

Елементи от функционалното програмиране

Делегати, функции, действия. Предаване на функции и действия като параметри



Учителски екип

Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning/>



```
private static void PrintFilteredStudent(  
    Dictionary<string, int> people,  
    Func<int, bool> tester,  
    Action<KeyValuePair<string, int>> printer)  
{  
    foreach (var person in people)  
    {  
        if (tester(person.Value))  
        {  
            printer(person);  
        }  
    }  
}
```



Съдържание

1. Променливи от тип "функция" в C#
2. Действия (процедури): Action<T>
3. Делегати
4. Предаване на функции към метод



Променливи от тип (референция към) функция в C# Func<T, TResult>

- Инициализация на функция

Lambda израз

```
Func<int, string> func = n => n.ToString();
```

Входен тип

Изходен тип

Име

Входен параметър

Връщан резултат

- ВХОДНИЯТ И ИЗХОДНИЯТ ТИП МОГАТ ДА СА **различни**
- ВХОДНИЯТ И ИЗХОДНИЯТ ТИП **може** ДА СА ОТ **тип**, КОЙТО НИЕ СМЕ **декларирали**

Действия (процедури): Action<T>

- В .NET Action<T> е метод, който не връща резултат:

```
private void Print(string message)
{ Console.WriteLine(message); }
```

- Вместо да пишем метода можем да напишем:

```
Action<string> print = message => Console.WriteLine(message);
```

- Тогава ние го използваме така:

```
print("pesho");
print(5.ToString());
```


Задача: Сбор на числа

- Въведете числа от клавиатурата
- Използвайте собствена функция за парсване
- Изведете броя на числата
- Изведете сбора им

4, 2, 1, 3, 5, 7, 1, 4, 2, 12



10
41

Делегати

- Func и Action се реализират чрез **делегати** и сами по себе си са вградени делегати. Допускат **до 15 параметъра**, които са напълно достатъчни за практическа работа.
- Ако желаем да работим **с повече** параметри се **ползват** делегати

Решение: Сбор на числа

```
string input = Console.ReadLine();  
  
Func<string, int> parser = n => int.Parse(n);  
  
int[] numbers = input.Split(new string[] {",", " "},  
                             StringSplitOptions.RemoveEmptyEntries)  
                        .Select(parser).ToArray();  
  
Console.WriteLine(numbers.Length);  
Console.WriteLine(numbers.Sum());
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/597#1>

Задача: Пребройте думите, започващи с главна буква

- Въведете текст от конзолата (клавиатурата)
- Пребройте колко думи започват с главна буква
- Изведете броя на думите
- Използвайте предикати



The following example
shows how to use
Predicate



The
Predicate

Решение: Пребройте думите, започващи с главна буква

```
var words = Console.ReadLine().Split(new string[] {" "},  
                                     StringSplitOptions.RemoveEmptyEntries);  
  
Func<string, bool> checker = n => n[0] == n.ToUpper()[0];  
  
words.Where(checker)  
    .ToList()  
    .ForEach(n => Console.WriteLine(n));
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/597#2>

Задача: начисляване на ДДС (VAT)

- Въведете от клавиатурата няколко цени на стоки
- Добавете **ДДС (VAT) 20%** на всяка стока
- Използвайте унарна (едноаргументна) операция



1.38, 2.56, 4.4



Цени с ДДС (VAT):

1,66

3,07

5,28

Решение: Добавяне на ДДС (VAT)

```
Console.ReadLine()  
    .Split(new string[] { ",", " " },  
           StringSplitOptions.RemoveEmptyEntries)  
    .Select(double.Parse)  
    .Select(n => n * 1.2)  
    .ToList()  
    .ForEach(n => Console.WriteLine($"{n:F2}"));
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/597#3>

Решение: Добавяне на ДДС (VAT) вариант 2 с функции

