Лабораторна робота № 1

Тема роботи: Інтерфейс віконного додатку.

Мета роботи: дослідження основних підходів та технологій реалізації інтерфейсу універсального додатку Windows.

Обладнання: ПК, Visual Studio не нижче v.2010 (.Net Framework 4.5)

1. Завдання на лабораторну роботу

Засобами інтегрованого середовища розробки Visual Studio побудувати текстовий редактор. Потрібно реалізувати наступні функції:

1. Створення, відкриття та збереження файлів
2. Копіювання, переміщення тексту.
3. Виділення тексту та знищення тексту.
4. Відміну останньої дії.
5. Стрічку, меню та меню швидкого доступу для управління функціональними компонентами додатку.
6. Персоналізувати параметри проекту (назву додатку, оригінальну піктограму, вікно AboutBox тощо).
7. Виконати індивідуальне завдання за варіантом.
8. Додатково виконати завдання на самостійну роботу.

Індивідуальні завдання

1. Додати Control та діалогове вікно що дозволяє зміну вирівнювання виділеного фрагмента тексту.
2. Додати Control та діалогове вікно що дозволяє зміну відступу від лівого краю першого рядка виділеного фрагмента тексту.
3. Додати пункт Control та діалогове вікно що дозволяє встановити маркери до виділеного фрагмента тексту.
4. Додати пункт меню та кнопку що дозволяє встановити нумерований список до виділеного фрагмента тексту.
5. Додати пункт меню та кнопку що дозволяє зміну відступу від лівого краю виділеного фрагмента тексту.
6. Додати пункт меню та кнопку що дозволяє зміну відступу від правого краю виділеного фрагмента тексту.
7. Додати пункт меню та кнопку що дозволяє зміну інтервалу перед та після фрагменту тексту.
8. Додати пункт меню та кнопку що дозволяє визначити позицію табуляції в тексті.

9. Додати пункт меню що визначає позицію курсору та відображує її у командному рядку.
10. Додати кнопки що дозволяють вставляти часто використані символи грецького алфавіту в текст (α,β ,μ).
11. Реалізувати пошук символів в тексті.
12. Реалізувати заміну символів в тексті.
13. Колір фону.
14. Колір тексту.
15. Границі тексту.

Індивідуальні завдання підвищенного рівня складності

1. Додати кнопку, що визначає параметри форматування фрагменту тексту та відображує її у діалоговому вікні.
2. Реалізувати друк документів з редактору.
3. Реалізувати кнопку відміни форматування для виділеного фрагменту тексту.
4. Реалізувати журнали операцій для відміни та повернення останньої дії.
5. Вставку об’єкта.
6. Багаторівневий список.
7. Колонтитули.

2. Завдання на самостійну роботу

1. Реалізувати MDI інтерфейс текстового редактора
2. реалізувати системи контекстних меню додатку
3. реалізувати довідникову систему додатку.

3. ХІД РОБОТИ

Взаємодія користувача з комп’ютером шляхом використання ним додатків для виконання своїх функціональних задач. Фокусом взаємодії в таких випадках є інтерфейс додатку.

Класичний інтерфейс WIMP додатку має наступні елементи:

• *робоча область:* внутрішня частина вікна, призначена для вдображення інформації та виконання завдань користувача;

• *границі:* рамка, що обмежує вікно із чотирьох сторін. Розміри вікна можна змінювати, переміщаючи границю мишею;

• *заголовок:* рядок безпосередньо під верхньою границею вікна, що містить назву вікна;

• *значок системного меню:* кнопка ліворуч у рядку заголовка відкриває меню переміщення й зміни розмірів вікна;

• *рядок горизонтального меню:* розташовується безпосередньо під заголовком, містить пункти меню, забезпечує доступ до команд;

• *панель інструментів:* розташовується під рядком меню, являє собою набір кнопок, забезпечує швидкий доступ до деяких команд;

• *кнопки Згорнути, Розгорнути/Відновити, Закрити* розташовані у верхній правій частині вікна.

Більш сучасна версія інтерфейсу замість горизонтального меню та панелів інструментів використовує елемент керування *Стрічка (Ribbon).*

До стандартних можливостей будь-якого додатку є:

- Створення, відкриття та збереження вмісту робочої області.
- Попередній перегляд та друк вмісту робочої області;
- Робота з буфером обміну;

До функціональних можливостей текстових редакторів відносяться:

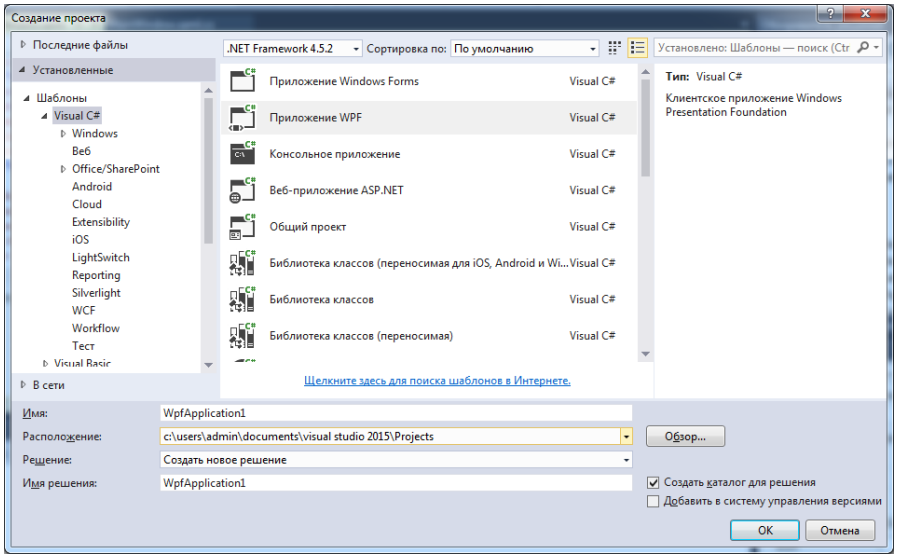
- керування виглядом текстових символів (керування шрифтами)
- керування розташуванням тексту (форматування абзаців);
- операції пошуку та заміни символів;
- вставка зовнішніх об’єктів (зображень, таблиць, формул тощо)

Будь-який додаток має гнучку систему допомоги що складається із централізованої довідки, підказок, що спливають та повідомлень в рядку стану.

