**Operadores de Colecciones**

**Where**:

Se utiliza para filtrar elementos de una secuencia basándose en una condición especificada.

**SelectMany**:

Se utiliza para realizar proyecciones y aplanar secuencias anidadas.

**Select**:

Se utiliza para realizar proyecciones, es decir, transformar cada elemento de una secuencia de acuerdo con una función.

**Any:**

Verifica si al menos un elemento de una secuencia cumple con una condición dada.

**First:**

Devuelve el primer elemento de una secuencia que cumple con una condición específica.

**Last:**

Devuelve el último elemento de una secuencia que cumple con una condición específica.

**Take:**

Devuelve un número especificado de elementos desde el principio de una secuencia.

**Skip:**

Omite un número especificado de elementos desde el principio de una secuencia y devuelve los restantes.

**OrderBy:**

Ordena los elementos de una secuencia en orden ascendente o descendente según una clave especificada.

**OfType:**

Filtra los elementos de una secuencia basándose en un tipo específico.

**ToList:**

Convierte una secuencia en una lista.

**ToArray:**

Convierte una secuencia en un array.

**ToDictionary:**

Convierte una secuencia en un diccionario basándose en una función de clave y una función de valor.

**GroupBy:**

Agrupa los elementos de una secuencia según una clave específica.

**EjemplosOperadoresColecciones:**

using System;

using System.Linq;

using System.Collections.Generic;

class Program

{

static void Main()

{

// Crear una lista de números

List<int> numeros = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

// Filtrar números mayores que 5

var numerosMayoresQueCinco = numeros.Where(n => n > 5);

// Proyectar el cuadrado de cada número

var cuadrados = numeros.Select(n => n \* n);

// Verificar si hay algún número par

var hayNumeroPar = numeros.Any(n => n % 2 == 0);

// Obtener el primer número mayor que 3

var primerNumeroMayorQueTres = numeros.First(n => n > 3);

// Ordenar los números en orden descendente

var numerosOrdenadosDescendentes = numeros.OrderByDescending(n => n);

// Tomar los primeros 3 números

var primerosTresNumeros = numeros.Take(3);

// Saltar los primeros 2 números

var numerosDespuesDelSegundo = numeros.Skip(2);

// Convertir la lista a un array

var arrayDeNumeros = numeros.ToArray();

// Crear un diccionario con el número como clave y su cuadrado como valor

var diccionarioDeCuadrados = numeros.ToDictionary(n => n, n => n \* n);

// Imprimir resultados

Console.WriteLine("Números mayores que 5: " + string.Join(", ", numerosMayoresQueCinco));

Console.WriteLine("Cuadrados de los números: " + string.Join(", ", cuadrados));

Console.WriteLine("¿Hay algún número par? " + hayNumeroPar);

Console.WriteLine("Primer número mayor que 3: " + primerNumeroMayorQueTres);

Console.WriteLine("Números ordenados descendentes: " + string.Join(", ", numerosOrdenadosDescendentes));

Console.WriteLine("Primeros tres números: " + string.Join(", ", primerosTresNumeros));

Console.WriteLine("Números después del segundo: " + string.Join(", ", numerosDespuesDelSegundo));

Console.WriteLine("Array de números: " + string.Join(", ", arrayDeNumeros));

Console.WriteLine("Diccionario de cuadrados: " + string.Join(", ", diccionarioDeCuadrados.Select(kv => $"{kv.Key}^2 = {kv.Value}")));

Console.ReadLine();

}

}

**Para Crear threads, no se recomienda el uso de threads, en su defecto se recomienda el uso de Sync/Await, usar juntos en caso contrario pierde efectividad**

**Ejemplo:**

using System;

using System.Threading.Tasks;

class Program

{

static async Task Main()

{

Console.WriteLine("Inicio del programa en el hilo principal.");

// Iniciar una tarea asincrónica en un hilo separado

await RealizarTareaAsincronica();

Console.WriteLine("Fin del programa en el hilo principal.");

Console.ReadLine();

}

static async Task RealizarTareaAsincronica()

{

Console.WriteLine("Inicio de la tarea asincrónica.");

// Simular una operación asincrónica, como una llamada a una API o una operación de E/S

await Task.Delay(2000); // Simulando una espera de 2 segundos

Console.WriteLine("Fin de la tarea asincrónica.");

}

}

**System.reflection para obtener información sobre un tipo y sus miembros**

**Ejemplo:**

using System;

using System.Reflection;

class Program

{

static void Main()

{

// Obtener información sobre el tipo de la clase Program

Type tipoProgram = typeof(Program);

// Imprimir el nombre del tipo

Console.WriteLine("Nombre del tipo: " + tipoProgram.FullName);

// Obtener información sobre los métodos del tipo

MethodInfo[] metodos = tipoProgram.GetMethods();

foreach (MethodInfo metodo in metodos)

{

Console.WriteLine("Método: " + metodo.Name);

}

// Obtener información sobre las propiedades del tipo

PropertyInfo[] propiedades = tipoProgram.GetProperties();

foreach (PropertyInfo propiedad in propiedades)

{

Console.WriteLine("Propiedad: " + propiedad.Name);

}

// ... Puedes continuar explorando otros miembros del tipo

Console.ReadLine();

}

}