Министерство образования Республики Беларусь

Оршанский колледж ВГУ имени П.М.Машерова

**Отчет**

по учебной практике по программированию

«Модуль 3. Задание по делегатам»

по предмету «Конструирование программ и языки программирования»

Выполнил учащийся Казаченко Н.С.

группы 3ПОИС23 1.10.2025 г.

Проверил Алейников М.А.

1.10.2025 г.

Орша, 2025

**Цель:** закрепить знания по делегатам в языке программирования С#.

**Модуль 3 Задания по делегатам:**

1. Создайте базовый класс "Фигура" с методом для вычисления площади. Затем создайте производные классы для разных геометрических фигур (круг, прямоугольник, треугольник) и используйте делегат для динамического вызова метода вычисления площади.
2. Реализуйте систему событий для мобильного приложения. Создайте класс "Уведомление" с событиями для отправки уведомлений (сообщения, звонки, электронные письма). Зарегистрируйте обработчики событий для разных типов уведомлений.
3. Создайте приложение для управления задачами с использованием делегатов. Пользователь должен иметь возможность добавлять задачи и выбирать делегата для выполнения каждой задачи (например, отправка уведомления или запись в журнал).
4. Разработайте систему фильтрации данных с использованием делегатов. Пользователь должен иметь возможность выбрать фильтр для списка данных (например, фильтр по дате или по ключевым словам).
5. Создайте приложение для сортировки числовых данных. Пользователь должен иметь возможность выбрать метод сортировки (например, сортировка пузырьком или быстрая сортировка) с помощью делегатов.

**Ход работы**

**Выполнения задания 1**

**Листинг 1.** Код из файла Class1.cs

using System;

namespace Задание\_1

{

// Базовый класс "Фигура"

public abstract class Figure

{

// Метод для вычисления площади (будет переопределён в наследниках)

public abstract double CalculateArea();

}

// Производный класс "Круг"

public class Circle : Figure

{

public double Radius { get; set; }

public Circle(double radius)

{

Radius = radius;

}

public override double CalculateArea()

{

return Math.PI \* Radius \* Radius;

}

}

// Производный класс "Прямоугольник"

public class Rectangle : Figure

{

public double Width { get; set; }

public double Height { get; set; }

public Rectangle(double width, double height)

{

Width = width;

Height = height;

}

public override double CalculateArea()

{

return Width \* Height;

}

}

// Производный класс "Треугольник"

public class Triangle : Figure

{

public double Base { get; set; }

public double Height { get; set; }

public Triangle(double baseLength, double height)

{

Base = baseLength;

Height = height;

}

public override double CalculateArea()

{

return 0.5 \* Base \* Height;

}

}

}

**Листинг 2.** Код из файла MainWindow.xaml

<Window x:Class="Задание\_1.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:Задание\_1"

mc:Ignorable="d"

Title="Калькулятор площади геометрических фигур" Height="450" Width="800">

<Window.Background>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="LightBlue" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid>

<StackPanel Margin="20">

<TextBlock Text="Выберите фигуру:" FontWeight="Bold" FontSize="16" Margin="0,0,0,10"/>

<ComboBox x:Name="ShapeSelector" Width="200" Margin="0,0,0,20">

<ComboBoxItem Content="Круг" />

<ComboBoxItem Content="Прямоугольник" />

<ComboBoxItem Content="Треугольник" />

</ComboBox>

<StackPanel x:Name="InputFields" Margin="0,0,0,20"></StackPanel>

<Button Content="Рассчитать площадь" Width="200" Click="CalculateAreaButton\_Click" Margin="0,0,0,20"/>

<TextBlock x:Name="ResultTextBlock" FontSize="16" FontWeight="Bold"/>

</StackPanel>

</Grid>

</Window>

**Листинг 3.** Код из файла MainWindow.xaml.cs

using System;

using System.Windows;

namespace Задание\_1

{

public partial class MainWindow : Window

{

// Делегат для вызова метода вычисления площади

private delegate double AreaCalculatorDelegate();

public MainWindow()

{

InitializeComponent();

ShapeSelector.SelectionChanged += ShapeSelector\_SelectionChanged;

}

private void ShapeSelector\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

InputFields.Children.Clear();

if (ShapeSelector.SelectedItem is System.Windows.Controls.ComboBoxItem selectedShape)

{

switch (selectedShape.Content.ToString())

{

case "Круг":

AddInputField("Радиус:");

break;

case "Прямоугольник":

AddInputField("Ширина:");

AddInputField("Высота:");

break;

case "Треугольник":

AddInputField("Основание:");

AddInputField("Высота:");

break;

}

}

}

private void AddInputField(string label)