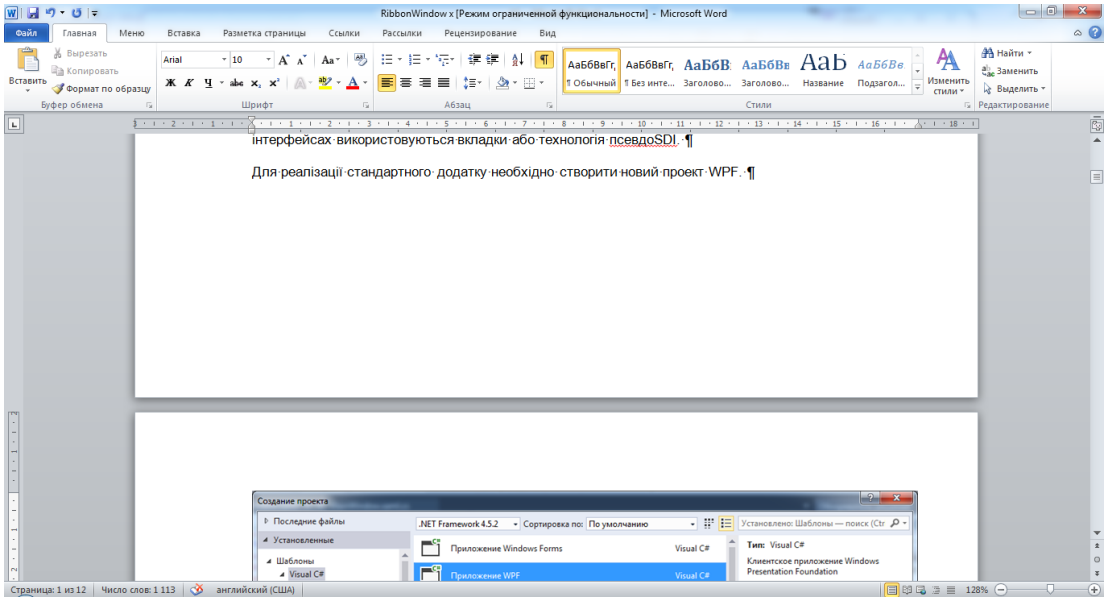
Додатки поділяються на SPI (однодокументні) та MDI (багатодокументні). Класичний MDI реалізується багатовіконним додатком на основі системи Parent-Child Windows. В більш сучасних в сучасних інтерфейсах використовуються вкладки або технологія псевдоSDI.

Windows Presentation Foundation (WPF [1]) - система для побудови клієнтських додатків для Windows з візуально привабливими можливостями взаємодії з користувачем, графічна (презентаційна) підсистема у складі .NET Framework (починаючи з версії 3.0), яка використовує мову XAML [2].

Для реалізації стандартного додатку необхідно створити новий проект WPF.



За зразок візьмемо інтерфейс текстового процесора Microsoft Word 2010.

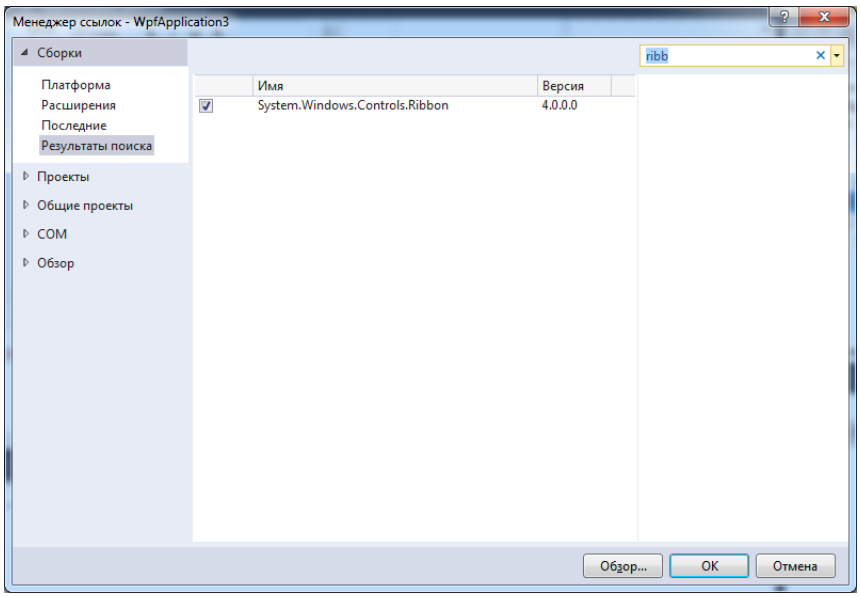


Послідовність створення SDI додатку в нашому випадку наступна:

- реалізації системи керування додатком (стрічки);
- реалізація робочої області;
- реалізація функціональних можливостей додатку.

Реалізація стрічки (Ribbon).

