**华 南 农 业 大 学 数 学 与 信 息（软 件） 学 院**

《.net 软件开发课程实验》成绩单

开设时间：2019学年第一学期

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 小组成员、组内分工、工作量比例、各成员个人成绩 | | | | | | | | | | | |
| **学号** | 201725110105  201716030117 | **姓名** | 邓泽骏  刘柏麟 | **分工** | 邓:界面交互设计  刘:底层接口实现 | | | **工作量比例** | 邓：50%  刘：50% | **成绩** |  |
| 实 验 题 目 | 富文本编辑器实现 | | | | | | | | | | |
| 自 我 评 价 | 本次课程设计中，开发组基于C#语言中的特性按照要求开发一个富文本编辑器，本软件通过wpf(windows presentation foundation)简单的实现了图形界面，用户可以通过使用窗体的功能栏与软件进行交互，完成新建文件，打开文件，关闭文件等指令。图形界面中还能修改字体颜色，大小，字体种类，还有查询和替换功能。  小组成员在开发过程中，编程水平都有着一定程度的提高。在合作期间，小组成员通过Git工具及GitHub网站管理项目。  小组成员在开发过程中遇到了一些问题，如基于.xaml文件完成wpf界面设计及触发函数的使用。  未来，小组成员将会吸取本次开发过程中的经验，努力应用更多的所学知识，使编写的程序功能更加完备、用户界面风格更加友好，提高程序的应用价值。 | | | | | | | | | | |
| 教 师 评 语 | 评价指标：   * 题目内容和要求完成情况 优 □ 良 □ 中 □ 差 □ * 对算法原理的理解程度 优 □ 良 □ 中 □ 差 □ * 程序设计水平 优 □ 良 □ 中 □ 差 □ * 程序运行效果及正确性 优 □ 良 □ 中 □ 差 □ * 课程设计报告结构清晰 优 □ 良 □ 中 □ 差 □ * 报告中总结和分析详尽 优 □ 良 □ 中 □ 差 □ | | | | | | | | | | |
| **成绩** |  | | | | | **教师签名** |  | | | | |

**目录**

[1 需求分析 2](#_Toc26620107)

[2 概要设计 3](#_Toc26620108)

[2.1 架构选用 3](#_Toc26620109)

[2.2 模块设计 3](#_Toc26620110)

[2.3 数据结构设计 3](#_Toc26620111)

[2.4 方法设计 3](#_Toc26620112)

[2.5 图形用户界面设计 4](#_Toc26620113)

[3 详细设计（本人主要完成文件操作相关功能） 4](#_Toc26620114)

[3.1 界面设计 4](#_Toc26620115)

[3.2 代码交互设计 6](#_Toc26620116)

[3.3 图形用户界面设计 9](#_Toc26620117)

[4 调试分析 9](#_Toc26620118)

[4.1 调试方法 9](#_Toc26620119)

[5 总结与展望 10](#_Toc26620120)

[6参考资料 10](#_Toc26620121)

# 1 需**求分**析

（1）类似于文本编辑器对于文件的，新建，打开，保存，关闭等操作

（2）文本编辑器编辑操作，剪切，复制，粘贴等操作

（3）查找，替换等文本整体操作

（4）字体，颜色，字体大小等修改文字属性操作

（5）文件拖拽打开

（6）MDI多文档

# 概要设计

## 2.1 架构选用

本程序使用C#语言编写，其既可以进行类似于C语言的过程化程序设计，还可以以委托等特性完成的面对对象程序设计，对于Windows系统的支持很好。

程序使用wpf(Windows Presentation Foundation)框架进行GUI(Graphic User Interface)设计。界面使用.xaml文件，使用类似于html文件的标记语言进行渲染，功能实现使用C#，是一个很好的MFC框架。

