МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

Кафедра современных технологий программирования

ГУЗОВ АЛЕКСАНДР ГЕННАДЬЕВИЧ

**Приложение для работы с текстом**

Курсовая работа   
по дисциплине «Алгоритмизация и программирование»

студента 2 курса специальности

1-26 03 01 «Управление информационными ресурсами»

дневной формы получения образования

|  |  |
| --- | --- |
|  | Научный руководитель  Гуща Юлия Вальдемаровна,  старший преподаватель  кафедры современных технологий программирования |

**РЕЗЮМЕ**

**Курсовая работа**: 38 с., 11 рис., 1 табл., 3 прил., 4 источника.

Текстовый редактор, TextEditor, C#, Windows Forms.

**Объект исследования:** язык программирования С#.

**Предмет исследования:** возможности языка программирования С# в области разработки программ с графическим интерфейсом и работы с текстом и текстовыми файлами.

**Цель работы:** изучить возможные способы реализации основных функций, необходимых для создания текстового редактора, научиться использованию необходимых инструментов и разработать приложение, соответствующее всем поставленным требованиям.

**Методы исследования:** сравнительного анализа, синтеза, группировок, моделирования.

**Исследования и разработки:** изучены основные инструменты для создания графических приложений и работы с текстом и текстовыми файлами в языке программирования C#; разработано графическое приложение для работы с текстом и текстовыми файлами TextEditor.

**Элементы научной новизны:** разработана программа, предоставляющая пользователю возможность работы с текстом.

**Область возможного практического применения:** работа с текстовыми файлами в условиях домашнего использования и на работе или в офисе.

Автор работы подтверждает, что приведённый в курсовой работе расчётно-аналитический материал правильно и объективно отражает состояние исследуемого процесса, а все заимствованные из литературных и других источников теоретические, методологические и методические положения и концепции сопровождаются ссылками на их авторов.

**SUMMARY**

**Graduate work**: 38 pp, 11 fig., 1 tabl., 3 app., 4 ref.

Текстовый редактор, TextEditor, C#, Windows Forms.

**The object of the research:** язык программирования С#.

**The subject of the research:** возможности языка программирования С# в области разработки программ с графическим интерфейсом и работы с текстом и текстовыми файлами.

**The research objective:** изучить возможные способы реализации основных функций, необходимых для создания текстового редактора, научиться использованию необходимых инструментов и разработать приложение, соответствующее всем поставленным требованиям.

**The methods of research:** сравнительного анализа, синтеза, группировок, моделирования.

**Researches and developments:** изучены основные инструменты для создания графических приложений и работы с текстом и текстовыми файлами в языке программирования C#; разработано графическое приложение для работы с текстом и текстовыми файлами TextEditor.

**The elements scientific novelty:** разработана программа, предоставляющая пользователю возможность работы с текстом.

**The field of possible practical use:** работа с текстовыми файлами в условиях домашнего использования и на работе или в офисе.

The author of work confirms that the design and analytical materials of graduate work reflect the state of the process under research correctly and impartially, and all theoretical, methodological and methodical statements and conceptions borrowed from the literature and other sources are accompanied by references to their authors.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc9791061)

[1 ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ 7](#_Toc9791062)

[1.1 Постановка задачи 7](#_Toc9791063)

[1.2 Классы 7](#_Toc9791064)

[1.2.1 Класс Program 7](#_Toc9791065)

[2.2 Класс Main\_Form 8](#_Toc9791066)

[1.2.3 Класс Find 10](#_Toc9791067)

[1.3 Формы 11](#_Toc9791068)

[1.3.1 Форма Main\_Form 11](#_Toc9791069)

[1.3.2 Форма Find 12](#_Toc9791070)

[2 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 13](#_Toc9791071)

[2.1 Главная форма 13](#_Toc9791072)

[2.2 Вспомогательная форма 17](#_Toc9791073)

[ЗАКЛЮЧЕНИЕ 19](#_Toc9791074)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc9791075)

[ПРИЛОЖЕНИЯ 21](#_Toc9791076)

# 

# **ВВЕДЕНИЕ**

Работа с текстовыми файлами – одна из самых часто выполняемых пользователями ПК работ. Это связано с тем, что текст является основным источником информации для человека, и, в 21 веке он зачастую представлен не в виде книг или журналов, а в виде текстовых файлов компьютера, что позволяет выполнять его быструю передачу и обработку. Следовательно, возникает большая потребность в разного рода программах, способных выполнять чтение текста из текстового файла, изменять его и сохранять внесенные изменения.

Соответственно, **цель курсовой работы** – изучить возможные способы реализации основных функций, необходимых для создания текстового редактора, научиться использованию необходимых инструментов и разработать приложение, соответствующее всем поставленным требованиям.

Для разработки приложения был выбран язык программирования С# на платформе .Net, отлично подходящей для разработки универсальных приложений высокого качества, использующих все возможности компьютеров с Windows, и среда программирования Visual Studio 2017, обладающая всеми необходимыми функциями и возможностями и позволяющая реализовать все запланированные возможности приложения.