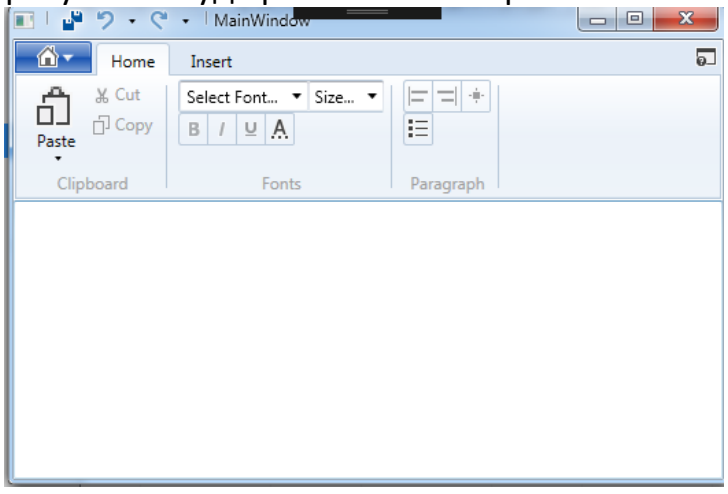
Для реалізації системи керування в проект необхідно додати посилання на зборку System.Windows.Controls.Ribbon:



Додатково про System.Widows.Controls.Ribbon дивимось за посиланнями:

https://www.codeproject.com/Articles/100081/Introducing-Ribbon-UI-Control-for-WPF
https://thebeyond.ru/2011/03/10/microsoft-ribbon-for-wpf/
http://www.c-sharpcorner.com/UploadFile/0b73e1/ribbon-control-in-wpf-4-5/
https://msdn.microsoft.com/en-us/library/system.windows.controls.ribbon.ribbon(v=vs.110).aspx
http://www.drdobbs.com/windows/whats-new-in-wpf-45/240009614?pgno=1

Дизайн інтерфейсу додатку буде виконано із використанням мови XAML. У результаті буде реалізований простий текстовий редактор, типу:



Для цього переходимо у файл з XAML розміткою головного вікна додатку (в даному випадку MainWindow.xaml) та першим кроком робимо заміну типу вікна з Windows на RibbonWindows:

```
<RibbonWindow x:Class="WpfApplication3.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApplication3"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
<Grid>
</Grid>
</RibbonWindow>
```

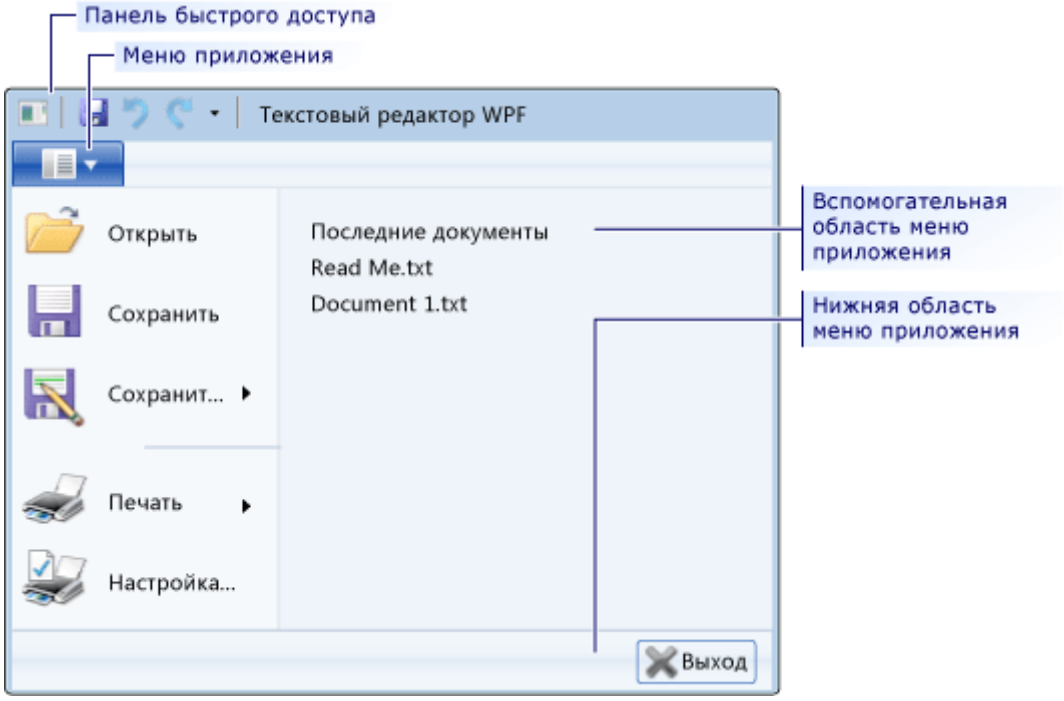
Наступним кроком в межах Grid визначаємо для розміточної сітки 2 рядки – перший для стрічки, другий для робочої області.

```
  <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
```

Далі розташовуємо стрічку в першому рядку:

```
<Ribbon Grid.Row="0" Margin="1,1,1,1">
</Ribbon>
```

Основними елементами ленти є:



Далі реалізуємо інтерфейс *Довідникового меню*:

```
<Ribbon.HelpPaneContent>
                <RibbonButton SmallImageSource="images\HelpApplication.png" KeyTip="F"/>
            </Ribbon.HelpPaneContent>
```

Для роботи з зобрженням кнопки HelpPane та іншими кнопками необхідно виконати наступні підготовчі дії:

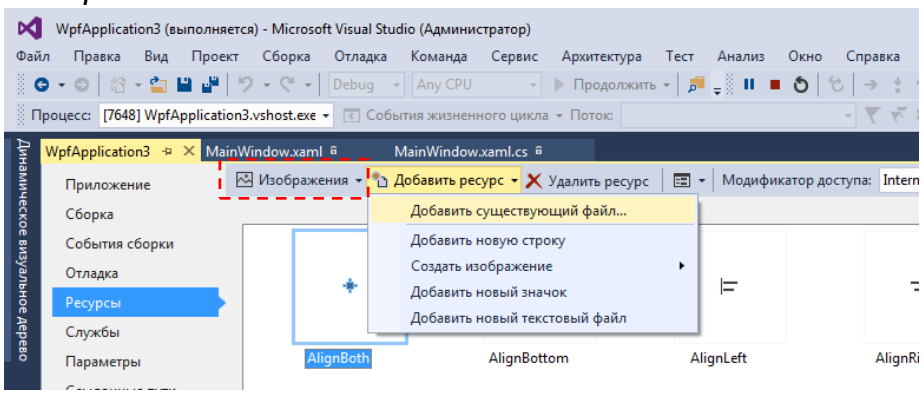
1. Створити в папці проекту папку images.

