https://www.tutorialsteacher.com/linq/linq-tutorials
Kirjutada lühidalt mõistetest:

1. LINQ (Language Integrated Query) is uniform query syntax in C# and VB.NET to retrieve data from different sources and formats. It is integrated in C# or VB, thereby eliminating the mismatch between programming languages and databases, as well as providing a single querying interface for different types of data sources.

Points to Remember:

Use System.Linq namespace to use LINQ.
LINQ api includes two main static class Enumerable & Queryable.
The static Enumerable class includes extension methods for classes that implements the IEnumerable<T> interface.
IEnumerable<T> type of collections are in-memory collection like List, Dictionary, SortedList, Queue, HashSet, LinkedList.
The static Queryable class includes extension methods for classes that implements the IQueryable<T> interface.
Remote query provider implements e.g. Linq-to-SQL, LINQ-to-Amazon etc.

2. mida tähendab märk: => Lambda Parameter (Expression in C#)

Lambda Expression is a shorter way of representing anonymous method.
Lambda Expression syntax: parameters => body expression
Lambda Expression can have zero parameter.
Lambda Expression can have multiple parameters in parenthesis ().
Lambda Expression can have multiple statements in body expression in curly brackets {}.
Lambda Expression can be assigned to Func, Action or Predicate delegate.
Lambda Expression can be invoked in a similar way to delegate.

3. Where-Filtering Operator(Linqi laiendusmeetod)

Where is used for filtering the collection based on given criteria.
Where extension method has two overload methods. Use a second overload method to know the index of current element in the collection.
Method Syntax requires the whole lambda expression in Where extension method whereas Query syntax requires only expression body.
Multiple Where extension methods are valid in a single LINQ query.

4. OfType The OfType operator filters the collection based on the ability to cast an element in a collection to a specified type.

The Where operator filters the collection based on a predicate function.

The OfType operator filters the collection based on a given type
Where and OfType extension methods can be called multiple times in a single LINQ query.

5. ThenBy
The ThenBy (and ThenByDescending) extension methods are used for sorting on multiple fields.
OrderBy and ThenBy sorts collections in ascending order by default.
ThenBy or ThenByDescending is used for second level sorting in method syntax.
ThenByDescending method sorts the collection in decending order on another field.
ThenBy or ThenByDescending is NOT applicable in Query syntax.
Apply secondary sorting in query syntax by separating fields using comma.

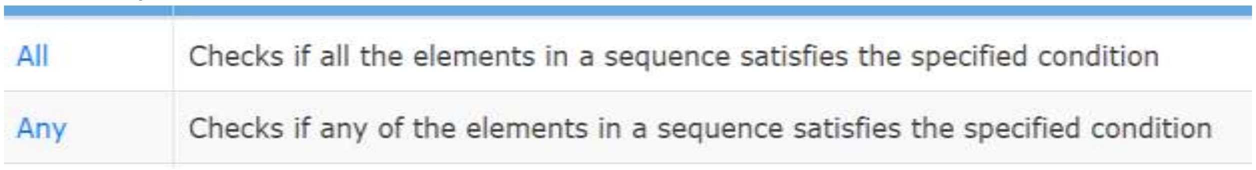
6. GroupBy, ToLookUp
The grouping operators do the same thing as the GroupBy clause of SQL query. The grouping operators create a group of elements based on the given key. This group is contained in a special type of collection that implements an IGrouping<TKey,TSource> interface where TKey is a key value, on which the group has been formed and TSource is the collection of elements that matches with the grouping key value.
GroupBy & ToLookup return a collection that has a key and an inner collection based on a key field value.
The execution of GroupBy is deferred whereas that of ToLookup is immediate.
A LINQ query syntax can be end with the GroupBy or Select clause.

