# Создание текстового редактора

**Часть I. Проектирование интерфейса главного окна**

## Разработайте приложение с графическим интерфейсом пользователя, позволяющее создавать и просматривать текстовые документы.

1. Создайте новое приложение Windows Forms с именем **BlokNote**
2. Переименуйте форму Form1 в **blokNoteForm**.

## Разместите на форме главное меню – компонент **MenuStrip**.

1. Спроектируйте пункты главного меню и присвойте им имена согласно таблице 1.

Таблица 1. Подписи и имена пунктов главного меню BlokNot

|  |  |  |
| --- | --- | --- |
| **Пункт** | **Подпункт** | **Имя (св-во Name)** |
| Файл |  | tsmiFile |
|  | Создать | tsmiNew |
| Открыть | tsmiOpen |
| Сохранить | tsmiSave |
| Выход | tsmiExit |
| Формат |  | tsmiFormat |
| Шрифт | tsmiFont |
| Заливка | tsmiFill |
| Выравнивание | tsmiAlignment |
| Маркер | tsmiMarker |
| Редактирование |  | tsmiEdit |
|  | Отменить действие | tsmiUndo |
|  | Повторить действие | tsmiRedo |
|  | Копировать | tsmiCopy |
|  | Вырезать | tsmiCut |
|  | Вставить | tsmiPaste |
|  | Удалить | tsmiDelete |
| Справка |  | tsmiHelp |

## Разместите на форме строку состояния – компонент **StatusStrip**.

1. В свойстве **Items** компонента **statusStrip1** добавьте элемент **StatusLabel**, в котором будет отображаться информация, сохранен документ или нет.
2. Добавьте на форму текстовое поле **RichTextBox**.

## Установите стиль выравнивания Заполнение для текстового поля (свойство **Dock** должно быть равно **Fill**).

1. В классе формы **blokNoteForm** объявите переменные, содержащие информацию о текущем файле:

private bool changed = false; //были ли изменения в файле

private string fileName = null; //имя файла на диске

## В конструкторе формы выведите информацию в заголовок окна и в строку состояния:

public BlokNoteForm()

{

InitializeComponent();

this.Text = "Новый документ - BlokNote"; //заголовок окна – имя файла и программы

statusStrip1.Items[0].Text = "Не сохранено"; //стока состояния - сохранение

…

## При изменении содержимого текстового поля необходимо установить флаг изменения:

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

{

changed = true;

statusStrip1.Items[0].Text = "Не сохранено";

}

**Часть II. Сохранение, открытие и создание текстовых документов**

1. Разместите на форме невизуальный компонент **SaveFileDialog**, предоставляющий функционал диалога сохранения.
2. Настройте фильтр компонента **saveFileDialog1**, позволяющий отображать файлы определенного формата. Для этого в свойстве **Filter** разместите строку:

Простой текст|\*.txt|RTF-текст|\*.rtf|Все файлы|\*.\*

1. Добавьте в класс формы метод сохранения файла **«Сохранить как»**:

private void saveFileAs() //метод "сохранить как"

{

//если пользователь выбрал имя файла и нажал кнопку "ОК" в диалоге сохранения if (saveFileDialog1.ShowDialog() == DialogResult.OK && saveFileDialog1.FileName != null)

{

fileName = saveFileDialog1.FileName; //запоминаем имя файла

saveFile(); //вызываем метод "сохранить"

}

}

## Добавьте в класс формы метод сохранения файла **«Сохранить»**:

private void saveFile() //метод "сохранить"

{

try

{

}

//сохраняем содержимое текстового поля под выбранным именем

richTextBox1.SaveFile(fileName, RichTextBoxStreamType.RichText);

catch (Exception ex)

{

MessageBox.Show("Не удалось сохранить файл! Информация об ошибке: "+ ex.Message,"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

1. Создайте обработчик события **Click** для пункта главного меню **Сохранить**:

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

{

if (fileName == null) //если имя файла пусто

{

}

else

{

}

saveFileAs(); //вызываем метод "сохранить как"

saveFile(); //вызываем метод "сохранить"

//выносим в заголовок окна имя файла без пути

this.Text = Path.GetFileName(fileName) + " - BlokNot"; statusStrip1.Items[0].Text = "Сохранено"; //статус сохранения в строку состояния changed = false; //снимаем флаг изменений

}

## Подключите пространство имен **System.IO**. В нем находится класс Path, выполняющий операции с путями к файлам или каталогам.

