**Министерство науки и высшего образования российской федерации**

**ФГАОУ ВО «Пермский государственный национальный исследовательский университет»**

Физический факультет

Технологии .NET Framework

Лабораторная работа №11 на тему

**«****Организация интерфейса windows-приложений. Часть 2.»**

Выполнил:

студент 2 курса физического факультета

группа ПМИ-2-2021

направление «01.03.02 Прикладная математика и информатика»

профиль «Инженерия программного обеспечения»

Макарова Полина Фёдоровна

Пермь, 2023 г.

1. Цель работы

Цель лабораторной работы: научиться создавать приложение Windows в среде MS Visual Studio и программировать алгоритмы с использованием простых элементов управления.

• разработка приложения с использованием WPF (Windows Presentation Foundation);

• изучение основ XAML (eXtensible Application Markup Language);

• обработка событий от элементов управления.

1. Листинг программы

|  |
| --- |
| *Листинг 1 — MainWindow.xaml.cs* |
| /\* Макарова Полина ПМИ-2  \* Вариант 10  \* Лабораторная работа 11  \*/  using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  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;  namespace WPF\_CSH11  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : Window  {  public MainWindow()  {  InitializeComponent();  ((Button)RootLayout.FindName("confirmButton")).Click += ConfirmButton\_Click;  }  private void ConfirmButton\_Click(object sender, RoutedEventArgs e)  {  double x1, x2, x3, x4, x5, results;  //Проверка на правильность введённыхх данных  try  {  //Присвоение соответствующим переменным введённых данных  x1 = Convert.ToDouble(((TextBox)RootLayout.FindName("TextBox1")).Text);  x2 = Convert.ToDouble(((TextBox)RootLayout.FindName("TextBox2")).Text);  x3 = Convert.ToDouble(((TextBox)RootLayout.FindName("TextBox3")).Text);  x4 = Convert.ToDouble(((TextBox)RootLayout.FindName("TextBox4")).Text);  x5 = Convert.ToDouble(((TextBox)RootLayout.FindName("TextBox5")).Text);  results = Math.Round((x1 - x2 + Math.Abs(Math.Cos(x3) + Math.Sqrt(Math.Abs(x4)))) / (Math.Pow(x1, x5) - Math.Log(x2)), 3);  // Проверяем вторую кнопку на вывод  if (RadioButton2.IsChecked == true)  {  // Проверка на вывод  if (CheckBox1.IsChecked == true)  {  // Вывод на листбокс  ListBox1.Items.Add(string.Format("arg01 = " + x1 +  " | arg02 = " + x2 +  " | arg03 = " + x3 +  " | arg04 = " + x4 +  " | arg05 = " + x5 +  " | Результат = " + results.ToString()));  }  if (CheckBox2.IsChecked == true)  {  // Вывод в файл  string str = "arg01 = " + x1 +  " | arg02 = " + x2 +  " | arg03 = " + x3 +  " | arg04 = " + x4 +  " | arg05 = " + x5 +  " | Результат = " + results.ToString() + Environment.NewLine;  StreamWriter SW = new StreamWriter(((TextBox)RootLayout.FindName("TextBox7")).Text, true); // Открываем файл на дозапись  SW.Write(string.Format(str)); //Записываем строку  SW.Close(); //Закрываем файл  }  }  }  catch (FormatException)  {  ListBox1.Items.Add("Неверно введены данные!");  }  }  }  } |