7. Join- The joining operators joins the two sequences (collections) and produce a result.
Join and GroupJoin are joining operators.
Join is like inner join of SQL. It returns a new collection that contains common elements from two collections whosh keys matches.
Join operates on two sequences inner sequence and outer sequence and produces a result sequence.
Join query syntax:
```
from... in outerSequence
join... in innerSequence
on  outerKey equals innerKey
        select ...
```

8. GroupJoin
The GroupJoin operator performs the same task as Join operator except that GroupJoin returns a result in group based on specified group key. The GroupJoin operator joins two sequences based on key and groups the result by matching key and then returns the collection of grouped result and key.

9. Select - Projection Operator
The Select operator always returns an IEnumerable collection which contains elements based on a transformation function. It is similar to the Select clause of SQL that produces a flat result set.
LINQ query syntax must end with a Select or GroupBy clause.

10. All, Any



All	Checks if all the elements in a sequence satisfies the specified condition
Any	Checks if any of the elements in a sequence satisfies the specified condition

The All operator evalutes each elements in the given collection on a specified condition and returns True if all the elements satisfy a condition.
Any checks whether any element satisfy given condition or not? In the following example, Any operation is used to check whether any student is teen ager or not.

All, Any are quantifier operators in LINQ.
All checks if all the elements in a sequence satisfies the specified condition.
Any check if any of the elements in a sequence satisfies the specified condition
All, Any & Contains are not supported in query syntax in C# or VB.Net.

11. Contains
The Contains operator checks whether a specified element exists in the collection or not and returns a boolean.
The Contains() extension method has following two overloads. The first overload method requires a value to check in the collection and the second overload method requires additional parameter of IEqualityComparer type for custom equalality comparison.

Contains are quantifier operators in LINQ.
Contains operator checks whether specified element exists in the collection or not.
Use custom class that derives IEqualityOperator with Contains to check for the object in the collection.
Contains is not supported in query syntax in C# or VB.Net.

12. Aggregate -Aggregation Operators
Performs a custom aggregation operation on the values in the collection.

13. Avarage- Aggregation Operator
Average extension method calculates the average of the numeric items in the collection.
Average method returns nullable or non-nullable decimal, double or float value.

14. Count-aggregation Operator
The Count operator returns the number of elements in the collection or number of elements that have satisfied the given condition.
C# Query Syntax doesn't support aggregation operators. However, you can wrap the query into brackets and use an aggregation functions as shown below.

Example: Count operator in query syntax C#
```
var totalAge = (from s in studentList select s.age).Count();
```

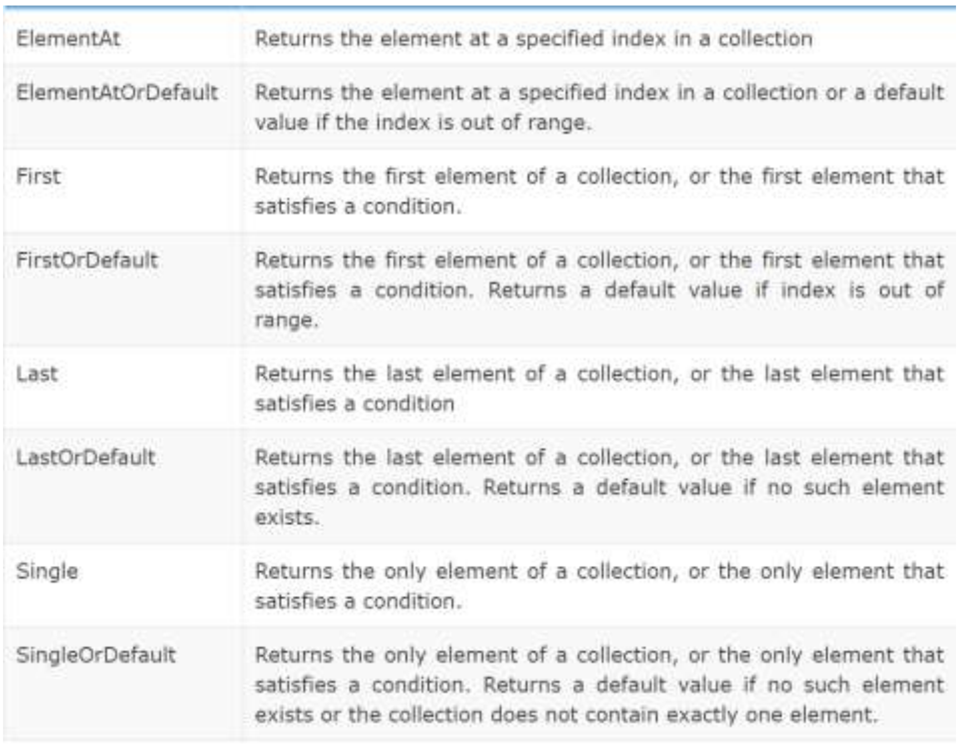
15. Max- Aggregation Operator
The Max() method returns the largest numeric element from a collection.
Max operator is Not Supported in C# Query syntax. However, it is supported in VB.Net query syntax

