**25 Технология Windows Presentation Foundation**

Задание №1. Изучить теоретический материал. Выполнить на все примеры создания WPF приложения.

Листинг программы:

public MainWindow()

{

InitializeComponent();

ButtonRun.Click += ButtonRun\_Click\_1;

ButtonAbout.Click += ButtonAbout\_Click;

}

private void ButtonRun\_Click\_1(object sender, RoutedEventArgs e)

{

String hello = "Hello, ";

String input = string.Empty;

if (string.IsNullOrEmpty(TextBoxEnterName.Text) || string.IsNullOrWhiteSpace(TextBoxEnterName.Text))

{

input = "World!";

}

else

{

input = TextBoxEnterName.Text;

}

TextBlockHello.Text = $"{hello} {input}";

}

private void ButtonAbout\_Click(object sender, RoutedEventArgs e)

{

string info = $"About program{this.Title}";

AboutApp aboutApp = new AboutApp(info);

aboutApp.ShowDialog();

}

Таблица 25.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Aleksey | Hello, Aleksey |

Источник: собственная разработка

Анализ результатов представлен на рисунке 25.1.

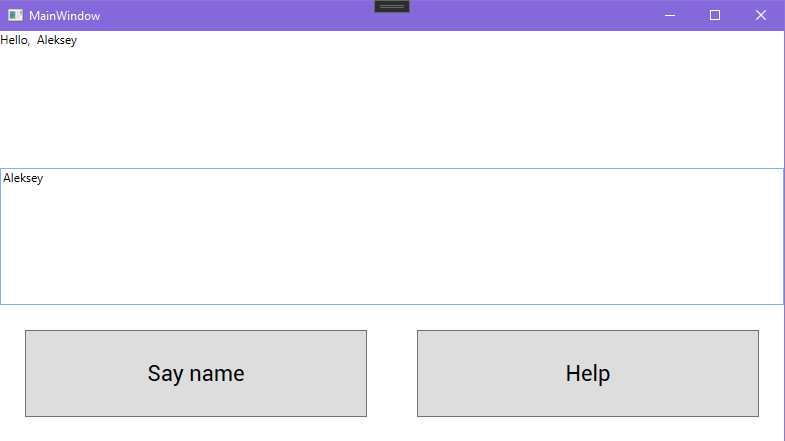


Рисунок 25.1 – Результат работы программы

Источник: собственная разработка

Задание №2. Переделать задание 1 таким образом, чтобы главная форма приняла следующий вид и все элементы выполняли необходимые действия. Изменить название главной формы на “Приложение Hello Name”.

Листинг программы:

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void TextBox1\_GotFocus(object sender, RoutedEventArgs e)

{

if (TextBox1.Text == "Введите имя")

{

TextBox1.Text = "";

}

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

LabelName.Text = "Hello, " + TextBox1.Text;

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

MessageBox.Show("Enter your name and click button 'Enter name'");

}

Таблица 25.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| James | Hello, James |

Источник: собственная разработка

Анализ результатов представлен на рисунке 25.2.

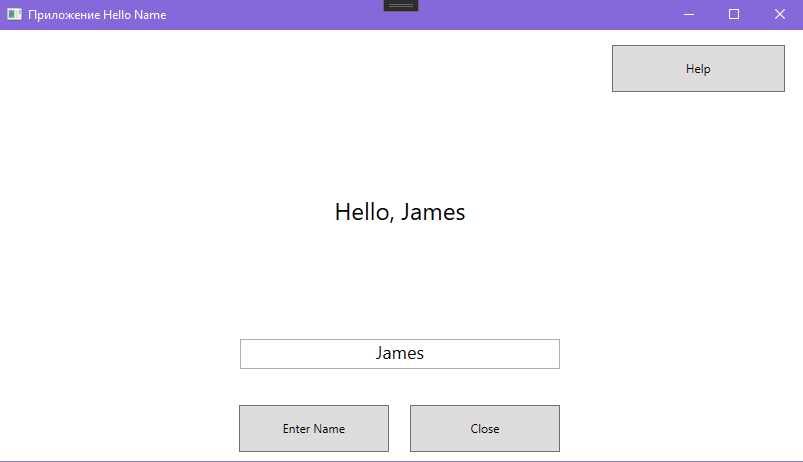


Рисунок 25.2 – Результат работы программы

Источник: собственная разработка

Задание №3. Создайте проект WPF и выполните решение алгоритма y = sin2x. Элементы управления Button TextBlock, TextBox. Расположение элементов продумайте самотоятельно.

Построить график функции. Таблицу данных получить путём изменения параметра X с шагом h. Самостоятельно выбрать удобные параметры настройки.

