**ЗАДАНИЕ 13**

Задание 1. Создать 3 метода, возвращающих три различны сообщения. Создать делегат. В программе вызывать все три метода при помощи делегата.

Листинг программы:

using System;

public class Program

{

public delegate string GetMessage();

public static string GetMessage1()

{

return "Первое сообщение";

}

public static string GetMessage2()

{

return "Второе сообщение";

}

public static string GetMessage3()

{

return "Третье сообщение";

}

public static void Main()

{

GetMessage getMessageDelegate = GetMessage1;

getMessageDelegate += GetMessage2;

getMessageDelegate += GetMessage3;

foreach (GetMessage getMessage in getMessageDelegate.GetInvocationList())

{

Console.WriteLine(getMessage());

}

}

}

Таблица 13.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| - | Первое сообщение, Второе сообщение, Третье сообщение |

Анализ результатов:

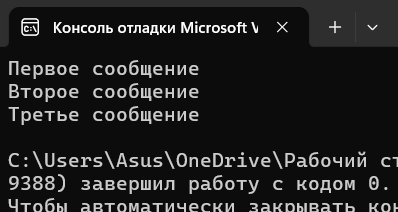


Рисунок 13.1 – Результат работы программы

Задание 2. На основе своей программы создать метод, в котором делегат будет параметром.

Листинг программы:

using System;

public class Program

{

public delegate string GetMessage();

public static string GetMessage1()

{

return "Первое сообщение";

}

public static string GetMessage2()

{

return "Второе сообщение";

}

public static string GetMessage3()

{

return "Третье сообщение";

}

public static void Main()

{

GetMessage getMessageDelegate = GetMessage1;

getMessageDelegate += GetMessage2;

getMessageDelegate += GetMessage3;

ExecuteDelegate(getMessageDelegate);

}

public static void ExecuteDelegate(GetMessage getMessage)

{

foreach (GetMessage messageGetter in getMessage.GetInvocationList())

{

Console.WriteLine(messageGetter());

}

}

}

Таблица 13.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| - | Первое сообщение, Второе сообщение, Третье сообщение |

Анализ результатов:

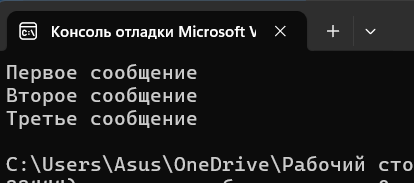


Рисунок 13.2 – Результат работы программы

Задание 3. Для класса MyInfo содержащего поле name (Ваше имя) создать событие выводящее оповещение в случае изменения значения поля name.

Листинг программы:

using System;

public class MyInfo

{

private string name;

public event EventHandler NameChanged;

public string Name

{

get { return name; }

set

{

if (name != value)

{

name = value;

OnNameChanged(EventArgs.Empty);

}

}

}

protected virtual void OnNameChanged(EventArgs e)

{

NameChanged?.Invoke(this, e);

}

}

public class Program

{

public static void Main()

{

MyInfo info = new MyInfo();

info.NameChanged += (sender, e) =>

{

Console.WriteLine("Имя изменилось.");

};

info.Name = "Новое имя";

}

}

Таблица 13.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Имя | Имя изменилось |

Анализ результатов:

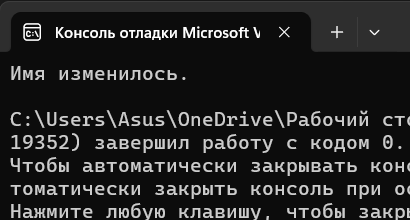


Рисунок 13.3 – Результат работы программы

Задание 4. Создать класс, в нем делегат и соответствующее ему событие. Создать два класса-наблюдателя, в методах которых будет описана их реакция на событие. Добавить 2 обработчика к событию из первого класса и один из второго. Вывести результат на экран. Удалить один обработчик события и вывести результат на экран.

Листинг программы:

using System;

public class Publisher

{

public delegate void MyEventHandler(object sender, EventArgs e);

public event MyEventHandler MyEvent;

public void TriggerEvent()

{

OnMyEvent(EventArgs.Empty);

}

protected virtual void OnMyEvent(EventArgs e)

{

MyEvent?.Invoke(this, e);

}

}

public class Observer1

{

public void Handler1(object sender, EventArgs e)

{

Console.WriteLine("Наблюдатель1 Обработчик1: Событие обработано.");

}

public void Handler2(object sender, EventArgs e)

{

Console.WriteLine("Наблюдатель1 Обработчик2: Событие обработано.");

}

}

public class Observer2

{

public void Handler3(object sender, EventArgs e)

{

Console.WriteLine("Наблюдатель2 Обработчик3: Событие обработано.");

}

}

public class Program

{

public static void Main()

{

Publisher publisher = new Publisher();

Observer1 observer1 = new Observer1();

Observer2 observer2 = new Observer2();

publisher.MyEvent += observer1.Handler1;

publisher.MyEvent += observer1.Handler2;

publisher.MyEvent += observer2.Handler3;

Console.WriteLine("Триггер события...");

publisher.TriggerEvent();

Console.WriteLine("\nУдаление одного обработчика у Наблюдателя1...");

publisher.MyEvent -= observer1.Handler2;

Console.WriteLine("Триггер события снова...");

publisher.TriggerEvent();

}

}

Таблица 13.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| - | Событие обработано. |

Анализ результатов:

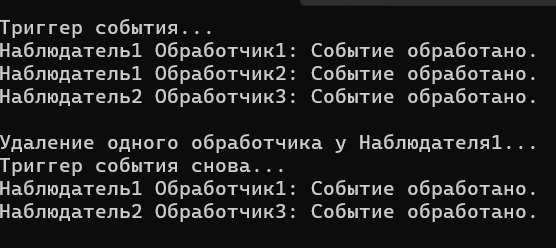


Рисунок 13.4 – Результат работы программы