2. Зберегти в ній піктограми для наступних дій:
   Create, Open, Close, Save, Propetrty, Print
   Copy, Paste, Cut
   FontColor, Bold, Italic, Underline
   AlignLeft, AlignRight, AlignBoth, AlignCenter, BulletList
   InsertObject, InsertTable, InsertImage та для індивідуального завдання.

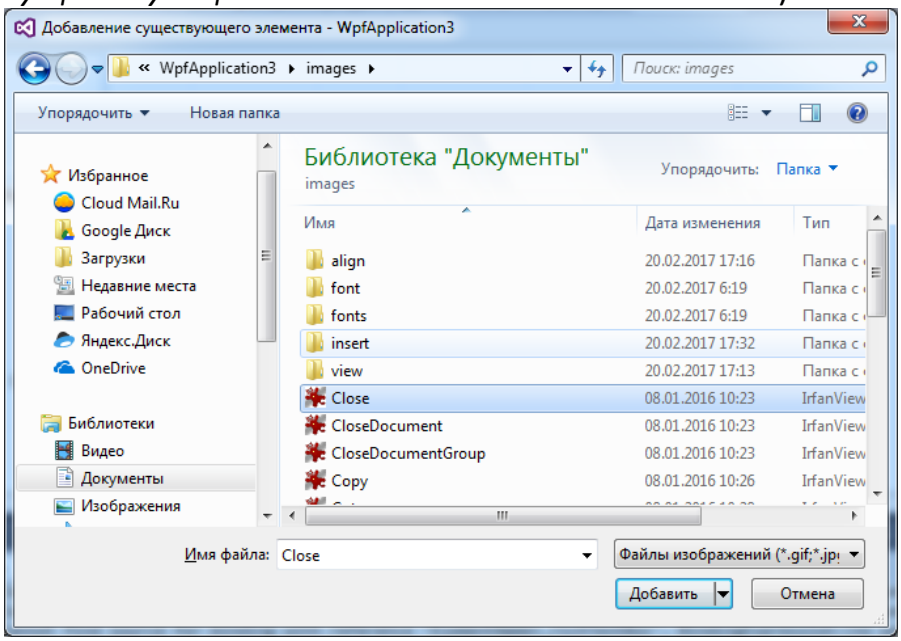
Для виконання даного завдання слід виконати пошук в Інтернеті колекції пікторграм або скористатись VSImageLibrary за посиланням https://www.microsoft.com/en-us/download/details.aspx?id=35825 або використати типовий набір кнопок із VSImageLibrary, що використані в даному зразку за посиланням

3. Додати проект ресурси зображень

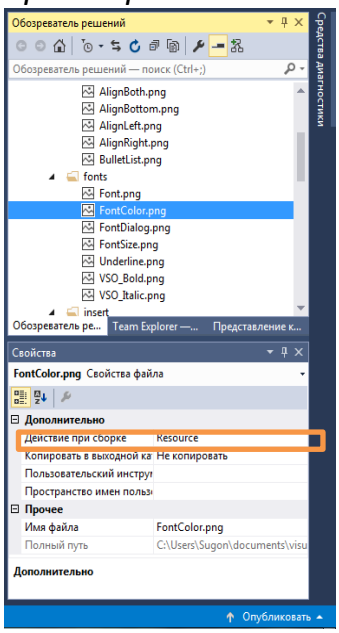
Для цього перейти в меню *Проект->Свойства*: на вкладку *Ресурси* та обрати *Изображения*:



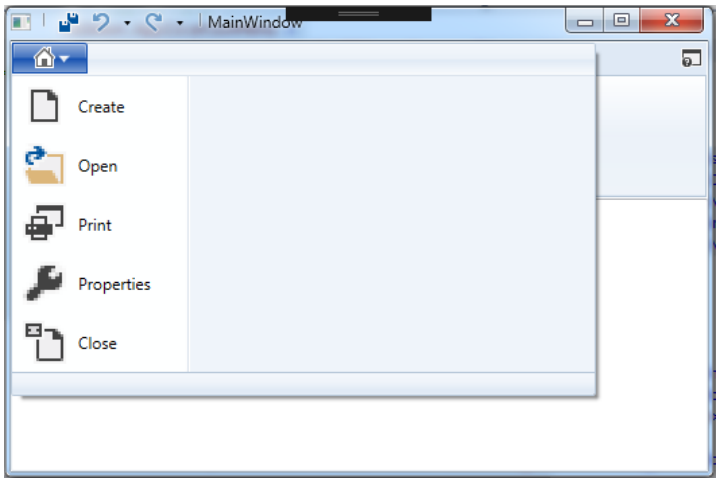
або за допомогою контекстного меню проекта у *Обозреватели решений* создать папку images и добавить зображення (команда *Добавить существующий елемент*… в контекстном меню):



Обов’язково перевірити у властивостях всіх рисунків, що параметр *Действие при сборке* має значення Recource.

