Задание 1. Обработка текстовых файлов

Технологии: WPF, File, BackgroundWorker/SynchronizationContext, LINQ/PLINQ

Задачи:

1. Реализовать обработку текстовых файлов с помощью технологий LINQ и PLINQ.

Пользователь выбирает папку с текстовыми файлами. Необходимо найти:

- распределение слов по длине (для построения гистограммы);

- 5-самых часто встречающихся слов с учетом фильтра по длине;

- 5 самых длинных слов;

2. Разработать оконный интерфейс с помощью WPF для запуска вычислений и вывода результатов с возможностью досрочной отмены вычислений и отображением прогресса в процессе обработки. В конце работы отображается время последовательной обработки и параллельной.

3. Обеспечить асинхронность вычислений с помощью объектов BackgroundWorker (или SynchonizationContext, Task).

Рекомендации

Для получения списка файлов в виде перечислимой структуры можно воспользоваться конструкцией: var files = Directory.EnumerateFiles("E:\\Books", "*.txt");

Поиск часто встречающихся слов с помощью LINQ-технологии выполняется как последовательность операторов Select, SelectMany, Where, GroupBy, OrderBy, Take.

Для замера времени обработки можно использовать объект Stopwatch из пространства имен System.Diagnostics. Методы Start и Stop управляют секундомером. Свойство `ElapsedMilliseconds` содержит общее время работы секундомера в миллисекундах.

Для параллельного выполнения LINQ-запроса необходимо использовать преобразование входной последовательности:

```
files.AsParallel().Select (.. )
```

При реализации Windows-интерфейса необходимо учитывать следующие моменты: обработка данных не должна блокировать интерфейс пользователя, поэтому ее необходимо запускать в отдельном потоке. При этом обновление интерфейса пользователя (изменение надписей, цвета элементов и др.) должно происходить только в главном потоке, связанном с Windows-приложением.

Один из простых способов реализации обработки в приложении с оконным интерфейсом является использование компонента **BackgroundWorker**. Компонент запускает длительную обработку в фоновом потоке и позволяет назначить обработчик для события **RunWorkerCompleted**, который будет вызван после завершения фоновой задачи в главном

потоке. Дополнительный обработчик события **ProgressChanged** позволяет отслеживать прогресс выполнения фоновой задачи.

Для построения гистограммы распределения количества слов по длине можно использовать объект Chart из Windows Forms. В WPF приложении нужно подключить сборки System.Windows.Forms, System.Windows.Forms.DataVisualizaion и WindowsFormsIntegration. В разметке XAML необходимо подключить пространства имен:

```
xmlns:wf="clr-namespace:System.Windows.Forms;assembly=System.Windows.Forms"

xmlns:wfcharting="clr-namespace:System.Windows.Forms.DataVisualization.Charting;  …
```

После этого элемент Chart можно использовать в XAML разметке следующим образом:

```
<WindowsFormsHost>
       <wfcharting:Chart x:Name="LengthChart">
       </wfcharting:Chart>
</WindowsFormsHost>
```

Для отображения гистограммы необходимо создать область построения (ChartArea), ряд данных (Series) и передать данные для отображения.

```
// Создаем область построения ChartArea
chart.ChartAreas.Add(new ChartArea("defaultArea"));

// Добавляем ряд данных "defaultArea"
chart.Series.Add(new Series("Series1"));
chart.Series["Series1"].ChartArea = "defaultArea";
chart.Series["Series1"].ChartType = SeriesChartType.Column;

// добавляем данные
int[] axisXData = new int[] {1, 2, 3, 4, 5};
int[] axisYData = new int[] {56, 34, 15, 11, 5};
chart.Series["Series1"].Points.DataBindXY(axisXData, axisYData);
```

Для вывода списков слов (самых часто встречающихся и самых длинных) можно использовать элемент ListBox. Заполнение списка можно выполнить через свойство ItemsSource:

```
top10List.ItemsSource = new string[] { "Hello: 567", "Goodbye: 367", "Welcome: 331"};
```