ЛАБОРАТОРНАЯ РАБОТА №3

Работа с элементами управления ComboBox и ListBox. Валидация данных

1 Цель работы

Изучить возможности использования элементов управления ListBox и ComboBox. Освоить способы проверки ввода данных.

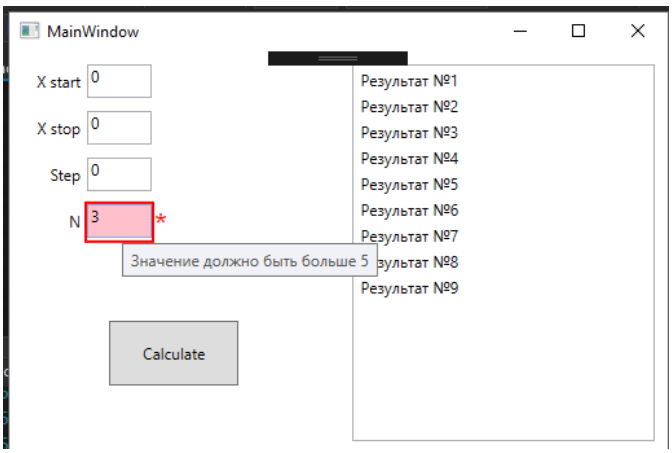
2 Задание 1

Требуется создать программу для вычисления двух функций, одна из которых задана при помощи ряда. В качестве типа проекта использовать приложение типа WPF.

В элементы управления TextBox следует вводить: начальное значение аргумента, конечное значение, шаг и число членов суммы. При вводе недопустимых данных, например, набора букв вместо цифр, следует изменить внешний вид соответствующего TextBox и вывести подсказку при наведении на него курсора мыши.

При нажатии на кнопку Calculate выполняется вычисление функций и отображение расчетных значений в элементе ListBox.

Вариант интерфейса:



2.1 Индивидуальные задания

1. $S(x) = \sum_{k=0}^{n} \frac{ln^k}{k!} x^k$, $Y(x) = 3^x - 1$

2. $S(x) = \sum_{k=1}^{n} \frac{\cos kx}{k},\ Y(x) = -ln\left|2\sin\frac{x}{2}\right|$

3. $S(x) = \sum_{k=0}^{n} (-1)^k \frac{x^{2k+1}}{(2k+1)!},\ Y(x) = \sin(x)$

4. $S(x) = \sum_{k=0}^{n} \frac{x^k}{k!}, Y(x) = e^x$

5. $S(x) = \sum_{k=1}^{n} (-1)^{k+1} \frac{\sin(kx)}{k}, Y(x) = \frac{x}{2}$

6. $S(x) = \sum_{k=0}^{n} \frac{\cos(\frac{k\pi}{4})}{k!} x^k,\ Y(x) = e^{x\cos\frac{\pi}{4}} \cos(x\sin(\frac{\pi}{4}))$

7. $S(x) = \sum_{k=0}^{n} (-1)^k \frac{x^{2k}}{(2k)!},\ Y(x) = \cos x$

8. $S(x) = \sum_{k=1}^{n} x^k \sin^k\left(\frac{\pi}{4}\right),\ Y(x) = \frac{x\sin(\pi/4)}{1-2x\cos(\pi/4)+x^2}$

9. $S(x) = \sum_{k=0}^{n} \frac{x^{4k+1}}{4k+1},\ Y(x) = \frac{1}{4} ln\frac{1+x}{1-x} + \frac{1}{2} arctg(x)$

10. $S(x) = \sum_{k=0}^{n} \frac{\cos(kx)}{k!},\ Y(x) = e^{\cos x}\cos(\sin x)$

11. $S(x) = \sum_{k=0}^{n} \frac{2k+1}{k!} x^{2k},\ Y(x) = (1+2x^2)e^{x^2}$

12. $S(x) = \sum_{k=1}^{n} \frac{x^k \cos(\frac{k\pi}{3})}{k},\ Y(x) = -\frac{1}{2}\ln(1-2x\cos\frac{\pi}{3} + x^2)$

13. $S(x) = \sum_{k=0}^{n} \frac{1}{2k+1}\left(\frac{x-1}{x+1}\right)^{2k+1},\ Y(x) = \frac{1}{2}\ln(x)$

14. $S(x) = \sum_{k=1}^{n} (-1)^k \frac{\cos(kx)}{k^2},\ Y(x) = \frac{1}{4}(x^2 - \pi^2/3)$

15. $S(x) = \sum_{k=1}^{n} (-1)^{k+1} \frac{x^{2k+1}}{4k^2-1},\ Y(x) = \frac{1+x^2}{2} arctg(x) - \frac{x}{2}$

3 Задание 2.