16. Sum-Aggregation Operator
The Sum() method calculates the sum of numeric items in the collection.

17. ElementAt, ElementAtOrdefault

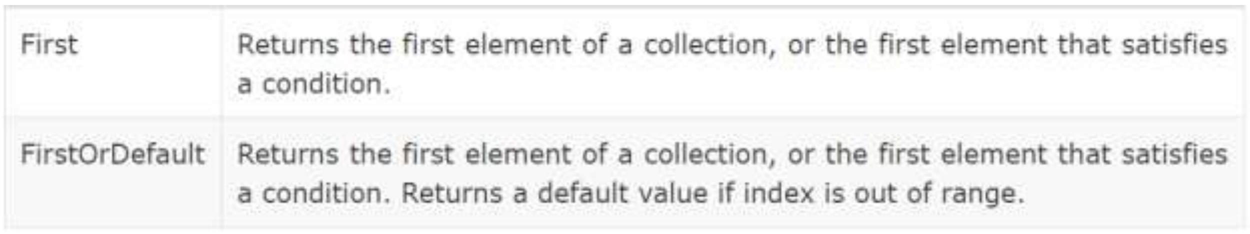
Element Operators

Element operators return a particular element from a sequence (collection).



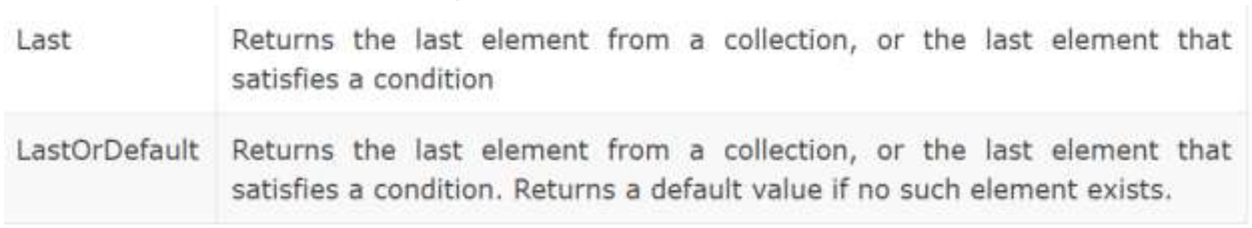
ElementAt	Returns the element at a specified index in a collection
ElementAtOrDefault	Returns the element at a specified index in a collection or a default value if the index is out of range.
First	Returns the first element of a collection, or the first element that satisfies a condition.
FirstOrDefault	Returns the first element of a collection, or the first element that satisfies a condition. Returns a default value if index is out of range.
Last	Returns the last element of a collection, or the last element that satisfies a condition
LastOrDefault	Returns the last element of a collection, or the last element that satisfies a condition. Returns a default value if no such element exists.
Single	Returns the only element of a collection, or the only element that satisfies a condition.
SingleOrDefault	Returns the only element of a collection, or the only element that satisfies a condition. Returns a default value if no such element exists or the collection does not contain exactly one element.

18. First, FirstOrDefault- Element Operators
The First and FirstOrDefault method returns an element from the zeroth index in the collection i.e. the first element. Also, it returns an element that satisfies the specified condition.



