**Лабораторная работа № 10   
«Текстовый редактор»**

Прежде чем начинать разработку текстового редактора, рассмотрим необходимый для этого элемент управления RichTextBox (). Он предназначен для ввода и отображения форматированного текста. Этот элемент управления поддерживает форматирование символов и абзацев для любого фрагмента текста, позволяет сохранять и открывать файлы в формате *RTF*, а также вставлять в текст картинки, ссылки и другие объекты.

Некоторые свойства элемента управления RichTextBox приведены в табл. 10.1.

*Таблица 10.1*

**Некоторые свойства элемента управления RichTextBox**

|  |  |
| --- | --- |
| Свойство | Описание |
| *CanRedo, CanUndo* | Логические свойства, определяющие, были ли в элементе управления произведены такие действия, которые можно было бы применить заново / отменить с помощью метода *Redo*()/*Undo*() |
| *RedoActionName,*  *UndoActionName* | Возвращает имя действия, которое можно повторно применить/отменить |
| *Rtf* | Получает или задает текст, содержащийся в элементе управления, в формате *RTF* |
| *HideSelection* | Получает или задает значение, показывающее, остается ли выделенный текст таковым, когда фокус ввода на форме переходит с данного элемента управления на другой |
| *Lines* | Возвращает массив строк, содержащихся в элементе управления |
| *SelectedRtf* | Получает или задает выделенный в настоящий момент текст в формате *RTF* |
| *SelectedText* | Получает или задает выделенный в элементе управления текст |
| *SelectionAlignment* | Получает или задает выравнивание, которое будет применено к текущему выделению или положению курсора |
| *SelectionBackColor* | Получает или задает цвет выделенного текста (заливки), который будет применен к текущему выделению или положению курсора |
| *SelectionColor* | Получает или задает цвет текста, который будет применен к текущему выделению или положению курсора |
| *SelectionFont* | Получает или задает шрифт текста, который будет применен к текущему выделению или положению курсора |
| *SelectionLength* | Получает или задает число символов, выделенных в элементе управления |
| *ShowSelectionMargin* | Получает или задает значение, указывающее, отображается ли поле выделения в элементе управления |

В табл. 10.2 перечислены методы элемента управления RichTextBox.

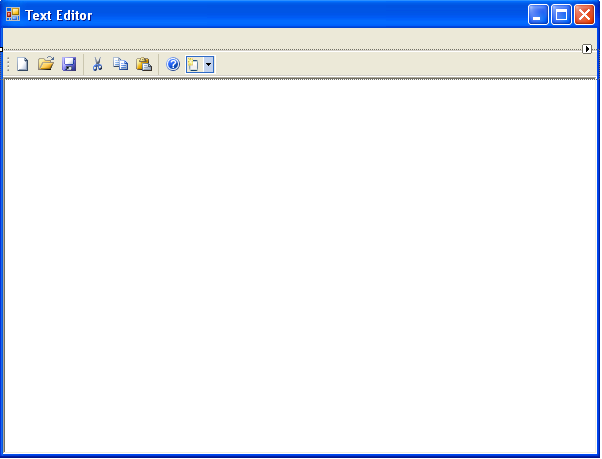
*Таблица 10.2*

**Методы элемента управления RichTextBox**

|  |  |
| --- | --- |
| Метод | Описание |
| *Clear*() | Удаляет весь текст из элемента управления |
| *Copy*() | Копирует текущее выделение из текстового поля в буфер обмена |
| *Cut*() | Перемещает текущее выделение из текстового поля в буфер обмена |
| *Focus*() | Задает фокус ввода элемента управления |
| *LoadFile*() | Загружает файл в формате *RTF* или стандартный текстовый файл |
| *Redo*() | Заново применяет выполненное последним действие, результаты которого были отменены в элементе управления |
| *SaveFile*() | Сохраняет содержимое элемента управления в *RTF*‑файл |
| *Undo* | Отменяет последнюю операцию редактирования |

При изменении пользователем выделенного текста или положения курсора в тексте генерируется событие SelectionChanged.

**Задание.** Напишите приложение «Текстовый редактор», позволяющее пользователю вводить, редактировать и форматировать текст, открывать текст из заданного файла и сохранять текст в файл.



*Рис. 10.1.* Форма из упражнения 10.1.

**Упражнение 10.1.** Напишите приложение «Текстовый редактор», позволяющее пользователю открывать и редактировать файлы в формате *RTF*.

