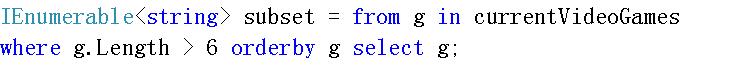
# LINQ: Language Integrated Query

A consistent, symmetrical manner  to  obtain and manipulate “data”

SQL-like query operations

* Interact with numerous types of data
  + IEnumerable<T>
  + RDB, DataSet, XML….

Give me the items inside of currentVideoGames that have more than six characters, ordered alphabetically



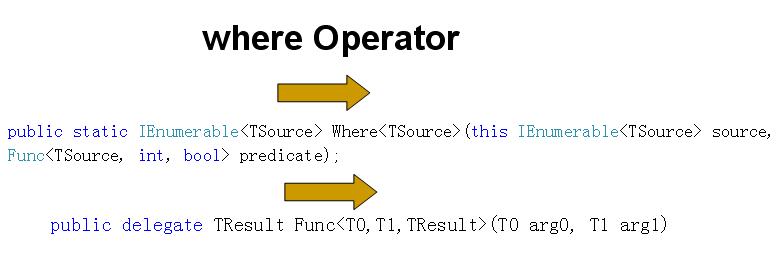
System.Array doesn’t Impl IEnumerable<T>, yet Static System.Linq.Enumerable extends its behavior.

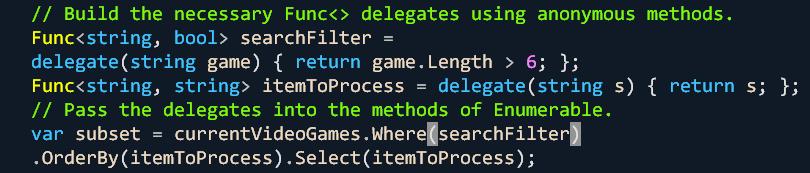
No execution until iterating over the contents, e.g. ToArray()

If an Enumerable contains different data types, filter data using OfType<T>(). It can also convert non-generic data type(e.g. ArrayList) to typed enumerable.

// The following query shows that the standard query operators such as // Where() can be applied to the ArrayList type after calling OfType(). IEnumerable<string> query2 = fruits.OfType<string>().Where(fruit => fruit.ToLower().Contains("n"));

C# query operators (such as from, in,  where, orderby, and select) compiles to various methods of the System.Linq.Enumerable or other types





* Query expressions are created using various C# query operators.
* Query operators are shorthand notations for invoking methods defined by the Enumerable type.
* Any method requiring a delegate parameter can instead be passed a lambda expression.
* Lambda expressions are simply anonymous methods in disguise.
* Anonymous methods are shorthand notations for allocating a raw delegate.

LINQ can do almost every thing SQL can do: sort, order, group, join, equals, distinct, count, max....

* Limitation of var
  + Can’t be parameters, return values, or fields of a class type
* Abandon var for strong typed data
  + E.g. IEnumerable<string>
* Transforming Query Results to Array Types