|  |
| --- |
| *Листинг 1 — XAML* |
| <Window x:Class="WPF\_CSH11.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:WPF\_CSH11"  mc:Ignorable="d"  Title="MainWindow" Height="560" Width="800">  <Grid x:Name="RootLayout" Background="DarkSeaGreen">  <Grid.RowDefinitions>  <RowDefinition Height="10"/>  <RowDefinition Height="Auto"/>  <RowDefinition Height="Auto"/>  <RowDefinition Height="Auto"/>  <RowDefinition Height="Auto"/>  <RowDefinition Height="10"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="10"/>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="10"/>  </Grid.ColumnDefinitions>  <Image Source="C:\Users\User\source\repos\WPF\_CSH11\in\Photo.png" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2"/>  <ListBox x:Name="ListBox1" Grid.Row="2" Grid.Column="1" Grid.ColumnSpan="2" Height="150"/>    <GroupBox x:Name="GroupBox1" Header="Задание и входные параметры" Background="CadetBlue" Grid.Row="3" Grid.Column="1" Grid.ColumnSpan="1">  <StackPanel>  <StackPanel Orientation="Horizontal">  <Label Content=" arg01 = " Width="70"/>  <TextBox x:Name="TextBox1" Width="250"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Label Content=" arg02 = " Width="70"/>  <TextBox x:Name="TextBox2" Width="250"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Label Content=" arg03 = " Width="70"/>  <TextBox x:Name="TextBox3" Width="250"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Label Content=" arg04 = " Width="70"/>  <TextBox x:Name="TextBox4" Width="250"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Label Content=" arg05 = " Width="70"/>  <TextBox x:Name="TextBox5" Width="250"/>  </StackPanel>  </StackPanel>  </GroupBox>  <GroupBox x:Name="GroupBox2" Header="Вывод лога тестов" Background="CadetBlue" Grid.Row="3" Grid.Column="2">  <StackPanel x:Name="writeStackPanel">  <RadioButton x:Name="RadioButton1" Content="Не выводить" IsChecked="True" VerticalAlignment="Center"/>  <RadioButton x:Name="RadioButton2" Content="Выводить" VerticalAlignment="Center"/>  <CheckBox x:Name="CheckBox1" Content="На экран" VerticalAlignment="Center"/>  <CheckBox x:Name="CheckBox2" Content="В файл" VerticalAlignment="Center"/>  <StackPanel x:Name="PathStackPanel">  <Label Content="Путь до файла"/>  <TextBox x:Name="TextBox7" Height="25"/>  </StackPanel>  </StackPanel>  </GroupBox>  <Button x:Name="confirmButton" Content="Результат" Height="40" Grid.Row="4" Grid.Column="1"/>  </Grid>  </Window> |

