Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА**

О Т Ч Е Т

по лабораторной работе № 6

Название: Делегаты

Дисциплина: Разработка приложений на языке C#

Студент ИУ6-75Б _____ И.Ю. Жосан
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ А.М. Минитаева
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Цель работы: изучить основы работы с делегатами в языке программирования C#.

Задание: Разработать программу, реализующую делегаты.

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 входных 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.

Выполнение задания:

Был определен делегат, принимающий 2 параметра (число int, строка string) и возвращающий значение произвольного типа:

```
public delegate Object myDelegate(int a, string b);
```

Был написан метод конкатенации числа и строки, соответствующий данному делегату MyDelegate:

```
private static Object concate_int_string(int a, string b)
{
    return a.ToString() + b;
}
```

Был написан метод, принимающий делегат MyDelegate в качестве одного из параметров. Второй параметр – число:

```
private static Object delegate_in_parameters(myDelegate d, int a)
{
    return d(a, " - string" + a.ToString());
}
```

Ниже приведен код основной программы, содержащий вызовы методов delegate_in_parameters, принимающих в качестве параметров:

— метод concate_int_string;

— лямбда-выражение `(int a,string b) => { return b.Length + a; }`, возвращающее длину предыдущей строки + a – 1;

— обобщенный делегат:
`Func<int,string,Object> func_delegate = (a,b) => b.Length==a,` возвращающий true, если длина предыдущей строки равна a, в остальных случаях – false.

```
static void Main()
        {
            int i = 5;
            myDelegate delegate_1 = new myDelegate(concate_int_string);
            Object obj = delegate_in_parameters(delegate_1, i);
            Console.WriteLine($"Concated int {i} and string.");
            Console.WriteLine(obj);

            obj = delegate_in_parameters((int a, string b) =>
                                    { return b.Length + a; }, i);
            Console.WriteLine($"\nLength of previous string + {i}- 1");
            Console.WriteLine(obj);

            Func<int, string, Object> func_delegate = (a, b) =>
                                                  b.Length == a;
            myDelegate delegate_2 = new myDelegate(func_delegate);
            obj = delegate_in_parameters(delegate_2, i);
            Console.WriteLine($"\nIs length of previous sting =={i}?");
            Console.WriteLine(obj);

            Console.ReadLine();
        }
```

Пример выполнения программы приведен на рисунке 1.

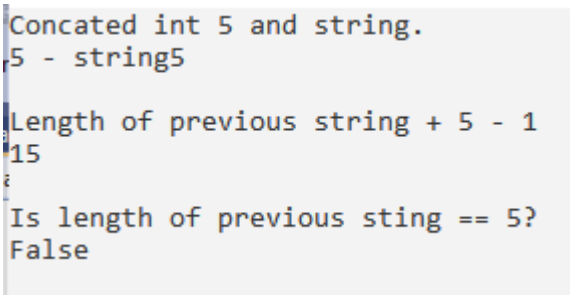

Рисунок 1 – Выполнение программы

Вывод: в процессе выполнения лабораторной работы были изучены средства работы с делегатами, лямбда-выражениями и обобщенным делегатом Func< >.