**Решение**

Создайте новый проект *TextEditor* в решении *Lab10*.

**1‑й этап.** Создадим визуальный интерфейс приложения (рис. 10.1).

Положите на форму элементы управления MenuStrip, ToolStrip и RichTextBox. Свойства компонентов установите следующим образом:

|  |  |  |
| --- | --- | --- |
| Компонент | Свойство | Значение |
| *form1* | *Name* | *MainForm* |
| *Text* | *Текстовый редактор* |
| *menuStrip1* | *Name* | *MainMenu* |
| *RenderMode* | *System* |
| *toolStrip1* | *Name* | *ToolStrip* |
| *RenderMode* | *System* |
| *richTextBox1* | *Name* | *TextEdt* |
| *Dock* | *Fill* |

Элемент управления ToolStrip предназначен для создания панелей инструментов, предоставляющих пользователю возможность быстрого доступа к наиболее часто используемым функциям приложения.

Панель инструментов ToolStrip может включать несколько видов различных элементов, они перечислены в табл. 10.3.

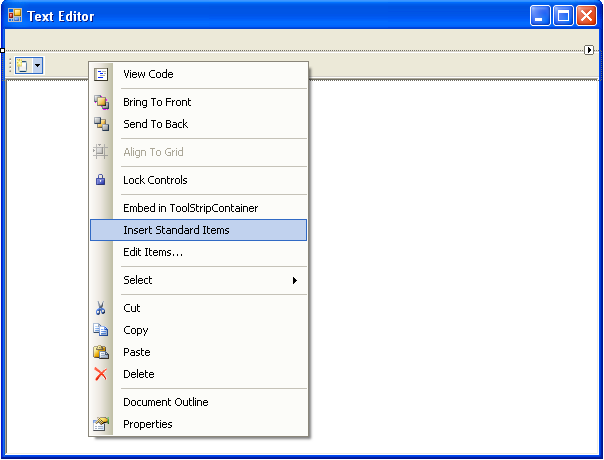
*Таблица 10.3*

**Элементы панели инструментов ToolStrip**

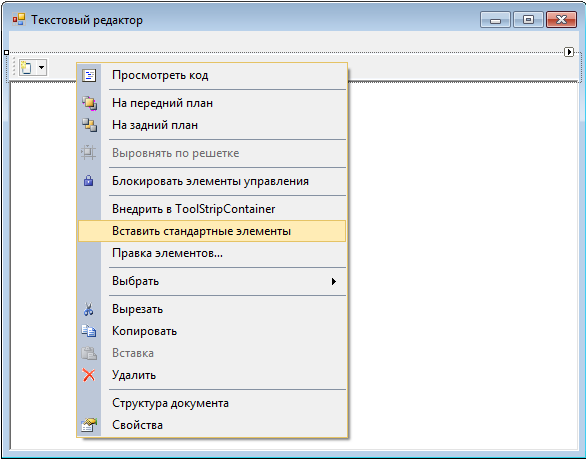
|  |  |
| --- | --- |
| Элемент | Описание |
| *ToolStripButton* | Кнопка с картинкой и/или текстом |
| *ToolStripLabel* | Надпись |
| *ToolStripSplitButton* | Кнопка с выпадающим меню, меню отображается при щелчке по кнопке развертывания в виде треугольника в правой части элемента управления |
| *ToolStrpDropDownButton* | Кнопка с выпадающим меню, меню отображается при щелчке по любой части элемента управления |
| *ToolStripComboBox* | Поле со списком |
| *ToolStripProgressBar* | Индикатор протекания процесса |
| *ToolStripTextBox* | Текстовое поле |
| *ToolStripSeparator* | Горизонтальный или вертикальный разделитель элементов |

Выберите элемент управления ToolStrip, выполните щелчок правой кнопкой мыши по свободной части элемента и в появившемся контекстном меню выберите пункт *Вставить стандартные элементы* (рис. 10.2)*.* В результате на панели инструментов появятся кнопки, соответствующие стандартным функциональным возможностям большинства приложений. Удалите кнопку вывода документа на печать.

Поместите на форму элемент OpenFileDialog.