{

var stackPanel = new System.Windows.Controls.StackPanel { Orientation = System.Windows.Controls.Orientation.Horizontal, Margin = new Thickness(0, 5, 0, 5) };

stackPanel.Children.Add(new System.Windows.Controls.TextBlock { Text = label, Width = 100 });

stackPanel.Children.Add(new System.Windows.Controls.TextBox { Width = 100 });

InputFields.Children.Add(stackPanel);

}

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

{

if (ShapeSelector.SelectedItem is System.Windows.Controls.ComboBoxItem selectedShape)

{

AreaCalculatorDelegate areaCalculator = null;

Figure figure = null;

try

{

switch (selectedShape.Content.ToString())

{

case "Круг":

double radius = double.Parse(((System.Windows.Controls.TextBox)((System.Windows.Controls.StackPanel)InputFields.Children[0]).Children[1]).Text);

figure = new Circle(radius);

break;

case "Прямоугольник":

double width = double.Parse(((System.Windows.Controls.TextBox)((System.Windows.Controls.StackPanel)InputFields.Children[0]).Children[1]).Text);

double height = double.Parse(((System.Windows.Controls.TextBox)((System.Windows.Controls.StackPanel)InputFields.Children[1]).Children[1]).Text);

figure = new Rectangle(width, height);

break;

case "Треугольник":

double baseLength = double.Parse(((System.Windows.Controls.TextBox)((System.Windows.Controls.StackPanel)InputFields.Children[0]).Children[1]).Text);

double triangleHeight = double.Parse(((System.Windows.Controls.TextBox)((System.Windows.Controls.StackPanel)InputFields.Children[1]).Children[1]).Text);

figure = new Triangle(baseLength, triangleHeight);

break;

}

if (figure != null)

{

areaCalculator = figure.CalculateArea;

ResultTextBlock.Text = $"Площадь: {areaCalculator()}";

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка ввода данных: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

}

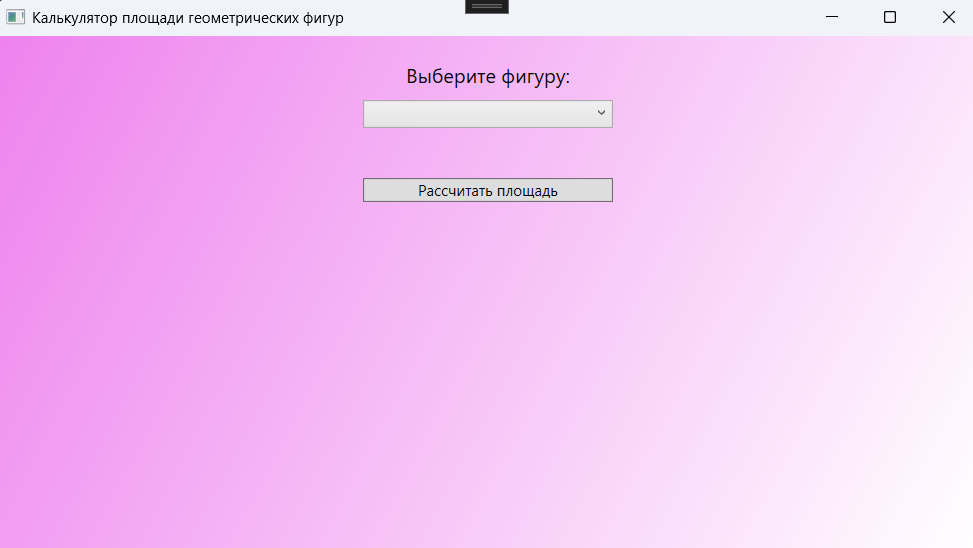
На рисунках 1–4 представлено выполнение задания создания базового класса Фигура с методом для вычисления площади.

Рисунок 1 – Запуск программы из задания 1

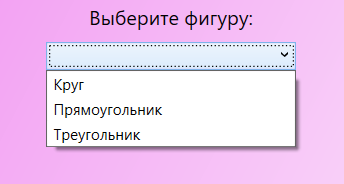


Рисунок 2 – Выбор фигуры

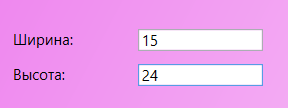


Рисунок 3 – Ввод данных о фигуре

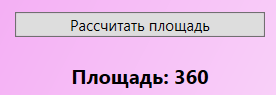


Рисунок 4 – Вывод результата

**Выполнения задания 2**

**Листинг 4.** Код из файла Class1.cs

using System;

namespace Задание\_2

{

public class Notification

{

// События для отправки уведомлений

public event Action<string> OnMessageSent;

public event Action<string> OnCallMade;

public event Action<string> OnEmailSent;

// Метод для отправки сообщения

public void SendMessage(string message)

