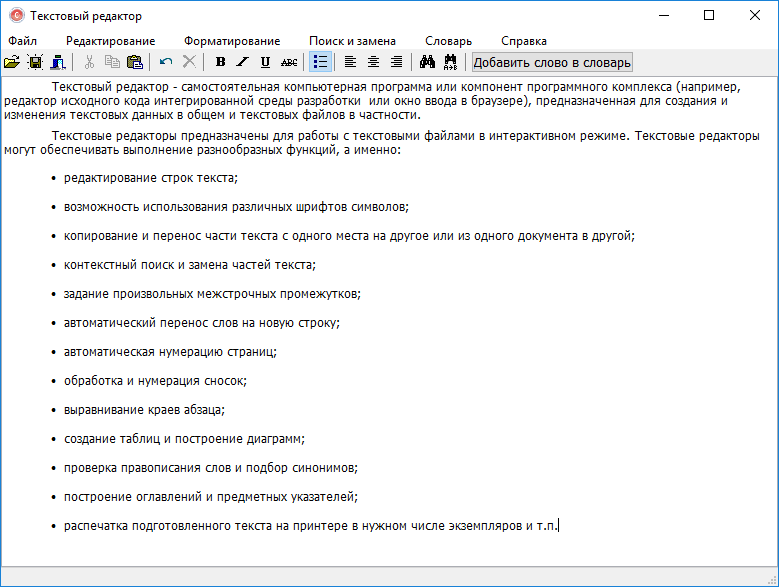
Поиск фрагментов текста осуществляется при помощи диалога с пользователем. Ему предлагается ввести слово, которое он хотел бы найти в поле для ввода и указать, вверх или вниз от текущего положения курсора должна программа попытаться найти нужное слово. Имеется возможность найти слово целиком и найти слово без учёта регистра.

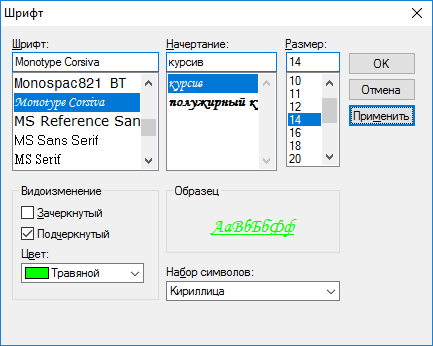


Замена фрагмента текста выполняется подобно поиску. При нажатии на кнопку «Замена» на панели быстрого доступа или на кнопку подменю «Замена» меню «Поиск и замена» всплывает диалоговое окно с двумя полями для ввода. Первое поле предназначено для слова, которое хотят заменить в тексте, второе – для слова, которое хотят увидеть в тексте вместо заменяемого. Для лучшего эффекта рекомендуется поставить курсор в начало документа.

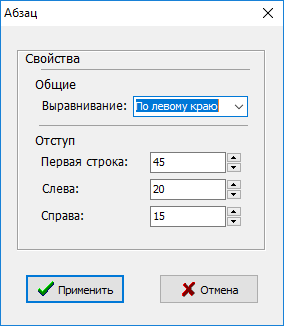


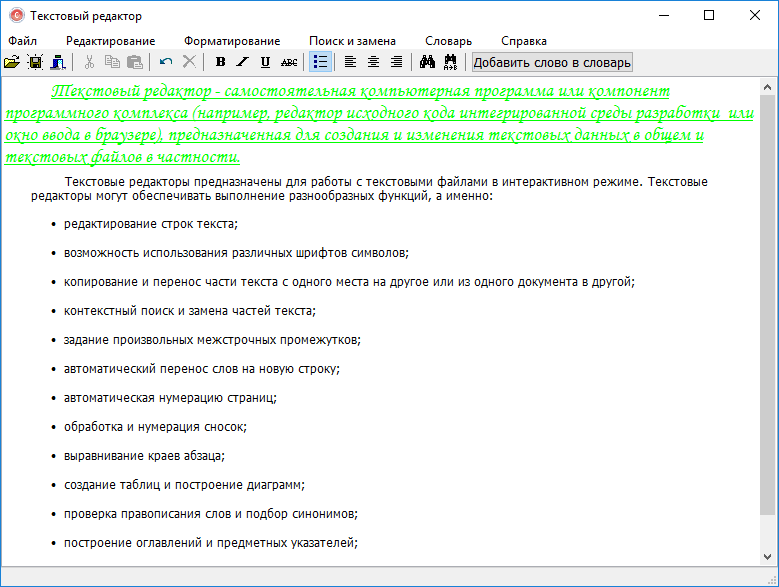
1. Пример работы с текстовым редактором

Запишем некоторый текст и отредактируем его.  


