# Ejecutar PLINQ

Para ejecutar una instrucción LINQ paralela mediante PLINQ, tendremos que agregar en la parte del `from` después del nombre del objeto, el método `AsParallel()`:

```
public UsoParallelLINQ() {
    var source = Enumerable.Range(100, 20000);

    // Result sequence might be out of order.
    var parallelQuery = from num in source.AsParallel()
                        where num % 10 == 0
                        select num;

    // Para leer una ParallelLinQ es muy recomendable usar el metodo ForAll.
    parallelQuery.ForAll((e) => {
        Console.WriteLine($"PARALLEL LINQ - FOR ALL - {e}");
    });

    // Or use foreach to merge results first.
    foreach (int n in parallelQuery)
        Console.WriteLine($"PARALLEL LINQ - parallelQuery - {n}");

    // You can also use ToArray, ToList, etc as with LINQ to Objects.
    var parallelQuery2 = (from num in source.AsParallel()
                        where num % 10 == 0
                        select num).ToArray();

    foreach (var x in parallelQuery2)
        Console.WriteLine($"PARALLEL LINQ - parallelQuery2 - {x}");

    // Method syntax is also supported
    var parallelQuery3 = source.AsParallel().Where(n => n % 10 == 0).Select(n =>
n);

    foreach (var z in parallelQuery3)
        Console.WriteLine($"PARALLEL LINQ - parallelQuery3 - {z}");

    Console.WriteLine("\nPress any key to exit...");
    Console.ReadLine();
}
```

La consulta crea particiones del origen, dividiéndolo en tareas que se ejecutan de forma asíncrona en varios subprocesos. El orden en el que se completa cada tarea no depende solo de la cantidad de trabajo necesario para procesar los elementos de la partición.

Hay que tener en cuenta que puede ser que esta consulta PLINQ puede que vaya mas lenta que una consulta LINQ secuencial, para eso es mejor comprobar el rendimiento de la instrucción con herramientas como las que dispone VS

## `ForAll()` o `Parallel.ForEach/Parallel.For`

`ParallelLinQ` se ejecuta en varios subprocesos, pero cuando se ejecuta un bucle `for/foreach` los resultados se deben de fusionar en un solo subproceso para poder ser leidos, en cambio, con `ForAll()` (siempre que sea posible) intentará que cada subproceso de la consulta genere por si solo su propio resultado, por tanto, por lo general se ejecutará mas rápido.