{

OnMessageSent?.Invoke(message);

}

// Метод для совершения звонка

public void MakeCall(string phoneNumber)

{

OnCallMade?.Invoke(phoneNumber);

}

// Метод для отправки электронного письма

public void SendEmail(string email)

{

OnEmailSent?.Invoke(email);

}

}

}

**Листинг 5.** Код из файла MainWindow.xaml

<Window x:Class="Задание\_2.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:Задание\_2"

mc:Ignorable="d"

Title="Notification System" Height="500" Width="800">

<Window.Background>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="LightCoral" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid>

<StackPanel Margin="20">

<TextBlock Text="Система уведомлений" FontWeight="Bold" FontSize="18" Margin="0,0,0,20"/>

<TextBlock Text="Отправить сообщение:" FontWeight="Bold"/>

<TextBox x:Name="MessageTextBox" Width="300" Margin="0,5,0,10"/>

<Button Content="Отправить сообщение" Width="200" Click="SendMessageButton\_Click" Margin="0,0,0,20"/>

<TextBlock Text="Совершить звонок:" FontWeight="Bold"/>

<TextBox x:Name="CallTextBox" Width="300" Margin="0,5,0,10"/>

<Button Content="Совершить звонок" Width="200" Click="MakeCallButton\_Click" Margin="0,0,0,20"/>

<TextBlock Text="Отправить электронное письмо:" FontWeight="Bold"/>

<TextBox x:Name="EmailTextBox" Width="300" Margin="0,5,0,10"/>

<Button Content="Отправить письмо" Width="200" Click="SendEmailButton\_Click" Margin="0,0,0,20"/>

<TextBlock x:Name="NotificationLogTextBlock" FontSize="14" FontWeight="Bold" Margin="0,20,0,0"/>

</StackPanel>

</Grid>

</Window>

**Листинг 6.** Код из файла MainWindow.xaml.cs

using System.Windows;

namespace Задание\_2

{

public partial class MainWindow : Window

{

private Notification \_notification;

public MainWindow()

{

InitializeComponent();

\_notification = new Notification();

// Регистрация обработчиков событий

\_notification.OnMessageSent += MessageSentHandler;

\_notification.OnCallMade += CallMadeHandler;

\_notification.OnEmailSent += EmailSentHandler;

}

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

{

string message = MessageTextBox.Text;

if (!string.IsNullOrWhiteSpace(message))

{

\_notification.SendMessage(message);

}

else

{

MessageBox.Show("Введите текст сообщения.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

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

{

string phoneNumber = CallTextBox.Text;

if (!string.IsNullOrWhiteSpace(phoneNumber))

{

\_notification.MakeCall(phoneNumber);

}

else

{

MessageBox.Show("Введите номер телефона.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

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

{

string email = EmailTextBox.Text;

if (!string.IsNullOrWhiteSpace(email))

{

\_notification.SendEmail(email);

}

else

{

MessageBox.Show("Введите адрес электронной почты.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

// Обработчики событий

private void MessageSentHandler(string message)

{

NotificationLogTextBlock.Text = $"Сообщение отправлено: {message}";

}

private void CallMadeHandler(string phoneNumber)

{

NotificationLogTextBlock.Text = $"Звонок совершен на номер: {phoneNumber}";

}

private void EmailSentHandler(string email)

{

NotificationLogTextBlock.Text = $"Письмо отправлено на: {email}";

}

}

}

На рисунках 5-7 представлено выполнение задания создания класса Уведомление с событиями для отправки уведомлений (сообщения, звонки, электронные письма).

