Dictionaries, Lambda and LINQ

Collections and Queries

greenwich.edu.vn

UNIVERSITY of GREENWICH

Alliance with FPT Education

UNIVERSITY of GREENWICH
Alliance with FPT Education

Table of Contents

- Associative Arrays
  - Dictionary <key, value>
  - SortedDictionary <key, value>
- Lambda
- LINQ
  - Filtering
  - Mapping
  - Ordering

ASSOCIATIVE ARRAYS

Collection of Key and Value Pairs

UNIVERSITY of GREENWICH
Alliance with FPT Education

Associative Arrays (Maps, Dictionaries)

- Associative arrays are arrays indexed by keys
  - Not by the numbers 0, 1, 2, … (like arrays)
- Hold a set of pairs

Key	Value
John Smith	+1-555-8976
Lisa Smith	+1-555-1234
Sam Doe	+1-555-5030

UNIVERSITY of GREENWICH
Alliance with FPT Education

Dictionary

- Dictionary<K, V> - collection of key and value pairs
- Keys are unique
- Keeps the keys in their order of addition
- Uses a hash-table + list

```
var fruits = new SortedDictionary<string, double>();
fruits["kiwi"] = 4.50;
fruits["orange"] = 2.50;
fruits["banana"] = 2.20;
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Sorted Dictionary

- SortedDictionary<K, V>
- Keeps its keys always sorted
- Uses a balanced search tree

```
var fruits = new SortedDictionary<string, double>();
fruits["kiwi"] = 4.50;
fruits["orange"] = 2.50;
fruits["banana"] = 2.20;
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Built-In Methods

- **Add(key, value)** method

```
var airplanes = new Dictionary<string, int>();
airplanes.Add("Boeing 737", 130);
airplanes.Add("Airbus A320", 150);
```

- **Remove(key)** method

```
var airplanes = new Dictionary<string, int>();
airplanes.Add("Boeing 737", 130);
airplanes.Remove("Boeing 737");
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Built-In Methods (2)

- ContainsKey(key)

```
var dictionary = new Dictionary<string, int>();
dictionary.Add("Airbus A320", 150);
if (dictionary.ContainsKey("Airbus A320"))
   Console.WriteLine($"Airbus A320 key exists");
```

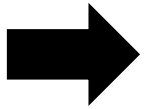
- ContainsValue(value)

```
var dictionary = new Dictionary<string, int>();
dictionary.Add("Airbus A320", 150);
Console.WriteLine(airplanes.ContainsValue(150)); //true
Console.WriteLine(airplanes.ContainsValue(100)); //false
```

Problem: Count Real Numbers

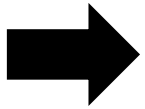
- Read a list of real numbers and print them in ascending order along with their number of occurrences

```
8 2.5 2.5 8 2.5
```



```
2.5 -> 3
8 -> 2
```

```
1.5 5 1.5 3
```



```
1.5 -> 2
3 -> 1
5 -> 1
```

Traditional Dictionary: Add()

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

Hash Function

Dictionary<string, string>

Key	Value

Dictionary: Remove()

Pesho

Hash Function

Dictionary<string, string>

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

SortedDictionary<K, V> – Example

Pesho	0881-123-987
Alice	+359-899-55-592

Comparator Function

SortedDictionary <string, string>

Key	Value

UNIVERSITY of GREENWICH
Alliance with FPT Education

Iterating through Dictionary

- Using foreach loop
- Iterate through objects of type KeyValuePair<K, V>
- Cannot modify the dictionary (read-only)

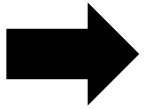
```
var fruits = new Dictionary<string, double>();
fruits.Add("banana", 2.20);
fruits.Add("kiwi", 4.50);
foreach (var fruit in fruits)
  Console.WriteLine($"{fruit.Key} -> {fruit.Value}");
```

fruit.Key -> fruit name
fruit.Value -> fruit price

Problem: Word Synonyms

- Read 2 * N lines of pairs word and synonym
- Each word may have many synonyms

