

ЛАБОРАТОРНАЯ РАБОТА №6

Разработка многопоточных приложений с графическим интерфейсом

1 Цель работы

Освоить принципы взаимодействия с элементами графического интерфейса в многопоточном приложении.

2 Постановка задачи

Разработать приложение для вычисления интеграла по методу прямоугольников:

$$\int_a^b f(x)dx \approx h * \sum_{i=1}^N f(x_i), \text{ где } x_i = a + h * i; h = (b - a)/N;$$

Поместить алгоритм вычисления интеграла в отдельный поток приложения. В главном окне при помощи элемента ProgressBar отображать ход выполнения процесса вычисления. Верхний и нижний предел интегрирования, а также число разбиений области интегрирования N вводить в отдельном диалоговом окне.

Для запуска процесса вычисления предусмотреть две кнопки. Одна кнопка должна реализовать поставленную задачу с помощью объекта Dispatcher, вторая кнопка — с помощью компонента System.ComponentModel.BackgroundWorker.

Во время вычисления интеграла обе кнопки должны быть недоступны.

3 Индивидуальные задания

1. $\int_0^{2\pi} \sin(x) dx = 0$

2. $\int_0^1 \sqrt{x} dx = 2/3$

3. $\int_0^1 (1 + x) dx = 1,5$

4. $\int_0^1 x^3 dx = 0,25$

5. $\int_0^1 x^4 dx = 0,2$

6. $\int_0^1 x^3 dx = 0,25$

$$7. \int_0^{\pi} \cos(2x) dx = 0$$

$$8. \int_0^1 x^9 dx = 0,1$$

$$9. \int_0^1 x^4 dx = 0,2$$

$$10. \int_0^1 2x dx = 1$$

4 Рекомендации к выполнению задания

4.1 Использование DispatcherObject

Запуск вычисления в отдельном потоке

```
private void btnStart_Click(object sender, RoutedEventArgs e)
{
    Thread t = new Thread(Calculate);
    t.Start();
}
```

Реализация метода Calculate:

```
private void Calculate()
{
    var step = Math.Round((double)(n / 100));

    for (int i = 0; i <= n; i++)
    {
        ... Здесь поместите код вычисления интеграла
        if (i % step == 0)
        {
            Dispatcher.BeginInvoke(DispatcherPriority.Normal,
                new Action(() => pBar.Value = i / step));
        }
    }
};
}
```

Если доступен .NET версии 4.5 и выше, можно реализовать асинхронные операции следующим образом:

```
private async void btnStart_Click(object sender, RoutedEventArgs e)
{
    await CalculateAsync();
}
```

Реализация метода CalculateAsync:

```
private Task CalculateAsync()
{
    var step = Math.Round((double)(n / 100));
    return Task.Run(() =>
    {
        for (int i = 0; i <= n; i++)
```

```

        {
            ... Здесь поместите код вычисления интеграла
            if(i % step==0)
            {
                Dispatcher.BeginInvoke(DispatcherPriority.Normal,
                    new Action(() => pBar.Value = i/step));
            }
        }
    });
}

```

4.2 Использование BackgroundWorker

В разметке окна подключите пространство имен System.ComponentModel и добавьте ресурс BackgroundWorker:

```

. . .
xmlns:thrd="clr-namespace:System.ComponentModel;assembly=System"
mc:Ignorable="d"
Title="MainWindow" Height="350" Width="525">
<Window.Resources>
    <thrd:BackgroundWorker x:Key="worker"
        DoWork="BackgroundWorker_DoWork"
        ProgressChanged="BackgroundWorker_ProgressChanged"
        RunWorkerCompleted="BackgroundWorker_RunWorkerCompleted"
        WorkerReportsProgress="True"
    />
</Window.Resources>

```

В коде окна подключите пространство имен System.ComponentModel, опишите переменную типа BackgroundWorker и в конструкторе окна проинициализируйте ее данными из ресурса окна:

```

BackgroundWorker backgroundWorker;

public MainWindow()
{
    InitializeComponent();
    backgroundWorker = (BackgroundWorker)this.Resources["worker"];
}

```

В обработчик события BackgroundWorker_DoWork поместите код вычисления интеграла. Внутри кода нужно сгенерировать событие ProgressChanged для управления элементом ProgressBar. Для этого используется метод backgroundWorker.ReportProgress(int progress):

```

private void BackgroundWorker_DoWork(object sender, DoWorkEventArgs e)
{
    var step = (int)Math.Round((double)(n / 100));
    for (int i = 0; i <= n; i++)
    {
        ... Здесь разместите код вычисления интеграла
        if (i % step == 0)
        {
            if(backgroundWorker!=null
                && backgroundWorker.WorkerReportsProgress)
            {
                backgroundWorker.ReportProgress(i / step);
            }
        }
    };
}

```

В обработчик события BackgroundWorker_ProgressChanged разместите код, управляющий показаниями элемента ProgressBar. Используйте свойство аргумента события e.ProgressPercentage.

В обработчике события нажатия кнопки запуска вычисления сделайте кнопки запуска недоступными и запустите backgroundWorker с помощью функции backgroundWorker.RunWorkerAsync();

В обработчике события BackgroundWorker_RunWorkerCompleted разрешите использование кнопок запуска вычисления.