*Рис. 10.2.* Добавление стандартных элементов на панель ToolStrip



**2‑й этап**. Создание программного кода.

Создайте обработчик события Click кнопки открытия файла *open­Tool­StripButton* и поместите туда следующие операторы:

//устанавливаем фильтр

openFileDialog1.Filter = "RTF files|\*.rtf";

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

{

//загружаем текст из файла

TextEdt.LoadFile(openFileDialog1.FileName);

//отображаем полное имя файла в заголовке формы

Text = "Text Editor ‑ " + openFileDialog1.FileName;

}

Поясним код. Здесь мы отображаем диалоговое окно открытия файла с помощью метода *ShowDialog*(). Если пользователь нажал кнопку *Открыть*, загружаем текст из выбранного файла в элемент RichTextBox, используя метод *LoadFile*(). Полное имя открытого файла отображаем в заголовке формы.

Запустите приложение, убедитесь в правильности его работы.

**Задания для самостоятельного выполнения**

* 1. Напишите обработчик события нажатия на кнопку «Сохранить». Если текущий текстовый документ не связан с конкретным файлом, то вывести стандартное диалоговое окно сохранения файла, иначе просто сохранить документ в заданный файл.
  2. Напишите обработчик события нажатия на кнопку создания нового текстового документа. При этом если в текущий текстовый документ были внесены изменения, то выводится диалоговое окно, спрашивающее у пользователя о необходимости сохранения документа.
  3. Напишите обработчик события нажатия на кнопку вызова окна помощи  на панели инструментов. При этом выводится диалоговое окно, содержащее информацию о создателе программы.

**Упражнение 10.2.** Дополним текстовый редактор возможностью форматирования символов.

**Решение**

**1‑й этап.** Доработаем визуальный интерфейс приложения.

Добавьте на панель инструментов ToolStrip еще три элемента ToolStripButton и элемент ToolStripComboBox. Установите их свойства следующим образом:

|  |  |  |
| --- | --- | --- |
| Компонент | Свойство | Значение |
| *toolStripButton1* | *Name* | *BoldToolStripButton* |
| *CheckOnClick* | *True* |
| *DisplayStyle* | *Text* |
| *Text* | Ж |
| *Font.Bold* | *True* |
| *toolStripButton2* | *Name* | *ItalicToolStripButton* |
| *CheckOnClick* | *True* |
| *DisplayStyle* | *Text* |
| *Text* | К |
| *Font.Italic* | *True* |
| *toolStripButton3* | *Name* | *UnderlineToolStripButton* |
| *CheckOnClick* | *True* |
| *DisplayStyle* | *Text* |
| *Text* | Ч |
| *Font.Underline* | *True* |
| *toolStripComboBox1* | *Name* | *SizeTSComboBox* |
| *AutoSize* | *False* |
| *Items* | Задайте коллекцию строк, соответствующих возможным значениям размера шрифта в пикселях |
| *Text* | 12 |
| *Size.Width* | 40 |

**2‑й этап**. Создание программного кода.

Создайте обработчик события Click кнопки изменения стиля начертания символов на полужирный *BoldToolStripButton* и поместите туда следующие операторы:

if (BoldToolStripButton.Checked)

//добавляем полужирный стиль начертания

TextEdt.SelectionFont =

new Font(TextEdt.SelectionFont,

TextEdt.SelectionFont.Style | FontStyle.Bold);

else //отменяем полужирный стиль начертания

TextEdt.SelectionFont =

new Font(TextEdt.SelectionFont,

TextEdt.SelectionFont.Style & ~FontStyle.Bold);

Создайте обработчик события SelectionChanged для элемента управления RichTextBox. Напомним, что это событие возникает при перемещении курсора в тексте. В этом случае кнопки отображения стиля начертания должны отражать стиль начертания текущего символа. Добавьте в обработчик события следующие операторы:

if (TextEdt.SelectionFont == null) return;

BoldToolStripButton.Checked = TextEdt.SelectionFont.Bold;

Запустите приложение, удостоверьтесь в том, что стиль начертания символов изменяется на полужирный.

Для поля со списком, предоставляющего пользователю возможность выбрать размер шрифта, *SizeTSComboBox* создайте обработчик события DropDownClosed, возникающего после закрытия раскрывающегося списка. Поместите в метод следующий программный код:

TextEdt.SelectionFont =

new Font(TextEdt.SelectionFont.FontFamily, Convert.ToInt16(SizeTSComboBox.SelectedItem.ToString()));

TextEdt.Focus();

Обратите внимание, здесь переопределяется значение свойства *SelectionFont*, задающего шрифт выделенного текста. Новый экземпляр класса *Font* формируется на основе текущего значения шрифта, с указанием нового значения размера шрифта. После этого фокус ввода возвращается элементу *TextEdt*.

