

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

```
class AnyClass    {
    public static int AnyMethod(int AnyParameter)    {
        return AnyParameter + 1;
    }
}
class Program    {
    delegate int AnyDelegateType(int TramPamPam); //Объявление делегата-типа
    static void Main()    {
        AnyDelegateType AnyDelegate = // Объявление переменной-ссылки
            new AnyDelegateType(AnyClass.AnyMethod); //Инстанцирование
        int p;
        p = AnyDelegate(3); //Вызов метода через 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

```
class AnyClass    {
    public int AnyMethod(int AnyParameter)        {
        return AnyParameter + 1;
    }
}
class Program    {
    //Объявление делегата-типа
    delegate int AnyDelegateType(int TramPamPam);
    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
    }
}
```

 AnyDelegate	{Method = {Int32 AnyMethod(Int32)}}
 base	{Method = {Int32 AnyMethod(Int32)}}
 base	{Method = {Int32 AnyMethod(Int32)}}
 Method	{Int32 AnyMethod(Int32)}
 Target	{ConsoleApplication1.AnyClass}

Использование анонимного метода

```
class Program {  
    //Объявление делегата-типа  
    delegate int AnyDelegateType(int TramPamPam);  
  
    static void Main() {  
        // Объявление переменной и инстанцирование:  
        AnyDelegateType Anonim = delegate(int AnyParameter)  
        { return AnyParameter + 1; };  
  
        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

```
class Program {  
    //Объявление делегата-типа:  
    delegate int AnyDelegateType(int TramPamPam);  
  
    static void Main()  
    { // Объявление переменной с типом делегата и инстанцирование:  
        AnyDelegateType Lambda = x => x + 1;  
        int p = 5;  
        p = Lambda(p); // Вычисление лямбда-выражения  
        Console.WriteLine(p); // Результат: 6  
    }  
}
```

Синтаксис лямбда-выражения

(<спецификация_параметров>) =>
{ <операторы> }

Задача 1

```
using System;
delegate int Cast(double x);    // делегат-тип
class Program {
    static void Main() {
        double test = 15.83;

        // ближайшее четное целое
        Cast cast1 = delegate (double z)
        { return (int)(Math.Round(z / 2) * 2); };

        // порядок положительного числа
        Cast cast2 = delegate (double z)
        { return (int)Math.Log10(z); };

        Console.WriteLine("cast1(test)={0}, cast2(test)= {1}",
            cast1(test), cast2(test));
        Console.WriteLine("cast1(4.46)={0}, cast2(4.46)= {1}",
            cast1(4.46), cast2(4.46));
    }
}
```

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**.

Задача 2. Библиотека классов

```
public delegate int[] Row(int num); // делегат-тип
public delegate void Print(int[] ar); // делегат-тип
public class Example {
    // Метод возвращает массив цифр целого числа-параметра.
    static public int[] GetDigits(int num) {
        int arLen = (int)Math.Log10(num) + 1;
        int[] res = new int[arLen];
        for (int i = arLen - 1; i >= 0; i--) {
            res[i] = num % 10;
            num /= 10;
        }
        return res;
    }
    // Вывод значений элементов массива-параметра.
    static public void Display(int[] ar) {
        for (int i = 0; i < ar.Length; i++)
            Console.Write("{0}\t", ar[i]);
        Console.WriteLine();
    }
} // end of Example
```

Задача 2. Метод Main() консольного приложения

```
Row delRow; // Ссылка на делегат
Print delPrint; // Ссылка на делегат

delRow = new Row(Example.GetDigits); // Экземпляр делегата
delPrint = new Print(Example.Display); // Экземпляр делегата

int[] myAr = delRow(13579); // Вызов метода через делегат
delPrint(myAr); // Вызов метода через делегат
int[] newAr = { 11, 22, 33, 44, 55, 66 };
delPrint(newAr); // Вызов метода через делегат
Example.Display(myAr); // Явное обращение к методу
Console.WriteLine("delRow casts {0}.", delRow.Method);
Console.WriteLine("delPrint casts {0}.", delPrint.Method);
Console.WriteLine("delRow.Target: {0}.", delRow.Target);
Console.WriteLine("delPrint.Target: {0}.", delPrint.Target);
```

Задача 3

- Опишите делегат-тип **delegateConvertTemperature**, представляющий методы с одним вещественным параметром и возвращающий вещественное значение.
- В классе **TemperatureConverterImp** опишите методы, преобразующие вещественное значение температуры из градусов Цельсия в градусы Фаренгейта и наоборот.
- В основной программе свяжите с методами класса **TemperatureConverterImp** делегаты типа **delegateConvertTemperature** и протестируйте их.

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/>]

Задача 3

```
// Part 1 - Explicit declaration of a delegate
// (helps a compiler ensure type safety)
public delegate double delegateConvertTemperature(double
sourceTemp);

// A sample class to play with
class TemperatureConverterImp {
    // Part 2 - Will be attached to a delegate later in the code
    public double ConvertToFahrenheit(double celsius) {
        return (celsius * 9.0 / 5.0) + 32.0;
    }

    // Part 3 - Will be attached to a delegate later in the code
    public double ConvertToCelsius(double fahrenheit) {
        return (fahrenheit - 32.0) * 5.0 / 9.0;
    }
}
```

Задача 3

```
// Part 4 - Instantiate the main object
TemperatureConverterImp obj = new TemperatureConverterImp();
// Part 5 - Intantiate delegate #1
delegateConvertTemperature delConvertToFahrenheit =
    new delegateConvertTemperature(obj.ConvertToFahrenheit);
// Part 6 - Intantiate delegate #2
delegateConvertTemperature delConvertToCelsius =
    new delegateConvertTemperature(obj.ConvertToCelsius);

// Part 7 - delegate #1
double celsius = 0.0;
double fahrenheit = delConvertToFahrenheit(celsius);
string msg1 = string.Format("Celsius = {0}, Fahrenheit = {1}",
                             celsius, fahrenheit);
Console.WriteLine(msg1);

// Part 8 - delegate #2
fahrenheit = 212.0;
celsius = delConvertToCelsius(fahrenheit);
string msg2 = string.Format("Celsius = {0}, Fahrenheit = {1}",
                             celsius, fahrenheit);
Console.WriteLine(msg2);
```

Задание к задаче 3

- Замените в основном приложении два делегата – массивом делегатов.
- Опишите класс **StaticTempConverters** со статическими методами перевода температур из градусов Цельсия в Кельвины, Ранкины и Реомюры и наоборот.
- В основном приложении 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 сохранять 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:

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

Задача 5

```
using System;
class Program {
    delegate double Proizv(int i);
    delegate double Sum();
    public static void Main() {
        const int Max = 5;
        double x;
        do {
            Console.Write("Введите x - ");
        } while (!double.TryParse(Console.ReadLine(), out x));
        Proizv prF = (i) => {
            double p = 1;
            for (int j = 1; j <= Max; j++)
                p = p * i * x / j;
            return p;
        };
        Sum sumF = () => {
            double S = 0;
            for (int i = 1; i <= Max; i++)
                S += prF(i);
            return S;
        };
        Console.WriteLine(sumF());
    }
}
```

Задание для самостоятельного решения

- Опишите 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 $a_j = \frac{1}{j}$; $a_j = \frac{1}{2^j}$