Ход работы

1. Описание функционала программы
   1. Создание нового документа

Алгоритм создания нового документа:

1. Открытие диалогового окна *SaveFileDialog*.

if (fod.ShowDialog() != DialogResult.Cancel) {

1. Используя полученный с помощью *SaveFileDialog* путь к файлу, создаем файл, очищаем поле *RichTextBox* и меняем свойство *Enabled* последнего на *true*.

filename = fod.FileName;

File.Create(filename);

mainTextBox.Clear();

mainTextBox.Enabled = true;

}

* 1. Открытие существующего документа

1. Открытие диалогового окна *OpenFileDialog*.

if (fod.ShowDialog() != DialogResult.Cancel) {

1. Используя полученный путь к файлу, вызываем метод *LoadFile* компонента *RichTextBox* и меняем свойство *Enabled* на *true*.

filename = fod.FileName;

mainTextBox.LoadFile(filename);

mainTextBox.Enabled = true;

}

* 1. Сохранение документа

1. Открытие диалогового окна *SaveFileDialog*.

if (sfd.ShowDialog() != DialogResult.Cancel) {

1. Используя полученный путь к файлу, вызываем метод *SaveFile* компонента *RichTextBox*.

filename = sfd.FileName;

mainTextBox.SaveFile(filename);

}

* 1. Задание стиля текста (жирный, курсив, подчеркнутый)

1. Получаем шрифт выделенного текста:

Font currentFont = mainTextBox.SelectionFont;

1. Если стиль текста не жирный, то создаем стиль *FontStyle.Bold*, если нет, то *FontStyle.Regular*:

if (!currentFont.Bold)

style = FontStyle.Bold;

else

style = FontStyle.Regular;

1. Задаем выделенному тексту новый класс шрифта, включая в него свойства выделенного текста со сменой стиля на полученный на шаге 2:

mainTextBox.SelectionFont = new Font(currentFont.FontFamily, currentFont.Size, style);

Для остальных стилей алгоритм повторяется, с изменением класса стиля.

* 1. Задание размера шрифта текста

1. Получаем значение *value* компонента *NumericUpDown*.

float size = (float)this.FontSizeUpDown.Value;

1. Задаем полученное значение свойству *SelectionFont* компонента *RichTextBox*:

var currentFont = mainTextBox.SelectionFont;

mainTextBox.SelectionFont = new Font(currentFont.FontFamily, size, currentFont.Style);

* 1. Сохранение актуальными значений компонентов при изменении выделения текста

1. В конструкторе компоненту *RichTextBox* событию *SelectionChanged* добавляем обработчик:

mainTextBox.SelectionChanged += selectionChanged;

1. В обработчике получаем значение размера выделенного текста и стиля. Полученные значения синхронизируем с соответствующими значениями компонентов, отвечающих за эти свойства:

var font = mainTextBox.SelectionFont;

FontSizeUpDown.Value = (decimal)font.Size;

if (font.Bold)

boldBtn.Font = new Font(boldBtn.Font.FontFamily, boldBtn.Font.Size, font.Style);

if (font.Italic)

italicBtn.Font = new Font(italicBtn.Font.FontFamily, italicBtn.Font.Size, font.Style);

if (font.Underline)

underlineBtn.Font = new Font(underlineBtn.Font.FontFamily, underlineBtn.Font.Size, font.Style);

* 1. Задание цвета текста

1. Открытие диалогового окна *ColorDialog*.

ColorDialog cd = new ColorDialog();

if (cd.ShowDialog() != DialogResult.Cancel)

1. Используя полученное значение цвета, задаем цвет выделенного текста:

mainTextBox.SelectionColor = cd.Color;

* 1. Печать документа

1. Открываем диалоговое окно печати, и используя полученные параметры печати, запускаем печать:

pd = new PrintDialog();

pdoc = new PrintDocument();

pdoc.BeginPrint += OnBeginPrint;

pdoc.PrintPage += OnPrintPage;

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

{

pdoc.Print();

}

1. Определяем обработчик события начала печати:

char[] param = { '\n' };

if (pd.PrinterSettings.PrintRange == PrintRange.Selection)

{

lines = mainTextBox.SelectedText.Split(param);

}

else

{

lines = mainTextBox.Text.Split(param);

}

int i = 0;

char[] trimParam = { '\r' };

foreach (string s in lines)

{

lines[i++] = s.TrimEnd(trimParam);

}