1. Во избежание потери несохраенной информации при открытии или создании нового документа, добавьте в класс формы метод **closeFileQuery**, возвращающий результат «истина», если файл можно закрыть и «ложь» в противном случае:

private bool closeFileQuery() //запрос на закрытие файла

{

if (changed) //если файл изменен

{

//открываем диалоговое окно с предупреждением

DialogResult result = MessageBox.Show("Сохранить изменения?", "Предупреждение", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);

if (result == DialogResult.Yes) //если нажата кнопка "да", сохраняем

{

if (fileName == null) //если имя файла пустое

{

}

else

}

else

saveFileAs(); //вызываем метод "сохранить как" saveFile(); //вызываем метод "сохранить"

if (result == DialogResult.Cancel) //если нажата кнопка "отмена"

{

return false; //запрещаем закрытие файла

}

}

return true; //разрешаем закрытие файла

}

## Разместите на форме невизуальный компонент **OpenFileDialog**, предоставляющий функционал диалога открытия.

1. Выполните настройку фильтра аналогично п.2.
2. Добавьте в класс формы метод, обеспечивающий открытие файла:

private void openFile() //метод открытия файла

{

if (openFileDialog1.ShowDialog() == DialogResult.OK && openFileDialog1.FileName != null)

{

try

{

//загружаем в текстовое поле содержимое файла в формате rtf richTextBox1.LoadFile(openFileDialog1.FileName, RichTextBoxStreamType.RichText);

}

catch (System.FormatException) //если формат файла отличен от rtf

{

//загружаем в текстовое поле содержимое файла в формате txt richTextBox1.LoadFile(openFileDialog1.FileName, RichTextBoxStreamType.PlainText);

}

changed = false;

fileName = openFileDialog1.FileName;

}

}

## В обработчике события **Click** пункта меню **Открыть** вызовите метод открытия:

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

{

if (closeFileQuery()) //если разрешено закрыть предыдущий документ

{

openFile(); //вызов метода открытия файла

this.Text = Path.GetFileName(fileName) + " - BlokNot"; statusStrip1.Items[0].Text = "Сохранено";

}

}

## Реализуйте метод **newFile**, вызываемый при необходимости создания нового документа:

private void newFile()

{

richTextBox1.Clear(); //очистка текстового поля

changed = false; fileName = null;

}

## Оформите программный код пункта меню **Создать**:

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

{

if (closeFileQuery())//если разрешено закрыть предыдущий документ

{

newFile(); //вызов метода создания

this.Text = "Новый документ - BlokNote"; statusStrip1.Items[0].Text = "Не сохранено";

}

}

## Запустите программу на выполнение и проверьте правильность ее работы.

**Часть III. Форматирование текста в RichTextBox**

1. Перетащите на форму с панели элементов компонент **FontDialog**, выполняющий функции диалога выбора параметров шрифта.
2. В свойстве **ShowColor** компонента **fontDialog1** установите значение **True**. Это свойство указывает, показывать панель выбора цвета шрифта или нет.
3. Для подпункта меню **Шрифт** создайте обработчик события **Click**:

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

{

//если пользователь нажал кнопку ОК в диалоге

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

{

//изменяем шрифт выделенного фрагмента на указанный в диалоге

richTextBox1.SelectionFont = fontDialog1.Font;

//изменяем цвет выделенного фрагмента на указанный в диалоге

richTextBox1.SelectionColor = fontDialog1.Color;

}

}

1. Создайте следующие подпункты для пункта главного меню **Выравнивание**: **По левому краю – tsmiToLeft**, **По центру – tsmiToCenter** и **По правому краю – tsmiToRight**.

## При нажатии на подпункт **По левому краю** выполните соответствующее выравнивание текста:

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

{

richTextBox1.SelectionAlignment = System.Windows.Forms.HorizontalAlignment.Left;

}

## **Самостоятельно** определите программный код для остальных типов выравнивания текста.

1. Программный код подпункта меню Маркер должен устанавливать или удалять маркер списка:

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

{

//инвертируем свойство, отвечающее за наличие маркера

richTextBox1.SelectionBullet = !richTextBox1.SelectionBullet;

}

## Запустите программу и проверьте правильность работы функций форматирования текста.

**Часть IV. Методы редактирования текста**

1. Пункты меню **Отменить действие** и **Повторить действие** должны быть доступны только, если список действий пользователя не пуст. Эту проверку осуществите при выборе пункта меню **Редактирование**:

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

{

//доступность подпунктов меню определяется возможностью отмены (повтора)

//последней операции

tsmiUndo.Enabled = richTextBox1.CanUndo; tsmiRedo.Enabled = richTextBox1.CanRedo;

}

## Удаление выделенного фрагмента текста можно производить нажатием клавиши Delete или с помощью пункта меню **Удалить**:

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

{

//присваиваем выделенному фрагменту текста пустое значение

richTextBox1.SelectedText = "";

}

## Запустите программу на выполнение и проверьте новые функции.

**Часть V. Задания для самостоятельной работы**

**Задание 1:** Разместите на форме компонент **ColorDialog** и реализуйте программный код пункта меню Заливка, изменяющий цвет фона выделенного фрагмента текста.

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

{

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

return;

// установка цвета

richTextBox1.SelectionBackColor = colorDialog1.Color;

}

**Задание 2:** Напишите программный код для пунктов меню **Копировать**, **Вырезать** и **Вставить**.

## **Задание 3:** Обеспечьте закрытие формы по нажатию на «Выход»

**Дополнительно (по желанию):**

**Задание 4:** Изучите функционал компонента **ImageList** и дополните пункты главного меню иконками.