First	Returns the first element of a collection, or the first element that satisfies a condition.
FirstOrDefault	Returns the first element of a collection, or the first element that satisfies a condition. Returns a default value if index is out of range.

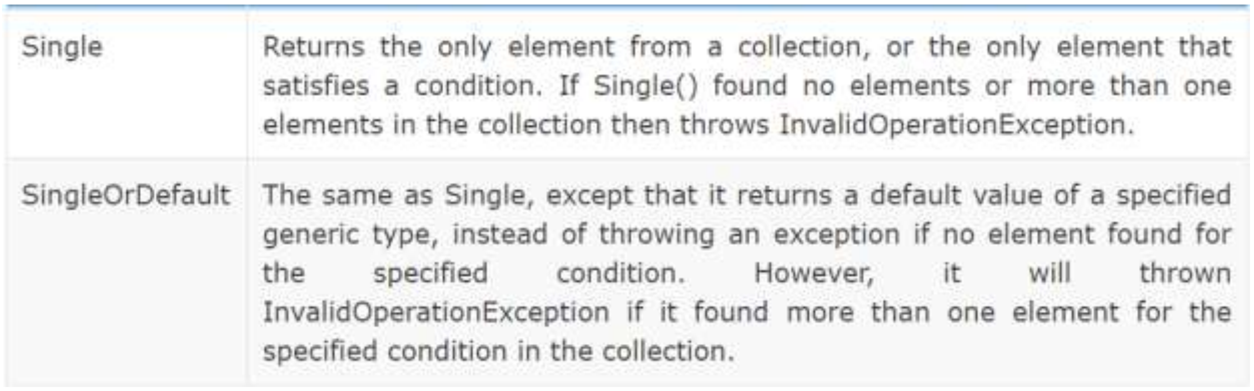
19. Last, LastOrDefault-Element Operators



Last	Returns the last element from a collection, or the last element that satisfies a condition
LastOrDefault	Returns the last element from a collection, or the last element that satisfies a condition. Returns a default value if no such element exists.

Last and LastOrDefault has two overload methods. One overload method doesn't take any input parameter and returns last element from the collection. Second overload method takes a lambda expression to specify a condition and returns last element that satisfies the specified condition.

20. Single, SingleOrDefault-Element Operators



Single	Returns the only element from a collection, or the only element that satisfies a condition. If Single() found no elements or more than one elements in the collection then throws InvalidOperationException.
SingleOrDefault	The same as Single, except that it returns a default value of a specified generic type, instead of throwing an exception if no element found for the specified condition. However, it will thrown InvalidOperationException if it found more than one element for the specified condition in the collection.

Single and SingleOrDefault have two overload methods. The first overload method doesn't take any input parameter and returns a single element in the collection. The second overload method takes the lambda expression as a predicate delegate that specifies the condition and returns a single element that satisfies the specified condition.

Single() expects one and only one element in the collection.
Single() throws an exception when it gets no element or more than one elements in the collection.
If specified a condition in Single() and result contains no element or more than one elements then it throws an exception.
SingleOrDefault() will return default value of a data type of generic collection if there is no elements in a colection or for the specified condition.
SingleOrDefault() will throw an exception if there is more than one elements in a colection or for the specified condition.

21. SequenceEquel- LINQ Equality Operator
There is only one equality operator: SequenceEqual. The SequenceEqual method checks whether the number of elements, value of each element and order of elements in two collections are equal or not.
If the collection contains elements of primitive data types then it compares the values and number of elements, whereas collection with complex type elements, checks the references of the objects. So, if the objects have the same reference then they considered as equal otherwise they are considered not equal.

The SequenceEqual method compares the number of items and their values for primitive data types.