В пояснительной записке разобраны основные возможности программы и их реализация, а также описана работа с интерфейсом пользователя.

В данной программе реализованы следующие функциональные возможности:

* создание нового документа;
* открытие существующего документа;
* ввод текста;
* сохранение текста;
* отмена последних действий;
* повторение отмененных действий;
* копирование/вырезание/вставка текста;
* изменение шрифта и цвета текста;
* изменение стилей текста (жирный, курсивный, зачеркнутый, подчеркнутый);
* выравнивание текста (по левой стороне, по центру, по правому краю);
* поиск слова

В данной программе использовались следующие разделы C#:

* Управляющие структуры (условие, циклы);
* Символы и строки;
* Коллекции;
* Методы;
* Классы;
* Работа с файлами;
* Обработка событий;

# **1 ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ**

## 1.1 Постановка задачи

Тема курсовой работы: «Создание программы - текстового редактора».

Программа должна уметь работать с текстом(должна иметь такие базовые функции как сохранение файла, открытие файла, поиск заданного слова), производить базовые операции по форматированию текста(изменение стилей текста, его размера, положения по горизонтали, цвета) и обладать приятным и удобным интерфейсом.

Должна быть реализлвана возможность сохранить файлы в 2 форматах: RTF и TXT.

Программа должна быть реализована в среде Visual Studio 2017 с использованием языка программирования C# на платформе .Net.

## 1.2 Классы

Всего в программе реализовано 3 класса: Program, Main\_Form и Find.

### 1.2.1 Класс Program

Является стартовой точкой входа в приложение.

Сначала программой загружается данный класс (см. приложение А) и запускается метод Main, затем с помощью выражения Application.Run(new Main\_Form()) он запускает форму Main\_Form. Для изменения текущей стартовой формы в приложении на другую нужно изменить в этом выражении Main\_Form на соответствующий класс формы.

### 2.2 Класс Main\_Form

Служит для описания обработки событий.

Используемые переменные (см. листинг 2.2.1):

string path = "";

string filename = "";

int change = 0;

**Листинг 2.2.1** – Переменные

Примечание ‑ Источник: собственная разработка на основе [3, с.15].

Переменная path содержит путь к файлу. Переменная filename хранит в себе имя текущего файла. Переменная change хранит в себе числовое значение, которое указывает был изменен текст в элементе richTextBox1 формы Main\_Form или нет.

Также для хранения предыдущих и предшествующих версий текста используется встроенный класс Stack<T> (см. листинг 2.2.2), представляющий коллекцию, которая использует алгоритм LIFO ("последний вошел - первый вышел"). Данные коллекций затем используются в обработчике событий, вызываемых при нажатии на кнопки Undo/Redo.

Stack<string> undoActions = new Stack<string>();

Stack<string> redoActions = new Stack<string>();

**Листинг 2.2.2** – Объекты класса Stack

Примечание ‑ Источник: собственная разработка на основе [3, с.15].

В классе были реализованы следующие методы:

Таблица 2.1 – Методы класса Main\_Form

|  |  |
| --- | --- |
| Название метода | Описание метода |
| Proverka (см. листинг Б.1) | запрос на сохранение изменений в документе (выводит диалоговое окно, спрашивающее, хочет ли пользователь сохранить свои данные; возвращает значение диалогового окна |

Продолжение таблицы 2.1

|  |  |
| --- | --- |
| newToolStripMenuItem\_Click (см. листинг Б.2) | создание нового файла (вызывает метод Proverka и в зависимости от ответа пользователя, вызывает метод сохранения изменений в файле или создает новый файл |
| openToolStripMenuItem (см. листинг Б.3) | открытие файла (вызывает метод Proverka и в зависимости от ответа пользователя, вызывает метод сохранения изменений в файле или выполняет вызов диалогового окна открытия нового файла |
| saveAsToolStripMenuItem\_Click (см. листинг Б.4) | сохранение с созданием нового файла или перезаписью существующего (вызывает диалоговое окно сохранения файла, сохраняет значения пути к файлу и имя файла, а также изменяет отображаемое в форме имя файла |
| saveToolStripMenuItem\_Click (см. листинг Б.5) | сохранение изменений в текущий документ (если значение переменной path, хранящей путь к текущему фалу, не null, то данные записываются в текущий файл, иначе вызывается метод saveAsToolStripMenuItem\_Click |
| Main\_Form\_FormClosing (см. листинг Б.6) | закрытие формы по нажатию на крестик (если текст был изменен (change != 0), то происходит вызов метода Proverka и, в зависимости от результата, либо вызывается метод сохранения файла, либо отменяется закрытие окна формы путем изменения значения свойства Сancel формы |
| quitToolStripMenuItem\_Click (см. листинг Б.7) | закрытие формы при нажатии кнопки Quit в меню File (сперва выполняется проверка на наличие текста и изменений в нем, и, если текст есть и был изменен, в зависимости от ответа пользователя на запрос о сохранении данных, сперва вызывается или не вызывается метод saveToolStripMenuItem\_Click, а затем только происходит закрытие формы (или не происходит, если пользователь отменил действие |
| richTextBox1\_KeyDown (см. листинг Б.8) | при нажатии клавиши записывает текущую версию текста в стек undoActions |
| undoToolStripMenuItem\_Click (см. листинг Б.9) | отмена последнего действия (если стек Undo пустой, прерывает выполнение метода, иначе, если в форме есть текст, добавляет текущий текст в стек и изменяет текущий текст на верхний объект в стеке Undo |

