СВПП. Лабораторная работа 1.

Введение в WPF. Контейнеры компоновки

Цель работы. Научиться создавать приложения с графическим интерфейсом, используя технологию WPF. Ознакомиться со структурой файлов разметки и заоконного кода. Научиться использовать один из контейнеров компоновки (Grid). Ознакомиться с простейшими элементами управления, их свойствами, обработчиками событий.

Задание 1. Калькулятор

Разработайте калькулятор согласно макету. Используйте контейнер компоновки Grid. Калькулятор должен выполнять основные арифметические операции с целыми и вещественными числами.

<table>
<tr><td colspan="4">0,00</td></tr>
<tr><td>C</td><td><-</td><td>*</td><td>/</td></tr>
<tr><td>7</td><td>8</td><td>9</td><td>-</td></tr>
<tr><td>4</td><td>5</td><td>6</td><td>+</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td rowspan="2">=</td></tr>
<tr><td colspan="2">0</td><td>,</td></tr>
</table>

Описание решения.

1. Создайте проект WPF Calculator

В окне разметки настройте параметры окна. Можно сразу добавить размер шрифта для всех вложенных элементов

```
Title="Калькулятор" Height="450" Width="300" FontSize="28"
```

2. Добавьте сетку

Создадим сетку 4x6

```
<Grid>
     <Grid.RowDefinitions>
         <RowDefinition/>
         <RowDefinition/>
         <RowDefinition/>
         <RowDefinition/>
         <RowDefinition/>
         <RowDefinition/>
     </Grid.RowDefinitions>
     <Grid.ColumnDefinitions>
         <ColumnDefinition/ >
         <ColumnDefinition/ >
         <ColumnDefinition/ >
         <ColumnDefinition/ >
     </Grid.ColumnDefinitions>
 </Grid>
```

3. Добавьте текстовое поле и кнопки в соответствующие ячейки

```
     <TextBox Grid.Row="0" Grid.ColumnSpan="4" />

     <Button Grid.Column="0" Grid.Row="1" Content="C"  />
     <Button Grid.Column="1" Grid.Row="1" Content="&lt;-"  />
     <Button Grid.Column="2" Grid.Row="1" Content="*"  />
```

```
        <Button Grid.Column="3" Grid.Row="1" Content="/"  />
```

Обратите внимание, как экранируется символ "<"

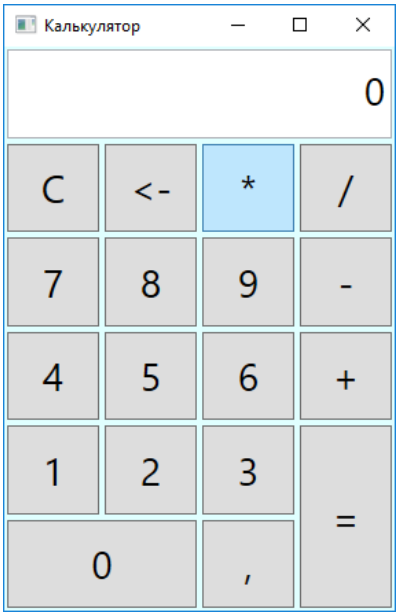
Знак "=" занимает 2 строки
```
<Button Grid.Column="3" Grid.Row="3" Content="=" Grid.RowSpan="2" />
```

Символ "0" занимает 2 столбца
```
<Button Grid.Column="0" Grid.Row="5" Content="0" Grid.ColumnSpan="2" />
```

4. Отформатируем изображение

Добавьте к текстовому полю текст "0", который должен присутствовать по умолчанию. Добейтесь, чтобы текст был прижат к правому краю, вертикальное выравнивание – по центру. Сделайте небольшой отступ внутрь (padding). Выберите моноширинный шрифт (например, `"Lucida Console"`).

Для кнопок можно сделать небольшой отступ (margin). Надписи на кнопках должны располагаться по центру.



5. Перейдем к обработчикам событий. Цифровые кнопки

Дважды щелкнем в конструкторе по кнопке "1".

Обработчик будет создан автоматически

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

```
        }
```

Поскольку для всех кнопок с цифрами будет выполняться одно и то же действие – цифра будет просто добавляться к текстовому полю, им всем подойдет один обработчик.

Переименуем его:

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


        }
```

Теперь нужно извлечь цифру из контента кнопки и записать ее в текстовое поле.

Для этого сначала добавим имя текстовому полю в окне свойств:

Или так

```
<TextBox x:Name="textBox"
```

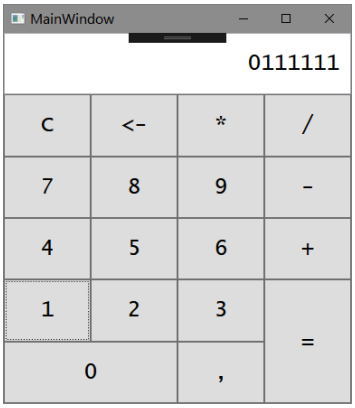
Теперь добавим код к обработчику:

```
private void ButtonNumber_Click(object sender, RoutedEventArgs e)
{
    textBox.Text += (sender as Button)?.Content;