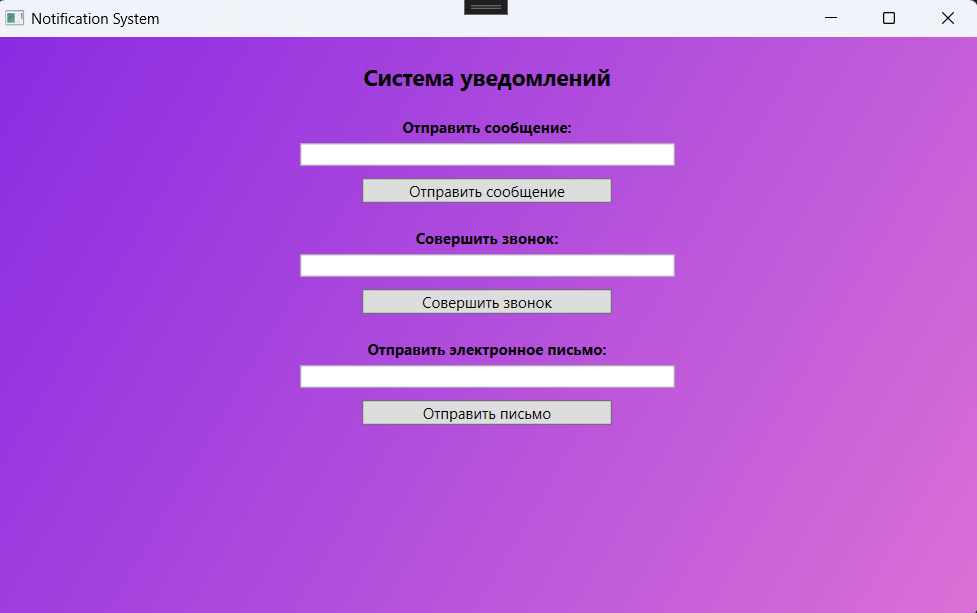


Рисунок 5 – Запуск программы из задания 2

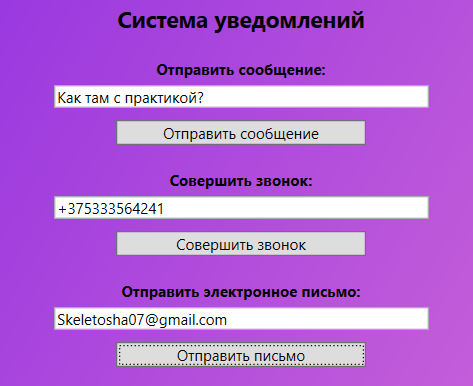


Рисунок 6 – Ввод данных

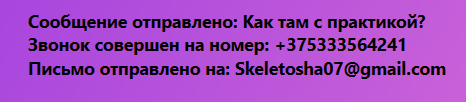


Рисунок 7 – Вывод данных

**Выполнения задания 3**

**Листинг 7.** Код из файла Class1.cs

using System;

namespace Задание\_3

{

public class TaskItem

{

public string Description { get; set; }

// Делегат для выполнения задачи

public Action<string> TaskAction { get; set; }

public TaskItem(string description, Action<string> taskAction)

{

Description = description;

TaskAction = taskAction;

}

// Метод для выполнения задачи

public void Execute()

{

TaskAction?.Invoke(Description);

}

}

}

**Листинг 8.** Код из файла MainWindow.xaml

<Window x:Class="Задание\_3.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:Задание\_3"

mc:Ignorable="d"

Title="Task Management App" Height="600" Width="800">

<Window.Background>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="LightGreen" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid>

<StackPanel Margin="20">

<TextBlock Text="Управление задачами" FontWeight="Bold" FontSize="18" Margin="0,0,0,20"/>

<TextBlock Text="Описание задачи:" FontWeight="Bold"/>

<TextBox x:Name="TaskDescriptionTextBox" Width="400" Margin="0,5,0,10"/>

<TextBlock Text="Выберите действие для задачи:" FontWeight="Bold"/>

<ComboBox x:Name="ActionSelector" Width="400" Margin="0,5,0,10">

<ComboBoxItem Content="Отправить уведомление"/>

<ComboBoxItem Content="Записать в журнал"/>

</ComboBox>

<Button Content="Добавить задачу" Width="200" Click="AddTaskButton\_Click" Margin="0,0,0,20"/>

<TextBlock Text="Список задач:" FontWeight="Bold" Margin="0,20,0,10"/>

<ListBox x:Name="TaskListBox" Width="600" Height="200" Margin="0,5,0,20"/>

<Button Content="Выполнить выбранную задачу" Width="200" Click="ExecuteTaskButton\_Click" Margin="0,0,0,20"/>

<TextBlock x:Name="LogTextBlock" FontSize="14" FontWeight="Bold" Margin="0,20,0,0"/>

</StackPanel>

</Grid>

</Window>

**Листинг 9.** Код из файла MainWindow.xaml.cs

using System;

using System.Windows;

using System.Windows.Controls;

namespace Задание\_3

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

// Добавление задачи в список

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

{

string description = TaskDescriptionTextBox.Text;

if (!string.IsNullOrWhiteSpace(description) && ActionSelector.SelectedItem is ComboBoxItem selectedAction)

{

Action<string> taskAction = null;

switch (selectedAction.Content.ToString())

{

case "Отправить уведомление":

taskAction = SendNotification;

break;

case "Записать в журнал":

taskAction = WriteToLog;

break;

}

TaskItem task = new TaskItem(description, taskAction);

TaskListBox.Items.Add(task);

TaskDescriptionTextBox.Clear();

ActionSelector.SelectedIndex = -1;

}

else

{

MessageBox.Show("Введите описание задачи и выберите действие.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

// Выполнение выбранной задачи

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

{

if (TaskListBox.SelectedItem is TaskItem selectedTask)

{

selectedTask.Execute();