Окончание таблицы 2.1

|  |  |
| --- | --- |
| redoToolStripMenuItem\_Click (см. листинг Б.10) | повторение последнего действия (если стек Redo пустой, прерывает выполнение метода, иначе, если в форме есть текст, добавляет текущий текст в стек Undo и изменяет текущий текст на верхний объект в стеке Redo |
| richTextBox1\_TextChanged (см. листинг Б.11) | метод, отвечающий за проверка на возможность выполнения поиска по словам и действий Undo/Redo |
| cutToolStripMenuItem\_Click (см. листинг Б.12) | методы, позволяющие выполнить вырезание, копирование и вставку текста |
| copyToolStripMenuItem\_Click |
| pasteToolStripMenuItem\_Click |
| fontToolStripMenuItem\_Click (см. листинг Б.13) | изменяет щрифт текста |
| colorToolStripMenuItem\_Click (см. листинг Б.14) | изменяет цвет текста |
| leftToolStripMenuItem\_Click (см. листинг Б.15) | группа методов, выполняющих изменение выравнивания выделенного текста |
| rightToolStripMenuItem\_Click |
| centerToolStripMenuItem\_Click |
| toolStripButton1\_Click (см. листинг Б.16) | группа методов, выполняющих изменение стилей выделенного текста |
| toolStripButton2\_Click |
| toolStripButton3\_Click |
| и toolStripButton4\_Click |
| richTextBox1\_Click (см. листинг Б.17) | метод, отвечающий за определение состояния кнопок изменения стиля |
| findToolStripMenuItem\_Click (см. листинг Б.18) | метод, вызывающий вспомогательную форму Find |
| FindSubStr (см. листинг Б.19) | метод, выполняющий поиск заданного слова и возвращающий индекс первого символа первого вхождения или -1 (если совпадений найдено не было) |
| Finding (см. листинг Б.20) | метод, возвращающий начальную позицию поиска для следующего поиска совпадения |

### 1.2.3 Класс Find

Используется для обработки событий формы Find.

Состоит из инициализатора объекта класса, переменной start, хранящей позицию начала поиска и обработчиков событий кнопок формы Find\_Further (вызов метода для поиска следующего нахождения слова в тексте) и Cancel (закрытие формы).

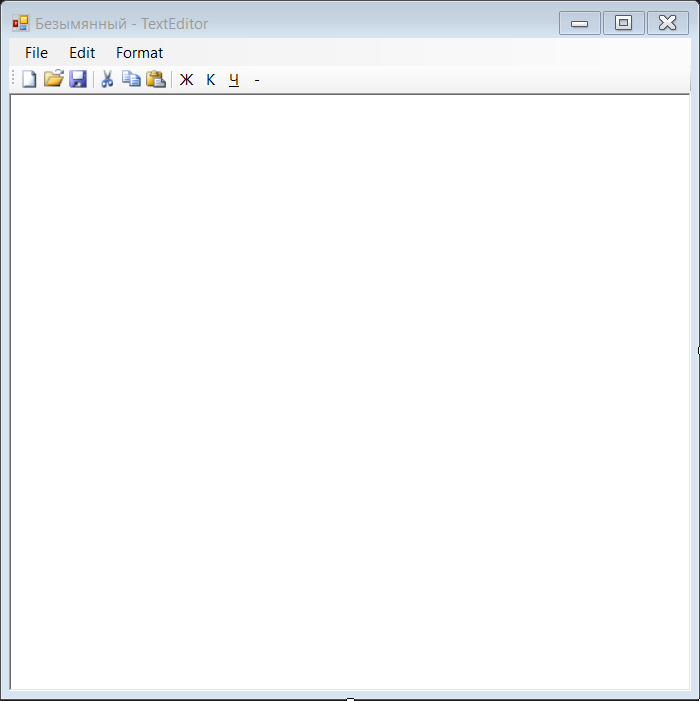
**1.3 Формы**

Для реализации взаимодействия программы с пользователем были использованы 2 формы: Main\_Form и Find.

**1.3.1 Форма Main\_Form**

Main\_Form является главной формой и вызывается при запуске программы.

Основное назначение – обеспечение возможности ввода и форматирования текста пользователем с возможностью последующего сохранения.

Выглядит форма следующим образом (рис. 1.1):

**Рисунок 1.1** – Главная форма приложения

Примечание ‑ Источник: собственная разработка.

