Parallel LinQ (P LinQ)

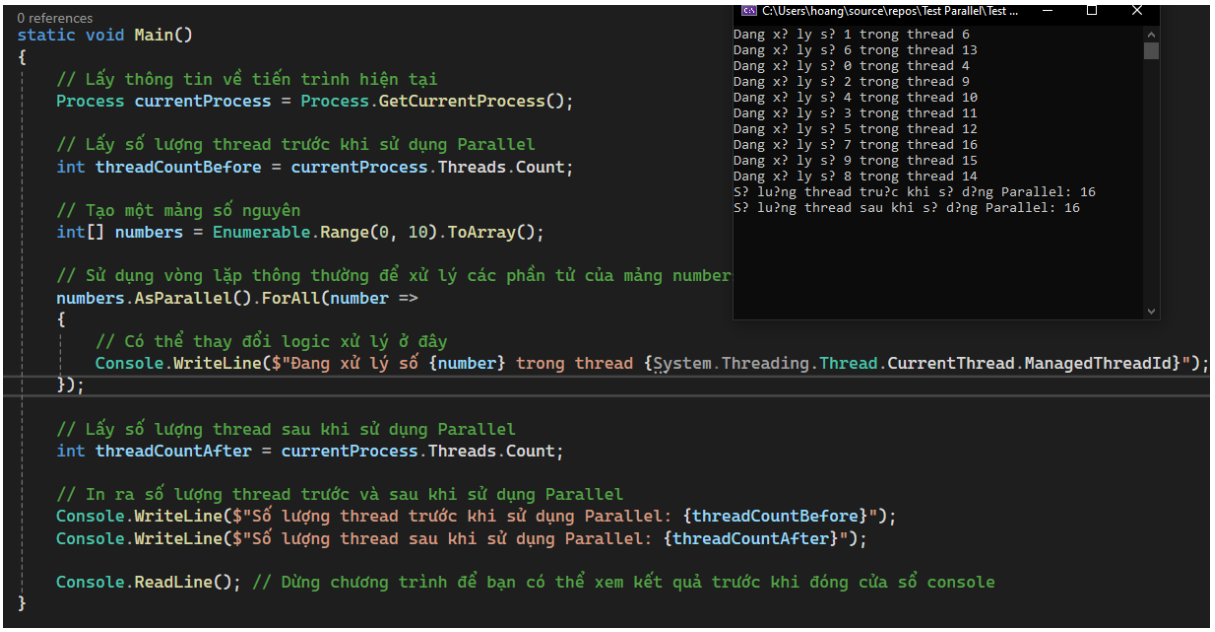
. AsParallel

TH : Ko dùng AsParallel sẽ là 1 thread

```
Program                                                    Main()

eferences
atic void Main()

  // Lấy thông tin về tiến trình hiện tại
  Process currentProcess = Process.GetCurrentProcess();

  // Lấy số lượng thread trước khi sử dụng Parallel
  int threadCountBefore = currentProcess.Threads.Count;

  // Tạo một mảng số nguyên
  int[] numbers = Enumerable.Range(0, 10).ToArray();

  // Sử dụng vòng lặp thông thường để xử lý các phần tử của mảng numbers
  foreach (var number in numbers)
  {
      // Có thể thay đổi logic xử lý ở đây
      Console.WriteLine($"Đang xử lý số {number} trong thread {System.Threading.Thread.CurrentThread.ManagedThreadId}");
  }

  // Lấy số lượng thread sau khi sử dụng Parallel
  int threadCountAfter = currentProcess.Threads.Count;

  // In ra số lượng thread trước và sau khi sử dụng Parallel
  Console.WriteLine($"Số lượng thread trước khi sử dụng Parallel: {threadCountBefore}");
  Console.WriteLine($"Số lượng thread sau khi sử dụng Parallel: {threadCountAfter}");

  Console.ReadLine(); // Dừng chương trình để bạn có thể xem kết quả trước khi đóng cửa sổ console
```

```
C:\Users\hoang\source\repos\Test Parallel\Test ...
Dang x? ly s? 0 trong thread 1
Dang x? ly s? 1 trong thread 1
Dang x? ly s? 2 trong thread 1
Dang x? ly s? 3 trong thread 1
Dang x? ly s? 4 trong thread 1
Dang x? ly s? 5 trong thread 1
Dang x? ly s? 6 trong thread 1
Dang x? ly s? 7 trong thread 1
Dang x? ly s? 8 trong thread 1
Dang x? ly s? 9 trong thread 1
S? lu?ng thread tru?c khi s? d?ng Parallel: 17
S? lu?ng thread sau khi s? d?ng Parallel: 17
```