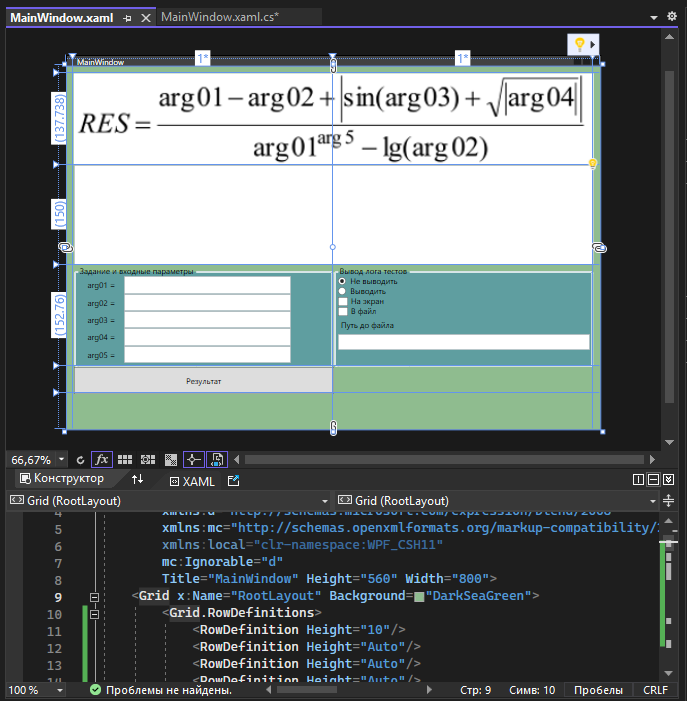


Рисунок — Вид приложения в программе

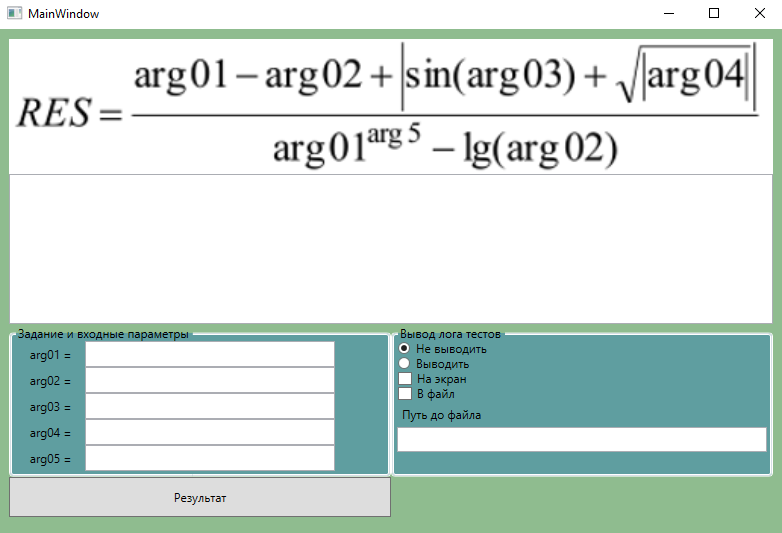


Рисунок — Вид программы после запуска

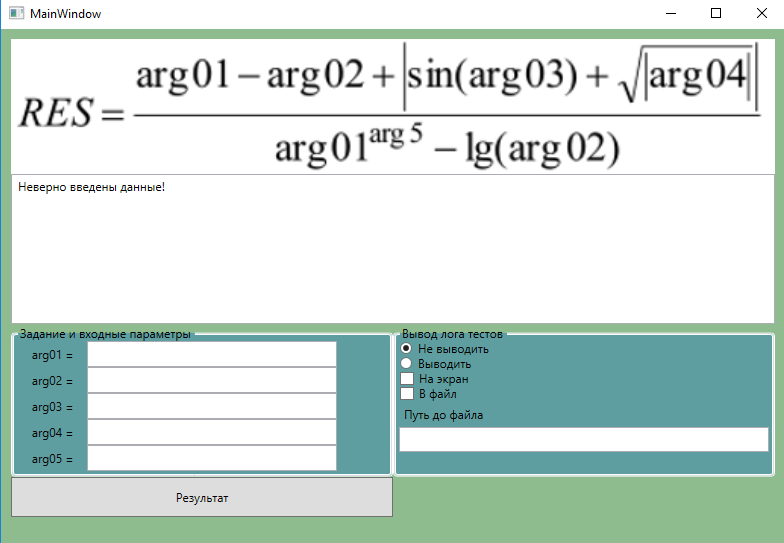


Рисунок — Результат выполнения программы 1

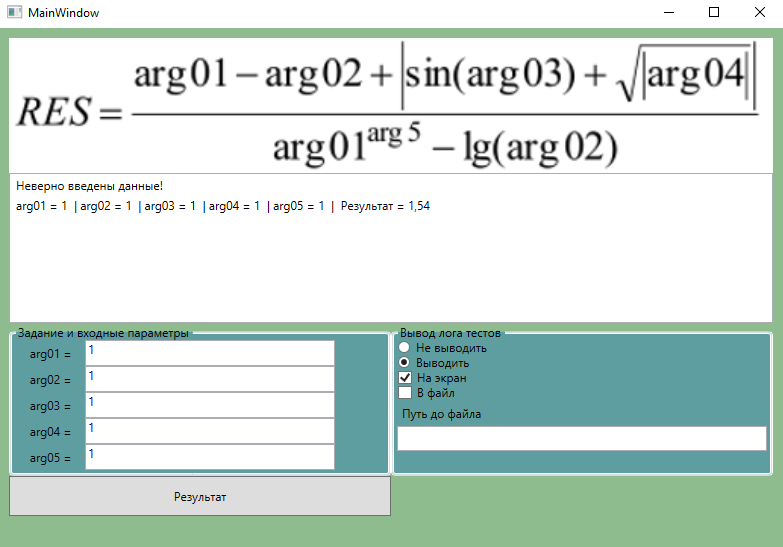


Рисунок — Результат выполнения программы 2

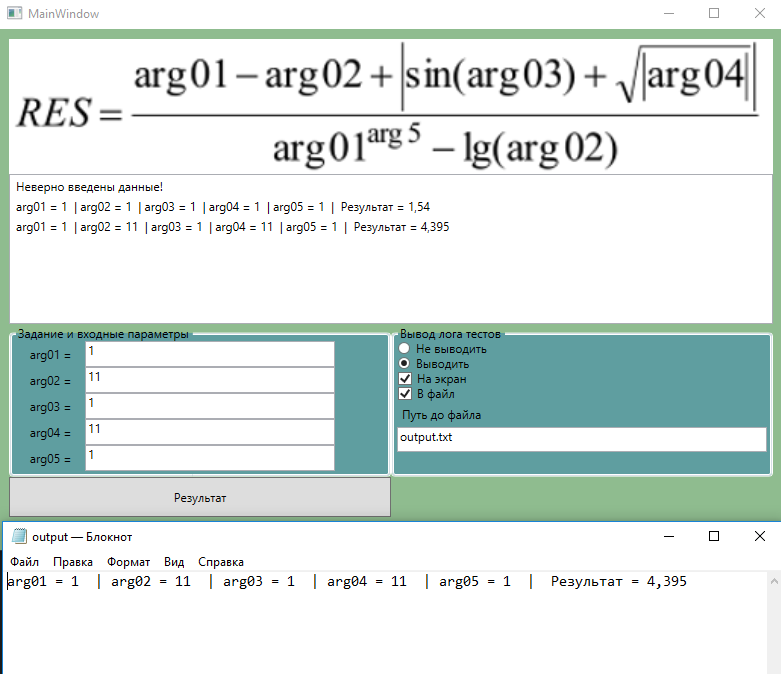


Рисунок — Результат выполнения программы 3

1. Ответы на вопросы

***1. Перечислите специальные контейнеры WPF?***

Grid, UniformGrid, StackPanel, WrapPanel, DockPanel, Canvas.

