|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 09.03.01 **Прикладная информатика**

**Отчет**

**по лабораторной работе № 6**

**Дисциплина:** Разработка приложений на языке C#

**Название работы:** Делегаты



Студент гр. ИУ6-72Б **\_\_**01.10.2022**\_\_\_\_\_\_\_** И.С. Марчук

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** А.М. Минитаева

(Подпись, дата) (И.О. Фамилия)

Москва, 2022

**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.

Код программы:

using System;

namespace \_6

{

class Program

{

delegate int MetodDel(int x, string str, bool b);

static void Main(string[] args)

{

MetodDel del = Metod;//экз делегата

Console.WriteLine($"метод1, соответствующий созданному делегату:\t{del(12, "zazaz", true)}");//выводим metod

Console.WriteLine($"метод, принимающий разработанный делегат(параметр-делегата - метод1):\t{Metod2(del, 7)}");//выводим metod2 передавая ему делегат

Console.WriteLine($"метод, принимающий разработанный делегат(параметр-делегата - лямда-выражение:\t{Metod2((int x, string str, bool b) => x + str.Length, 7)}");//выводим metod2 передавая ему лямда-выражение

Func<int, string, bool, int> dFub = Metod;

Console.WriteLine($"Обобщенный метод, принимающий разработанный делегат(параметр-делегата - метод1 :{Metod3(dFub, 5)}");//выводим metod3 передавая ему об. делегат

Console.WriteLine($"метод, принимающий разработанный делегат(параметр-делегата - лямда-выражение:\t{Metod3((x, str, b) => x + str.Length, 5)}");//выводим лямда-выражение

Console.ReadKey();

}

static int Metod(int x, string str, bool b)//первый метод

{

int n = 0;

foreach (char o in str)

if (o == 'z')

n++;

int M = b ? n + x + str.Length : n + x;

return M;

}

static int Metod2(MetodDel f, int x)//второй метод принимающий в качестве параметра делегат

{

return x + f(x, "zez", false); ;

}

static int Metod3(Func<int, string, bool, int> F, int x)//третий метод принимающий в качестве параметра обобщенный делегат

{

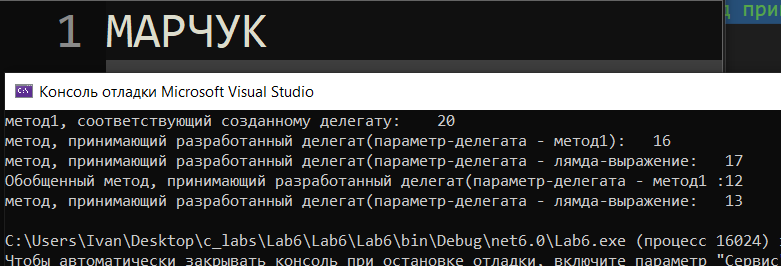
return x + F(x, "zez", false);

}

}

}

Работа программы показана на рисунке 1.



*Рисунок 1 – Работа программы*

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