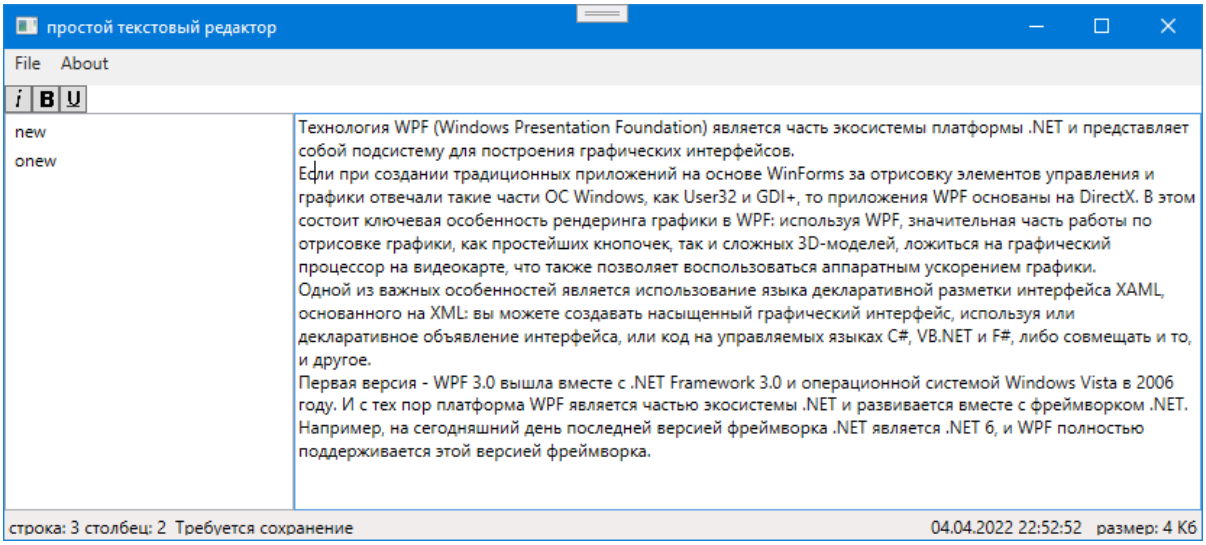
ПЗ 14. Создание проекта с использованием различных управляющих компонентов.

Задание: разработайте проект WPF простого текстового редактора:

В statusBar расположенном в нижней части окна необходимо разместить информацию о:

- Позиции курсора в тексте
- Статусе файла
    - Файл имеет статус «требуется сохранение» при создании нового файла, внесении информации, до того момента, как данные сбросятся в файл на диске
    - Файл имеет статус «сохранено» при успешном сбросе информации в файл на диске
- Дате создания/изменения файла
- Размере файла в Кб



Теоретическая часть:

Работа со StatusBar

StatusBar удерживает любое содержимое (упаковывая его неявным образом в объекты StatusBarItem) и переопределяет используемые по умолчанию стили некоторых элементов для обеспечения более подходящей визуализации. Однако поддержки для перегруппирования элементов за счет перетаскивания и создания дополнительного меню элемент управления StatusBar не включает. В основном его применяют для отображения текста и графических индикаторов (изредка — панели хода выполнения).

Обычно элемент управления StatusBar компонует свои дочерние элементы слева направо с помощью объекта StackPanel с горизонтальной ориентацией. Однако в приложениях элементы строки состояния довольно часто имеют пропорциональные размеры или прикреплены к правой стороне строки состояния. Реализовать такой дизайн можно за счет указания того, что в строке состояния должна использоваться другая панель, с помощью свойства ItemsPanelTemplate.

Один из способов получить пропорциональные или правильно выровненные элементы — применить в качестве контейнера компоновки элемент управления Grid. Единственной сложностью является то, что для установки свойства GridColumn подходящим образом дочерний элемент нужно обязательно упаковать в объект StatusBarItem. Ниже приведен пример, в котором один элемент TextBlock размещается в левой части StatusBar, а другой — в правой:

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition></RowDefinition>
        <RowDefinition Height="auto"></RowDefinition>
    </Grid.RowDefinitions>
    <StatusBar Grid.Row="1">
        <StatusBar.ItemsPanel>
            <ItemsPanelTemplate>
                <Grid>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*"></ColumnDefinition>
                        <ColumnDefinition Width="auto"></ColumnDefinition>
                    </Grid.ColumnDefinitions>
                </Grid>
            </ItemsPanelTemplate>
        </StatusBar.ItemsPanel>
        <TextBlock>Текст слева</TextBlock>
        <StatusBarItem Grid.Column="1">
            <TextBlock>Текст справа</TextBlock>
        </StatusBarItem>
    </StatusBar>
</Grid>
```

Здесь становится очевидным одно из главных преимуществ WPF другие элементы управления могут извлекать пользу из базовой модели компоновки без необходимости ее воссоздания.

Получение позиции курсора:

Для получения значения строки, в котором находится в текущий момент курсор можно воспользоваться готовым функционалом - методом **GetLineIndexFromCharacterIndex**(), который в качестве параметра принимает указанный индекс символа (в данном случае textField.CaretIndex) и возвращает индекс нужной строки (отсчет с нуля)
Для получения значения столбца можно вычесть из значения текущей позиции каретки значение позиции начала строки, передав в метод **GetLineIndexFromCharacterIndex** значение нужной строки (помните о нумерации строк с нуля)

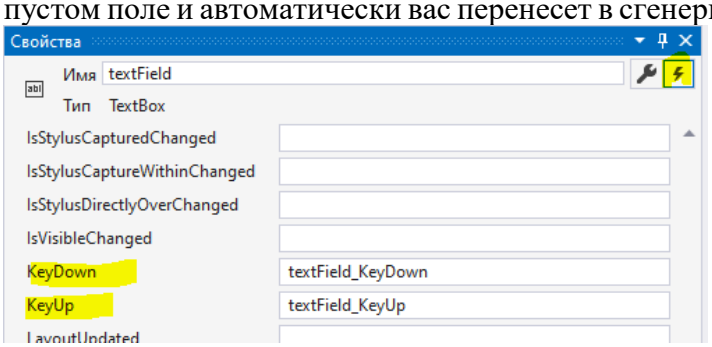
Например:

```
        private void UpdateCursorPosition()
        {
            int row = textField.GetLineIndexFromCharacterIndex(textField.CaretIndex);
            int col = textField.CaretIndex - textField.GetLineIndexFromCharacterIndex(row);

            cursorPosition.Text = $"строка: {row + 1} столбец: {col + 1}";
        }
```

Здесь реализован отдельный метод, который обновляет информацию о позиции курсора. Данный метод можно вызывать при любом событии изменении позиции курсора, например SelectionChanged, KeyDown, KeyUp

Для создания события выберите нужный элемент (к примеру TextBox) нажмите F4 для появления панели свойств и перейдите в список событий. Напротив нужного события щелкните дважды в пустом поле и автоматически вас перенесет в сгенерированное событие:



Для получения информации о времени создания и обновления файла и его размере стоит воспользоваться функционалом классов File или FileInfo (1 семестр)