**ИИТ БГУИР**

**Факультет повышения квалификации и переподготовки**   
  
**Дисциплина: Средства визуального программирования приложений**

**Лабораторная работа №6**

Разработка многопоточных приложений

с графическим интерфейсом

Вариант 3

**Выполнил слушатель гр. 40322-1** Проверила

**Гончаров Максим Евгеньевич Желакович Ирина Миролюбовна**

Минск 2025

*Цели лабораторной работы*:

Освоить принципы взаимодействия с элементами графического интерфейса в многопоточном приложении.

*Выполнение работы:*

Индивидуальное задание: интеграл ( 0 ; 1) от (1 + x)dx = 1.5

Разработано приложение для вычисления интеграла по методу прямоугольников.

<Window x:Class="LB6.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:LB6"

mc:Ignorable="d"

Title="Вычисление интеграла" Height="500" Width="700">

<Window.Resources>

<ComponentModel:BackgroundWorker x:Key="MyBackgroundWorker"

xmlns:ComponentModel="clr-namespace:System.ComponentModel;assembly=System"

WorkerReportsProgress="True"

WorkerSupportsCancellation="False"

DoWork="BackgroundWorker\_DoWork"

ProgressChanged="BackgroundWorker\_ProgressChanged"

RunWorkerCompleted="BackgroundWorker\_RunWorkerCompleted"/>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<!-- Панель параметров -->

<StackPanel Grid.Row="0" Orientation="Horizontal" HorizontalAlignment="Center" Margin="10">

<TextBlock Text="Параметры: " VerticalAlignment="Center" Margin="0,0,10,0"/>

<TextBlock x:Name="ParamsInfoText" Text="a=0, b=1, N=1000" VerticalAlignment="Center" Margin="0,0,10,0"/>

<Button x:Name="ButtonParams" Content="Изменить параметры" Width="120" Click="ButtonParams\_Click"/>

</StackPanel>

<ProgressBar x:Name="ProgressBar" Grid.Row="1" Width="600" Height="20" Margin="10" Minimum="0" Maximum="100"/>

<!-- Кнопки методов -->

<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center" Margin="10">

<Button x:Name="Button\_Dispatcher" Content="Dispatcher" Width="120" Margin="5" Click="Button\_Dispatcher\_Click"/>

<Button x:Name="Button\_BackgroundWorker" Content="BackgroundWorker" Width="120" Margin="5" Click="Button\_BackgroundWorker\_Click"/>

<Button x:Name="Button\_AsyncStream" Content="Async Stream" Width="120" Margin="5" Click="Button\_AsyncStream\_Click"/>

</StackPanel>

<!-- ListBox для отображения хода вычислений -->

<ListBox x:Name="ResultsListBox" Grid.Row="3" Margin="10" Height="200"/>

<!-- Итоговый результат -->

<TextBlock x:Name="ResultTextBlock" Grid.Row="4" Text="Результат: " Margin="10" HorizontalAlignment="Center" FontWeight="Bold"/>

</Grid>

</Window>

using LB6;

using System.ComponentModel;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Threading;

namespace LB6

{

public partial class MainWindow : Window

{

private double a = 0;

private double b = 1;

private int n = 1000;

public MainWindow()

{

InitializeComponent();

UpdateParamsInfo();

}

private void UpdateParamsInfo()

{

ParamsInfoText.Text = $"a={a}, b={b}, N={n}";

}

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

{

var paramsWindow = new ParametersWindow(a, b, n);

if (paramsWindow.ShowDialog() == true)

{

a = paramsWindow.A;

b = paramsWindow.B;

n = paramsWindow.N;

UpdateParamsInfo();

}

}

private void ClearResults()

{

ResultsListBox.Items.Clear();

ProgressBar.Value = 0;

ResultTextBlock.Text = "Вычисление...";

}

private void SetButtonsEnabled(bool enabled)

{

Button\_Dispatcher.IsEnabled = enabled;

Button\_BackgroundWorker.IsEnabled = enabled;

Button\_AsyncStream.IsEnabled = enabled;

ButtonParams.IsEnabled = enabled;

}

// === Dispatcher реализация ===

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

{

ClearResults();

SetButtonsEnabled(false);

Thread calculationThread = new Thread(new ThreadStart(CalculateWithDispatcher));

calculationThread.Start();

}

private void CalculateWithDispatcher()

{

double h = (b - a) / n;

double sum = 0.0;

for (int i = 0; i <= n; i++)

{

double x = a + h \* i;

if (i < n) sum += Function(x);

double currentIntegral = sum \* h;

double progress = (double)i / n;

// Обновляем UI через Dispatcher

Dispatcher.BeginInvoke(new Action(() =>

{

ProgressBar.Value = progress \* 100;

if (i % 10 == 0) // Чтобы не засорять ListBox, обновляем каждые 10 итераций

{

ResultsListBox.Items.Add($"Dispatcher: x={x:F2}, S={currentIntegral:F6}, progress={progress:P0}");

ResultsListBox.ScrollIntoView(ResultsListBox.Items[ResultsListBox.Items.Count - 1]);

}

}), DispatcherPriority.Normal);

}

double result = sum \* h;

Dispatcher.BeginInvoke(new Action(() =>

{

ResultTextBlock.Text = $"Результат (Dispatcher): {result:F6}";

SetButtonsEnabled(true);

}), DispatcherPriority.Normal);

}

// === BackgroundWorker реализация ===

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

{

ClearResults();

SetButtonsEnabled(false);

BackgroundWorker worker = (BackgroundWorker)this.FindResource("MyBackgroundWorker");

worker.RunWorkerAsync();

}

private void BackgroundWorker\_DoWork(object sender, DoWorkEventArgs e)