```
Func<string, double> costumDoubleParser =  
    str => double.Parse(str);  
Func<double, double> taxVAT =  
    (double price) => {return price*1.2;};  
Console.ReadLine()  
    .Split(new string[] { "," },  
        StringSplitOptions.RemoveEmptyEntries)  
    .Select(costumDoubleParser)  
    .Select(taxVAT)  
    .ToList()  
    .ForEach(n => Console.WriteLine($"{n:F2}"));
```

Очевидно, вариант 1 е по-кратък и прегледен, тъй като за случая има синтактична захар

Предаване на функции към метод

- Може да предаваме `Func<T>` към методи:

```
private int Operation(int number, Func<int, int> operation)
{
    return operation(number);
}
```

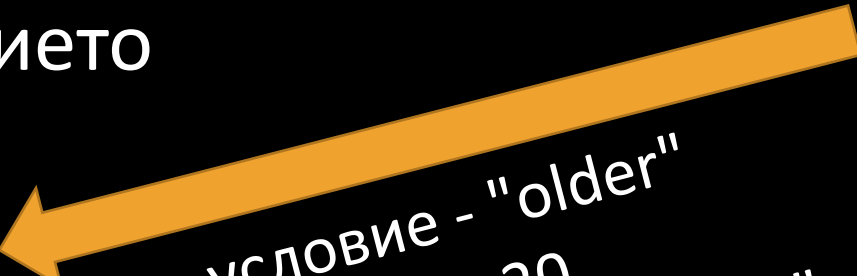
- Може да използваме метод като този:

```
int a = 5;
int b = Operation(a, number => number * 5);
int c = Operation(a, number => number - 3);
int d = Operation(b, number => number % 2);
```

Задача: Филтриране по възраст

- Въведете **n** души с тяхната възраст
- Въведете условие и възраст за филтър
- Въведете начина на форматиране на изхода
- Изведете всички хора, които удовлетворяват условието

Pesho	20
Radka	29
Mara	32

- 
- условие - "older"
 - Възраст - 20
 - формат- "name age"

Pesho	20
Gosho	18
Radka	29
Mara	32
Izdislav	16

Решение: Филтриране по възраст

```
//TODO: Въвеждане на данни от клавиатурата  
//Реализация на методите на следващите слайдове  
  
Func<int, bool> tester = CreateTester(condition, age);  
Action<KeyValuePair<string, int>> printer =  
    CreatePrinter(format);  
  
PrintFilteredStudent(people, tester, printer);
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/597#4>

Решение: Филтриране по възраст(2)

```
public static Func<int, bool> CreateTester  
    (string condition, int age)  
{  
    switch (condition)  
    {  
        case "younger": return x => x < age;  
        case "older": return x => x >= age;  
        default: return null;  
    }  
}
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/597#4>

Решение: Филтриране по възраст(3)

```
public static Action<KeyValuePair<string, int>>
    CreatePrinter(string format)
{
    switch (format)
    {
        case "name":
            return person => Console.WriteLine($"{person.Key}");
        case "age":
            return person => Console.WriteLine($"{person.Value}");
        case "name age":
            return person =>
                Console.WriteLine($"{person.Key} - {person.Value}");
        default: return null;
    }
}
```

Решение: Филтриране по възраст(4)

```
public static void PrintFilteredStudent(  
    Dictionary<string, int> people,  
    Func<int, bool> tester,  
    Action<KeyValuePair<string, int>> printer)  
{  
    foreach (var person in people)  
        if (tester(person.Value))  
            printer(person);  
}
```

За **всеки** човек в речника
с предварително
определения **tester**

се проверява от
tester дали отговаря
на **condition**

и ако отговаря се **извежда** информация,
подредена в реда, указан от **формата**

Какво научихме?

- `Action<T>` е функция, която не връща резултат
- `Func<T, TResult>` е функция, която връща резултат от тип `TResult`

`Func<T, TResult>` и `Action<T>` може да се предават като параметри на метод

Реализират се чрез делегати.

С тази технология можем да направим
кода **динамичен**.



Елементи от функционалното програмиране



Въпроси?



Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"



Министерство
на образованието
и науката



Национална
програма
„Обучение за
ИТ кариера“

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**



SoftUni
Foundation

