

Речници

(карти, асоциативни масиви)

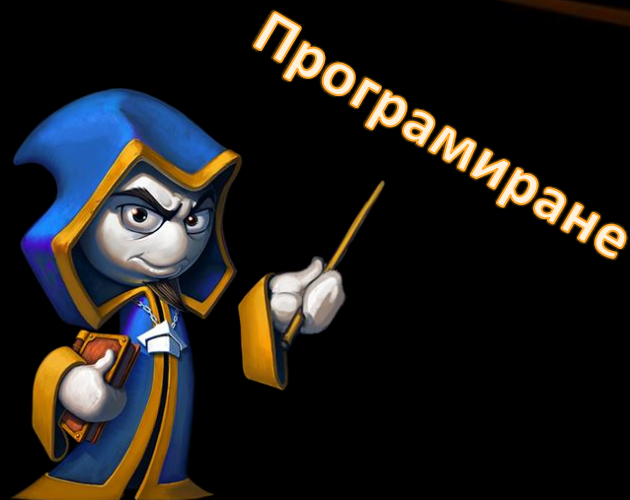


Учителски екип

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
3. Сортирани речници
4. Примерни задачи

Асоциативни масиви (Карти, Речници)

- Асоциативните масиви са масиви, чиито индекси са **ключове**
 - Ключовете са пак уникални, но 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"; // Добавяне  
phonebook["Ivan"] = "+359-2-981-9819"; // Замяна  
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>();  
... // добавяме елементи  
foreach(var key in dict.Keys)  
    Console.WriteLine(key);  
Console.WriteLine(String.Join(", ", dict.Values));
```

Речници: Функционалност

- **Add(key, value)** – добавя елемент
- **Remove(key)** – премахва елемент
- **this[key] = value** – добавя или подменя елемент
- **this[key]** – извлича елемент / хвърля изключение ако го няма
- **Keys** – връща всички ключове (по реда на добавяне)
- **Values** – връща всички стойности (по реда на добавяне)
- **Count** – пази броя на двойките от ключ-стойност


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(key, out value)** – търси **стойност** по **ключ**
 - Ако намери **key**, записва стойността му във **value** и връща **true**
 - Иначе връща **false**

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

SortedDictionary<K, V>


- Сортирани речници
 - Използват балансирано дърво за претърстване
 - SortedDictionary<K, V>
 - Пазят ключовете си сортирани в техния естествен ред

```
var sortedDict = new SortedDictionary<int,int>();
```


SortedDictionary<K, V> – Пример

Имя	Телефон
Palstra	+388-892-55872

Comparator
Function



SortedDictionary <string, string>	

Ключ

Стойность

Пример: SortedDictionary – Събития

```
var events = new SortedDictionary<DateTime, string>();
events[new DateTime(1998, 9, 4)] = "Google's birth date";
events[new DateTime(2013, 11, 5)] = "SoftUni's birth date";
events[new DateTime(1975, 4, 4)] = "Microsoft's birth date";
events[new DateTime(2004, 2, 4)] = "Facebook's birth date";
events[new DateTime(2013, 11, 5)] = "SoftUni was founded";
foreach (var entry in events)
{
    Console.WriteLine("{0:dd-MMM-yyyy}: {1}",
        entry.Key, entry.Value);
}
```

Задача: Пребройте реалните числа

- Въведете списък от реални числа и ги изведете в нарастващ ред заедно с техния брой срещания

8 2.5 2.5 8 2.5



2.5 -> 3 times
8 -> 2 times

1.5 5 1.5 3



1.5 -> 2 times
3 -> 1 times
5 -> 1 times

-2 0.33 0.33 2



-2 -> 1 times
0.33 -> 2 times
2 -> 1 times

Тествайте в Judge: <https://judge.softuni.bg/Contests/2670>

Решение: Пребройте реалните числа