{

BackgroundWorker worker = sender as BackgroundWorker;

double h = (b - a) / n;

double sum = 0.0;

for (int i = 0; i <= n; i++)

{

double x = a + h \* i;

if (i < n) sum += Function(x);

double currentIntegral = sum \* h;

double progress = (double)i / n;

// Отправляем прогресс и промежуточные данные

worker.ReportProgress((int)(progress \* 100), new { x, currentIntegral, progress });

}

e.Result = sum \* h;

}

private void BackgroundWorker\_ProgressChanged(object sender, ProgressChangedEventArgs e)

{

ProgressBar.Value = e.ProgressPercentage;

// Получаем промежуточные данные

var data = e.UserState as dynamic;

if (data != null && (int)(data.progress \* n) % 10 == 0)

{

ResultsListBox.Items.Add($"BgWorker: x={data.x:F2}, S={data.currentIntegral:F6}, progress={data.progress:P0}");

ResultsListBox.ScrollIntoView(ResultsListBox.Items[ResultsListBox.Items.Count - 1]);

}

}

private void BackgroundWorker\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

if (e.Error != null)

{

ResultTextBlock.Text = $"Ошибка: {e.Error.Message}";

}

else

{

double result = (double)e.Result;

ResultTextBlock.Text = $"Результат (BgWorker): {result:F6}";

}

SetButtonsEnabled(true);

}

// === Асинхронный стрим ===

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

{

ClearResults();

SetButtonsEnabled(false);

try

{

var calculator = new IntegralCalculator(a, b, n);

IAsyncEnumerable<(double x, double S, double progress)> data = calculator.CalculateAsync();

double finalResult = 0;

await foreach (var (x, currentIntegral, progress) in data)

{

ProgressBar.Value = progress \* 100;

if ((int)(progress \* n) % 10 == 0)

{

ResultsListBox.Items.Add($"Async: x={x:F2}, S={currentIntegral:F6}, progress={progress:P0}");

ResultsListBox.ScrollIntoView(ResultsListBox.Items[ResultsListBox.Items.Count - 1]);

}

finalResult = currentIntegral;

}

ResultTextBlock.Text = $"Результат (Async Stream): {finalResult:F6}";

}

catch (Exception ex)

{

ResultTextBlock.Text = $"Ошибка: {ex.Message}";

}

finally

{

SetButtonsEnabled(true);

}

}

private double Function(double x)

{

return 1 + x;

}

}

}

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.

Алгоритм вычисления интеграла помещён в отдельный поток приложения.

В главном окне при помощи элемента ProgressBar отображается ход выполнения процесса вычисления.

<ProgressBar x:Name="ProgressBar" Grid.Row="1" Width="600" Height="20" Margin="10" Minimum="0" Maximum="100"/>

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Верхний и нижний предел интегрирования, а также число разбиений области интегрирования N вводится в отдельном диалоговом окне.

<Window x:Class="LB6.ParametersWindow"

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"

mc:Ignorable="d"

Title="Параметры интегрирования" Height="250" Width="300"

WindowStartupLocation="CenterOwner">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Нижний предел (a):" Grid.Row="0" Grid.Column="0" Margin="5" VerticalAlignment="Center"/>

<TextBox x:Name="TextBoxA" Grid.Row="0" Grid.Column="1" Margin="5" Text="0"/>

<TextBlock Text="Верхний предел (b):" Grid.Row="1" Grid.Column="0" Margin="5" VerticalAlignment="Center"/>

<TextBox x:Name="TextBoxB" Grid.Row="1" Grid.Column="1" Margin="5" Text="1"/>

<TextBlock Text="Число разбиений (N):" Grid.Row="2" Grid.Column="0" Margin="5" VerticalAlignment="Center"/>

<TextBox x:Name="TextBoxN" Grid.Row="2" Grid.Column="1" Margin="5" Text="1000"/>

<StackPanel Grid.Row="4" Grid.ColumnSpan="2" Orientation="Horizontal" HorizontalAlignment="Right" Margin="5">

<Button Content="OK" Width="80" Margin="5" Click="OkButton\_Click"/>

<Button Content="Отмена" Width="80" Margin="5" Click="CancelButton\_Click"/>

</StackPanel>

</Grid>

</Window>

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Для запуска процесса вычисления предусмотреть три кнопки. Одна кнопка должна реализовать поставленную задачу с помощью объекта Dispatcher, вторая кнопка - с помощью компонента System.ComponentModel.BackgroundWorker, третья - с помощью асинхронного стрима.

namespace LB6

{

public class IntegralCalculator

{

private double a;

private double b;

private int n;

public IntegralCalculator(double a, double b, int n)

{

this.a = a;

this.b = b;

this.n = n;

}

public async IAsyncEnumerable<(double x, double S, double progress)> CalculateAsync()

{

double h = (b - a) / n;

double sum = 0.0;

for (int i = 0; i <= n; i++)

{

double x = a + h \* i;

if (i < n)

{

sum += Function(x);

}

double currentIntegral = sum \* h;

double progress = (double)i / n;

// Небольшая задержка для наглядности

if (i % 10 == 0) // Обновляем каждые 10 итераций

{

await Task.Delay(1);

}

yield return (x, currentIntegral, progress);

}

}

private double Function(double x)

{

return 1 + x;

}

}

}

Во время вычисления интеграла кнопки недоступны:

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Код проектов всех лабораторных работ, задания и отчеты находятся в публичном доступе на сайте GitHub: <https://github.com/Maxevgen555/SVPP_LB>. Все проекты находятся в одном решении и выполнены в VS2022. Файл решения SVPP\_LB.sln находится в папке \ЛБ1\LB1\_Calc\.

*Вывод:*

Я получил навыки работы с принципами взаимодействия с элементами графического интерфейса в многопоточном приложении.