TaskListBox.Items.Remove(selectedTask);

}

else

{

MessageBox.Show("Выберите задачу для выполнения.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

// Действие: Отправка уведомления

private void SendNotification(string message)

{

LogTextBlock.Text = $"Уведомление отправлено: {message}";

}

// Действие: Запись в журнал

private void WriteToLog(string message)

{

LogTextBlock.Text = $"Запись в журнал: {message}";

}

}

}

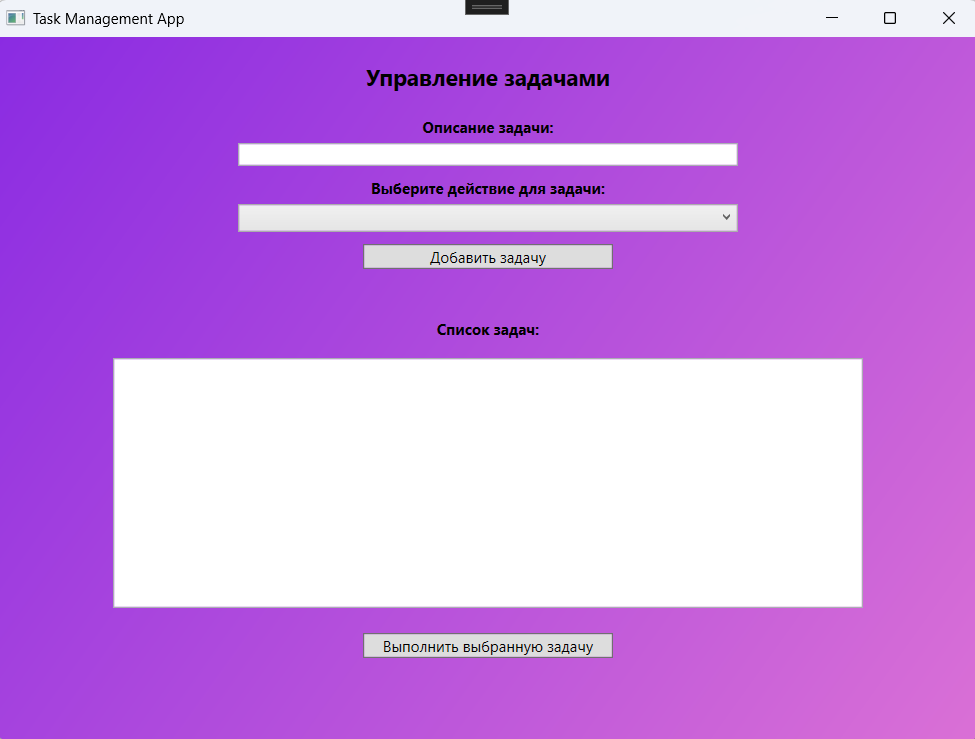
На рисунках 8–12 представлено выполнение задания создания приложения для управления задачами с использованием делегатов. 

Рисунок 8 – Запуск программы из задания 3



Рисунок 9 – Описание задачи

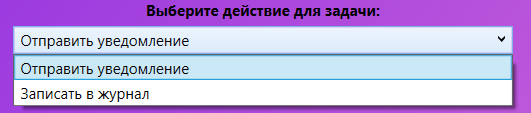


Рисунок 10 – Добавление задачи в список

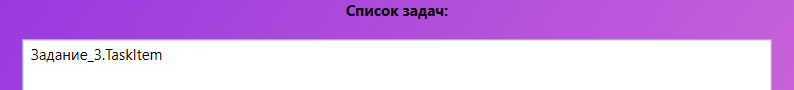


Рисунок 11 – Список задач после добавления



Рисунок 12 – Выполнение задачи из списка

**Выполнения задания 4**

**Листинг 10.** Код из файла Class1.cs

using System;

namespace Задание\_4

{

public class DataItem

{

public string Title { get; set; }

public DateTime Date { get; set; }

public DataItem(string title, DateTime date)

{

Title = title;

Date = date;

}

public override string ToString()

{

return $"{Title} - {Date.ToShortDateString()}";

}

}

}

**Листинг 11.** Код из файла MainWindow.xaml

<Window x:Class="Задание\_4.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:Задание\_4"

mc:Ignorable="d"

Title="Data Filtering App" Height="600" Width="800">

<Window.Background>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="LightBlue" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid>

<StackPanel Margin="20">

<TextBlock Text="Фильтрация данных" FontWeight="Bold" FontSize="18" Margin="0,0,0,20"/>

