

# LINQ Cheat Sheet (Español – Inglés)

## SINTAXIS DE CONSULTA / QUERY SYNTAX

### Filtrado / Filtering

#### CONSULTA / QUERY

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

#### LAMBDA

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

### Ordenamiento / Ordering

#### CONSULTA / QUERY

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

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

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

var col5 = from o in orders
            orderby o.Cost descending
            orderby o.CustomerID
            select o; // Nota: orden importa
```

#### LAMBDA

```
var col2 = orders.OrderBy(o => o.Cost);
var col4 = orders.OrderByDescending(o => o.Cost);

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

### Agrupación / Grouping

#### CONSULTA / QUERY

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

NOTA: la clave del grupo (g.Key) tiene el mismo tipo que la expresión usada para agrupar.
```

#### LAMBDA

```
var orderCounts1 = orders
    .GroupBy(o => o.CustomerID)
    .Select(g => new
    {
        CustomerID = g.Key,
        TotalOrders = g.Count()
    });
```

## SINTAXIS LAMBDA / LAMBDA SYNTAX

### Tipo Anónimo / Anonymous Type

#### CONSULTA / QUERY

```
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
});
```

#### LAMBDA

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

### Unión / Joining

#### CONSULTA / QUERY

```
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
    });
```

#### LAMBDA

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

### Paginación / Paging (Skip & Take)

#### CONSULTA / QUERY

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

// saltar 2 y tomar 2
var col3 = (from o in orders
            where o.CustomerID == 84
            orderby o.Cost
            select o).Skip(2).Take(2);
```

#### LAMBDA

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

## Operadores de Elementos / Element Operators

### CONSULTA / QUERY

```
// Single - excepción si no hay elementos
var c1 = (from c in customers
        where c.CustomerID == 84
        select c).Single();

// SingleOrDefault - null si no hay elementos
var c2 = (from c in customers
        where c.CustomerID == 84
        select c).SingleOrDefault();

// DefaultIfEmpty + Single
var c3 = (from c in customers
        where c.CustomerID == 85
        select c)
        .DefaultIfEmpty(new Customer())
        .Single();

// Last con orden
var o1 = (from o in orders
        where o.CustomerID == 84
        orderby o.Cost
        select o).Last();
```

### LAMBDA

```
var c4 = customers.Single(c => c.CustomerID == 84);
var c5 = customers.SingleOrDefault(c => c.CustomerID == 84);

var c6 = customers.Where(c => c.CustomerID == 85)
                  .DefaultIfEmpty(new Customer())
                  .Single();

var o2 = orders.Where(o => o.CustomerID == 84)
               .OrderBy(o => o.Cost)
               .Last();

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

NOTA: Single/Last/First/ElementAt lanzan excepción si la secuencia está vacía. Las variantes \*OrDefault devuelven default(T).

## Conversiones / Conversions

### CONSULTA / QUERY

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

// 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);
```

### LAMBDA

```
// ToDictionary simple
var dict = customers.ToDictionary(
    c => c.CustomerID);

// ToDictionary compuesto
var 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));
```

## Notas generales LINQ / General LINQ Notes

- **Single, First, Last, ElementAt:** lanzan excepción si no hay elemento válido.
- **\*OrDefault:** devuelven default(T) si no hay elemento.
- Tipos de referencia → null; tipos de valor → valor por defecto (0, false, etc.).
- Combina filtros (where) antes de estos operadores para evitar excepciones.