}
```

Что означает знак вопроса в коде?

Проверим работу кнопки "1" (пока мы назначили обработчик только ей):

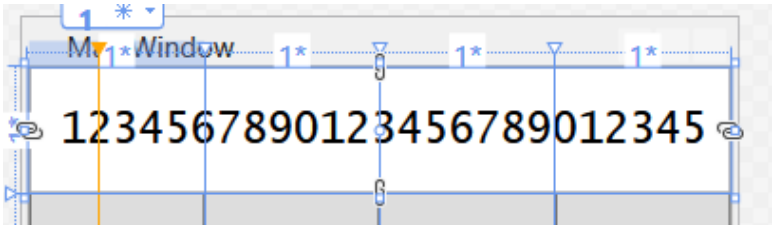


Очевидно, что "0" не нужно оставлять в начале строки. Если в строке содержится только "0", заменим строку, иначе – добавим:

```
private void ButtonNumber_Click(object sender, RoutedEventArgs e)
{
    if(textBox.Text == "0")
        textBox.Text = (sender as Button)?.Content.ToString();
    else
        textBox.Text += (sender as Button)?.Content;

}
```

И ограничим длину строки разумными пределами.
Проверим длину текстового поля в конструкторе – у меня помещается 25 символов.



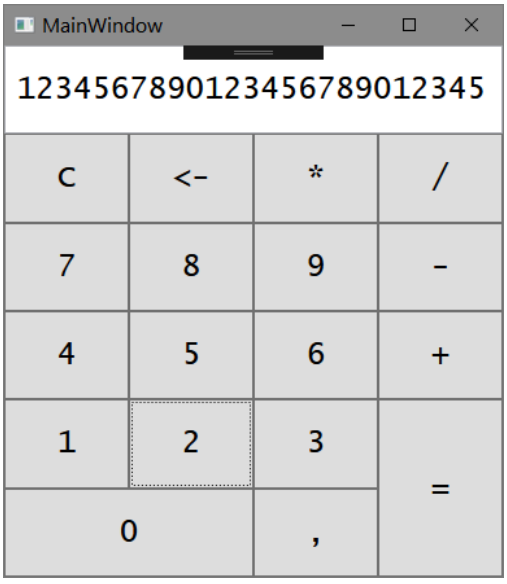
Сделаем так, чтобы новые символы добавлялись, только если в строке есть место:

```
private void ButtonNumber_Click(object sender, RoutedEventArgs e)
{
    if(textBox.Text == "0")
        textBox.Text = (sender as Button)?.Content.ToString();
    else
        if(textBox.Text.Length < 25)
            textBox.Text += (sender as Button)?.Content;

}
```

Теперь назначим этот обработчик всем цифровым кнопкам:

```
Click="ButtonNumber_Click"
```



Убедимся, что лишние символы не добавляются.

6. Сделаем отдельный обработчик для запятой.

Запятая может быть в строке только одна. Поэтому запятую добавляем, только если в строке она не найдена.

```
private void ButtonPoint_Click(object sender, RoutedEventArgs e)
{
    if (textBox.Text.IndexOf(",") < 0)
        textBox.Text +=  ",";
}
```

Что возвращает IndexOf?

7. Теперь научим наш калькулятор считать. Обработчики для кнопок с операциями

Мы реализуем простейший алгоритм калькулятора. Пользователь вводит число, потом знак операции (число при этом исчезает из текстового поля), а затем второе число. При нажатии на "=" получаем результат, который можно использовать в дальнейших вычислениях.

При нажатии на клавишу с операцией число из текстового поля должно где-то сохраняться. Заведем для этого переменную в виде поля класса

```
public partial class MainWindow : Window
{
    double x;
```

Так же должен сохраняться знак операции

```
public partial class MainWindow : Window
{
    double x;
    char oper;
```

И само текстовое поле должно очищаться

```
private void ButtonOper_Click(object sender, RoutedEventArgs e)
{
    x = Convert.ToDouble(textBox.Text);
    oper = (sender as Button).Content.ToString()[0];
    textBox.Text = "0";
}
```

Назначим этот обработчик всем кнопкам с операциями

8. Вычисления выполняются при нажатии на кнопку "=".

При этом извлекаются сохраненное число и знак операции.

```
private void ButtonEnter_Click(object sender, RoutedEventArgs e)
{
    double y = Convert.ToDouble(textBox.Text);
    double result = 0;
    switch (oper)
    {
        case '+': result = x + y; break;
        case '-': result = x - y; break;
        case '*': result = x * y; break;
        case '/': result = x / y; break;
    }
    textBox.Text = result.ToString();
}
```

Проверьте работу калькулятора.
Протестируйте деление на 0, попытку вычисления без знака операции, попытайтесь перемножить очень большие числа.

9. Самостоятельная часть

Самостоятельно запрограммируйте кнопку "C" – полная очистка памяти и поля калькулятора.
Кнопка "<-" удаляет последний введенный символ.

Задание 2. Конвертеры значений (по вариантам)

Разработать приложение, которое будет конвертировать указанные величины. Приложение должно содержать 2 строки – конвертируемое значение и результат. Должно использоваться не менее 5 видов конвертируемых величин (например, 5 видов валют). Выбор конвертируемой величины осуществляется при помощи ComboBox. Значение величины вводится в текстовое поле.

Вариант 1. Конвертер валют.
Вариант 2. Конвертер длины.
Вариант 3. Конвертер площади.
Вариант 4. Конвертер массы.
Вариант 5. Конвертер объема.
Вариант 6. Конвертер времени.
Вариант 7. Конвертер скорости.
Вариант 8. Конвертер температуры.
Вариант 9. Конвертер системы счисления.
Вариант 10. Конвертер данных.