

Речници, ламбда изрази и LINQ

Колекции и заявки

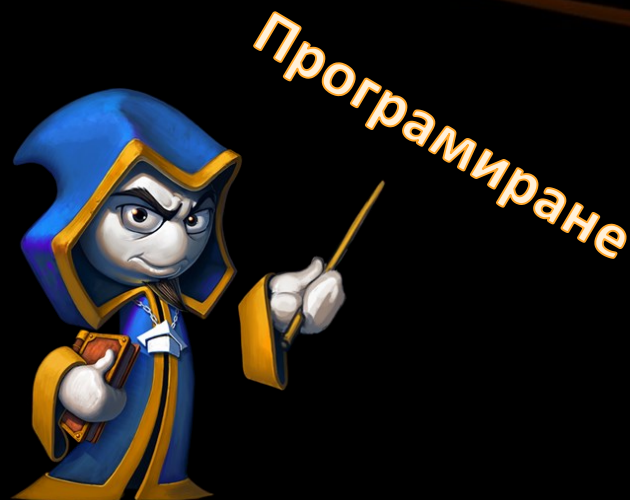


Учителски екип

Обучение за ИТ кариера

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

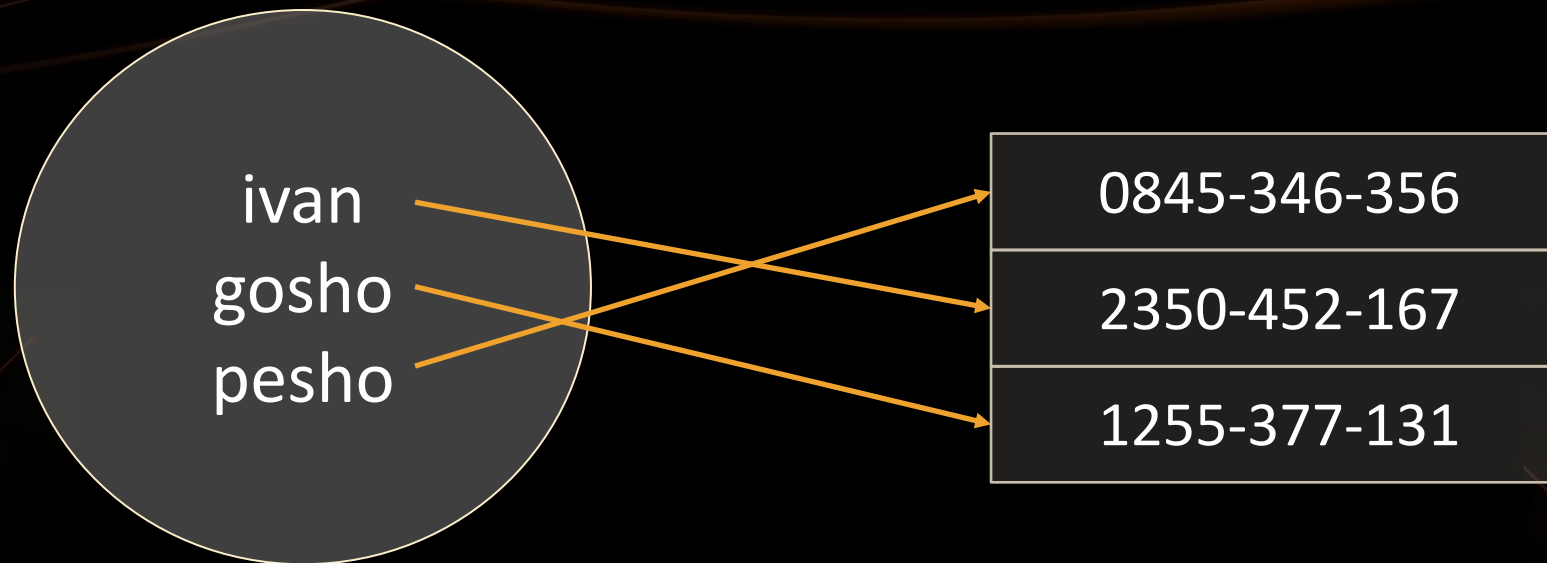
<https://github.com/BG-IT-Edu/School-Programming/tree/main/Courses/Applied-Programmer/Programming-Fundamentals>



