

# Модуль 3, практическое занятие 1

**Делегаты**

**Анонимные методы**

**Лямбда-выражения,**

**Массивы делегатов, Многоадресные делегаты**

# Делегат

Метод:

`public static int AnyMethod(int AnyParameter)`

Тип делегата:






`delegate int AnyDelegateType(int TramPamPam)`

```
using System;
// Объявление делегата-типа:
delegate void SimpleDelegate();

class Program {
    static void F() {
        System.Console.WriteLine("Test.F");
    }
    static void Main() {
        // Инстанцирование делегата:
        SimpleDelegate d = new SimpleDelegate(F);
        d(); // Обращение к делегату и тем самым вызов метода
    }
}
```

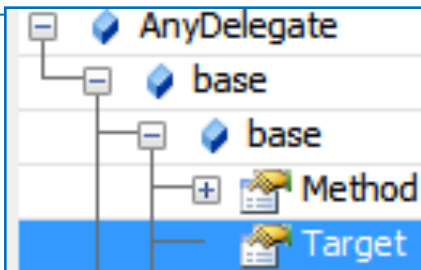
# Вызов статического метода через делегат

```
using System;
class AnyClass {
    public static int AnyMethod(int AnyParameter) { return ++AnyParameter; }
}
delegate int AnyDelegateType(int TramPamPam); // делегат-тип
class Program {
    static void Main() {
        AnyDelegateType AnyDelegate = // Объявление переменной-ссылки
            new AnyDelegateType(AnyClass.AnyMethod); //Инстанцирование делата
        int p;
        p = AnyDelegate(3); //Вызов метода через ссылку на экземпляр делегата
        Console.WriteLine(p); //4
    }
}
```

 AnyDelegate	{Method = {Int32 AnyMethod(Int32)}}
 base	{Method = {Int32 AnyMethod(Int32)}}
 base	{Method = {Int32 AnyMethod(Int32)}}
 Method	{Int32 AnyMethod(Int32)}
 Target	null

# Вызов нестатического метода через делегат

```
using System;
class AnyClass {
    public int AnyMethod(int AnyParameter) { return ++AnyParameter; }
}
delegate int AnyDelegateType(int TramPamPam);
class Program {
    static void Main() {
        AnyClass AnyObj = new AnyClass(); //Объявление объекта
        AnyDelegateType AnyDelegate = // Объявление переменной-ссылки
            new AnyDelegateType(AnyObj.AnyMethod); //Инстанцирование
        int p;
        p = AnyDelegate(3); //Вызов метода через экземпляр делегата
        Console.WriteLine(p); // Результат: 4
    }
}
```

	<pre>{Method = {Int32 AnyMethod(Int32)}} {Method = {Int32 AnyMethod(Int32)}} {Method = {Int32 AnyMethod(Int32)}} {Int32 AnyMethod(Int32)} {ConsoleApplication1.AnyClass}</pre>
--	--

# Использование анонимного метода

```
using System;
class Program {
    //Объявление делегата-типа
    delegate int AnyDelegateType(int TramPamPam);

    static void Main() {
        // Объявление переменной и инстанцирование делегата:
        AnyDelegateType Anonim = delegate (int AnyParameter) {
            return ++AnyParameter;
        };
        int p;
        p = Anonim(3); // Вызов анонимного метода через экземпляр делегата
        Console.WriteLine(p);
    }
}
```

Синтаксис  
анонимного  
метода

```
delegate
(<спецификация_параметров>) {
    <операторы_тела_метода>
}
```

# Использование лямбда-выражения

```
using System;
class Program {
    //Объявление делегата-типа
    delegate int AnyDelegateType(int TramPamPam);
    static void Main() {
        // Объявление переменной и инстанцирование:
        AnyDelegateType Lambda = x => ++x;
        int p;
        // Вызов лямбда-выражения через экземпляр делегата
        p = Lambda(3);
        Console.WriteLine(p);
    }
}
```