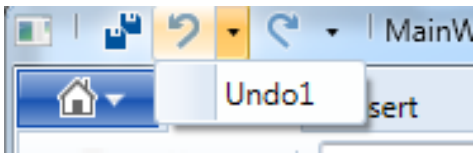


Наступним кроком є створення *Головного меню* додатку:



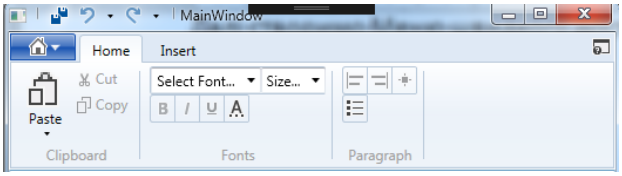

```
<Ribbon.ApplicationMenu >
   <RibbonApplicationMenu SmallImageSource="images\icon-home.png"  >
   <RibbonApplicationMenuItem Header="Create" ImageSource="images\Document.png"
KeyTip="C" />
   <RibbonApplicationMenuItem Header="Open" ImageSource="images\OpenFolder.png"
KeyTip="O" />
   <RibbonApplicationMenuItem Header="Print" ImageSource="images\PrintDialog.png"
      KeyTip="P"/>
   <RibbonApplicationMenuItem Header="Properties" ImageSource="images\Property.png" />
   <RibbonApplicationMenuItem Header="Close" ImageSource="images\CloseDocument.png"
      KeyTip="C" Click="btnClose_Click"/>
</RibbonApplicationMenu>
</Ribbon.ApplicationMenu>
```

Далі створимо *Меню швидкого доступу*:




```
<Ribbon.QuickAccessToolBar>
            <RibbonQuickAccessToolBar>
                <RibbonButton SmallImageSource="images\SaveAll.png" />
                <RibbonSplitButton x:Name="Undo" SmallImageSource="images\Undo.png" >
                    <RibbonSplitMenuItem Header="Undo1"></RibbonSplitMenuItem>
                </RibbonSplitButton>
                <RibbonSplitButton x:Name="Redo" SmallImageSource="images\Redo.png" >
                    <RibbonSplitMenuItem Header="Redo1"></RibbonSplitMenuItem>
                </RibbonSplitButton>
            </RibbonQuickAccessToolBar>
        </Ribbon.QuickAccessToolBar>
```

Наступним кроком є створення та заповнення вкладки *Home:*



Зверніть увагу, що дана вкладка поділена на три групи: Clipboard, Fonts, Paragraph.

```
<RibbonTab Header="Home">
    <!-- Home/Clipboard group-->
    <RibbonGroup Header="Clipboard">
     <RibbonMenuButton LargeImageSource="Images\paste.png" Label="Paste" KeyTip="V" >
     <RibbonMenuItem ImageSource="Images\Paste.png" Header="Keep Text Only" KeyTip="T" />
      <RibbonMenuItem ImageSource="Images\Paste.png" Header="Paste Special..."
          KeyTip="S"/>
     </RibbonMenuButton>
     <RibbonButton SmallImageSource="Images\Cut.png" Label="Cut" KeyTip="X" />
     <RibbonButton SmallImageSource="Images\Copy.png" Label="Copy" KeyTip="C" />
    </RibbonGroup>

    <!-- Home/Colors group-->
    <RibbonGroup x:Name="fonts" Header="Fonts" Width="Auto" >
     <RibbonControlGroup>
      <ComboBox ItemsSource="{Binding Source={x:Static Fonts.SystemFontFamilies}}"
          Text="Select Font..." IsEditable="True"/>
      <ComboBox x:Name="_fontSize" Text="Size..." IsEditable="True"></ComboBox>
     </RibbonControlGroup>
     <RibbonControlGroup>
      <RibbonButton SmallImageSource="Images\fonts\VSO_Bold.png" KeyTip="B" />
      <RibbonButton SmallImageSource="Images\fonts\VSO_Italic.png" KeyTip="I" />
      <RibbonButton SmallImageSource="Images\fonts\Underline.png" KeyTip="U" />
      <RibbonButton SmallImageSource="Images\fonts\FontColor.png" KeyTip="A"/>
     </RibbonControlGroup>
    </RibbonGroup>

    <RibbonGroup x:Name="paragraph" Header="Paragraph">
     <RibbonControlGroup>
      <RibbonButton SmallImageSource="Images\align\AlignLeft.png" />
      <RibbonButton SmallImageSource="Images\align\AlignRight.png" />
      <RibbonButton SmallImageSource="Images\align\AlignBoth.png" />
     </RibbonControlGroup>

     <RibbonControlGroup>
      <RibbonButton SmallImageSource="Images\align\BulletList.png"/>
     </RibbonControlGroup>
    </RibbonGroup>

   </RibbonTab>
```

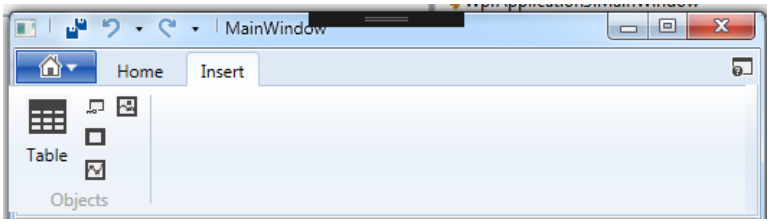
Для коректного відображення розмірів шрифта у відповідному ComboBox необхідно у файлі коду головного вікна (в даному випадку MainWindow.xaml.cs) визначити структуру даних, приблизно такого вмісту:

```
        public double[] FontSizes
        {
            get
            {
                return new double[] { 3.0, 4.0, 5.0, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0,
9.5, 10.0, 10.5, 11.0, 11.5, 12.0, 12.5,13.0,13.5,14.0, 15.0,16.0, 17.0, 18.0, 19.0,
20.0, 22.0, 24.0, 26.0, 28.0, 30.0,32.0, 34.0, 36.0, 38.0, 40.0, 44.0, 48.0, 52.0, 56.0,
60.0, 64.0, 68.0, 72.0, 76.0,80.0, 88.0, 96.0, 104.0, 112.0, 120.0, 128.0, 136.0, 144.0
};
            }
              }
```

та визначити її як джерело даних для елемента керування ComboBox

```
public MainWindow()
      {
         InitializeComponent();
         _fontSize.ItemsSource = FontSizes;
      }
```

Останнім кроком є створення вкладки Insert:




```
<RibbonTab Header="Insert" Margin="-8,0,8,0" UseLayoutRounding="True"
     ScrollViewer.VerticalScrollBarVisibility="Auto">
    <RibbonGroup x:Name="objects" Header="Objects">
     <RibbonButton LargeImageSource="Images\insert\Table.png" Label="Table"/>
     <RibbonButton SmallImageSource="Images\insert\ApplicationAccess.png"/>
     <RibbonButton SmallImageSource="Images\insert\Rectangle.png"/>
     <RibbonButton SmallImageSource="Images\insert\LineChart.png"/>
     <RibbonButton SmallImageSource="Images\insert\Image.png"/>
    </RibbonGroup>
   </RibbonTab>
```

Компілюємо проект та перевіряємо помилки (сподіваємось, що це вже 3 або 4 раз). Дивимось на результат.