<TextBlock Text="Данные для фильтрации:" FontWeight="Bold"/>

<ListBox x:Name="DataListBox" Width="600" Height="200" Margin="0,5,0,20"/>

<TextBlock Text="Выберите фильтр:" FontWeight="Bold"/>

<ComboBox x:Name="FilterSelector" Width="400" Margin="0,5,0,10">

<ComboBoxItem Content="Фильтр по дате (после)"/>

<ComboBoxItem Content="Фильтр по ключевому слову"/>

</ComboBox>

<TextBlock Text="Введите значение фильтра:" FontWeight="Bold"/>

<TextBox x:Name="FilterValueTextBox" Width="400" Margin="0,5,0,20"/>

<Button Content="Применить фильтр" Width="200" Click="ApplyFilterButton\_Click" Margin="0,0,0,20"/>

<TextBlock Text="Отфильтрованные данные:" FontWeight="Bold" Margin="0,20,0,10"/>

<ListBox x:Name="FilteredDataListBox" Width="600" Height="200"/>

</StackPanel>

</Grid>

</Window>

**Листинг 12.** Код из файла MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace Задание\_4

{

public partial class MainWindow : Window

{

private List<DataItem> \_dataItems;

private List<DataItem> \_filteredDataItems;

public MainWindow() {

InitializeComponent();

InitializeData();

}

private void InitializeData()

{

// Инициализация данных

\_dataItems = new List<DataItem>

{

new DataItem("Событие A", new DateTime(2024, 10, 1)),

new DataItem("Событие B", new DateTime(2024, 10, 5)),

new DataItem("Событие C", new DateTime(2024, 10, 8)),

new DataItem("Событие D", new DateTime(2024, 9, 25)),

new DataItem("Конференция", new DateTime(2024, 11, 12))

};

\_filteredDataItems = new List<DataItem>(\_dataItems);

UpdateDataList();

}

private void UpdateDataList()

{

DataListBox.Items.Clear();

foreach (var item in \_dataItems)

{

DataListBox.Items.Add(item);

}

FilteredDataListBox.Items.Clear();

foreach (var item in \_filteredDataItems)

{

FilteredDataListBox.Items.Add(item);

}

}

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

{

if (FilterSelector.SelectedItem is ComboBoxItem selectedFilter && !string.IsNullOrWhiteSpace(FilterValueTextBox.Text))

{

Func<DataItem, bool> filter = null;

switch (selectedFilter.Content.ToString())

{

case "Фильтр по дате (после)":

if (DateTime.TryParse(FilterValueTextBox.Text, out DateTime filterDate))

{

filter = item => item.Date > filterDate;

}

else

{

MessageBox.Show("Введите корректную дату.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

break;

case "Фильтр по ключевому слову":

string keyword = FilterValueTextBox.Text.ToLower();

filter = item => item.Title.ToLower().Contains(keyword);

break;

}

if (filter != null)

{

\_filteredDataItems = \_dataItems.Where(filter).ToList();

UpdateDataList();

}

}

else

{

MessageBox.Show("Выберите фильтр и введите значение.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

}

}

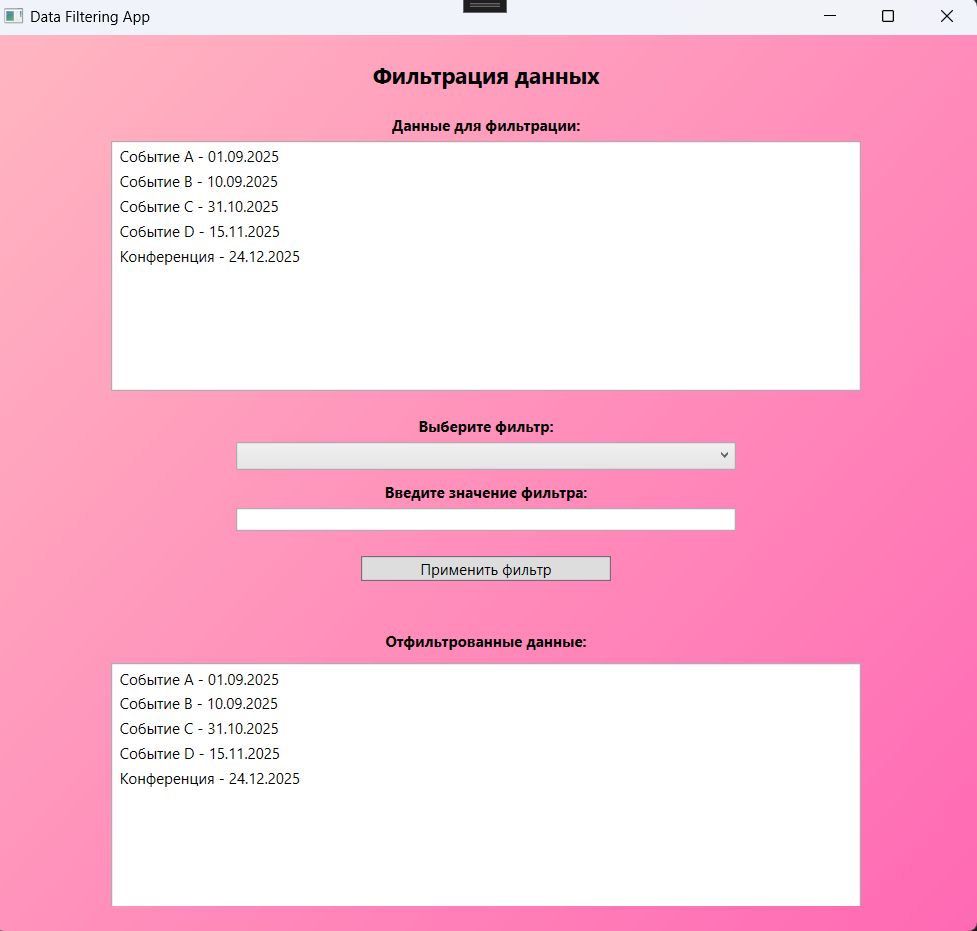
На рисунке 13–16 представлено выполнение задания создания системы фильтрации данных с использованием делегатов.