Установим шрифт первого абзаца «Monotype Corsiva», начертание – курсив, размер шрифта 16, стиль – подчёркнутый и цвет – травяной.  


У второго абзаца отредактируем поля. Поставим выравнивание текста по левому краю, сделаем 45 пикселей отступ первой строки, 20 пикселей – слева и 15 – справа. Нажмём кнопку «Применить».



Полученный текстовый документ будет иметь следующий вид:  


# **Заключение**

Результатом данного курсового проекта стал текстовый редактор типа WordPad.

Реализация проекта потребовала предварительной подготовки. Были изучены теоретические материалы по текстовым редакторам (особенности текстовых редакторов как программных средств, их основные различия, популярные представители и т.п.) Потребовалось ознакомиться с возможностями и принципами редактирования текстов (форматирование, редактирование и проверка орфографии) в BuilderXE на языке C++. Самые удобные и подходящие алгоритмы были отобраны, а затем внедрены в созданный текстовый редактор.

Интерфейс программы простой и удобный, он не требует освоения специальных навыков. Пользователь может использовать различные шрифты, стили и размер текста, менять цвет и фон, отменять последнее действие, копировать, вставлять и вырезать текстовые фрагменты, сохранять и открывать текстовые файлы в стандартных форматах \*.TXT и \*.RTF.

Разработанный текстовый редактор является цельным законченным продуктом. В дальнейшем его можно использовать как в рамках тех возможностей, которые он предоставляет, так и в качестве шаблона, в структуру которого можно добавлять любой дополнительный функционал, учитывающий ту или иную специфику работы пользователя с текстами.

# **Список используемой литературы**

1. Жвакина, А. В. Разработка Windows-приложений в среде визуального программирования: Электронное учебное издание (Пособие). / А. В. Жвакина. – Минск: ВА РБ, 2016.
2. Шилдт Г. С++ базовый курс, третье издание. Перевод англ. – Издательский дом «Вильямс» 2012 г.
3. <https://ru.wikipedia.org/wiki/Текстовый_редактор>
4. https://ru.wikipedia.org/wiki/WordPad

# **Приложение 1. Исходный код программы**

Код основного модуля Unit1.cpp, где находится форма самого текстового редактора.

#include <vcl.h>  
#pragma hdrstop  
#include "Unit1.h"  
#include "Unit2.h"  
#include "Unit3.h"  
#include "Unit4.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)  
#pragma resource "\*.dfm"

TForm1 \*Form1;  
TFileStream \*file;  
AnsiString filename;  
Dictionary dict;  
int words[30];  
int i = 0;  
bool separator = true;