TH2 : Trường hợp có AsParallel

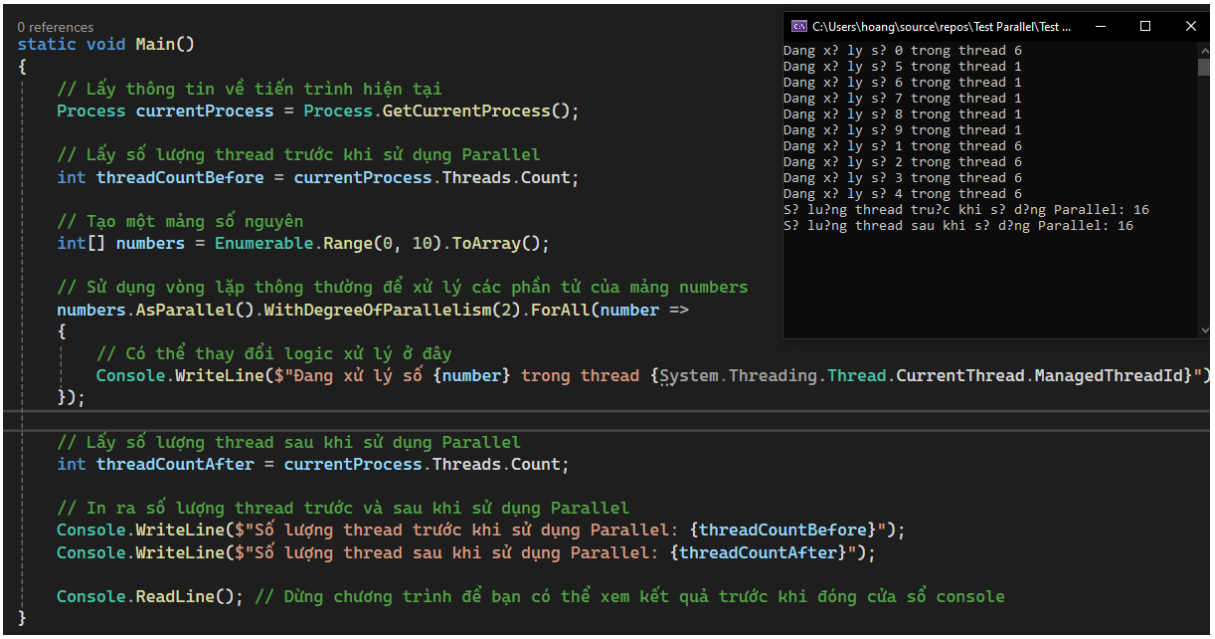


```
0 references
static void Main()
{
    // Lấy thông tin về tiến trình hiện tại
    Process currentProcess = Process.GetCurrentProcess();

    // Lấy số lượng thread trước khi sử dụng Parallel
    int threadCountBefore = currentProcess.Threads.Count;

    // Tạo một mảng số nguyên
    int[] numbers = Enumerable.Range(0, 10).ToArray();

    // Sử dụng vòng lặp thông thường để xử lý các phần tử của mảng number
    numbers.AsParallel().ForAll(number =>
    {
        // Có thể thay đổi logic xử lý ở đây
        Console.WriteLine($"Đang xử lý số {number} trong thread {System.Threading.Thread.CurrentThread.ManagedThreadId}");
    });

    // Lấy số lượng thread sau khi sử dụng Parallel
    int threadCountAfter = currentProcess.Threads.Count;

    // In ra số lượng thread trước và sau khi sử dụng Parallel
    Console.WriteLine($"Số lượng thread trước khi sử dụng Parallel: {threadCountBefore}");
    Console.WriteLine($"Số lượng thread sau khi sử dụng Parallel: {threadCountAfter}");

    Console.ReadLine(); // Dừng chương trình để bạn có thể xem kết quả trước khi đóng cửa sổ console
}
```

```
C:\Users\hoang\source\repos\Test Parallel\Test ...
Dang x? ly s? 1 trong thread 6
Dang x? ly s? 6 trong thread 13
Dang x? ly s? 0 trong thread 4
Dang x? ly s? 2 trong thread 9
Dang x? ly s? 4 trong thread 10
Dang x? ly s? 3 trong thread 11
Dang x? ly s? 5 trong thread 12
Dang x? ly s? 7 trong thread 16
Dang x? ly s? 9 trong thread 15
Dang x? ly s? 8 trong thread 14
S? lu?ng thread tru?c khi s? d?ng Parallel: 16
S? lu?ng thread sau khi s? d?ng Parallel: 16
```