***2. Какое назначение метода InitializeComponent() в коде класса?***

Метод InitializeComponent() осуществляет вызов метода LoadComponent() класса System.Windows.Application, который извлекает код из BAML (скомпилированный XAML) из сборки и использует его для построения пользовательского интерфейса.

***3. Из каких объектов можно сформировать основное меню приложения?***

Нужно использовать объект класса Window с различными объявленными свойствами и пространствами имен.

Атрибут x:Class задает имя частичного класса, который будет сгенерирован в code-behind файле для этого окна.

Атрибуты xmlns объявляют пространтсва имен xml для ядра WPF, языка xaml и локального пространства имен для текущего проекта.

Свойства Title, Width, Height задают текст, отображаемый в заголовке окна, и его размеры.

Элемент-контейнер <Grid> включен для размещения макета и других элементов пользовательского интерфейса, которые будут составлять визуальную иерархию окна.

***4. Приведите примеры простых и сложных атрибутов элементов***

Простой атрибут элемента состоит из значения, присвоенного свойству элемента.

Пример:



Сложные атрибуты – атрибут, который содержит несколько частей или вложенных элементов.

Пример:



***5. Перечислите способы как организовать взаимодействия между визуализацией xaml и кодом C#?***

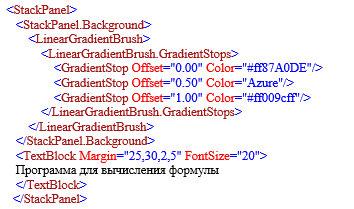
Code-behind. Можно использовать code-behind для определения обработчиков событий для элементов в XAML.

Команды - вместо того чтобы напрямую обрабатывать событие в code-behind, вы можете привязать команду к элементу в XAML и указать логику команды в коде C#,

Привязка данных для соединения элементов пользовательского интерфейса со свойствами или коллекциями в коде C# и автоматически обновлять пользовательский интерфейс при изменении данных.

Прикрепленное поведение нужно чтобы прикрепить пользовательскую логику к любому визуальному элементу пользовательского интерфейса. Они могут быть определены в коде C# и затем прикреплены к элементам в XAML с помощью синтаксиса прикрепленного свойства.

***6. Поясните приведенный ниже код, куда его можно вставить (файл, контейнер), прокомментируйте свойства и атрибуты.***



Приведенный код можно вставить в файл xaml, в контейнер Window или UserControl.

Код создает StackPanel, который имеет цвет фона с помощью градиентной кисти. Используемый тип кисти - LinearGradientBrush, который создает эффект градиентного цвета от одной точки к другой. В этом случае цвета определяются с помощью свойства GradientStop, которое задает несколько остановок цвета вдоль линии градиента. Определены три элемента GradientStop с различными значениями Offset и Color.

Внутри панели StackPanel есть также элемент TextBlock, который отображает текст "Программа для вычисления формулы". Margin текстового блока задает расстояние между текстовым блоком и краями панели StackPanel. FontSize изменяет размер отображаемого текста.

***7. Для чего в проекте есть файлы App.xaml и App.xaml.cs? Их назначение, как можно задействовать?***

Файлы App.xaml и App.xaml.cs являются основными элементами WPF-приложения.

Файл App.xaml содержит глобальный ресурс, который достпен во всем проекте. Он обеспечивает центральное место для хранения ресурсов и настроек, которые могут применяться во всем приложении. Файл описывает модель приложения.

Файл App.xaml.cs содержит логику code-behind для класса приложения. Он начинается с частичного объявления класса, предоставляющего доступ к экземпляру объекта Application, который управляет событиями времени жизни, такими как запуск, закрытие.

***8. Можно ли поместить изображение и текст на кнопку? В каком файле? Если да, приведите код.***

Да, на кнопку можно поместить текст и изображение. Для этого необходимо изменить файл App.xaml. Одним из способов является изменение шаблона кнопки. Еще нужно определить стиль ImageButton.

Пример:

<Button Content="Click Me!" Style="{StaticResource ImageButton}" />

<Style x:Key="ImageButton" TargetType="Button">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Grid Width="{TemplateBinding Width}"

Height="{TemplateBinding Height}">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Image Source="<image\_path>"

VerticalAlignment="Center"

HorizontalAlignment="Left"/>

<ContentPresenter Grid.Column="1"

VerticalAlignment="Center"

HorizontalAlignment="Stretch"

RecognizesAccessKey="True"/>

</Grid>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>