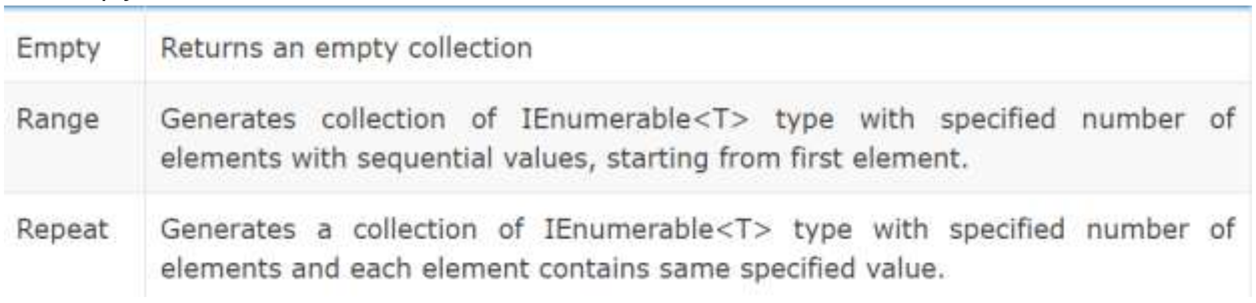
The SequenceEqual method compares the reference of objects for complex data types.

Use IEqualityComparer class to compare two colection of complex type using SequenceEqual method.

22. Concat-Concatenation Operator
The Concat() method appends two sequences of the same type and returns a new sequence (collection).

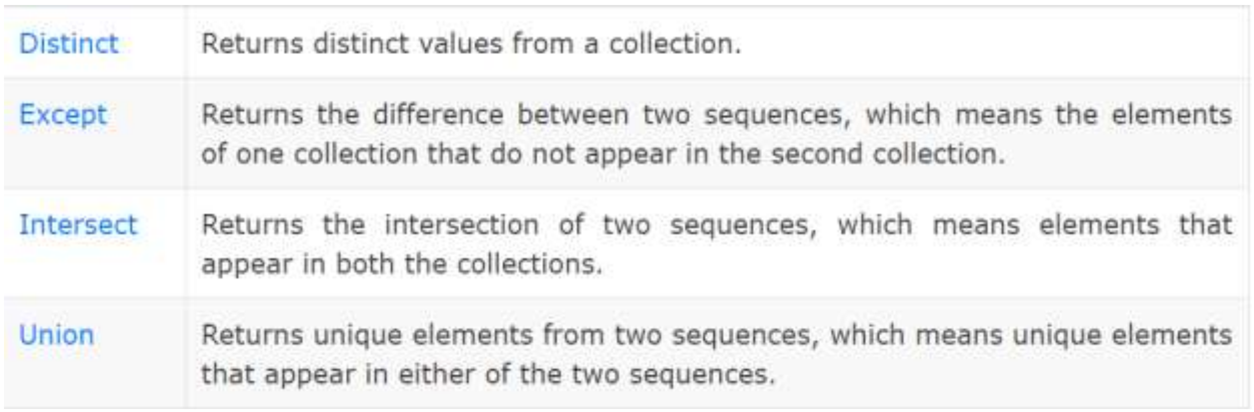
23. DefaultEmpty-Generation Operator
The DefaultIfEmpty() method returns a new collection with the default value if the given collection on which DefaultIfEmpty() is invoked is empty.
Another overload method of DefaultIfEmpty() takes a value parameter that should be replaced with default value.

24. Empty, Range, Repeat-Generation Operators:
LINQ includes generation operators DefaultIfEmpty, Empty, Range & Repeat. The Empty, Range & Repeat methods are not extension methods for IEnumerable or IQueryable but they are simply static methods defined in a static class Enumerable.



Empty	Returns an empty collection
Range	Generates collection of IEnumerable<T> type with specified number of elements with sequential values, starting from first element.
Repeat	Generates a collection of IEnumerable<T> type with specified number of elements and each element contains same specified value.