**Задания для самостоятельного выполнения**

* 1. Обеспечьте возможность задания пользователем с клавиатуры такого значения размера шрифта, которое не указано в раскрывающемся списке.
  2. Напишите обработчик события нажатия на кнопки изменения стиля начертания на курсив и подчеркнутый. Удостоверьтесь в том, что при перемещении по тексту кнопки отображения начертания символов правильно отображают стиль начертания текущего символа.
  3. Добавьте на панель инструментов кнопки изменения цвета текста и цвета выделенного текста (заливки) и обеспечьте их функциональность. Для выбора цвета используйте элемент управления ColorDialog.
  4. Добавьте на панель инструментов кнопку вызова стандартного диалогового окна изменения шрифта, обеспечьте ее функциональность. Используйте элемент управления FontDialog.

**Упражнение 10.3.** Дополним текстовый редактор возможностью форматирования абзацев.

**Решение**

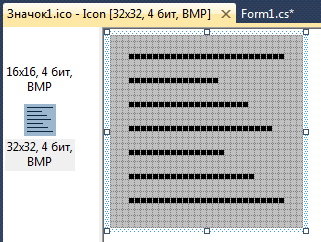
**1‑й этап.** Доработаем визуальный интерфейс приложения.

Добавьте на панель инструментов ToolStrip еще три элемента ToolStripButton и установите их свойства следующим образом:

|  |  |  |
| --- | --- | --- |
| Компонент | Свойство | Значение |
| *toolStripButton1* | *Name* | *CenterToolStripButton* |
| *CheckOnClick* | *True* |
| *toolStripButton2* | *Name* | *RightToolStripButton* |
| *CheckOnClick* | *True* |
| *toolStripButton3* | *Name* | *LeftToolStripButton* |
| *CheckOnClick* | *True* |

Создайте файлы, содержащие иконки для кнопок форматирования абзаца. Для этого в среде *Visual Studio* создайте новый фал *Файл – Создать – Файл* и в появившемся диалоговом окне создания нового файла выберите *Файл значка*.

Задайте размер файла иконки *32×32, 4bit, BMP* (рис. 10.3). Последовательно создайте и сохраните три картинки, соответствующие способу выравнивания абзаца по центру, по левому и по правому краю.



размер 32×32

*Рис. 10.3.* Определение размера файла иконки

Для того чтобы отобразить созданные иконки на кнопках панели инструментов, воспользуемся компонентом ImageList, позволяющим хранить коллекцию изображений, которые могут использоваться другими элементами управления. Добавьте его на форму. Используя свойство *Images*, загрузите созданные файлы в компонент. Далее в конструкторе формы добавьте следующие строки кода:

toolStrip1.ImageList = imageList1;

CenterToolStripButton.ImageIndex = 0;

LeftToolStripButton.ImageIndex = 1;

RightToolStripButton.ImageIndex = 2;

Здесь мы связали панель инструментов *toolStrip1* с компонентом *imageList1* посредством свойства *ImageList*. Далее для каждой из кнопок на панели инструментов задали значение свойства *ImageIndex*, соответствующее номеру отображаемого изображения в компоненте ImageList.

Удостоверьтесь в правильности соотнесения изображений функциональности каждой из кнопок.

**2‑й этап**. Создание программного кода.

Создайте обработчик события Click кнопки выравнивания абзаца по центру *CenterToolStripButton* и поместите туда следующие операторы:

if (TextEdt.SelectionAlignment == HorizontalAlignment.Center)

TextEdt.SelectionAlignment = HorizontalAlignment.Left;

else TextEdt.SelectionAlignment = HorizontalAlignment.Center;

Запустите приложение, удостоверьтесь в его работоспособности.

**Задания для самостоятельного выполнения**

* 1. Обеспечьте функциональность оставшихся кнопок выравнивания абзаца. Удостоверьтесь в том, что при перемещении по тексту кнопки выравнивания правильно отображают способ выравнивания текущего абзаца.
  2. Создайте и отобразите иконки для кнопок изменения цвета текста и цвета выделенного текста (заливки).
  3. Обеспечьте функциональность кнопок редактирования текста: вырезать, копировать и вставить.
  4. Разработайте главное меню, реализующее все функциональные возможности приложения, обеспечьте его работоспособность.

***Баллы:*** полностью реализованное приложение оценивается в 6 баллов.

### 