Рисунок 13 – Запуск программы из задания 4

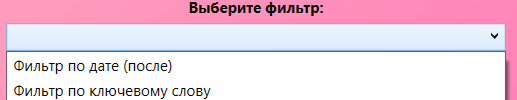


Рисунок 14 – Выбор фильтра

****

Рисунок 15 – Ввод значений фильтра

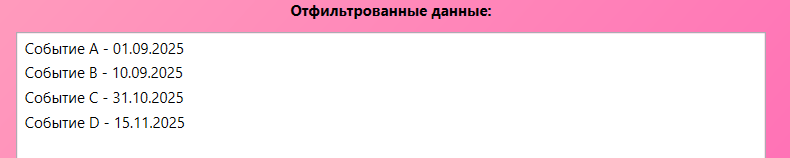
****

Рисунок 16 – Вывод отфильтрованных данных

**Выполнения задания 5**

**Листинг 13.** Код из файла Class1.cs

namespace Задание\_5

{

public class Sorter

{

// Метод сортировки пузырьком

public static void BubbleSort(int[] array)

{

int length = array.Length;

for (int i = 0; i < length - 1; i++)

{

for (int j = 0; j < length - i - 1; j++)

{

if (array[j] > array[j + 1])

{

// Обмен

int temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

// Метод быстрой сортировки

public static void QuickSort(int[] array, int left, int right)

{

int i = left, j = right;

int pivot = array[(left + right) / 2];

while (i <= j)

{

while (array[i] < pivot) i++;

while (array[j] > pivot) j--;

if (i <= j)

{

// Обмен

int temp = array[i];

array[i] = array[j];

array[j] = temp;

i++;

j--;

}

}

// Рекурсивные вызовы

if (left < j)

QuickSort(array, left, j);

if (i < right)

QuickSort(array, i, right);

}

}

}

**Листинг 14.** Код из файла MainWindow.xaml

<Window x:Class="Задание\_5.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:Задание\_5"

mc:Ignorable="d"

Title="Sorting App" Height="600" Width="800">

<Window.Background>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="LightSalmon" Offset="0" />

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Window.Background>

<Grid>

<StackPanel Margin="20">

<TextBlock Text="Сортировка числовых данных" FontWeight="Bold" FontSize="18" Margin="0,0,0,20"/>

<TextBlock Text="Введите числа (через запятую):" FontWeight="Bold"/>

<TextBox x:Name="NumbersTextBox" Width="400" Margin="0,5,0,10"/>

<TextBlock Text="Выберите метод сортировки:" FontWeight="Bold"/>

<ComboBox x:Name="SortMethodSelector" Width="400" Margin="0,5,0,10">

<ComboBoxItem Content="Сортировка пузырьком"/>

<ComboBoxItem Content="Быстрая сортировка"/>

</ComboBox>

<Button Content="Сортировать" Width="200" Click="SortButton\_Click" Margin="0,0,0,20"/>

<TextBlock Text="Результат:" FontWeight="Bold" Margin="0,20,0,10"/>

<TextBox x:Name="ResultTextBox" Width="400" Height="100" Margin="0,5,0,20" IsReadOnly="True" TextWrapping="Wrap"/>

</StackPanel>

</Grid>

</Window>

**Листинг 15.** Код из файла MainWindow.xaml.cs

using System;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace Задание\_5

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

string input = NumbersTextBox.Text;

if (string.IsNullOrWhiteSpace(input))