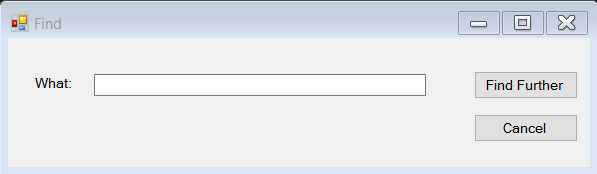
Первая строка формы – строка заголовка (содержит название текущего открытого документа). Вторая – строка меню (содержит основные группы команд для работы с текстом и текстовыми файлами). Ниже расположена панель инструментов**,** где в виде пиктограмм представлены наиболее часто используемые команды из меню. Также форма содержит рабочую область для ввода текста.

**1.3.2 Форма Find**

Find является вспомогательной формой и вызывается при нажатии на клавишу Find пункта меню Edit.

Назначение – обеспечение возможности поиска необходимого символа, слова или фразы.

Выглядит форма следующим образом (рис. 1.2):



**Рисунок 1.2** – Окно поиска

Примечание ‑ Источник: собственная разработка.

Первая строка формы – строка заголовка (содержит название формы). Ниже расположено поле для ввода текста, который необходимо найти, а также кнопки Find Further и Cancel. Первая служит для запуска поиска первого совпадения, вторая – для закрытия формы.

# **2 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

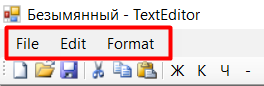
## 2.1 Главная форма

Программа “TextEditor” предназначена для выполнения различных записей по усмотрению пользователя, например, краткое описание файлов в различных папках, создания заметок, редактирование и сохранение небольшого фрагмента текста и др.

Каждая запись может быть скопирована, сохранена и др.

Вызов программы производится с помощью исполняемого файла “TextEditor.exe”.

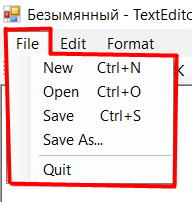
Строка меню редактора содержит пункты File, Edit, Format (рис. 2.1).



**Рисунок 2.1** – Строка меню

Примечание ‑ Источник: собственная разработка.

Пункт File разворачивается на меню New, Open, Save, Save As, и Quit (рис. 2.2).

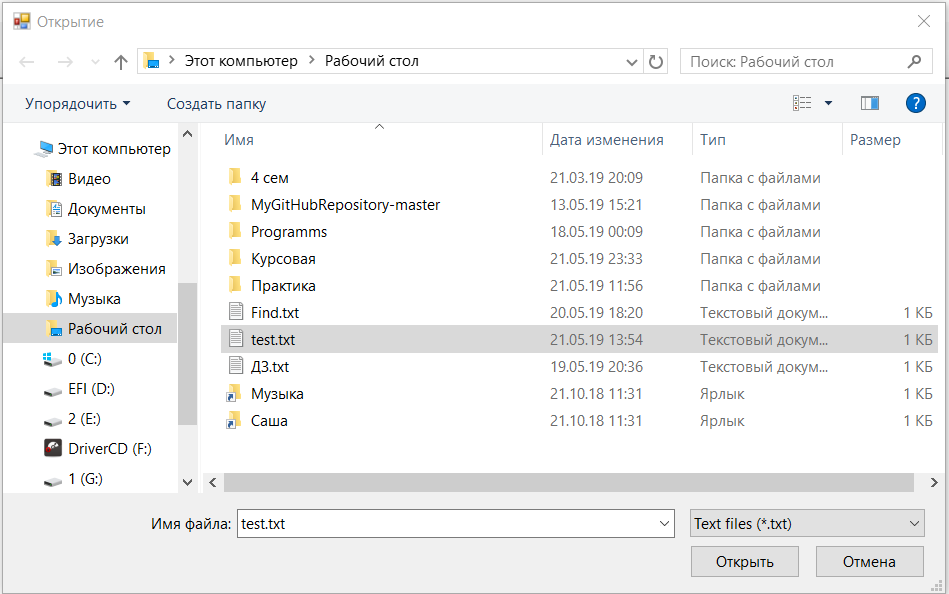


**Рисунок 2.2** – Пункт меню File

Примечание ‑ Источник: собственная разработка.

Команда New выводит на экран новый чистый документ.

Open раскрывает окно открытия документа (рис. 2.3), через которое можно найти и открыть любой документ на любом диске в любой папке.



**Рисунок 2.3** – Окна диалога открытия файла

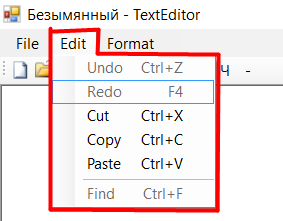
Примечание ‑ Источник: собственная разработка.

Save – записывает документ в память компьютера под прежним именем.

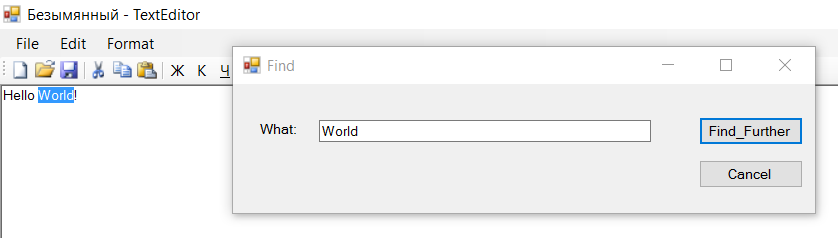
Save As – открывает окно сохранения документа, где указывается его имя, тип, диск и папка, в которую этот документ нужно записать.