. WithDegreeOfParallelism

Lưu ý : ở đây nó đã tập trung vào **2 thread** như ta đã yêu cầu



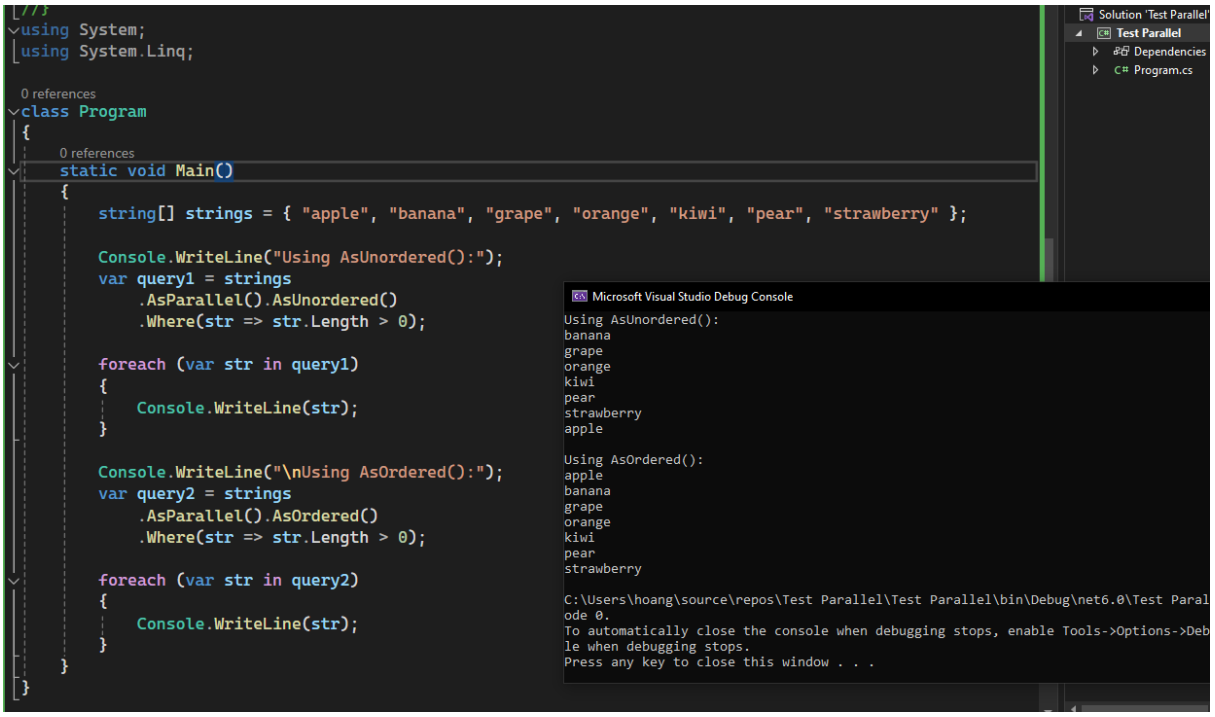
```
0 references
static void Main()
{
    // Lấy thông tin về tiến trình hiện tại
    Process currentProcess = Process.GetCurrentProcess();

    // Lấy số lượng thread trước khi sử dụng Parallel
    int threadCountBefore = currentProcess.Threads.Count;

    // Tạo một mảng số nguyên
    int[] numbers = Enumerable.Range(0, 10).ToArray();

    // Sử dụng vòng lặp thông thường để xử lý các phần tử của mảng numbers
    numbers.AsParallel().WithDegreeOfParallelism(2).ForAll(number =>
    {
        // Có thể thay đổi logic xử lý ở đây
        Console.WriteLine($"Đang xử lý số {number} trong thread {System.Threading.Thread.CurrentThread.ManagedThreadId}");
    });

    // Lấy số lượng thread sau khi sử dụng Parallel
    int threadCountAfter = currentProcess.Threads.Count;

    // In ra số lượng thread trước và sau khi sử dụng Parallel
    Console.WriteLine($"Số lượng thread trước khi sử dụng Parallel: {threadCountBefore}");
    Console.WriteLine($"Số lượng thread sau khi sử dụng Parallel: {threadCountAfter}");

    Console.ReadLine(); // Dừng chương trình để bạn có thể xem kết quả trước khi đóng cửa sổ console
}
```

```
C:\Users\hoang\source\repos\Test Parallel\Test ...
Dang x? ly s? 0 trong thread 6
Dang x? ly s? 5 trong thread 1
Dang x? ly s? 6 trong thread 1
Dang x? ly s? 7 trong thread 1
Dang x? ly s? 8 trong thread 1
Dang x? ly s? 9 trong thread 1
Dang x? ly s? 1 trong thread 6
Dang x? ly s? 2 trong thread 6
Dang x? ly s? 3 trong thread 6
Dang x? ly s? 4 trong thread 6
S? lu?ng thread tru?c khi s? d?ng Parallel: 16
S? lu?ng thread sau khi s? d?ng Parallel: 16
```