Синтаксис лямбда-выражения

(<спецификация\_параметров>) =>  
{ <операторы> }

## Задача 1

- Опишите делегат-тип **Cast** для представления методов с одним параметром типа **double** и возвращаемым значением типа **int**.
- Создайте два экземпляра типа **Cast**. Первый свяжите с анонимным методом, возвращающим ближайшее чётное целое к переданному в параметре вещественному числу. Второй – с анонимным методом, вычисляющим порядок переданного в параметре положительного числа.
- Протестируйте вызовы при помощи делегатов
  - на одном тестовом вещественном значении;
  - на нескольких тестовых вещественных значениях

## Задание к задаче 1

- Используя операцию += свяжите оба анонимных метода из задачи 1 с одним многоадресным делегатом. Вызовите методы через него.
- Замените анонимные методы лямбда-выражениями.



## Задача 2

В библиотеке классов определить два делегата-типа. Первый – для представления методов с целочисленным параметром, возвращающих ссылку на целочисленный массив. Второй для представления методов, параметр которых – ссылка на целочисленный массив, тип возвращаемого значения **void**.

В отдельной библиотеке классов объявить два статических метода.

- Первый метод формирует по введённому числу массив его цифр.
- Второй метод выводит массив на экран.

Для тестирования кода библиотеки разработайте консольное приложение.

В программе задайте пятизначное число и массив из 10 двузначных чисел. Создайте два экземпляра делегатов (первого и второго типа), свяжите с ними соответствующие методы из библиотеки.

Протестируйте их (для этого были созданы массив и число).

Для каждого делегата выведите на печать результаты вызова свойств **Method** и **Target**.

## Задача 3

- Декларируйте делегат-тип **delegateConvertTemperature**, представляющий методы с одним вещественным параметром и возвращающий вещественное значение.
- В нестатическом классе **TemperatureConverterImp** определите нестатические методы, преобразующие вещественное значение температуры из градусов Цельсия в градусы Фаренгейта и наоборот.

$$t_C = \frac{5}{9} \cdot (t_F - 32).$$

$$t_F = \frac{9}{5} \cdot t_C + 32.$$

- В основной программе свяжите с методами класса **TemperatureConverterImp** делегаты типа **delegateConvertTemperature** и протестируйте их (создайте экземпляры делегатов и выведите результат их применения к заданным в программе значениям).

Задача по материалам C# Delegates, Actions, Funcs, Lambdas—Keeping it super simple

[<https://blogs.msdn.microsoft.com/brunoterkaly/2012/03/02/c-delegates-actions-funcs-lambdaskeeping-it-super-simple/>]

## Задание к задаче 3

- Замените в методе **Main()** два делегата – массивом делегатов.
- Объявите класс **StaticTempConverters** со статическими методами перевода температур из градусов Цельсия в Кельвины, Ранкины и Реомюры и наоборот.
- В методе **Main()** в массив делегатов добавьте методы перевода температур из Цельсия в другие шкалы из классов **StaticTempConverters** и **TemperatureConverterImp**.
- Получая от пользователя значение температуры в Цельсиях выводить таблицу перевода в другие шкалы, с указанием шкалы.
- Таблица перевода между шкалами температур(<https://ru.wikipedia.org/wiki/Температура>)

# Задача 4

Применение массива делегатов для управления перемещением робота

*Код библиотеки классов...*

```
class Robot {  
    // класс для представления робота  
    int x, y; // положение робота на плоскости  
  
    public void Right() { x++; } // направо  
    public void Left() { x--; } // налево  
    public void Forward() { y++; } // вперед  
    public void Backward() { y--; } // назад  
  
    public string Position() { // сообщить координаты  
        return String.Format("The Robot position: x={0}, y={1}", x, y);  
    }  
}
```

```
delegate void Steps(); // делегат-тип
```