Команда Quit закрывает текстовый редактор.

Пункт меню Edit (рис. 2.4) содержи команды Undo(отмена последней операции), Redo(повторное выполнение последней операции), Cut(вырезает фрагмент текста, т.е. переписывает его в буферную память с удалением с прежнего места), Copy(копирование фрагмента текста в буферную память, при этом на прежнем месте он сохраняется), Paste(вставка текста из буфера на место, где установлен курсор), Find (поиск введенного в новую форму слова).

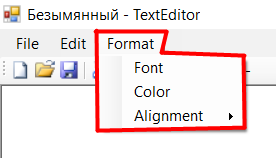
**Рисунок 2.4** – Пункт меню Edit

Примечание ‑ Источник: собственная разработка.



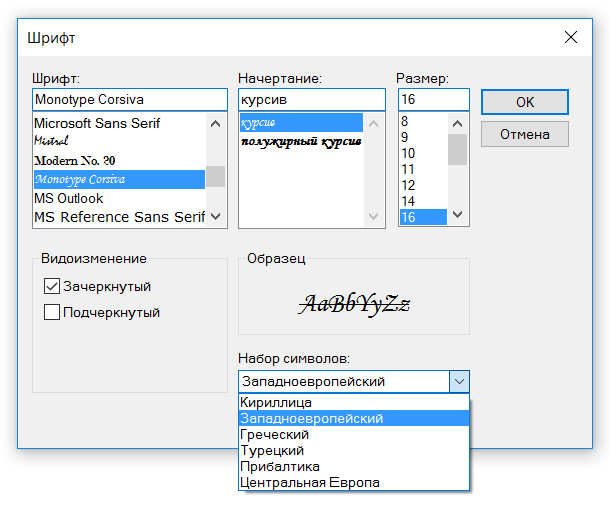
**Рисунок 2.5** – Окно поиска

Примечание ‑ Источник: собственная разработка.

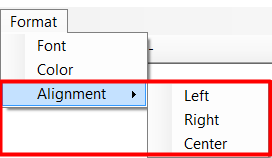
Пункт Format (рис. 2.6) содержит пункты Font, Color и Alignment(Left, Right, Center).

**Рисунок 2.6** – Пункт меню Format

Примечание ‑ Источник: собственная разработка.

Font – типоразмеры шрифтов (рис. 2.7).

**Рисунок 2.7** – Изменение шрифта

Alignment (Left, Right, Center) – выравнивание текста по левому, правому краю и по центру (рис. 2.8).

**Рисунок 2.8** – Изменение выравнивания

Примечание ‑ Источник: собственная разработка.

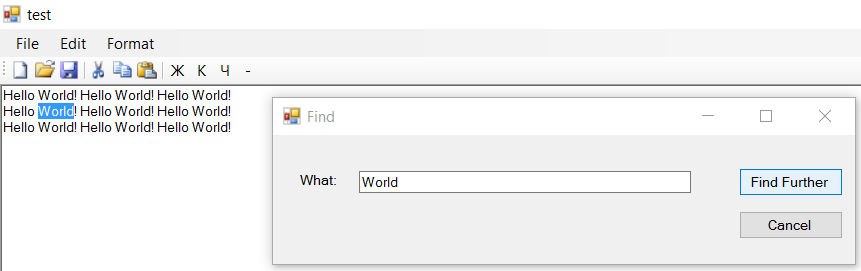
Также реализована панель инструментов (рис. 2.9) с быстрым доступом ко всем наиболее часто используемым функциям: New(Создать), Open(Открыть), Save(Сохранить), Cut(Вырезать), Copy(Копировать), Paste(Вставить), Bold(Жирный), Italic(Курсивный), UnderLine(Подчеркнутый), StriceOut(Зачеркнутый).



**Рисунок 2.9** – Панель инструментов

Примечание ‑ Источник: собственная разработка.

