LINQ CHEAT SHEET

Filtering

```
var col = from o in Orders
            where o.CustomerID == 84
            select o;
```

```
var col2 = Orders.Where(o => o.CustomerID == 84);
```

Return Anonymous Type

```
var col = from o in orders
            select new
            {
                OrderID = o.OrderID,
                Cost = o.Cost
            };
```

```
var col2 = orders.Select(o => new
            {
                OrderID = o.OrderID,
                Cost = o.Cost
            }
);
```

Ordering

```
var col = from o in orders
            orderby o.Cost ascending
            select o;
```

```
var col2 = orders.OrderBy(o => o.Cost);
```

```
var col3 = from o in orders
            orderby o.Cost descending
            select o;
```

```
var col4 = orders.OrderByDescending(o => o.Cost);
```

```
var col9 = from o in orders
            orderby o.CustomerID, o.Cost descending
            select o;
```

```
var col6 = orders.
                OrderBy(o => o.CustomerID).
                ThenByDescending(o => o.Cost);
```

```
//returns same results as above
var col5 = from o in orders
            orderby o.Cost descending
            orderby o.CustomerID
            select o;
//NOTE the ordering of the orderby's
```

Joining

```
var col = from c in customers
        join o in orders on
        c.CustomerID equals o.CustomerID
        select new
        {
            c.CustomerID,
            c.Name,
            o.OrderID,
            o.Cost
        };
```

```
var col2 = customers.Join(orders,
        c => c.CustomerID, o => o.CustomerID,
        (c, o) => new
        {
            c.CustomerID,
            c.Name,
            o.OrderID,
            o.Cost
        }
);
```

Grouping

```
var OrderCounts = from o in orders
                group o by o.CustomerID into g
                select new
                {
                    CustomerID = g.Key,
                    TotalOrders = g.Count()
                };
```

```
var OrderCounts1 = orders.GroupBy(
        o => o.CustomerID).
        Select(g => new
        {
            CustomerID = g.Key,
            TotalOrders = g.Count()
        }
);
```

LINQ CHEAT SHEET

Paging (using Skip & Take)

```
//select top 3
var col = (from o in orders
            where o.CustomerID == 84
            select o).Take(3);
```

```
var col2 = orders.Where(
    o => o.CustomerID == 84
).Take(3);
```

```
//skip first 2 and return the 2 after
var col3 = (from o in orders
            where o.CustomerID == 84
            orderby o.Cost
            select o).Skip(2).Take(2);
```

```
var col4 = (from o in orders
            where o.CustomerID == 84
            orderby o.Cost
            select o).Skip(2).Take(2);
```

Element Operators (Single, Last, First, ElementAt, Defaults)

```
//throws exception if no elements
var cust = (from c in customers
            where c.CustomerID == 84
            select c).Single();
```

```
var cust = customers.Single(
            c => c.CustomerID == 84
            );
```

```
//returns null if no elements
var cust = (from c in customers
            where c.CustomerID == 84
            select c).SingleOrDefault();
```

```
var cust = customers.SingleOrDefault(
            c => c.CustomerID == 84
            );
```

```
//returns a new customer instance if no elements
var cust = (from c in customers
            where c.CustomerID == 85
            select c).
            DefaultIfEmpty(new Customer()).
            Single();
```

```
var cust = customers.Where(c => c.CustomerID == 85).
            DefaultIfEmpty(new Customer()).
            Single();
```

```
//First, Last and ElementAt used in same way
var cust = (from o in orders
            where o.CustomerID == 84
            orderby o.Cost
            select o).Last();
```

```
var cust = orders.Where(o => o.CustomerID == 84).
            OrderBy(o => o.Cost).
            Last();
```

```
//returns 0 if no elements
var i = (from c in customers
         where c.CustomerID == 85
         select c.CustomerID).SingleOrDefault();
```

```
var j = customers.Where(c => c.CustomerID == 85).
         Select(o => o.CustomerID).
         SingleOrDefault();
```