## Задача 4

*Код метода Main()...*

```
Robot rob = new Robot();    // конкретный робот
Steps[] trace = { new Steps(rob.Backward),
                  new Steps(rob.Backward),
                  new Steps(rob.Left)
};
// сообщить координаты
Console.WriteLine("Start:" + rob.Position());

for (int i = 0; i < trace.Length; i++) {
    Console.WriteLine("Method={0}, Target={1}",
        trace[i].Method, trace[i].Target);
    trace[i]();
}

Console.WriteLine(rob.Position());    // сообщить координаты
```

# Задача 4

Применение многоадресных делегатов для управления перемещением робота

*Код метода Main()...*

```
Robot rob = new Robot();    // конкретный робот
Steps delR = new Steps(rob.Right);    // направо
Steps delL = new Steps(rob.Left);    // налево
Steps delF = new Steps(rob.Forward);    // вперед
Steps delB = new Steps(rob.Backward);    // назад
// шаги по диагоналям (многоадресные делегаты):
Steps delRF = delR + delF;
Steps delRB = delR + delB;
Steps delLF = delL + delF;
Steps delLB = delL + delB;
delLB();
Console.WriteLine(rob.Position());    // сообщить координаты
delRB();
Console.WriteLine(rob.Position());    // сообщить координаты
Console.WriteLine("Для завершения нажмите любую клавишу.");
Console.ReadKey();
```

## Задание к задаче 4

1. В основной программе получать программу для робота в виде строки **S**. Каждая команда кодируется заглавной латинской буквой:
  - **R** (Right)
  - **L** (Left)
  - **F** (Forward)
  - **B** (Backward)
2. В многоадресный делегат добавить методы, в порядке, определённом программой **S**.
3. Запускать программу и выводить исходные и конечные координаты.

## Задание к задаче 4

- Разработайте для робота консольный интерфейс. Клетки – позиции текстового курсора на экране.
- Ограничения координат на поле получать от пользователя перед запуском робота.
- Программу в виде строки **S** получать от пользователя (см. предыдущий слайд)
- Робот отображается символом '\*' красного цвета.
- Позиции, в которых побывал робот отмечаются символом '+' серого цвета.
- Если программа робота выводит его за пределы поля – останавливать выполнение программы и сообщать об этом.



## Задача 5

Представив вычисление произведения и суммы с помощью лямбда-выражений, вычислить значение выражения с использованием делегатов для вызова методов:

$$\sum_{i=1}^5 \prod_{j=1}^5 \frac{i * x}{j}$$

## Задание для самостоятельного решения

- В библиотеке классов:
  - Декларируйте делегат-тип **MyDel**, представляющий методы с двумя целочисленными параметрами (**int**) , возвращающие целочисленное значение (**int**);
  - Декларируйте статический класс **TestClass** со статическим методом **TestMethod()**, возвращающий максимальное из двух переданных в качестве параметров целых чисел (**int**).
- В проекте консольного приложения:
  - В методе **Main()** связать ссылку типа **MyDel** с методом **TestMethod()** и протестировать вызов делегата для пар чисел, которые вводит с клавиатуры пользователь. Результаты работы метода, вызванного через делегат, выводить на экран.

## Задание для самостоятельного решения

- В библиотеке классов:
  - Декларируйте делегат-тип **MyDel**, представляющий методы с двумя вещественными параметрами (**double**) и возвращающие целочисленное значение (**int**);
  - Декларируйте класс **TestClass** с нестатическим методом **TestMethod()**, возвращающим сумму целых частей из двух переданных в качестве параметров вещественных чисел (**double**).
- В проекте консольного приложения:
  - В методе **Main()** связать ссылку типа **MyDel** с методом **TestMethod()** и протестировать вызов делегата для пар вещественных чисел, которые вводит с клавиатуры пользователь. Результаты работы метода, вызванного через делегат, выводить на экран.

## Задание для самостоятельного решения

- Декларируйте тип-делегат **Sum** с одним целочисленным параметром ( **n**), тип возвращаемого значения **double**
- Используя **Sum**, организовать вычисление двойных сумм вида:

$$\sum_{i=1}^N \sum_{j=1}^i a_j$$

Протестировать делегат для  $a_j = \frac{1}{j}$ ;  $a_j = \frac{1}{2^j}$