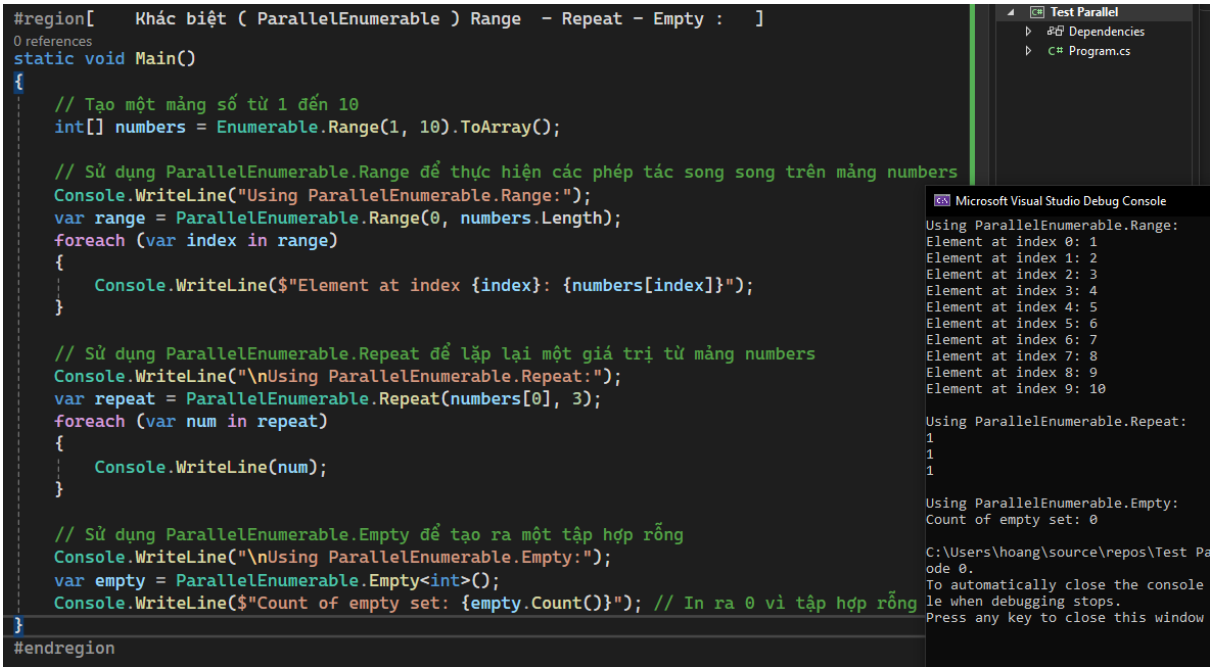
AsOrdered And AsUnordered

Khác biệt AsOrdered va AsUnordered:

- AsOrdered khi trả về kết quả sẽ được **sắp xếp thứ tự y chang kết quả đầu vào**
- AsUnOrdered khi trả về kết quả sẽ ko theo bất kỳ quy tắc nào cả và điều đó giúp tăng tốc câu lệnh