```
double[] nums = Console.ReadLine().Split(' ')
    .Select(double.Parse).ToArray();

var counts = new SortedDictionary<double, int>();
foreach (var num in nums)
    if (counts.ContainsKey(num))
        counts[num]++;
    else
        counts[num] = 1;

foreach (var num in counts.Keys)
    Console.WriteLine($"{num} -> {counts[num]}");
```

counts[num] пази
колко пъти **num** се
среща в **nums**

Тествайте в Judge: <https://judge.softuni.bg/Contests/2670>

Какво научихме този час?

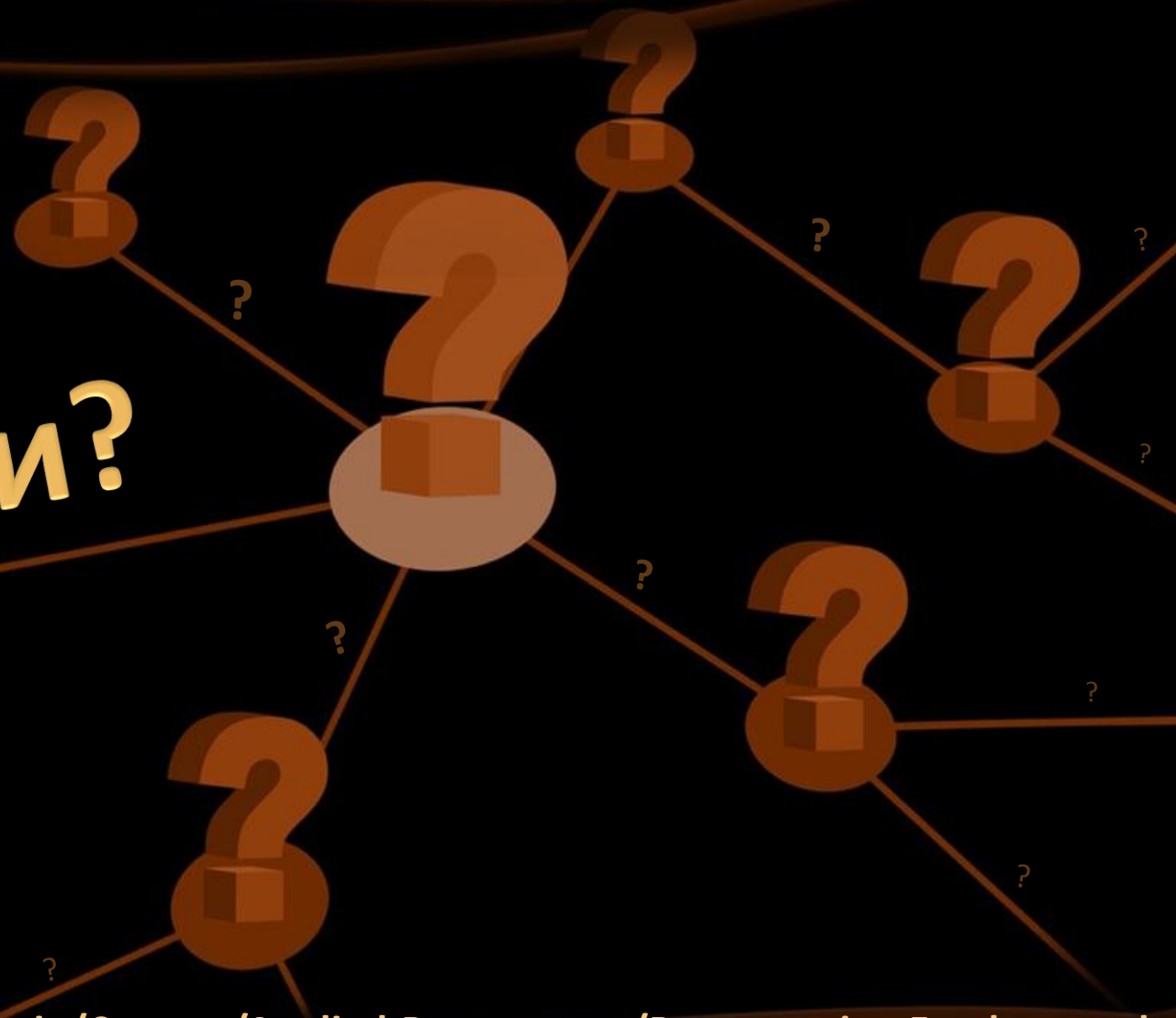
- Речниците съдържат двойки от {ключ (key) → стойност (value)}
 - **.Keys** съдържа уникални ключове
 - **.Values** съдържа колекция от стойности
 - Обхождането на речника разглежда записите като **KeyValuePair<K, V>**
- **Dictionary<K, V>** - елементите се пазят в реда на добавяне, докато при **SortedDictionary<K, V>** редът на добавяне няма значение, елементите са сортирани по ключа



Речници



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