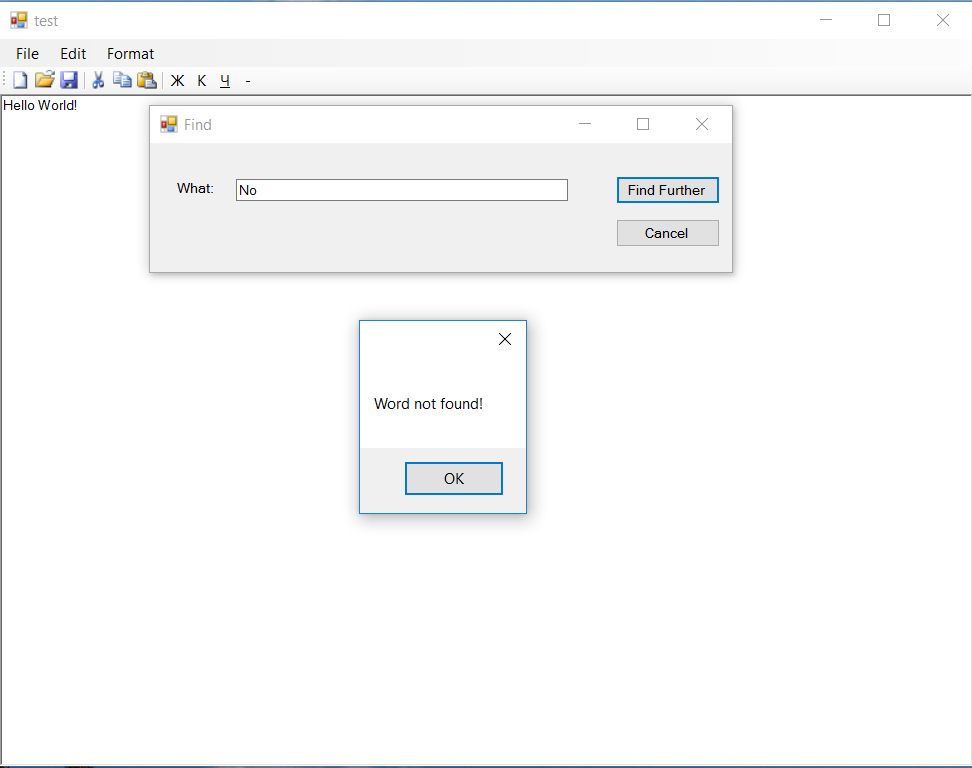
## 2.2 Вспомогательная форма

Вспомогательная форма Find служит для поиска совпадений в тексте.

**Рисунок 2.10** – Поиск слова World

Примечание ‑ Источник: собственная разработка.

Для поиска нужного слова или фразы необходимо ввести его в поле формы и нажать кнопку Find Further (рис. 2.10).



**Рисунок 2.11** – Поиск слова No

Примечание ‑ Источник: собственная разработка.

При первом нажатии на кнопку программа подсветит первое найденное совпадение в тексте, при повторном нажатии следующее найденное совпадение и т.д.

При отсутствии совпадений программа выведет соответствующее сообщение (рис. 2.11).

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы было создано работоспособное приложение TextEditor. Были изучены теоретические сведения и закреплены ранее полученные знание, в частности применение Windows Forms для работы с файлами. Также полученные знания были практически применены в ходе выполнения данной работы.

Для разработки был использован язык программирования C# на платформе .Net и среда программирования Visual Studio 2017 Community Edition.

В ходе написания пояснительной записки был проведен анализ задачи, были определены основные требования к программе, а также описана структура программы и её основные возможности.

Программа TextEditor служит как текстовый процессор, предоставляя базовые возможности обработки текста. Она может быть использована как для решения простейших задач по работе с текстом, так и более сложных работ с форматированием. Программа предназначена как для повседневного, домашнего использования, так и на предприятии или офисе.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. www.docs.microsoft.com/ru-ru – информационно-обучающий сайт [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> – Дата доступа: 06.04.2019
2. www.mycsharp.ru – уроки программирования [Электронный ресурс] – Режим доступа: http://mycsharp.ru – Дата доступа: 14.04.2019
3. www.metanit.com – сборник руководств и обучающих материалов по программированию [Электронный ресурс] – Режим доступа: https://metanit.com – Дата доступа: 15.04.2019
4. www.programmer-lib.ru/csharp – обучение программированию [Электронный ресурс] – Режим доступа: http://programmer-lib.ru/csharp – Дата доступа: 08.05.2019

# **ПРИЛОЖЕНИЯ**

**ПРИЛОЖЕНИЕ А**

**Класс Program**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TextEditor

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Main\_Form());

}

}

}

**Приложение Б**

**Класс Main\_Form**

private DialogResult Proverka() // Запрос на сохранение изменений в документе

{

DialogResult result = DialogResult.Ignore;

if (richTextBox1.Text.Trim() != "" && change == 1)

{

result = MessageBox.Show(

"Do you want to save changes to your document?",

"TextEditor",

MessageBoxButtons.YesNoCancel);

}

return result;

}

**Листинг Б.1** – Метод Proverka

Примечание ‑ Источник: собственная разработка на основе [2].

private void newToolStripMenuItem\_Click(object sender, EventArgs e) // Создание нового файла

{

DialogResult result = Proverka();

if (result == DialogResult.Yes)

{

saveToolStripMenuItem\_Click(sender, e);

}

else if (result == DialogResult.No)

{

richTextBox1.Clear();

path = "";//Удаление текста из элем. упр. текстовым полем

filename = "Безымянный";

this.Text = filename;

}

else if (richTextBox1.Text.Trim() != "" && change == 0)

{

richTextBox1.Clear();

path = "";//Удаление текста из элем. упр. текстовым полем

filename = "Безымянный";

this.Text = filename;

}

}

**Листинг Б.2** – Метод newToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [3].

private void openToolStripMenuItem\_Click(object sender, EventArgs e) // Открытие файла

{

DialogResult result = Proverka();

if (result == DialogResult.Yes)

{

saveToolStripMenuItem\_Click(sender, e);

}

else if (result == DialogResult.Cancel)

{

return;

}

else

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

{

return;