Съдържание

1. Асоциативни масиви

2. Речници



Асоциативни масиви
Dictionary<Key, Value>

Асоциативни масиви (Карти, Речници)

- Асоциативните масиви са масиви, чиито индекси са **ключове**
 - Ключовете могат да бъдат думи или пък реални числа, за разлика от индексите на обикновения масив
- Съдържат информация в **двойки {ключ → стойност}**

Обикновен масив

ключ	0	1	2	3	4
стойност	8	-3	12	408	33

Асоциативен масив

Ключ (Key)	Стойност (Value)
John Smith	+1-555-8976
Lisa Smith	+1-555-1234
Sam Doe	+1-555-5030

Пример за ползване на Dictionary – Телефонен указател

```
var phonebook = new Dictionary<string, string>();  
phonebook["John Smith"] = "+1-555-8976";  
phonebook["Lisa Smith"] = "+1-555-1234";  
phonebook["Sam Doe"] = "+1-555-5030";  
phonebook["Ivan"] = "+359-899-555-592";  
phonebook["Ivan"] = "+359-2-981-9819"; // Заменяне  
phonebook.Remove("John Smith");  
foreach (var pair in phonebook)  
    Console.WriteLine("{0} --> {1}",  
        pair.Key, pair.Value);
```

Dictionary<K, V>

- Обикновен речник
 - Използват хеш-таблица + списък
 - Dictionary<K, V>
 - Пазят ключовете си по реда на добавяне

```
var dict = new Dictionary<string, int>();
```

Речници: Функционалност

- **Count** – пази броя на двойките от ключ-стойност
- **Keys** – съдържа уникалните ключове

```
var dict = new Dictionary<string, int>();  
foreach(var key in dict.Keys)  
    Console.WriteLine(key);
```

- **Values** – съдържа всички стойности

```
Console.WriteLine(String.Join(", ", dict.Values));
```

- Основни операции: **Add()**, **[], Remove()**, **Clear()**


Речници: Функционалност (2)

- Намиране на ключ / стойност:
 - **ContainsKey()** – проверяваме дали даден **ключ** съществува в речника (бърза операция)
 - **ContainsValue()** – проверяваме дали дадена **стойност** съществува в речника (бавна операция)
 - **TryGetValue()** – проверяваме дали даден **ключ** съществува в речника и отпечатва стойността му

Обикновен речник: Add()

Pesho	0881-123-987
Gosho	0881-123-789
Alice	0881-123-978

Hash Function



Dictionary<string, string>	

Ключ

Стойност

Речник: Remove()

Pesho

Hash Function



Dictionary<string, string>	
Pesho	0881-123-987
Gosho	0881-123-789
Alice	0881-123-978

Ключ

Стойност

Обхождане на речника

foreach цикъл

`KeyValuePair<string, string> keyValuePair` `in`

`Dictionary<string, string>`

Pesho

0881-123-987

Gosho

0881-456-987

Alice

+359-899-55-592

`.Key`

`.Value`

Задача: Нечетни срещания

- Напишете програма, която извлича от **поредица от думи** всички елементи, които се срещат **нечетен брой пъти** (без значение от големината на буквите)
- Думите са въведени на един ред разделени с **интервал**
- Изведете получените думи с **малки букви**, по реда им на поява

Java C# PHP PHP JAVA C java



java, c#, c

3 5 5 hi pi NO hi 5 ho 3 hi pi



5, hi

a a A SQL xx a xx a A a XX c



a, sql, xx, c

Тествайте в Judge: <https://judge.softuni.bg/Contests/2669>

Решение: Нечетни срещания

```
string input = Console.ReadLine().ToLower();
string[] words = input.Split(' ');

var counts = new Dictionary<string, int>();
foreach (var word in words)
    if (counts.ContainsKey(word))
        counts[word]++;
    else counts[word] = 1;

var results = new List<string>();
foreach (var pair in counts)
    // TODO: добави pair.Key към резултатите ако pair.Value е нечетно
Console.WriteLine(string.Join(", ", results));
```

counts[word]
пази колко пъти
word се среща в
words

Какво научихме този час?