## 2.2 模块设计

基于需求分析的结果，程序设计有如下一些方法：

（1）trigger方法

与界面交互的触发器方法

（2）function方法

一般作为功能实现的主方法

（3）initialize方法

作为对界面的补充方法

## 2.3 数据结构设计

实验中主要是对于功能的实现，基于数据结构的实现几乎没有。主要以功能拼接实现。

## 2.4 方法设计

本程序底层实现的接口如下所述。

* + MainWindows类
    - Initialize初始化方法 RichTextEditorSample ()
    - 创建文件 NewFile(string fileName)
    - 打开文件 Open\_Executed() / Open\_File(string fileName)
    - 关闭文件 Close\_Executed(object sender,ExecutedRoutedEventArgs e)
    - 保存文件 Save\_Executed(object sender, ExecutedRoutedEventArgs e)
    - 修改字体 cmbFontFamily\_SelectionChanged(object sender, SelectionChangedEventArgs e)
    - 修改字体大小 cmbFontSize\_TextChanged(object sender, TextChangedEventArgs e)
    - 标记文件是否被修改 rtbEditor\_TextChanged(object sender, TextChangedEventArgs e)
    - 修改字体颜色 cmbColour\_SelectionChanged(object sender, SelectionChangedEventArgs e)
    - 文件拖拽打开 rtbEditor\_Drop(object sender, DragEventArgs e) / rtbEditor\_DragOver(object sender, DragEventArgs e)
    - 修改选项卡指定编辑器 tabControl\_SelectionChanged(object sender, SelectionChangedEventArgs e)
    - 清空搜索栏 ClearSearchResult()
    - 搜索功能 Search(TextPointer position, string word) / SearchButton\_Click(object sender, RoutedEventArgs e)
    - 查找下一个功能 ReplaceNext(string keyword) / ReplaceNextButton\_Click(object sender, RoutedEventArgs e)
    - 查找全部 ReplaceAll(string keyword) / ReplaceAllButton\_Click(object sender, RoutedEventArgs e)
    - 工具栏初始化 ToolBar\_Loaded(object sender, RoutedEventArgs e)
    - 关闭窗口 Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

## 2.5 图形用户界面设计

程序包含两个主要模块，主要为图像设计的.xaml和后台的.cs文件

主界面能够显示文本编辑器，工具栏，选项卡，等界面

文本编辑界面进行常用的文本编辑操作

程序另有使用对话框来增强交互性。

# 详细设计（本人主要完成文件操作相关功能）

## 3.1 界面设计

程序使用VS(visual studio)内置的 designer来完成主要的界面设计工作，通过对.xaml文件编辑调整界面细节设计

<Window x:Class="WpfTutorialSamples.Rich\_text\_controls.RichTextEditorSample"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="RichTextEditorSample" Height="395.275" Width="813.544" Closing="Window\_Closing">

<Window.CommandBindings>

<CommandBinding Command="ApplicationCommands.Open" Executed="Open\_Executed" />

<CommandBinding Command="ApplicationCommands.Save" Executed="Save\_Executed" />

<CommandBinding Command="ApplicationCommands.Close" Executed="Close\_Executed"/>

</Window.CommandBindings>

<DockPanel>

<ToolBar DockPanel.Dock="Top" Background="#FFEEF5FD" Loaded="ToolBar\_Loaded">

<Button Command="ApplicationCommands.Open">

<Image Source="Images\disk.png" Width="16" Height="16" />

</Button>

<Button Command="ApplicationCommands.Save">

<Image Source="Images\folder.png" Width="16" Height="16" />

</Button>

<Button Command="ApplicationCommands.Close">

<Image Source="Images\Close.jpg" Width="16" Height="16" />

</Button>

<Separator />

<ToggleButton Command="EditingCommands.ToggleBold" Name="btnBold">

<Image Source="Images\Bold.png" Width="16" Height="16" />

</ToggleButton>

<ToggleButton Command="EditingCommands.ToggleItalic" Name="btnItalic">

<Image Source="Images\Italic.jpg" Width="16" Height="16" />

</ToggleButton>

<ToggleButton Command="EditingCommands.ToggleUnderline" Name="btnUnderline">

<Image Source="Images\Underline.jpg" Width="16" Height="16" />

</ToggleButton>

<Separator />

<ComboBox Name="cmbFontFamily" Width="150" SelectionChanged="cmbFontFamily\_SelectionChanged" />

<ComboBox Name="cmbFontSize" Width="50" IsEditable="True" TextBoxBase.TextChanged="cmbFontSize\_TextChanged" />

<ComboBox Name="cmbColour" Width="50" Margin="0,2,0,0" VerticalAlignment="Top" SelectionChanged="cmbColour\_SelectionChanged">

<ComboBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal">

<Rectangle Fill="{Binding Name}" Width="16" Height="16"/>

<TextBlock Text="{Binding Name}"/>

</StackPanel>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<Button Name="btnSearch"/>

<TextBox x:Name="SearchTextBox" Height="22" Margin="0" TextWrapping="Wrap" VerticalAlignment="Top" Width="90" ToolTip="查找"/>

<Separator Height="22" Margin="0" VerticalAlignment="Top" Width="2"/>

<TextBox x:Name="ReplaceTextBox" Height="22" Margin="0" TextWrapping="Wrap" VerticalAlignment="Top" Width="90" ToolTip="替换"/>

<Button x:Name="SearchButton" Content="查找" Height="22" VerticalAlignment="Top" Width="33" Click="SearchButton\_Click"/>

<Button x:Name="ReplaceNextButton" Content="替换" Height="22" VerticalAlignment="Top" Width="31" Click="ReplaceNextButton\_Click"/>

<Button x:Name="ReplaceAllButton" Content="替换所有" Height="22" VerticalAlignment="Top" Width="55" Click="ReplaceAllButton\_Click"/>

</ToolBar>

<TabControl Name="tabControl" Margin="0,-0.333" SelectionChanged="tabControl\_SelectionChanged">

<TabItem Header="File1">

<RichTextBox Name="rtbEditor1" SelectionChanged="rtbEditor\_SelectionChanged" TextChanged="rtbEditor\_TextChanged" Drop="rtbEditor\_Drop" DragOver="rtbEditor\_DragOver" />

</TabItem>

</TabControl>

</DockPanel>

</Window>

## 3.2 代码交互设计

本程序大部分的代码用于功能的拼接和与界面的交互。

//Close File trigger

private void Close\_Executed(object sender,ExecutedRoutedEventArgs e)

{

if (flagChange)

{

MessageBoxResult messageBox = MessageBox.Show("This file has changed, Save?", "Save tip", MessageBoxButton.YesNoCancel); //Show the 'Save tip' Message

switch (messageBox)

{

case MessageBoxResult.Yes:

{

Save\_Executed(null, null);

break;

}

case MessageBoxResult.No:

{

break;

}

case MessageBoxResult.Cancel:

{

return;

}

}

}

rtbEditor.Document.Blocks.Clear(); //Clear the Editor Context

fileName = null;

// Modify the TabController

int length = tabControl.Items.Count;

if (length == 1)

{

(tabControl.SelectedItem as TabItem).Header = "File1";

}else

{

int itemIndex = tabControl.SelectedIndex;

if (itemIndex != 1)

{

tabControl.SelectedIndex = itemIndex - 1;

}

tabControl.Items.RemoveAt(itemIndex);

fileName = (tabControl.SelectedItem as TabItem).Header as string;

}

rtbEditor = ((tabControl.SelectedItem as TabItem).Content) as RichTextBox;

flagChange = false;

}

程序段3.2.1 文件关闭

伪程序段3.2.2展示了程序保存文件时操作

//Save trigger

private void Save\_Executed(object sender, ExecutedRoutedEventArgs e)

{

FileStream fileStream = null;

TextRange range = new TextRange(rtbEditor.Document.ContentStart, rtbEditor.Document.ContentEnd);

// If the file if created by program

if (fileName != null)

{

fileStream = new FileStream(fileName, FileMode.Create);

range.Save(fileStream, dataFormat);

fileStream.Close();

flagChange = false;

(tabControl.SelectedItem as TabItem).Header = Path.GetFileName(fileName);

return;

}

// If the file opened

SaveFileDialog dlg = new SaveFileDialog();

dlg.Filter = "Rich Text Format (\*.rtf)|\*.rtf|Plain Text (\*.txt)|\*.txt";

if (dlg.ShowDialog() == true)

{

fileStream = new FileStream(dlg.FileName, FileMode.Create);

(tabControl.SelectedItem as TabItem).Header = Path.GetFileName(dlg.FileName);

dataFormat = dlg.FilterIndex == 2 ? DataFormats.Text : DataFormats.Rtf;

range.Save(fileStream, dataFormat);

fileName = dlg.FileName;

}

if (fileStream != null)

{

fileStream.Close();

}

flagChange = false;