{

MessageBox.Show("Введите хотя бы одно число.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Преобразование строки в массив чисел

int[] numbers = input.Split(new[] { ',' }, StringSplitOptions.RemoveEmptyEntries)

.Select(n => int.TryParse(n.Trim(), out int num) ? num : 0)

.ToArray();

// Проверка на пустые значения

if (numbers.Length == 0)

{

MessageBox.Show("Введите корректные числа.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Определение метода сортировки

Action<int[]> sortingMethod = null;

if (SortMethodSelector.SelectedItem is ComboBoxItem selectedMethod)

{

switch (selectedMethod.Content.ToString())

{

case "Сортировка пузырьком":

sortingMethod = Sorter.BubbleSort;

break;

case "Быстрая сортировка":

sortingMethod = array => Sorter.QuickSort(array, 0, array.Length - 1);

break;

}

// Сортировка и вывод результата

sortingMethod?.Invoke(numbers);

ResultTextBox.Text = string.Join(", ", numbers);

}

else

{

MessageBox.Show("Выберите метод сортировки.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

}

}

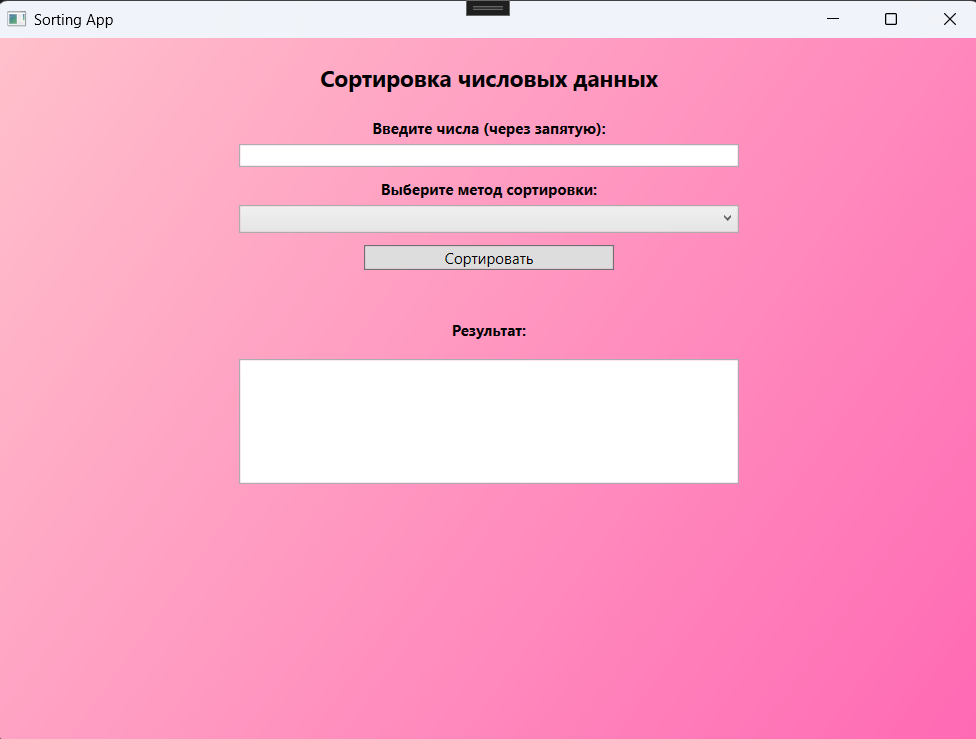
На рисунках 17–20 представлено выполнение задания создания приложения для сортировки числовых данных с использованием делегатов.

Рисунок 17 – Запуск программы из задания 5



Рисунок 18– Ввод данных(чисел)

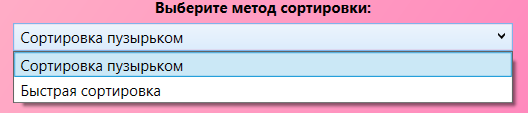


Рисунок 19 – Выбор метода сортировки

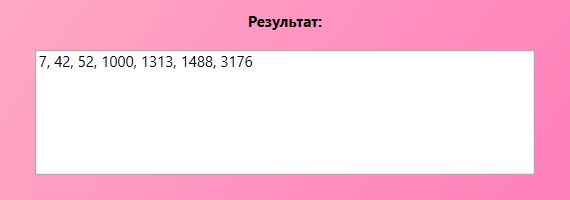


Рисунок 20 – Вывод результата сортировки

**Ссылка на репозиторий:** [**https://github.com/Seyko00/PraktikaPoProgrammirovaniu/tree/master/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%201.1**](https://github.com/Seyko00/PraktikaPoProgrammirovaniu/tree/master/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%201.1)

**Вывод:** в ходе выполнения практического задания модуля 3 были закреплены знания по делегатам в C#.