Часть 1. Разработать программу, использующую делегаты. (В качестве примера можно использовать проект «Delegates»).

1. Программа должна быть разработана в виде консольного приложения на языке C#.

2. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.

3. Напишите метод, соответствующий данному делегату.

4. Напишите метод, принимающий разработанный Вами делегат, в качестве одного из входным параметров. Осуществите вызов метода, передавая в качестве параметра-делегата:

- метод, разработанный в пункте 3;
- лямбда-выражение.

5. Повторите пункт 4, используя вместо разработанного Вами делегата, обобщенный делегат Func< > или Action< >, соответствующий сигнатуре разработанного Вами делегата.

```
using System;

// Шаг 2: Определение делегата
delegate T MyDelegate<T, U, V>(U param1, V param2);


class Program
{
    // Шаг 3: Метод, соответствующий делегату
    static int MyMethod(string str, double num)
    {
        Console.WriteLine($"Input parameters: {str}, {num}");
        return str.Length + (int)num;
    }

    // Шаг 4: Метод, принимающий делегат
    static void UseDelegate(MyDelegate<string, double, int> myDelegate, string str, double num)
    {
        // Шаг 4: Осуществление вызова метода, передавая делегат
        int result = myDelegate(str, num);
        Console.WriteLine($"Delegate result: {result}");
    }
```

```
    static void Main()
    {
        // Шаг 5: Использование делегата и лямбда-выражения
        UseDelegate(MyMethod, "Hello", 3.14);
        UseDelegate((s, d) => s.Length + (int)d, "Lambda", 2.71);

        // Шаг 5: Использование обобщенного делегата Func<>
        Func<string, double, int> funcDelegate = MyMethod;
        int funcResult = funcDelegate("Func", 1.23);
        Console.WriteLine($"Func delegate result: {funcResult}");
    }
}
```

Часть 2. Разработать программу, реализующую работу с рефлексией. (В качестве примера можно использовать проект «Reflection»).

1. Программа должна быть разработана в виде консольного приложения на языке C#.

2. Создайте класс, содержащий конструкторы, свойства, методы. 7

3. С использованием рефлексии выведите информацию о конструкторах, свойствах, методах.

4. Создайте класс атрибута (унаследован от класса System.Attribute).

5. Назначьте атрибут некоторым свойствам классам. Выведите только те свойства, которым назначен атрибут.

6. Вызовите один из методов класса с использованием рефлексии.

```
using System;
using System.Reflection;

// Шаг 2: Создание класса с конструкторами, свойствами и методами
class MyClass
{
    public MyClass()
    {
        Console.WriteLine("Default Constructor");
    }
    public MyClass(int value)
    {
```

```
        Console.WriteLine($"Parameterized Constructor: {value}");
    }

    public string MyProperty { get; set; }

    public int MyMethod()
    {
        Console.WriteLine("MyMethod called");
        return 42;
    }
}

// Шаг 4: Создание класса атрибута
[AttributeUsage(AttributeTargets.Property)]
class MyAttribute : Attribute
{
}

class Program
{
    static void Main()
    {
        // Шаг 3: Использование рефлексии для вывода информации
        Type type = typeof(MyClass);
        Console.WriteLine($"Type: {type.FullName}");

        ConstructorInfo[] constructors = type.GetConstructors();
        foreach (var constructor in constructors)
        {
            Console.WriteLine($"Constructor: {constructor}");
        }
```

```
        PropertyInfo[] properties = type.GetProperties();
        foreach (var property in properties)
        {
            Console.WriteLine($"Property: {property}");
        }

        MethodInfo[] methods = type.GetMethods();
        foreach (var method in methods)
        {
            Console.WriteLine($"Method: {method}");
        }

        // Шаг 5: Назначение атрибута и вывод свойств с атрибутом
        PropertyInfo propertyInfo = type.GetProperty("MyProperty");
        propertyInfo.SetCustomAttribute(new MyAttribute());

        Console.WriteLine($"Properties with MyAttribute:");
        foreach (var property in properties)
        {
            if (Attribute.IsDefined(property, typeof(MyAttribute)))
            {
                Console.WriteLine($"Property: {property}");
            }
        }

        // Шаг 6: Вызов метода с использованием рефлексии
        MethodInfo methodInfo = type.GetMethod("MyMethod");
        object instance = Activator.CreateInstance(type);
        int result = (int)methodInfo.Invoke(instance, null);
        Console.WriteLine($"Method invocation result: {result}");
    }
}
```