Листинг программы:

public ChartValues<double> Values1 { get; set; }

private void Button\_Click(object sender, RoutedEventArgs e)

{

try

{

double x = double.Parse(txX.Text);

double xMin = double.Parse(textboxXmin.Text);

double xMax = double.Parse(Xmax.Text);

double step = double.Parse(Step.Text);

Values1 = FunctionGraph(xMin, xMax, step, x);

DataContext = this;

}

catch (Exception ex)

{

MessageBox.Show($"An error occured - {ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private ChartValues<double> FunctionGraph(double xMin, double xMax, double step, double x)

{

ChartValues<double> values = new ChartValues<double>();

for (double i = xMin; i <= xMax; i += step)

{

double result = Math.Pow(Math.Sin(i),2);

values.Add(result);

}

return values;

}

Таблица 25.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| -10, 10, 3, 1 | Function graph |

Источник: собственная разработка

Анализ результатов представлен на рисунке 25.3.

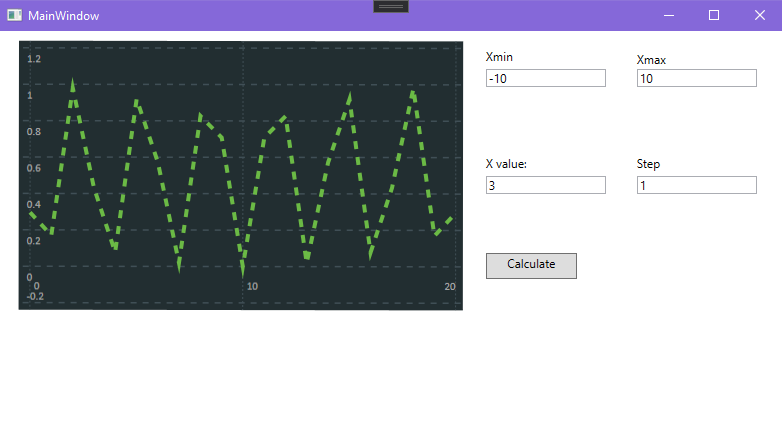


Рисунок 25.3 – Результат работы программы

Источник: собственная разработка

Задание №4. Написать программу с кнопкой. При наведении курсора на элемент управления он должен создавать новую кнопку, а при клике удалять себя.

Листинг программы:

public partial class MainWindow : Window

{

Random Random = new Random();

private int count = 0;

public MainWindow()

{

InitializeComponent();

}

private void Button\_MouseEnter(object sender, MouseEventArgs e)

{

Button btn = new Button();

btn.MouseEnter += MouseEnter;

btn.Click += Button\_Click;

btn.Content = "Child button " + Convert.ToString(count);

btn.Margin = new Thickness(Random.Next(0, 430), Random.Next(0, 400), Random.Next(0, 430), Random.Next(0, 500));

form1.Children.Add(btn);

count++;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

form1.Children.Remove(button);

}

private void MouseEnter(object sender, MouseEventArgs e)

{

Button button = new Button();

button.MouseEnter += MouseEnter;

button.Click += Button\_Click;

button.Content = "Child's buttun child";

button.Margin = new Thickness(Random.Next(0, 430), Random.Next(0, 400), Random.Next(0, 430), Random.Next(0, 500));

form1.Children.Add(button);

}

}

Таблица 25.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| MouseClick, MouseEnter | Button, Delete button |

Источник: собственная разработка

Анализ результатов представлен на рисунке 25.4.

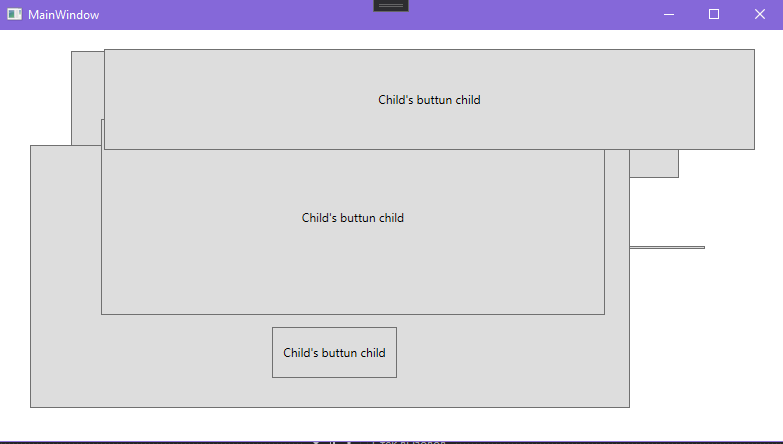


Рисунок 25.4 – Результат работы программы

Источник: собственная разработка