//---------------------------------------------------------------------------  
\_\_fastcall TForm1::TForm1(TComponent\* Owner): TForm(Owner)  
{  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N3Click(TObject \*Sender)  
{  
 if (OpenDialog1->Execute()) {  
 RichEdit1->Lines->LoadFromFile(OpenDialog1->FileName);  
 }  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::ToolButton16Click(TObject \*Sender)  
{  
 OpenDialog1->Filter = "Текстовый файл|\*.txt;|RTF-документ|\*.rtf;";  
 N3Click(Sender);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N4Click(TObject \*Sender)  
{  
 if (SaveDialog1->Execute()){  
 RichEdit1->Lines->SaveToFile(SaveDialog1->FileName);  
 }  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::ToolButton17Click(TObject \*Sender)  
{  
 SaveDialog1->Filter = "Текстовый файл|\*.txt;|RTF-документ|\*.rtf;";  
 N4Click(Sender);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::FontDialog1Apply(TObject \*Sender, HWND Wnd)  
{  
 RichEdit1->SelAttributes->Assign(FontDialog1->Font);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N15Click(TObject \*Sender)  
{  
 if (FontDialog1->Execute()) {  
 RichEdit1->SelAttributes->Assign(FontDialog1->Font);  
 }  
 else return;  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N29Click(TObject \*Sender)  
{  
 if (ColorDialog1->Execute()) {  
 RichEdit1->Color = ColorDialog1->Color;  
 }  
 else return;  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::FormCreate(TObject \*Sender)  
{  
 dict.CreateDict("Dictionary.txt", Memo1);  
 RichEdit1->PopupMenu = PopupMenu1;  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N26Click(TObject \*Sender)  
{  
 N15Click(Sender);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N27Click(TObject \*Sender)  
{  
 N29Click(Sender);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N25Click(TObject \*Sender)  
{  
 Form2->Show();  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N2Click(TObject \*Sender)  
{  
 Form3->Show();  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N5Click(TObject \*Sender)  
{  
 Form2->Show();  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N14Click(TObject \*Sender)  
{  
 RichEdit1->SelStart = 0;  
 int len = RichEdit1->Text.Length();  
 RichEdit1->SelLength = len;  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N11Click(TObject \*Sender)  
{  
 Application->Title = "Справка";  
 ShowMessage("Программа предназначена для работы с текстом.\n\nВерсия программы: 1.0\nРазработчик: Шимановская П.Н.");  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::RichEdit1KeyPress(TObject \*Sender, wchar\_t &Key)  
{  
 if ((int)Key == 8) {  
 if (i > 0)  
 i--;  
 }  
 else if ((Key == ' ' || Key == ',' || Key == '.' || Key == ';' || Key == '?' ||   
Key == '!' || Key == '\r' || Key == ')' || Key == '(' || Key == '"') && separator) {  
 AnsiString Word = "";  
 bool b = true;  
 if (i > 0) {  
 for (int j = 0; j < i; j++) {  
 if (words[j] >= 1072 && words[j] <= 1103)  
 Word = Word + (char)(words[j] - 1104);  
 else if (words[j] == 1105)  
 Word = Word + char(words[j] - 1177);  
 else  
 b = false;  
 }  
 }  
 if (b)  
 dict.FindWord(Word, RichEdit1);  
 i = 0;  
 separator = false;  
 }  
 else {  
 if (Key != ' ' && Key != ',' && Key != '.' && Key != ';' && Key != '?' &&  
 Key != '!' && Key != '\r' && Key != ')' && Key != '(' && Key != '"') {  
 words[i] = (int)Key;  
 i++;  
 separator = true;  
 }  
 }  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N22Click(TObject \*Sender)  
{  
 dict.CheckAllText(RichEdit1);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::Button1Click(TObject \*Sender)  
{  
 AnsiString Word = RichEdit1->SelText;  
 dict.AddNewWordInDictionary(Word);  
 N22Click(Sender);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::N21Click(TObject \*Sender)  
{  
 N22Click(Sender);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm1::FormClose(TObject \*Sender, TCloseAction &Action)  
{  
 dict.CloseDict(Memo1, "Dictionary.txt");  
}

//---------------------------------------------------------------------------

Код Unit1.h.

#ifndef Unit1H  
#define Unit1H

//---------------------------------------------------------------------------  
#include <Classes.hpp>  
#include <Controls.hpp>  
#include <StdCtrls.hpp>  
#include <Forms.hpp>  
#include <ActnList.hpp>  
#include <ComCtrls.hpp>  
#include <ExtActns.hpp>  
#include <ExtCtrls.hpp>  
#include <ImgList.hpp>  
#include <StdActns.hpp>  
#include <ToolWin.hpp>  
#include <Menus.hpp>  
#include <Dialogs.hpp>  
#include <OleCtnrs.hpp>