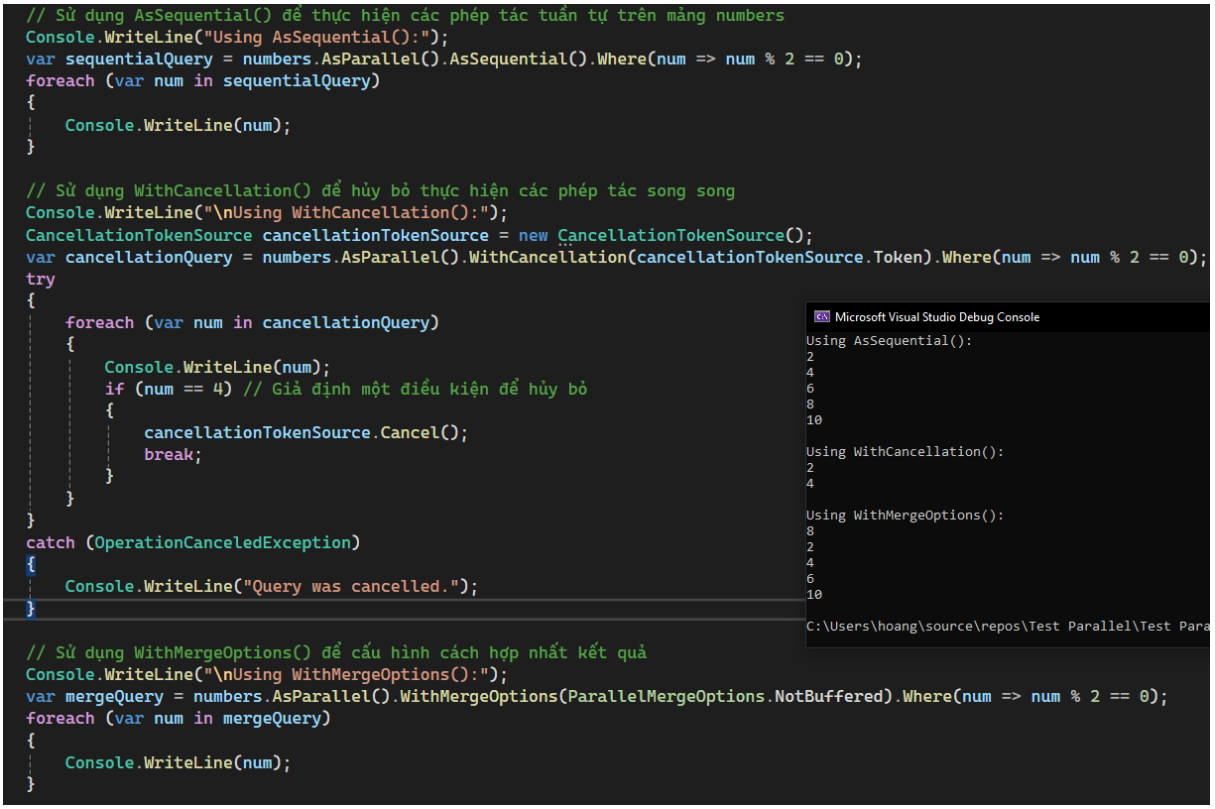


. ParallelEnumerable.Range vs ParallelEnumerable.Repeat vs ParallelEnumerable.Empty



. AsSequential vs WithCancellation vs WithMergeOptions

- AsSequential ở đây là Chuyển đổi PLINQ (Parallel LINQ) sang LINQ tuần tự (trái ngược với (PLINQ) Có thể kết hợp với AsParallel
- WithCancellation ở đây là sẽ cho phép quăng ra lỗi trước khi xuất ra đáp án
- WithMergeOptions



. Partitioning in PLINQ

29:35

. WithExecutionMode

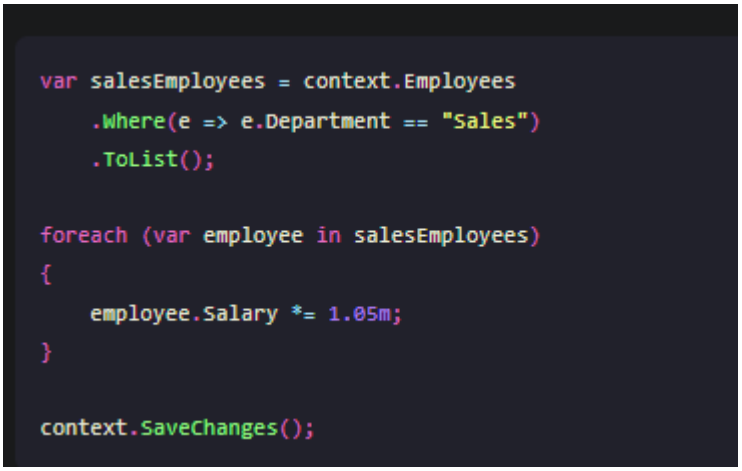
31:04

. ForAll

Query Splitting

```
List<Department> departments =
    context.Departments
        .Include(d => d.Teams)
        .ThenInclude(t => t.Employees)
        .ThenInclude(e => e.Tasks)
        .ToList();
```

Bulk Updates and Deletes



```
var salesEmployees = context.Employees
    .Where(e => e.Department == "Sales")
    .ToList();

foreach (var employee in salesEmployees)
{
    employee.Salary *= 1.05m;
}

context.SaveChanges();
```

thành thế này

```
context.Employees
    .Where(e => e.Department == "Sales")
    .ExecuteUpdate(s => s.SetProperty(e => e.Salary, e =>
e.Salary * 1.05m));
```

```
context.Carts
    .Where(o => o.CreatedOn < DateTime.Now.AddYears(-1))
    .ExecuteDelete();
```

Raw SQL Queries

```
public class ProductSummary
{
    public int ProductId { get; set; }
    public string ProductName { get; set; }
    public decimal TotalSales { get; set; }
}

var productSummaries = await context.Database
    .SqlQuery<ProductSummary>(
        @$"""
        SELECT p.ProductId, p.ProductName, SUM(oi.Quantity
* oi.UnitPrice) AS TotalSales
        FROM Products p
```

```
        JOIN OrderItems oi ON p.ProductId = oi.ProductId
        WHERE p.CategoryId = {categoryId}
        GROUP BY p.ProductId, p.ProductName
        """)
    .ToListAsync();
```

Query Filters

Eager Loading