Реалізація робочої області

Для реалізації робочої області використаємо компонет RichTextBox.

```
<RichTextBox x:Name="doc1" Grid.Row="1">
        </RichTextBox>
```

Даний компонент містить вбудований функціонал для підтримки стандартних операцій з текстом, таких як робота з буфером обміну та параметрами шрифта та абзацу.

Для використання вбудованих властивостей компоненту використаємо розширення розмітки x:Static (див. https://msdn.microsoft.com/ru-ru/library/ms742135(v=vs.110).aspx). За його допомоги реалізуємо наступні команди:

- команди рівня додатку: Cut, Copy, Paste, Undo, Rendo.
- команди редагування: Bold, Italic, Underline, AlignLeft, AlignRight, AlignJustify.

Додаємо відповідні дефініції в код розмітки вікна:

в *Меню швидкого доступу*:

```
 <RibbonSplitButton x:Name="Undo" SmallImageSource="images\Undo.png" Command="{x:Static
ApplicationCommands.Undo}" CommandTarget="{Binding ElementName=_richTextBox}">
…
 <RibbonSplitButton x:Name="Redo" SmallImageSource="images\Redo.png" Command="{x:Static
ApplicationCommands.Redo}" CommandTarget="{Binding ElementName=_richTextBox}">
```

на вкладку Insert

```
…
   <RibbonMenuItem ImageSource="Images\Paste.png" Header="Keep Text Only" KeyTip="T"
Command="{x:Static ApplicationCommands.Paste}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
…
   <RibbonButton SmallImageSource="Images\Cut.png" Label="Cut" KeyTip="X"
Command="{x:Static ApplicationCommands.Cut}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
   <RibbonButton SmallImageSource="Images\Copy.png" Label="Copy" KeyTip="C"
Command="{x:Static ApplicationCommands.Copy}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
```

…

```
<RibbonButton SmallImageSource="Images\fonts\VSO_Bold.png" KeyTip="B" Command="{x:Static
EditingCommands.ToggleBold}" CommandTarget="{Binding ElementName=_richTextBox}"/>
   <RibbonButton SmallImageSource="Images\fonts\VSO_Italic.png" KeyTip="I"
Command="{x:Static EditingCommands.ToggleItalic}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
   <RibbonButton SmallImageSource="Images\fonts\Underline.png" KeyTip="U"
Command="{x:Static EditingCommands.ToggleUnderline}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
```

…

```
 <RibbonButton SmallImageSource="Images\align\AlignLeft.png" Command="{x:Static
    EditingCommands.AlignLeft}" CommandTarget="{Binding ElementName=_richTextBox}"/>
<RibbonButton SmallImageSource="Images\align\AlignRight.png" Command="{x:Static
    EditingCommands.AlignRight}" CommandTarget="{Binding ElementName=_richTextBox}"/>
<RibbonButton SmallImageSource="Images\align\AlignBoth.png" Command="{x:Static
    EditingCommands.AlignJustify}" CommandTarget="{Binding ElementName=_richTextBox}"/>
```

Реалізація основних функцій додатку

Для реалізації функції відкриття файлів визначимо та реалізуємо процедуру btnOpen_Click:

```
<RibbonApplicationMenuItem Header="Open" ImageSource="images\OpenFolder.png" KeyTip="O"
Click="btnOpen_Click"/>
```

реалізація має наступний вид:

```
private void btnOpen_Click(object sender, RoutedEventArgs e)
{
 OpenFileDialog dlg = new OpenFileDialog();
 dlg.Filter = "Document files (*.rtf)|*.rtf";
 var result = dlg.ShowDialog();
 if (result.Value)
 {
 TextRange t = new TextRange(doc1.Document.ContentStart, doc1.Document.ContentEnd);
 FileStream file = new FileStream(dlg.FileName, FileMode.Open);
 t.Load(file, System.Windows.DataFormats.Rtf);
 }
}
```

Для реалізації функції збереження файлів визначимо та реалізуємо процедуру btnSave_Click:

```
<RibbonButton SmallImageSource="images\SaveAll.png" Click="btnSave_Click"/>
```

реалізація має наступний вид:

```
  private void btnSave_Click(object sender, RoutedEventArgs e)
  {

   SaveFileDialog savefile = new SaveFileDialog();
   // set a default file name
   savefile.FileName = "unknown.doc";
   // set filters - this can be done in properties as well
   savefile.Filter = "Document files (*.rtf)|*.rtf";
   if (savefile.ShowDialog() == true)
   {
    TextRange t = new TextRange(doc1.Document.ContentStart, doc1.Document.ContentEnd);
    this.Title = this.Title + " " + savefile.FileName;
    FileStream file = new FileStream(savefile.FileName, FileMode.Create);
    t.Save(file, System.Windows.DataFormats.Rtf);
    file.Close();
   }

   IsSaved = true;

  }
```

Змінна IsSaved визначається як глобальна і в подальшому вкористовується для контролю збереження файлу перед закриттям додатку.

```
  public partial class MainWindow : RibbonWindow
    {
        public bool IsSaved = false;
```

визначення процедури закриття додатку:

```
    <RibbonApplicationMenuItem Header="Close" ImageSource="images\CloseDocument.png"
KeyTip="C" Click="btnClose_Click"/>
```

реалізація:

```
 private void btnClose_Click(object sender, RoutedEventArgs e)
 { if (IsSaved == false)
  if (MessageBox.Show("Do you want save changes ?", "Message", MessageBoxButton.YesNo) ==
MessageBoxResult.Yes)
  //Если была нажата кнопка Yes, вызываем метод Save
  {
   this.btnSave_Click(sender, e);
  }

  this.Close();
 }
```

Останнім кроком є реалізація процедури визначення типу та розміру шрифта.