//---------------------------------------------------------------------------  
class TForm1 : public TForm  
{  
\_\_published: // IDE-managed Components  
 TRichEdit \*RichEdit1;  
 TActionList \*ActionList1;  
 TImageList \*ImageList1;  
 TEditCut \*EditCut1;  
 TEditCopy \*EditCopy1;  
 TEditPaste \*EditPaste1;  
 TEditSelectAll \*EditSelectAll1;  
 TEditUndo \*EditUndo1;  
 TRichEditBold \*RichEditBold1;  
 TRichEditItalic \*RichEditItalic1;  
 TRichEditUnderline \*RichEditUnderline1;  
 TRichEditStrikeOut \*RichEditStrikeOut1;  
 TRichEditBullets \*RichEditBullets1;  
 TRichEditAlignLeft \*RichEditAlignLeft1;  
 TRichEditAlignRight \*RichEditAlignRight1;  
 TRichEditAlignCenter \*RichEditAlignCenter1;  
 TFileExit \*FileExit1;  
 TFileOpen \*FileOpen1;  
 TFileSaveAs \*FileSaveAs1;  
 TSearchFind \*SearchFind1;  
 TSearchReplace \*SearchReplace1;  
 TMainMenu \*MainMenu1;  
 TMenuItem \*N1;  
 TMenuItem \*N2;  
 TMenuItem \*N3;  
 TMenuItem \*N4;  
 TMenuItem \*N6;  
 TMenuItem \*N7;  
 TMenuItem \*N8;  
 TMenuItem \*N9;  
 TMenuItem \*N10;  
 TMenuItem \*N11;  
 TMenuItem \*N12;  
 TMenuItem \*N13;  
 TMenuItem \*N15;  
 TMenuItem \*N16;  
 TMenuItem \*N17;  
 TMenuItem \*N18;  
 TMenuItem \*N19;  
 TMenuItem \*N29;  
 TToolBar \*ToolBar1;  
 TToolButton \*ToolButton1;  
 TToolButton \*ToolButton2;  
 TToolButton \*ToolButton3;  
 TToolButton \*ToolButton4;  
 TToolButton \*ToolButton6;  
 TToolButton \*ToolButton7;  
 TToolButton \*ToolButton8;  
 TToolButton \*ToolButton9;  
 TToolButton \*ToolButton10;  
 TToolButton \*ToolButton11;  
 TToolButton \*ToolButton12;  
 TToolButton \*ToolButton13;  
 TToolButton \*ToolButton15;  
 TToolButton \*ToolButton16;  
 TToolButton \*ToolButton17;  
 TToolButton \*ToolButton18;

TToolButton \*ToolButton19;   
 TStatusBar \*StatusBar1;  
 TEditDelete \*EditDelete1;  
 TToolButton \*ToolButton20;  
 TOpenDialog \*OpenDialog1;  
 TSaveDialog \*SaveDialog1;  
 TPopupMenu \*PopupMenu1;  
 TFontDialog \*FontDialog1;  
 TMenuItem \*N25;  
 TColorDialog \*ColorDialog1;  
 TMenuItem \*N26;  
 TMenuItem \*N27;  
 TMenuItem \*N28;  
 TMenuItem \*N30;  
 TMenuItem \*N31;  
 TMenuItem \*N32;  
 TMenuItem \*N33;  
 TToolButton \*ToolButton5;  
 TToolButton \*ToolButton21;  
 TToolButton \*ToolButton22;  
 TToolButton \*ToolButton23;  
 TToolButton \*ToolButton24;  
 TToolButton \*ToolButton25;  
 TMenuItem \*N5;  
 TMenuItem \*N14;  
 TMemo \*Memo1;  
 TFindDialog \*FindDialog1;  
 TButton \*Button1;  
 TToolButton \*ToolButton14;  
 TMenuItem \*N20;  
 TMenuItem \*N21;  
 TMenuItem \*N22;  
 void \_\_fastcall N3Click(TObject \*Sender);  
 void \_\_fastcall ToolButton16Click(TObject \*Sender);  
 void \_\_fastcall N4Click(TObject \*Sender);  
 void \_\_fastcall ToolButton17Click(TObject \*Sender);  
 void \_\_fastcall FontDialog1Apply(TObject \*Sender, HWND Wnd);  
 void \_\_fastcall N15Click(TObject \*Sender);  
 void \_\_fastcall N29Click(TObject \*Sender);  
 void \_\_fastcall FormCreate(TObject \*Sender);  
 void \_\_fastcall N26Click(TObject \*Sender);  
 void \_\_fastcall N27Click(TObject \*Sender);  
 void \_\_fastcall N25Click(TObject \*Sender);  
 void \_\_fastcall N2Click(TObject \*Sender);  
 void \_\_fastcall N5Click(TObject \*Sender);  
 void \_\_fastcall N14Click(TObject \*Sender);  
 void \_\_fastcall N11Click(TObject \*Sender);  
 void \_\_fastcall RichEdit1KeyPress(TObject \*Sender, wchar\_t &Key);  
 void \_\_fastcall N22Click(TObject \*Sender);  
 void \_\_fastcall Button1Click(TObject \*Sender);  
 void \_\_fastcall N21Click(TObject \*Sender);  
 void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);  
private: // User declarations  
public: // User declarations  
 \_\_fastcall TForm1(TComponent\* Owner);  
};  
//---------------------------------------------------------------------------  
extern PACKAGE TForm1 \*Form1;  
#endif

Код модуля Unit2.cpp - модуля, где находится форма для работы с абзацем.