LINQ CHEAT SHEET

Set Operators (Zip, Distinct, Except, Intersect, Union, Concat, SequenceEqual)

```csharp
var numbers = new[] { 1, 2, 3 };
var words = new[] { "one", "two", "three" };

var zipped = from pair in numbers.Zip(
        words, (n, w) => new { Number = n, Word = w }
    ) select $"{pair.Number}: {pair.Word}";

foreach (var item in zipped)
{
    Console.WriteLine(item);
}
// 1: one
// 2: two
// 3: three
```

```csharp
var numbers = new[] { 1, 2, 3 };
var words = new[] { "one", "two", "three" };

var zipped = numbers.Zip(words, (n, w) => $"{n}: {w}");

foreach (var item in zipped)
{
    Console.WriteLine(item);
}
// 1: one
// 2: two
// 3: three
```

```csharp
var numbers = new[] { 1, 2, 2, 3, 4, 4, 5 };
var result = (from n in numbers
            select n).Distinct();
// { 1, 2, 3, 4, 5 }
```

```csharp
var numbers = new[] { 1, 2, 2, 3, 4, 4, 5 };
var distinctNumbers = numbers.Distinct();
// { 1, 2, 3, 4, 5 }
```

```csharp
var numbers1 = new[] { 1, 2, 3 };
var numbers2 = new[] { 3, 4, 5 };
var result = (from n in numbers1
            select n)
        .Union(from n in numbers2
                select n);
// { 1, 2, 3, 4, 5 }
```

```csharp
var numbers1 = new[] { 1, 2, 3 };
var numbers2 = new[] { 3, 4, 5 };
var unionNumbers = numbers1.Union(numbers2);
// { 1, 2, 3, 4, 5 }
```

```csharp
var numbers1 = new[] { 1, 2, 3 };
var numbers2 = new[] { 3, 4, 5 };
var result = (from n in numbers1
            select n)
        .Intersect(from n in numbers2
                    select n);
// { 3 }
```

```csharp
var numbers1 = new[] { 1, 2, 3 };
var numbers2 = new[] { 3, 4, 5 };
var intersectNumbers = numbers1.Intersect(numbers2);
// { 3 }
```

```csharp
var numbers1 = new[] { 1, 2, 3, 4 };
var numbers2 = new[] { 3, 4, 5 };
var result = (from n in numbers1
            select n)
        .Except(from n in numbers2
                select n);
// { 1, 2 }
```

```csharp
var numbers1 = new[] { 1, 2, 3, 4 };
var numbers2 = new[] { 3, 4, 5 };
var exceptNumbers = numbers1.Except(numbers2);
// { 1, 2 }
```

```csharp
var numbers1 = new[] { 1, 2, 3, 4 };
var numbers2 = new[] { 3, 4, 5 };
var result = (from n in numbers1
            select n)
        .Concat(from n in numbers2
                select n);
// { 1, 2, 3, 3, 4, 5 }
```

```csharp
var numbers1 = new[] { 1, 2, 3 };
var numbers2 = new[] { 3, 4, 5 };
var concatenatedNumbers = numbers1.Concat(numbers2);
// { 1, 2, 3, 3, 4, 5 }
```

```csharp
var numbers1 = new[] { 1, 2, 3 };
var numbers2 = new[] { 1, 2, 3 };
bool areEqual = (from n in numbers1
                select n)
                .SequenceEqual(from n in numbers2
                                select n);
```

```csharp
var numbers1 = new[] { 1, 2, 3 };
var numbers2 = new[] { 1, 2, 3 };
var areEqual = numbers1.SequenceEqual(numbers2);
// true
```

```csharp
var people = new[]
{
    new { Name = "Alice", Age = 30 },
    new { Name = "Bob", Age = 40 },
    new { Name = "Charlie", Age = 30 },
    new { Name = "David", Age = 40 },
    new { Name = "Eve", Age = 30 }
};
var grpByAge = from person in people
                group person by person.Age into agegrp
                select agegrp;
```

```csharp
var people = new[] {
    new { Name = "Alice", Age = 30 },
    new { Name = "Bob", Age = 40 },
    new { Name = "Charlie", Age = 30 }
};
var groupedByAge = people.GroupBy(p => p.Age);
// Age group: 30
// Alice
// Charlie
// Age group: 40
// Bob
```

LINQ CHEAT SHEET

Conversions

ToArray

```
string[] names = (from c in customers select c.Name).ToArray();
```

ToDictionary

```
Dictionary<int, Customer> col = customers.ToDictionary(c => c.CustomerID);
Dictionary<string, double> customerOrdersWithMaxCost = (from oc in

    (from o in orders
     join c in customers on o.CustomerID equals c.CustomerID
     select new { c.Name, o.Cost })

    group oc by oc.Name into g
    select g).
    ToDictionary(g => g.Key, g => g.Max(oc => oc.Cost));
```

ToList

```
List<Order> ordersOver10 = (from o in orders
                            where o.Cost > 10
                            orderby o.Cost
                            select o).ToList();
```

ToLookup

```
ILookup<int, string> customerLookup = customers.ToLookup(c => c.CustomerID, c => c.Name);
```

Contenu tiré en grande partie de https://vslapp.wordpress.com/wp-content/uploads/2011/11/linq-cheatsheet.pdf