}

// получаем выбранный файл

path = openFileDialog1.FileName;

try

{

filename = Path.GetFileNameWithoutExtension(path);

// читаем файл

try

{

if (Path.GetExtension(path) == ".rtf")

richTextBox1.LoadFile(path);

else

File.ReadAllText(path, Encoding.Default);

}

catch (ArgumentException)

{

MessageBox.Show("This file is not RTF format!" + "\n" + "Reading in txt format!");

richTextBox1.Text = File.ReadAllText(path);

}

}

catch (DirectoryNotFoundException)

{

MessageBox.Show("File not found");

}

change = 0;

this.Text = filename;

}

}

**Листинг Б.3** – Метод openToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [3].

private void saveAsToolStripMenuItem\_Click(object sender, EventArgs e) // Сохранение с созданием нового файла или перезаписью существующего

{

if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

change = 0;

// получаем выбранный файл

path = saveFileDialog1.FileName;

try

{

filename = Path.GetFileNameWithoutExtension(path);

// читаем файл

}

catch (DirectoryNotFoundException)

{

MessageBox.Show("File not found");

}

// сохраняем текст в файл

try

{

if (Path.GetExtension(path) == ".rtf")

richTextBox1.LoadFile(path);

else

File.ReadAllText(path, Encoding.Default);

}

catch (IOException ex)

{

MessageBox.Show(ex.Message);

}

catch (ArgumentException ex)

{

MessageBox.Show(ex.Message);

}

this.Text = filename;

}

**Листинг Б.4** – Метод saveAsToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [3].

private void saveToolStripMenuItem\_Click(object sender, EventArgs e) // Сохранение изменений в текущий документ

{

if (path != "")

{

try

{

File.WriteAllText(path, richTextBox1.Rtf, Encoding.Default);

}

catch (ArgumentNullException ex)

{

MessageBox.Show(ex.Message);

}

change = 0;

}

else

{

saveAsToolStripMenuItem\_Click(sender, e);

}

}

**Листинг Б.5** – Метод saveToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [3].

private void Main\_Form\_FormClosing(object sender, FormClosingEventArgs e) // Закрытие формы по нажатию на крестик

{

if (change != 0)

{

DialogResult result = Proverka();

if (result == DialogResult.Yes)

{

saveToolStripMenuItem\_Click(sender, e);

}

else if (result == DialogResult.Cancel)

{

e.Cancel = true;

}

}

}

**Листинг Б.6** – Метод Main\_Form\_FormClosing

Примечание ‑ Источник: собственная разработка на основе [1].

private void quitToolStripMenuItem\_Click(object sender, EventArgs e) // Закрытие формы по нажатию на кнопку в меню

{

DialogResult result = Proverka();

if (result == DialogResult.Yes)

{

saveToolStripMenuItem\_Click(sender, e);

change = 0;

this.Close();

}

else if (result == DialogResult.Cancel){}

else

{

change = 0;

this.Close();

}

}

**Листинг Б.7** – Метод quitToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [1].

private void richTextBox1\_KeyDown(object sender, KeyEventArgs e)

{

// Запись в стек предыдущей версии текста

undoActions.Push(richTextBox1.Text);

}

**Листинг Б.8** – Метод richTextBox1\_KeyDown

Примечание ‑ Источник: собственная разработка на основе [3].

private void undoToolStripMenuItem\_Click(object sender, EventArgs e) // Отмена последнего действия

{

if (undoActions.Count < 1)

{

return;

}

if (redoActions.Count == 0 || richTextBox1.Text.Trim() != "" )

{

redoActions.Push(richTextBox1.Text);

}

richTextBox1.Text = undoActions.Pop();

}

**Листинг Б.9** – Метод undoToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [4].

private void redoToolStripMenuItem\_Click(object sender, EventArgs e) // Повторение последнего действия

{

if (redoActions.Count < 1)

{

return;

}

if (undoActions.Count == 0 || richTextBox1.Text != "")

{

undoActions.Push(richTextBox1.Text);

}

richTextBox1.Text = redoActions.Pop();

}

**Листинг Б.10** – Метод undoToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [4].

private void richTextBox1\_TextChanged(object sender, EventArgs e)

{

if (richTextBox1.Text.Trim() != "")

findToolStripMenuItem.Enabled = true;

else

findToolStripMenuItem.Enabled = false;

change = 1;

// Проверка на возможность выполнения действия Undo/Redo

if (undoActions.Count == 0)

undoToolStripMenuItem.Enabled = false;

else

{

undoToolStripMenuItem.Enabled = true;

}

if (redoActions.Count == 0)

redoToolStripMenuItem.Enabled = false;

else

{

redoToolStripMenuItem.Enabled = true;

}

}

**Листинг Б.11** – Метод richTextBox1\_TextChanged

Примечание ‑ Источник: собственная разработка на основе [3].

private void cutToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (richTextBox1.SelectedText != "")

richTextBox1.Cut(); //Вырезка

}

private void copyToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (richTextBox1.SelectionLength > 0)

richTextBox1.Copy(); // Копирование

}

private void pasteToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (Clipboard.GetDataObject().GetDataPresent(DataFormats.Text) == true)