25. Distinct-Set Operator

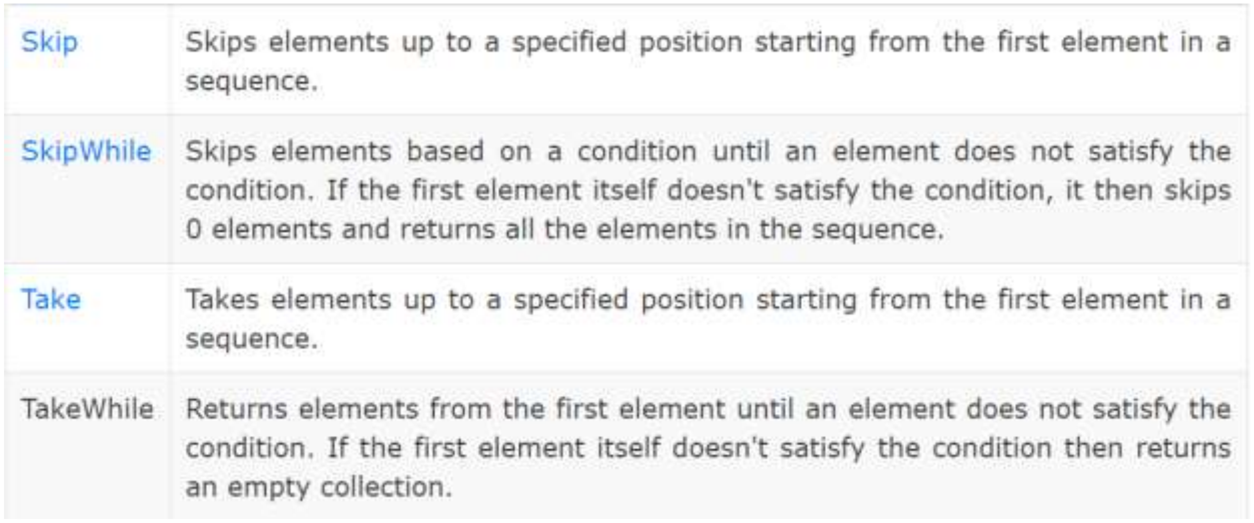


Distinct	Returns distinct values from a collection.
Except	Returns the difference between two sequences, which means the elements of one collection that do not appear in the second collection.
Intersect	Returns the intersection of two sequences, which means elements that appear in both the collections.
Union	Returns unique elements from two sequences, which means unique elements that appear in either of the two sequences.

The Distinct extension method doesn't compare values of complex type objects. You need to implement IEqualityComparer<T> interface in order to compare the values of complex types. In the following example, StudentComparer class implements IEqualityComparer<Student> to compare Student< objects.

26. Except The Except() method requires two collections. It returns a new collection with elements from the first collection which do not exist in the second collection (parameter collection).
27. Intersect-Set Operator
The Intersect extension method requires two collections. It returns a new collection that includes common elements that exists in both the collection.
28. Union-Set Operator
The Union extension method requires two collections and returns a new collection that includes distinct elements from both the collections.
he Union extension method doesn't return the correct result for the collection of complex types. You need to implement IEqualityComparer interface in order to get the correct result from Union method.

29. Skip, SkipWhile-Partitioning Operators
Partitioning operators split the sequence (collection) into two parts and return one of the parts.



Skip	Skips elements up to a specified position starting from the first element in a sequence.
SkipWhile	Skips elements based on a condition until an element does not satisfy the condition. If the first element itself doesn't satisfy the condition, it then skips 0 elements and returns all the elements in the sequence.
Take	Takes elements up to a specified position starting from the first element in a sequence.
TakeWhile	Returns elements from the first element until an element does not satisfy the condition. If the first element itself doesn't satisfy the condition then returns an empty collection.

30. Take, TakeWhile Partitioning Operators
Partitioning operators split the sequence (collection) into two parts and returns one of the parts.
The Take() extension method returns the specified number of elements starting from the first element.
The TakeWhile() extension method returns elements from the given collection until the specified condition is true. If the first element itself doesn't satisfy the condition then returns an empty collection.
The TakeWhile method has two overload methods. One method accepts the predicate of Func<TSource, bool> type and the other overload method accepts the predicate Func<TSource, int, bool> type that passes the index of element.
In the following example, TakeWhile() method returns a new collection that includes all the elements till it finds a string whose length less than 4 characters.