}

程序段3.2.2 文件保存

## 3.3 图形用户界面设计

程序的图形界面通过VS的designer设计实现，本次程序界面大部分是对于.xaml文件修改实现完成，而非通过拖动窗口修改，这主要是考虑对于界面修改的便捷性和对于新方法学习的热情。

主界面分为2个功能区，上侧功能区通过 Separator 分隔符划分细小区域。左上方三个功能键，之后是字体修改工具栏，最后是查找替换功能区。

文本编辑器在下侧的空白区域，对于文本的显示和修改等都在下方大的编辑区域显示，下方通过RichTextBox控件实现允许用户自由编辑内容。当关闭时，如果仍然有未保存的更改，系统提示用户保存修改。

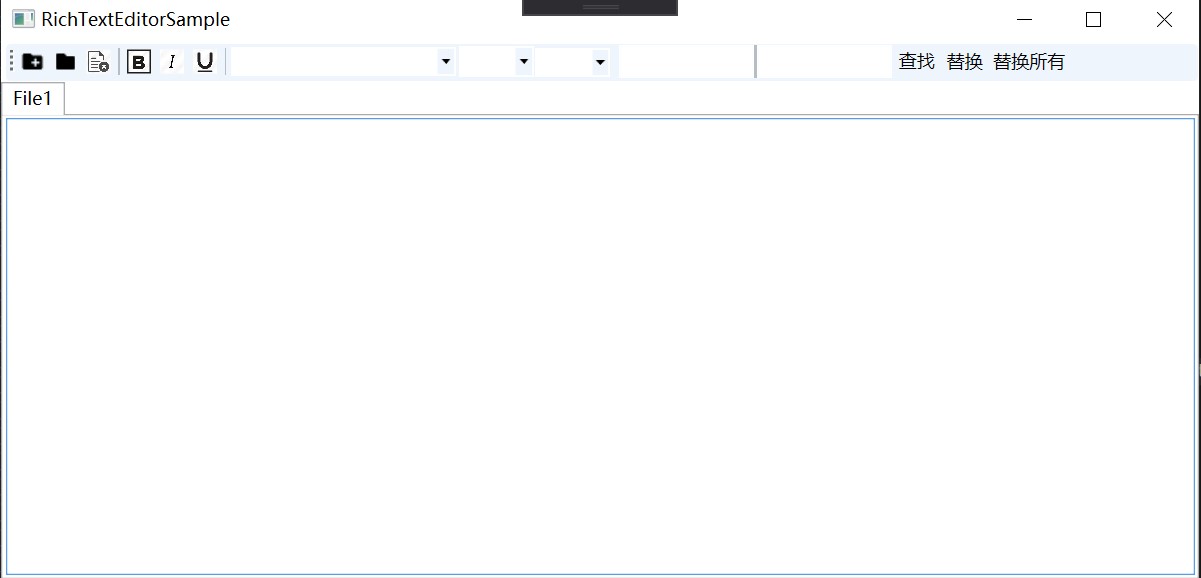


图3.2 文本编辑器界面

# 调试分析

## 4.1 调试方法

程序开发过程中，使用先模块单元测试，再整体调试的方法。函数在控制台观察到输出符合预期 输出后，推送到稳定版本的分支，并通过给定数据流，观察输出来作黑盒测试。函数内部调试主要使用断点调试法、单步调试法和中间结果输出法，即使用Console和窗体显示输出中间结果。

# 5 总结与展望

本次课程设计中，小组实现了一个基于wpf和C#的文本编辑器。有基本的图形界面，用户可以使用工具栏和右键菜单等方式直观地与程序交互，完成打开文件，新建文件，关闭文件等指令。

本程序诚然存在一些待改进之处，如根据光标显示当前字体属性等功能。

小组成员在开发过程中，对C#语言都有着一定的程度的进步，对于C#语言的熟悉程度加深。

# 6参考资料

[1]WPF Tutorial <https://www.wpf-tutorial.com/zh/62/rich-text%E6%8E%A7%E4%BB%B6/%E6%80%8E%E4%B9%88%E5%81%9A%E4%B8%80%E4%B8%AA%E5%AF%8C%E6%96%87%E6%9C%AC%E7%BC%96%E8%BE%91%E5%99%A8/> 2017