Dictionaries (maps) – асоциативни масиви. Нямат индекси като масивите, ключовете не са 0, 1, 2, а ние си ги създаваме. Условието е да са уникални, да няма 2 еднакви. Примерно име може да е ключ. Група от двойки данни – ключ -> стойност ( pairs : key -> value). Реално асоциативния масив са 2 отделни масива – един с ключове и един със стойности, който имат връзка помежду си. Това е основата на всички структури в програмирането=

Динамична структура като листа – могат да се махат и добавят стойности. Но ако се махат данни се премахва цялата двойка – не може да се махне само ключ или само стойност.

Има count.

String – ключове, int – стойности

Dictionary<string, int> dict = new Dictionary<string, int>()

Foreach(var key in dict.Keys)

Cw(key)

Foreach(var value in dict.Values)

Cw(values);

Команди – Add() Remove() Clear()

Тук махаме по ключ.

Boolean methods –

ContainsKey() – булев метод, връща дали го има

ContainsValue

TryGetValue() – ако има стойност я връща, ако няма – връща default стойността за определения тип данни. Реално дава и true/false?

Ключа минава през хешираща функция, която намира на ключа място в паметта. Създава хеш код, който отговаря на място в паметта.

Phonebook dictionary

Foreach – KeyValuePair<string, string>

Phonebook[“Nakov”] = “+359-”; - създаваш място и добавяш стойност

Ако го има вече – замества предишния

Phonebook.Remove(“John Smith”)

Foreach(KeyValuePair<string, string> pair in phonebook)

Cw() pair.Key, pair.Value

KeyValuePair<string, string> може да се замести с value

Foreach(string key in phonebook.Keys) – това взима само от ключовете

Cw($”{key}”)

Foreach(string value in phonebook.Values) – това взима само от стойностите

Cw($”{value}”)

SortedDictionary – двата ключа първо се сравняват и после се добавя втория – може да измести първия и да влезе на негово място, избутвайки го. Винаги е от най-малко към най-голямо.

ПО-БАВЕН от стандартните речници.

\*при Date.Time – MMM дава месеца като 3 букви – Jan, Feb…

Ламбда функцията е парче код, което изпълнява код, като метода, но не е именувано. Директно функция от LINQ библиотеката.

\*\* Нивото на един език – колко е близък до начина, по който ние говорим. По-ниско ниво – вземи това число и го постави в тази клетка на компютъра, примерно. Високо ниво – искам да сумирам – Sum().

Първо пишем using.System.Linq – зареди тази библиотека, за да използвам от нея Linq заявки

Remove и RemoveAt са методи вградени в листа. RemoveAll е linq заявка.

Min,Max,Sum,Average

Select – взимаме нещо и правим нещо с него. ИЗБЕРИ нещо от колекцията и направи нещо с него.

Double[] nums = Console.ReadLine()

.Split()

.Select(number => double.Parse(number)); - това е ламбда функция, взимам някакъв стринг примерно и връщам парса на тоя стринг през double.

.ToArray() – прави масив, иначе е някакъв тип колекция, която още не сме учили

Може Select(number +> number + Math.PI)

\*\*\*Ако искаме само да парснем може Select(double.Parse) – същото е като Select(number => double.Parse(number).

Освен ToArray имаме ToList, ToDictionary, ToCharArrays

OrderBy() и OrderByDescending()

Може по два критерия – OrderBy() и след това ThenBy()

Примерно OrderBy(pair => pair.Value).ThenBy(pair => pair.Key) – подреди по стойности, ако има 2 еднакви стойности – подреди ги по ключове

След това добавяме .ToDictionary(pair => Pair.Key, pair => pair.Value)

Take(3) – вземи първите 3

Skip(3)-прескочи 3 и вземи останалите

Nums = nums.Take(3) – реално взима стария масив, прави нов с 3 позиции и трие стария! Не премахва другите позиции, а просто първо прави новия масив, после трие стария.

Може Skip().Take()

* Означава goes to, отива към

() = > 42 – не приема нищо, но връща 42

Where() – проверява всеки елемент и връща само елементи, които отговарят на усовието вътре

.Where(num => num % 2 == 0) избира и връща само числата, които са четни

Count() – връща бройката на елементите, които отговарят на дадено условие. Колко са тия елементи, които са четни, примерно - count(num => num % 2 == 0)

Distinct() – връща елементи, които се повтарят само по веднъж, уникалните елементи.

Char[] separators = new char[] {}

Split(separators)

Where(word => word!= “”) – взима всичко, което не е празно

Take Single Element from Collection – First(), Last(), Single()

Примерно може първото четно – First(x => x% 2 == 0)

Other

Reverse()- обръща

Concat() – позволява да залепим 2 неща, може масив

Secelt((x, index)) – взима стойността, но връща и позицията!

row1.Select((x, index) => x + row2[index]);