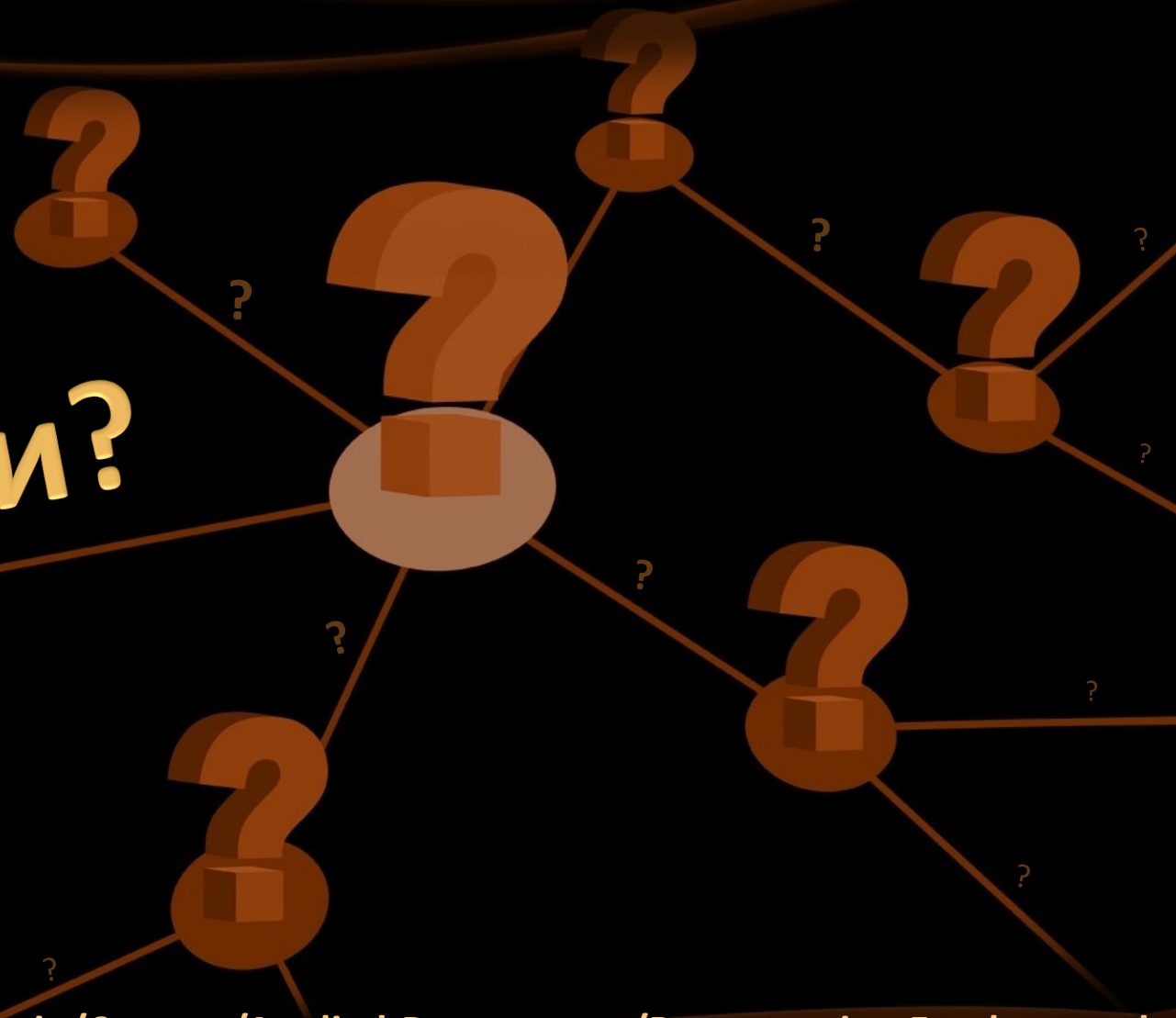
- Речниците съдържат двойки от {ключ (key) → стойност (value)}
- **.Keys** съдържа уникални ключове
- **.Values** съдържа колекция от стойности
- Обхождането на речника разглежда записите като **KeyValuePair<K, V>**



Речници, ламбда изрази и LINQ



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