Для цього визначимо наступні процедури:

```
 <ComboBox ItemsSource="{Binding Source={x:Static Fonts.SystemFontFamilies}}"
SelectionChanged="FontFamili_SelectionChange" Text="Select Font..." IsEditable="True"/>
 <ComboBox SelectionChanged="FontSize_SelectionChange" x:Name="_fontSize" Text="Size..."
IsEditable="True"></ComboBox>
```

та реалізуємо їх наступним чином:

```
void ApplyPropertyValueToSelectedText(DependencyProperty formattingProperty, object
value)
  {
   if (value == null)
    return;
   doc1.Selection.ApplyPropertyValue(formattingProperty, value);
  }

  private void FontFamili_SelectionChange(object sender, SelectionChangedEventArgs e)
  {
   try
   {
    FontFamily editValue = (FontFamily)e.AddedItems[0];
    ApplyPropertyValueToSelectedText(TextElement.FontFamilyProperty, editValue);
   }
   catch (Exception) { }
  }

  private void FontSize_SelectionChange(object sender, SelectionChangedEventArgs e)
  {

   try
   {


    ApplyPropertyValueToSelectedText(TextElement.FontSizeProperty, e.AddedItems[0]);
   }
   catch (Exception) { }
  }
 }
 }
```

Компілюємо, дивимось на результат та переходимо до виконання індивідуального завдання.

P.S. Не забудьте покликати викладача та отримати бали.

4. Лістинг коду додатку

Загальний вигляд файлу XAML розмітки:

```
<RibbonWindow x:Class="WpfApplication3.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:local="clr-namespace:WpfApplication3"
  mc:Ignorable="d"
  Title="MainWindow" Height="350" Width="525">
 <Grid>
  <Grid.RowDefinitions>
   <RowDefinition Height="Auto" />
   <RowDefinition Height="*" />
  </Grid.RowDefinitions>

  <Ribbon Grid.Row="0" Margin="1,1,1,1">
   <!--Конпка HELP-->
   <Ribbon.HelpPaneContent>
    <RibbonButton SmallImageSource="images\HelpApplication.png" KeyTip="F"/>
   </Ribbon.HelpPaneContent>
   <!--Главное меню-->
   <Ribbon.ApplicationMenu >
    <RibbonApplicationMenu SmallImageSource="images\icon-home.png" >
    <RibbonApplicationMenuItem Header="Create" ImageSource="images\Document.png"
KeyTip="C" Click="btnCreate_Click"/>
     <RibbonApplicationMenuItem Header="Open" ImageSource="images\OpenFolder.png"
KeyTip="O" Click="btnOpen_Click"/>
     <RibbonApplicationMenuItem Header="Print" ImageSource="images\PrintDialog.png"
KeyTip="P"/>
     <RibbonApplicationMenuItem Header="Properties" ImageSource="images\Property.png" />
     <RibbonApplicationMenuItem Header="Close" ImageSource="images\CloseDocument.png"
KeyTip="C" Click="btnClose_Click"/>
    </RibbonApplicationMenu>
   </Ribbon.ApplicationMenu>
   <!--быстрое меню-->
   <Ribbon.QuickAccessToolBar>
    <RibbonQuickAccessToolBar>
     <RibbonButton SmallImageSource="images\SaveAll.png" Click="btnSave_Click"/>
     <RibbonSplitButton x:Name="Undo" SmallImageSource="images\Undo.png"
Command="{x:Static ApplicationCommands.Undo}" CommandTarget="{Binding
ElementName=_richTextBox}">
      <RibbonSplitMenuItem Header="Undo1"></RibbonSplitMenuItem>
     </RibbonSplitButton>
     <RibbonSplitButton x:Name="Redo" SmallImageSource="images\Redo.png"
Command="{x:Static ApplicationCommands.Redo}" CommandTarget="{Binding
ElementName=_richTextBox}">
      <RibbonSplitMenuItem Header="Redo1"></RibbonSplitMenuItem>
     </RibbonSplitButton>
    </RibbonQuickAccessToolBar>
   </Ribbon.QuickAccessToolBar>
```

```
    <!--Вкладки ленты-->
    <RibbonTab Header="Home">
      <!-- Home/Clipboard group-->
      <RibbonGroup Header="Clipboard">
        <RibbonMenuButton LargeImageSource="Images\paste.png" Label="Paste" KeyTip="V" >
        <RibbonMenuItem ImageSource="Images\Paste.png" Header="Keep Text Only" KeyTip="T"
Command="{x:Static ApplicationCommands.Paste}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
        <RibbonMenuItem ImageSource="Images\Paste.png" Header="Paste Special..."
KeyTip="S"/>
        </RibbonMenuButton>
        <RibbonButton SmallImageSource="Images\Cut.png" Label="Cut" KeyTip="X"
Command="{x:Static ApplicationCommands.Cut}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
        <RibbonButton SmallImageSource="Images\Copy.png" Label="Copy" KeyTip="C"
Command="{x:Static ApplicationCommands.Copy}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
      </RibbonGroup>

      <!-- Home/Colors group-->
      <RibbonGroup x:Name="fonts" Header="Fonts" Width="Auto" >
        <RibbonControlGroup>
          <ComboBox ItemsSource="{Binding Source={x:Static Fonts.SystemFontFamilies}}"
SelectionChanged="FontFamili_SelectionChange" Text="Select Font..." IsEditable="True"/>
        <ComboBox SelectionChanged="FontSize_SelectionChange" x:Name="_fontSize"
Text="Size..." IsEditable="True"></ComboBox>
        </RibbonControlGroup>
        <RibbonControlGroup>
          <RibbonButton SmallImageSource="Images\fonts\VSO_Bold.png" KeyTip="B"
Command="{x:Static EditingCommands.ToggleBold}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
        <RibbonButton SmallImageSource="Images\fonts\VSO_Italic.png" KeyTip="I"
Command="{x:Static EditingCommands.ToggleItalic}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
        <RibbonButton SmallImageSource="Images\fonts\Underline.png" KeyTip="U"
Command="{x:Static EditingCommands.ToggleUnderline}" CommandTarget="{Binding
ElementName=_richTextBox}"/>
        <RibbonButton SmallImageSource="Images\fonts\FontColor.png" KeyTip="A"/>
        </RibbonControlGroup>
      </RibbonGroup>

      <RibbonGroup x:Name="paragraph" Header="Paragraph">
        <RibbonControlGroup>
          <RibbonButton SmallImageSource="Images\align\AlignLeft.png" Command="{x:Static
EditingCommands.AlignLeft}" CommandTarget="{Binding ElementName=_richTextBox}"/>
        <RibbonButton SmallImageSource="Images\align\AlignRight.png" Command="{x:Static
EditingCommands.AlignRight}" CommandTarget="{Binding ElementName=_richTextBox}"/>
        <RibbonButton SmallImageSource="Images\align\AlignBoth.png" Command="{x:Static
EditingCommands.AlignJustify}" CommandTarget="{Binding ElementName=_richTextBox}"/>
        </RibbonControlGroup>

        <RibbonControlGroup>
          <RibbonButton SmallImageSource="Images\align\BulletList.png"/>
        </RibbonControlGroup>
      </RibbonGroup>

    </RibbonTab>
```

```
    <RibbonTab Header="Insert" Margin="-8,0,8,0" UseLayoutRounding="True"
ScrollViewer.VerticalScrollBarVisibility="Auto">
      <RibbonGroup x:Name="objects" Header="Objects">
       <RibbonButton LargeImageSource="Images\insert\Table.png" Label="Table"/>
       <RibbonButton SmallImageSource="Images\insert\ApplicationAccess.png"/>
       <RibbonButton SmallImageSource="Images\insert\Rectangle.png"/>
       <RibbonButton SmallImageSource="Images\insert\LineChart.png"/>
       <RibbonButton SmallImageSource="Images\insert\Image.png"/>
      </RibbonGroup>
    </RibbonTab>
   </Ribbon>

    <RichTextBox x:Name="doc1" Grid.Row="1">
   </RichTextBox>


 </Grid>
</RibbonWindow>
```