#include <vcl.h>  
#pragma hdrstop  
#include "Unit2.h"  
#include "Unit1.h"

//---------------------------------------------------------------------------  
#pragma package(smart\_init)  
#pragma resource "\*.dfm"  
TForm2 \*Form2;

//---------------------------------------------------------------------------  
\_\_fastcall TForm2::TForm2(TComponent\* Owner) : TForm(Owner) { }

//---------------------------------------------------------------------------  
void \_\_fastcall TForm2::BitBtn1Click(TObject \*Sender)  
{  
 ComboBox1Click(Sender);  
 Edit1Change(Sender);  
 Edit2Change(Sender);  
 Edit3Change(Sender);  
 Form2->Close() ;  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm2::ComboBox1Click(TObject \*Sender)  
{  
 if (ComboBox1->ItemIndex == 0)  
 Form1->RichEdit1->Paragraph->Alignment = taLeftJustify;  
 else if (ComboBox1->ItemIndex == 1)  
 Form1->RichEdit1->Paragraph->Alignment = taCenter;  
 else if (ComboBox1->ItemIndex == 2)  
 Form1->RichEdit1->Paragraph->Alignment = taRightJustify;  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm2::Edit1Change(TObject \*Sender)  
{  
 if (StrToInt(Edit1->Text) != 0) {  
 Form1->RichEdit1->Paragraph->FirstIndent = StrToInt(Edit1->Text);  
 Form1->RichEdit1->Paragraph->LeftIndent = StrToInt(Edit2->Text) - StrToInt(Edit1->Text);  
 }  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm2::Edit2Change(TObject \*Sender)  
{  
 Form1->RichEdit1->Paragraph->LeftIndent =   
 StrToInt(Edit2->Text) - StrToInt(Edit1->Text);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm2::Edit3Change(TObject \*Sender)  
{  
 Form1->RichEdit1->Paragraph->RightIndent = StrToInt(Edit3->Text);  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm2::BitBtn2Click(TObject \*Sender)  
{  
 Form1->RichEdit1->Undo();  
 Form2->Close();  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm2::FormShow(TObject \*Sender)  
{  
 if (Form1->RichEdit1->Paragraph->Alignment == taLeftJustify)  
 ComboBox1->ItemIndex = 0;  
 else if (Form1->RichEdit1->Paragraph->Alignment == taCenter)  
 ComboBox1->ItemIndex = 1;  
 else if (Form1->RichEdit1->Paragraph->Alignment == taRightJustify)  
 ComboBox1->ItemIndex = 2;  
 Edit1->Text = Form1->RichEdit1->Paragraph->FirstIndent;  
 Edit2->Text = Form1->RichEdit1->Paragraph->LeftIndent +   
 Form1->RichEdit1->Paragraph->FirstIndent;  
 Edit3->Text = Form1->RichEdit1->Paragraph->RightIndent;  
}

//---------------------------------------------------------------------------

Код Unit2.h.

#ifndef Unit2H  
#define Unit2H  
#include "Unit1.h"  
//---------------------------------------------------------------------------  
#include <Classes.hpp>  
#include <Controls.hpp>  
#include <StdCtrls.hpp>  
#include <Forms.hpp>  
#include <Buttons.hpp>  
#include <ComCtrls.hpp>  
#include <ExtCtrls.hpp>  
//---------------------------------------------------------------------------  
class TForm2 : public TForm  
{  
\_\_published: // IDE-managed Components  
 TPanel \*Panel1;  
 TLabel \*Label1;  
 TLabel \*Label2;  
 TLabel \*Label3;  
 TLabel \*Label4;  
 TLabel \*Label5;  
 TLabel \*Label7;  
 TLabel \*Label8;  
 TComboBox \*ComboBox1;  
 TEdit \*Edit1;  
 TEdit \*Edit2;  
 TEdit \*Edit3;  
 TUpDown \*UpDown1;  
 TUpDown \*UpDown2;  
 TUpDown \*UpDown3;  
 TPanel \*Panel2;  
 TPanel \*Panel3;  
 TBitBtn \*BitBtn1;  
 TBitBtn \*BitBtn2;  
 void \_\_fastcall BitBtn1Click(TObject \*Sender);  
 void \_\_fastcall ComboBox1Click(TObject \*Sender);  
 void \_\_fastcall Edit1Change(TObject \*Sender);  
 void \_\_fastcall Edit2Change(TObject \*Sender);  
 void \_\_fastcall Edit3Change(TObject \*Sender);  
 void \_\_fastcall BitBtn2Click(TObject \*Sender);  
 void \_\_fastcall FormShow(TObject \*Sender);  
private: // User declarations  
public: // User declarations  
 \_\_fastcall TForm2(TComponent\* Owner);  
};  
//---------------------------------------------------------------------------  
extern PACKAGE TForm2 \*Form2;  
//---------------------------------------------------------------------------  
#endif