```
3
cute
adorable
cute
charming
smart
clever
```



```
cute - adorable, charming
smart - clever
```

Solution: Word Synonyms

```
int n = int.Parse(Console.ReadLine());
var words = new Dictionary<string, List<string>>();
for (int i = 0; i < 2 * n; i++) {
   string word = Console.ReadLine();
   string synonym = Console.ReadLine();
   if (words.ContainsKey(word) == false)
      words.Add(word, new List<string>());
   words[word].Add(synonym);
}
```

Adding the synonym to the list

LAMBDA EXPRESSIONS

Anonymous Functions

UNIVERSITY of GREENWICH
Alliance with FPT Education

Lambda Functions

- A lambda expression is an anonymous function containing expressions and statements

```
a => a > 5;
```

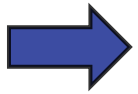
- Lambda expressions
- Use the lambda operator =>
  - Read as "goes to"
- The left side specifies the input parameters
- The right side holds the expression or statement

UNIVERSITY of GREENWICH
Alliance with FPT Education

Lambda Functions

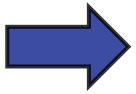
- Lambda functions are inline methods (functions) that take input parameters and return values:

```
x => x / 2
```



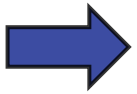
```
static int Func(int x) { return x / 2; }
```

```
x => x != 0
```



```
static bool Func(int x) { return x != 0; }
```

```
() => 42
```



```
static int Func() { return 42; }
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Processing Sequences with LINQ

- Min() – finds the smallest element in a collection

```
new List<int>() { 1, 2, 3, 4, -1, -5, 0, 50 }.Min()   //-5
```

- Max() – finds the largest element in a collection

```
new int[] { 1, 2, 3, 40, -1, -5, 0, 5 }.Max()         //40
```

- Sum() – finds the sum of all elements in a collection

```
new long[] {1, 2, 3, 4, -1, -5, 0, 50}.Sum()          //54
```

- Average() – finds the average of all elements in a collection

```
new int[] {1, 2, 3, 4, -1, -5, 0, 50}.Average()       //6.75
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Manipulating Collections

- Select() manipulates elements in a collection

```
var nums = Console.ReadLine()
    .Split()
    .Select(int.Parse);
```

```
string[] words = { "abc", "def" } ;
var result = words.Select(w => w + "x");
// words -> abcx, defx
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Converting Collections

- Using ToArray(), ToList() to convert collections:

```
int[] nums = Console.ReadLine()
  .Split()
  .Select(number => int.Parse(number))
  .ToArray();
```

```
List<double> nums = Console.ReadLine()
  .Split()
  .Select(double.Parse)
  .ToList();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Filtering Collections

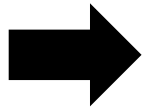
- Using Where()

```
int[] nums = Console.ReadLine()
   .Split()
   .Select(int.Parse)
   .Where(n => n > 0)
   .ToArray();
```

Problem: Word Filter

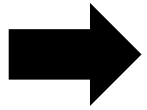
- Read a string array
- Print only words which length is

```
kiwi orange banana apple
```



```
kiwi
orange
banana
```

```
pizza cake pasta chips
```



```
cake
```

Solution: Word Filter

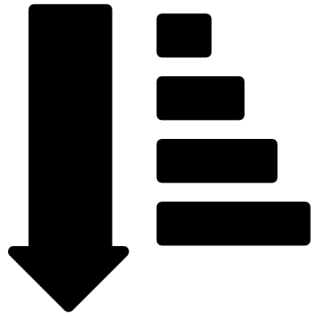
```
string[] words = Console.ReadLine()
                 .Split()
                 .Where(w => w.Length % 2 == 0)
                 .ToArray();

foreach (string word in words)
   Console.WriteLine(word);
```

Sorting Collections

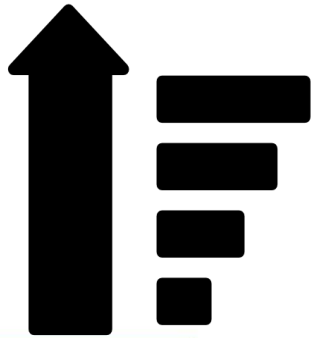
- Using OrderBy() to sort collections:

```
List<int> nums = { 1, 5, 2, 4, 3 };
nums = nums
   .OrderBy(num => num)
   .ToList();
```



- Using OrderByDescending() to sort collections:

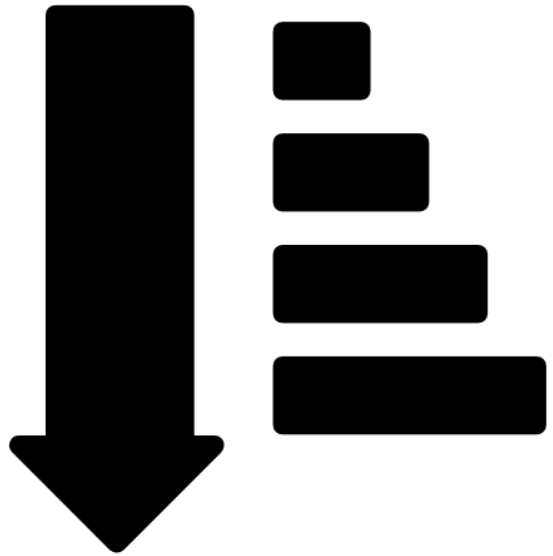
```
List<int> nums = { 1, 5, 2, 4, 3 };
nums = nums.OrderByDescending(num => num).ToList();
Console.WriteLine(String.Join(", ", nums));
```



Sorting Collections by Multiple Criteria

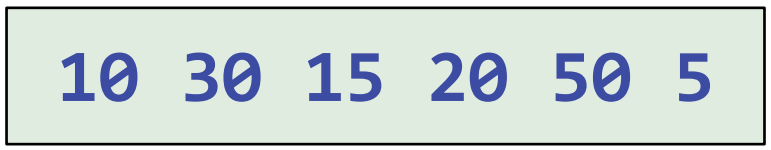
- Using ThenBy() to sort collections by multiple criteria:

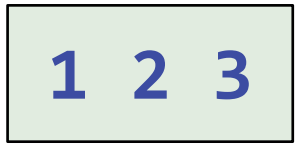
```
var products = new Dictionary<int, string>();
Dictionary<int, string> sortedDict = products
   .OrderBy(pair => pair.Value)
   .ThenBy(pair => pair.Key)
   .ToDictionary(pair => pair.Key,
                 pair => pair.Value);
```

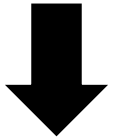


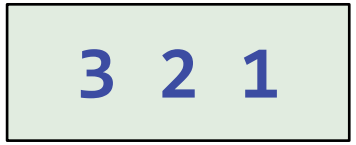
Problem: Largest 3 Numbers

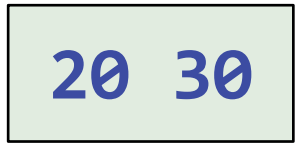
- Read a list of numbers
- Print largest 3, if there are less than 3, print all of them

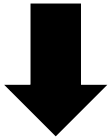


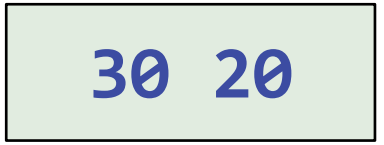












Solution: Largest 3 Numbers

```
int[] numbers = Console.ReadLine()
                .Split()
                .Select(int.Parse)
                .OrderByDescending(n => n)
                .ToArray();
int count = numbers.Length >= 3 ? 3 : numbers.Length;
for (int i = 0; i < count; i++)
   Console.Write($"{numbers[i]} ");
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Summary

- Dictionaries hold {key & value} pairs
  - Keys holds a set of unique keys
  - Values holds a collection of values
  - Iterating over dictionary
    takes the entries as KeyValuePair<K, V>
- Lambda and LINQ helps
  collection processing