{

richTextBox1.Paste(); //Вставка

}

}

**Листинг Б.12** – Методы cut/copy/pasteToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [1].

private void fontToolStripMenuItem\_Click(object sender, EventArgs e) //Изменение шрифта текста

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

richTextBox1.SelectionFont = fontDialog1.Font;

}

}

**Листинг Б.13** – Метод fontToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [1].

private void colorToolStripMenuItem\_Click(object sender, EventArgs e) // Изменение цвета текста

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

richTextBox1.SelectionColor = colorDialog1.Color;

}

}

**Листинг Б.14** – Метод colorToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [1].

private void leftToolStripMenuItem\_Click(object sender, EventArgs e) // Изменение выравнивания текста

{

string btn = (sender as ToolStripMenuItem).Text;

switch (btn)

{

case "Left":

richTextBox1.SelectionAlignment = HorizontalAlignment.Left;

break;

case "Right":

richTextBox1.SelectionAlignment = HorizontalAlignment.Right;

break;

case "Center":

richTextBox1.SelectionAlignment = HorizontalAlignment.Center;

break;

}

}

private void rightToolStripMenuItem\_Click(object sender, EventArgs e)

{

leftToolStripMenuItem\_Click(sender, e);

}

private void centerToolStripMenuItem\_Click(object sender, EventArgs e)

{

leftToolStripMenuItem\_Click(sender, e);

}

**Листинг Б.15** – Методы left/right/centerToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [3].

private void toolStripButton1\_Click(object sender, EventArgs e) // Изменение стиля текста

{

Font currentFont = richTextBox1.SelectionFont;

FontStyle newFontStyle = FontStyle.Regular;

if (toolStripButton1.Checked == true) newFontStyle |= FontStyle.Bold;

if (toolStripButton2.Checked == true) newFontStyle |= FontStyle.Italic; ;

if (toolStripButton3.Checked == true) newFontStyle |= FontStyle.Underline;

if (toolStripButton4.Checked == true) newFontStyle |= FontStyle.Strikeout;

richTextBox1.SelectionFont = new Font(currentFont.FontFamily, currentFont.Size, newFontStyle);

}

private void toolStripButton2\_Click(object sender, EventArgs e)

{

toolStripButton1\_Click(sender, e);

}

private void toolStripButton3\_Click(object sender, EventArgs e)

{

toolStripButton1\_Click(sender, e);

}

private void toolStripButton4\_Click(object sender, EventArgs e)

{

toolStripButton1\_Click(sender, e);

}

**Листинг Б.16** – Методы toolStripButton(1,2,3,4)\_Click

Примечание ‑ Источник: собственная разработка на основе [3].

private void richTextBox1\_Click(object sender, EventArgs e) // Определение состояния кнопок изменения стиля

{

Font currentFont = richTextBox1.SelectionFont;

if (currentFont.Bold) toolStripButton1.Checked = true; else toolStripButton1.Checked = false;

if (currentFont.Italic) toolStripButton2.Checked = true; else toolStripButton2.Checked = false;

if (currentFont.Underline) toolStripButton3.Checked = true; else toolStripButton3.Checked = false;

if (currentFont.Strikeout) toolStripButton4.Checked = true; else toolStripButton4.Checked = false;

}

**Листинг Б.17** – Метод richTextBox1\_Click

Примечание ‑ Источник: собственная разработка на основе [3].

private void findToolStripMenuItem\_Click(object sender, EventArgs e)

{

Find f = new Find();

f.Show();

}

**Листинг Б.18** – Метод findToolStripMenuItem\_Click

Примечание ‑ Источник: собственная разработка на основе [3].

public int FindSubStr(string substr, int start) //Поиск слова

{

int returnValue = -1;

if (substr.Length > 0 && start >= 0)

{

int indexToText = richTextBox1.Find(substr, start, RichTextBoxFinds.MatchCase);

if (indexToText >= 0)

{

returnValue = indexToText;

}

}

return returnValue;

}

**Листинг Б.19** – Метод FindSubStr

Примечание ‑ Источник: собственная разработка на основе [1].

public int Finding(string substr, int start)

{

if (FindSubStr(substr, start) != -1)

{

start = FindSubStr(substr, start) + substr.Length;

this.Activate();

}

else

{

MessageBox.Show("Word not found!");

}

return start;

}

**Листинг Б.20** – Метод Finding

Примечание ‑ Источник: собственная разработка на основе [4].

**Приложение В**

**Класс Find**

using System;

using System.Windows.Forms;

namespace TextEditor

{

public partial class Find : Form

{

public Find()

{

InitializeComponent();

}

private int start = 0;

private void Cancel\_Find\_Click(object sender, EventArgs e)

{

this.Close();

}

private void find\_furthr\_Click(object sender, EventArgs e)

{

string substr = textBox1.Text;

if (Application.OpenForms["Main\_Form"] != null)

{

start = (Application.OpenForms["Main\_Form"] as Main\_Form).Finding(substr, start);

}

}

}

}