Код модуля Unit3.cpp - модуля, где находится форма для диалога с пользователем о сохранении текущего документа, открывающаяся при нажатии на кнопку «Создать».

#include <vcl.h>  
#pragma hdrstop  
#include "Unit3.h"  
#include "Unit1.h"

//---------------------------------------------------------------------------  
#pragma package(smart\_init)  
#pragma resource "\*.dfm"  
TForm3 \*Form3;

//---------------------------------------------------------------------------  
\_\_fastcall TForm3::TForm3(TComponent\* Owner): TForm(Owner) { }

//---------------------------------------------------------------------------  
void \_\_fastcall TForm3::BitBtn1Click(TObject \*Sender)  
{  
 if (Form1->SaveDialog1->Execute()){  
 Form3->Close();  
 Form1->RichEdit1->Lines->SaveToFile(Form1->SaveDialog1  
 ->FileName);  
 }  
 Form1->RichEdit1->Clear();  
}

//---------------------------------------------------------------------------  
void \_\_fastcall TForm3::BitBtn2Click(TObject \*Sender)  
{  
 Form1->RichEdit1->Clear();  
 Form3->Close();  
}

//---------------------------------------------------------------------------

Код Unit3.h.

#ifndef Unit3H  
#define Unit3H  
//---------------------------------------------------------------------------  
#include <Classes.hpp>  
#include <Controls.hpp>  
#include <StdCtrls.hpp>  
#include <Forms.hpp>  
#include <Buttons.hpp>  
//---------------------------------------------------------------------------  
class TForm3 : public TForm  
{  
\_\_published: // IDE-managed Components  
 TLabel \*Label1;  
 TBitBtn \*BitBtn1;  
 TBitBtn \*BitBtn2;  
 TLabel \*Label2;  
 void \_\_fastcall BitBtn1Click(TObject \*Sender);  
 void \_\_fastcall BitBtn2Click(TObject \*Sender);  
private: // User declarations  
public: // User declarations  
 \_\_fastcall TForm3(TComponent\* Owner);  
};  
//---------------------------------------------------------------------------  
extern PACKAGE TForm3 \*Form3;  
//---------------------------------------------------------------------------  
#endif

Код модуля Unit4.h, где содержится класс для работы со словарём.

#ifndef Unit4H  
#define Unit4H  
#include <vcl.h>

struct element  
{  
 AnsiString word;  
 element \*next;  
};

class Dictionary  
{  
private:  
 element \*Head[30];  
public:  
 Dictionary();  
 ~Dictionary(){};  
 void CloseDict(TMemo \*Memo, AnsiString filename);  
 void Dictionary :: PrintDict(TMemo \*Memo);  
 void AddWord(char Key, AnsiString Word);  
 void CreateDict(AnsiString filename, TMemo \*Memo);  
 void FindWord(AnsiString Word, TRichEdit \*RichEdit);  
 void PrintHash(TMemo \*Memo);  
 void AddNewWordInDictionary(AnsiString Word);  
 void CheckAllText(TRichEdit \*RE);  
};

Код модуля Unit4.cpp, описывающий методы работы со словарём.