Загальий вигляд MainWindows.xaml.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls.Ribbon;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Win32;

namespace WpfApplication3
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : RibbonWindow
    {
        public bool IsSaved = false;
        public MainWindow()
        {
            InitializeComponent();
            _fontSize.ItemsSource = FontSizes;
        }
        public double[] FontSizes
        {
            get
            {
                return new double[] { 3.0, 4.0, 5.0, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0,
9.5, 10.0, 10.5, 11.0, 11.5, 12.0, 12.5,13.0,13.5,14.0, 15.0,16.0, 17.0, 18.0, 19.0,
20.0, 22.0, 24.0, 26.0, 28.0, 30.0,32.0, 34.0, 36.0, 38.0, 40.0, 44.0, 48.0, 52.0, 56.0,
60.0, 64.0, 68.0, 72.0, 76.0,80.0, 88.0, 96.0, 104.0, 112.0, 120.0, 128.0, 136.0, 144.0
};
            }
        }
```

```
        private void btnOpen_Click(object sender, RoutedEventArgs e)
        {
            OpenFileDialog dlg = new OpenFileDialog();
            dlg.Filter = "Document files (*.rtf)|*.rtf";
            var result = dlg.ShowDialog();
            if (result.Value)
            {
                TextRange t = new TextRange(doc1.Document.ContentStart,
doc1.Document.ContentEnd);
                FileStream file = new FileStream(dlg.FileName, FileMode.Open);
                t.Load(file, System.Windows.DataFormats.Rtf);
            }
        }

        private void btnCreate_Click(object sender, RoutedEventArgs e)
        {

        }

        private void btnSave_Click(object sender, RoutedEventArgs e)
        {

            SaveFileDialog savefile = new SaveFileDialog();
            // set a default file name
            savefile.FileName = "unknown.doc";
            // set filters - this can be done in properties as well
            savefile.Filter = "Document files (*.rtf)|*.rtf";
            if (savefile.ShowDialog() == true)
            {
                TextRange t = new TextRange(doc1.Document.ContentStart,
doc1.Document.ContentEnd);
                this.Title = this.Title + "  " + savefile.FileName;
                FileStream file = new FileStream(savefile.FileName, FileMode.Create);
                t.Save(file, System.Windows.DataFormats.Rtf);
                file.Close();
            }

            IsSaved = true;

        }

        private void btnClose_Click(object sender, RoutedEventArgs e)
        { if (IsSaved == false)
                if (MessageBox.Show("Do you want save changes ?", "Message",
MessageBoxButton.YesNo) == MessageBoxResult.Yes)
                //Если была нажата кнопка Yes, вызываем метод Save
                {
                    this.btnSave_Click(sender, e);
                }

            this.Close();
        }
```

```
        void ApplyPropertyValueToSelectedText(DependencyProperty formattingProperty,
object value)
        {
            if (value == null)
                return;
           doc1.Selection.ApplyPropertyValue(formattingProperty, value);
        }
        private void FontFamili_SelectionChange(object sender, SelectionChangedEventArgs
e)
        {
            try
            {
                FontFamily editValue = (FontFamily)e.AddedItems[0];
                ApplyPropertyValueToSelectedText(TextElement.FontFamilyProperty,
editValue);
            }
            catch (Exception) { }
        }

        private void FontSize_SelectionChange(object sender, SelectionChangedEventArgs e)
        {

            try
            {


               ApplyPropertyValueToSelectedText(TextElement.FontSizeProperty,
e.AddedItems[0]);
            }
            catch (Exception) { }
        }
    }
    }
```