Программа должна иметь возможность ввода информации о работнике (фамилия, зарплата, должность, город, улица, дом) и записывать введенные данные в текстовый файл. Список работников выводить в элемент ListBox.

Для ускорения ввода полей “должность”, “город”, “улица” использовать элемент ComboBox. При отсутствии требуемого значения в списке окна ComboBox предусмотреть возможность ввода нового значения. Добавить обработку ошибок ввода.

4 Рекомендации к выполнению задания 1

4.1 Определение объекта привязки

Опишите класс, инкапсулирующий исходные данные для расчетов. Например:

```
public class Values
{
   public double XStart { get; set;}
   public double XStop { get; set; }

      . . .
}
```

Добавьте объект созданного класса в контекст данных элемента, являющегося контейнером для элементов TextBox. Например:

```
Values values;

public MainWindow()
        {
            InitializeComponent();
            values = new Values();
            grid.DataContext = values;

 }
```

В разметке выполните привязку текста элементов TextBox к свойствам созданного класса. Например:

```
<TextBox x:Name="tbXStart" Width="50" Text="{Binding Path=XStart}">
</TextBox>
```

Запустите проект и введите некорректные данные (например, буквы) в один из элементов TextBox. При переключении фокуса на другой элемент вокруг элемента появляется красный контур.

4.2 Изменение стиля отображения некорректных данных

В ресурсах окна создайте стиль для элементов TextBox. В созданном стиле опишите триггер, отслеживающий свойство Validation.HasError:

```
<Window.Resources>
     <Style TargetType="{x:Type TextBox}">
```

```
            <Style.Triggers>
                 <Trigger Property="Validation.HasError" Value="true">
                     <Setter Property="ToolTip"
                       Value="{Binding RelativeSource={x:Static
                 RelativeSource.Self},
                       Path=(Validation.Errors)[0].ErrorContent}" />
                     <Setter Property="Background" Value="Pink"/>
                 </Trigger>
              </Style.Triggers>
          </Style>

</Window.Resources>
```

Запустите проект и проверьте, что у элемента с неверными данными меняется цвет фора, а при наведении курсора мыши появляется подсказка.

4.3 Отображение звездочки напротив элемента с неверными данными

В ресурсы окна добавьте шаблон элемента управления:

```
<ControlTemplate x:Key="ErrorTempl">
     <StackPanel Orientation="Horizontal">
          <Border BorderThickness="2" BorderBrush="Red">
              <AdornedElementPlaceholder></AdornedElementPlaceholder>
          </Border>
          <TextBlock Foreground="Red" FontSize="24">*</TextBlock
     </StackPanel>
</ControlTemplate>
```

Для использования данного шаблона измените разметку элементов TextBox:

```
<TextBox x:Name="tbXStart" Width="50"
          Text="{Binding
                Path=XStart,
                ValidatesOnExceptions=True,
                UpdateSourceTrigger=PropertyChanged}"
                Validation.ErrorTemplate="{StaticResource ErrorTempl}"
                >
</TextBox>
```

4.4 Добавление проверки правильности данных

В классе исходных данных для расчета добавьте свою проверку в метод get, например:

```
public class Values
{
      private int n;
      . . .
      . . .
      public int N
      {
            get { return n; }
            set
            {
                  if (value <= 5)
                        throw new ArgumentException("Значение должно быть
                  больше 5");
            }
      }
}
```

Запустите проект и проверьте результат

4.5 Привязка результатов вычисления к ListBox

Опишите коллекцию типа ObservableCollection<T> (требуется подключить пространство имен System.Collections.ObjectModel) и укажите ее в качестве DataContext элемента ListBox. Например:

```
public partial class MainWindow : Window
{
      ObservableCollection<string> results;
      Values values;
      public MainWindow()
      {
            InitializeComponent();
            values = new Values();
            grid.DataContext = values;

            results = new ObservableCollection<string>();
            lResult.DataContext = results;
      }

      . . .
}
```

В разметке укажите привязку для элементов списка:

```
<ListBox x:Name="lResult"
                  Grid.Column="1" Grid.RowSpan="2"
                  Margin="10"
                  ItemsSource="{Binding}">
</ListBox>
```

В обработчике события click кнопки заполните коллекцию результатами вычислений (не забудьте предварительно очистить коллекцию от предыдущих результатов методом Clear()). Исходные данные возьмите из объекта класса, созданного в начале (см. п 4.1).

Запустите проект и проверьте результат. Попробуйте заменить коллекцию на List<T>. В этом случае ListBox не отслеживает изменения в коллекции