#pragma hdrstop  
#include "Unit4.h"  
#include <vcl.h>

Dictionary :: Dictionary()  
{  
 for (int i = 0; i < 30; i++) {  
 Head[i] = NULL;  
 }  
}

void Dictionary :: PrintDict(TMemo \*Memo)  
{  
 Memo->Clear();  
 for (int hash = 0; hash < 30; hash++) {  
 if (Head[hash]) {  
 element \*temp = Head[hash];  
 while (temp){  
 Memo->Lines->Add(temp->word);  
 temp = temp->next;  
 }  
 }  
 }  
}

void Dictionary :: CloseDict(TMemo \*Memo, AnsiString filename)  
{  
 PrintDict(Memo);  
 if(FileExists(filename))  
 DeleteFile(filename);  
 TFileStream \*Stream = new TFileStream (filename, fmCreate);  
 delete Stream;  
 Memo->Lines->SaveToFile(filename);  
 for (int hash = 0; hash < 30; hash++) {  
 while (Head[hash]){  
 element \*temp = Head[hash];  
 Head[hash] = Head[hash]->next;  
 delete temp;  
 }  
 }  
}

int WhichHash(char Key)  
{  
 int hash;  
 if ((int)Key == -72)  
 hash = 6;  
 else if ((int)Key < -26)  
 hash = (int)Key + 32;  
 else if ((int)Key >= -3)  
 hash = (int)Key + 30  
 else  
 hash = (int)Key + 33;  
 return hash;  
}

void Dictionary :: AddWord(char Key, AnsiString Word)  
{  
 int hash = WhichHash(Key);  
 element \*temp = new element;  
 temp->next = Head[hash];  
 temp->word = Word;  
 Head[hash] = temp;  
}

void Dictionary :: CreateDict(AnsiString filename, TMemo \*Memo)  
{  
 Memo->Lines->Clear();  
 Memo->Lines->LoadFromFile(filename);  
 for (int i = 0; i < Memo->Lines->Count; i++) {  
 AnsiString Word = Memo->Lines->Strings[i];  
 char Key = Word.c\_str()[0];  
 AddWord(Key, Word);  
 }  
}

void Dictionary :: FindWord(AnsiString Word, TRichEdit \*RichEdit)  
{

char Key = Word.c\_str()[0];  
 int hash = WhichHash(Key);  
 if (Head[hash]) {  
 element \*item = Head[hash];  
 while (item){  
 if (item->word == Word) {  
 if (RichEdit->SelStart - Word.Length()) {  
 x = RichEdit->SelStart - Word.Length() - 1;  
 }  
 else  
 x = RichEdit->SelStart - Word.Length();  
 y = RichEdit->SelStart;  
 RichEdit->SelStart = x;  
 RichEdit->SelLength = Word.Length() + 1;  
 RichEdit->SelAttributes->Color = clBlack; RichEdit->SelStart = y;   
 return;  
 }  
 else  
 item = item->next;  
 }  
 }  
 int x, y;  
 if (RichEdit->SelStart - Word.Length()) {  
 x = RichEdit->SelStart - Word.Length() - 1;  
 }  
 else  
 x = RichEdit->SelStart - Word.Length();  
 y = RichEdit->SelStart;  
 RichEdit->SelStart = x;  
 RichEdit->SelLength = Word.Length() + 1;  
 RichEdit->SelAttributes->Color = clRed; //атрибуты неверного текста  
 RichEdit->SelStart = y;  
 RichEdit->SelAttributes->Color = clBlack;  
}

void Dictionary :: AddNewWordInDictionary(AnsiString Word)  
{  
 if (Word != "") {  
 char Key = Word.c\_str()[0];  
 AddWord(Key, Word);  
 }  
}

void Dictionary :: CheckAllText(TRichEdit \*RE)  
{  
 RE->SelStart = 0;  
 bool separator = true;  
 int words[30];  
 int k = 0;  
 AnsiString Text = RE->Text;  
 for (int i = 0; i < RE->Text.Length(); i++) {  
 RE->SelStart = i;  
 char Key = Text.c\_str()[i];  
 if ((Key == ' ' || Key == ',' || Key == '.' || Key == ';' || Key == '?'  
 || Key == '!' || Key == '\r' || Key == ')' || Key == '(' || Key == '"')   
 && separator) {  
 AnsiString Word = "";  
 bool b = true;  
 for (int j = 0; j < k; j++) {  
 if (words[j] >= -72 && words[j] <= 0)  
 Word = Word + (char)words[j];  
 else  
 b = false;  
 }  
 if (b)  
 FindWord(Word, RE);  
 k = 0;  
 separator = false;  
 }  
 else {  
 if (Key != ' ' && Key != ',' && Key != '.' && Key != ';' && Key != '?' && Key != '!' && Key != '\r' && Key != ')' && Key != '(' && Key != '"') {  
 words[k] = (int)Key;  
 k++;  
 separator = true;  
 }  
 }  
 }  
}

//---------------------------------------------------------------------------  
#pragma package(smart\_init)