



Código recursivo

1 Código recursivo

2 Inversión

3 Otro ejemplo

4 Bibliografía

Temas

- 1 Código recursivo
 - Recursividad estructural
 - Consumer
 - Ejecución en memoria

Lista recursiva en Java

Código: Lista recursiva

```
1 public class Lista<E>
2 {
3     private E _dato;
4     public E Dato
5     {
6         get => _dato;
7
8         [MemberNotNull(nameof(_dato))]
9         set
10        {
11            if (value == null) throw new NullReferenceException("No datos nulos.");
12            _dato = value;
13        }
14    }
15    public Lista<E>? Siguiente { get; private set; }
16
17    /** Construye una lista con un dato, seguida de otra lista. */
18    public Lista(E dato, Lista<E>? siguiente)
19    {
20        Dato = dato;
21        Siguiente = siguiente;
22    }
23 }
```

Temas

- 1 Código recursivo
 - Recursividad estructural
 - Consumer
 - Ejecución en memoria

Delegate Action

- En .NET, un delegate es un tipo de objeto que hace referencia a una función.
- Al especificar el delegate se debe dar el encabezado deseado de la función precedido por la palabra reservada `delegate`.

Ej:

```
public delegate void Action<in T>(T obj);
```

- El **delegate** `Action<in T>`, que ya viene definido en la API .NET, representa una operación que recibe un argumento y no devuelve resultados. Se espera que su efecto sea colateral.^[1]

^[1] [System Delegates](#) s.f.

Action

Código: Lista recursiva

```
1 public class Lista<E> {
2     // ...
3     /** Versión funcional de un método que trabaja con la lista. */
4     public static void ImprimeLista<T>(Lista<T>? l)
5     {
6         if(l == null) return;           // Caso base, escrito explícitamente.
7         else
8         {
9             Console.WriteLine(l.Dato);
10            ImprimeLista(l.Siguiente);
11        }
12    }
13
14    public static void Aplica<T, S> (Lista<T>? l, Action<S> op) where T : S
15    {
16        if(l != null)
17        {
18            op(l.Dato);
19            Aplica(l.Siguiente, op);
20        }
21    }
22 }
```

```
1 public class UsoLista
2 {
3     public static void Main()
4     {
5         Lista<string> l = new Lista<string>("Martinillo",
6             new Lista<string>("¿Dónde_", new Lista<string>("estás?", null)));
7         Lista.Lista<string>.Aplica<string, string>(l,
8             dato => Console.WriteLine(dato));
9     }
10 }
```

Código: Ejecución

```
1 Martinillo
2 ¿Dónde
3 estás?
```


Temas

- 1 Código recursivo
 - Recursividad estructural
 - Consumer
 - Ejecución en memoria

```

1 public static void Aplica<T, S>
2   (Lista<T>? l, Action<S> op)
3   where T : S
4 {
5     if(l != null)
6     {
7         op(l.Dato);
8         Aplica(l.Siguiente, op);
9     }
10 }

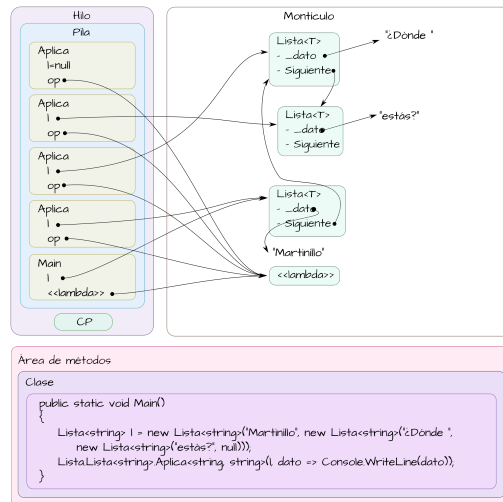
```

Código: Ejecución

```

1 Martinillo
2 ¿Dónde
3 estás?

```



Inversión

- 1 Código recursivo
- 2 Inversión
- 3 Otro ejemplo
- 4 Bibliografía

Temas

- 2 Inversión
 - Recursión invertida
 - Ejecución en memoria

Consumer

Código: Lista recursiva

```
1 public static void AplicaInversa<T, S> (Lista<T>? l, Action<S> op) where T : S
2 {
3     if(l != null)
4     {
5         Aplica(l.Siguiente, op);
6         op(l.Dato);
7     }
8 }
9
10 public static void Main()
11 {
12     Lista<string> l = new Lista<string>("Martinillo", new Lista<string>("¿Dónde",
13         new Lista<string>("estás?", null)));
14     AplicaInversa<string, string>(l, dato => Console.WriteLine(dato));
15 }
```

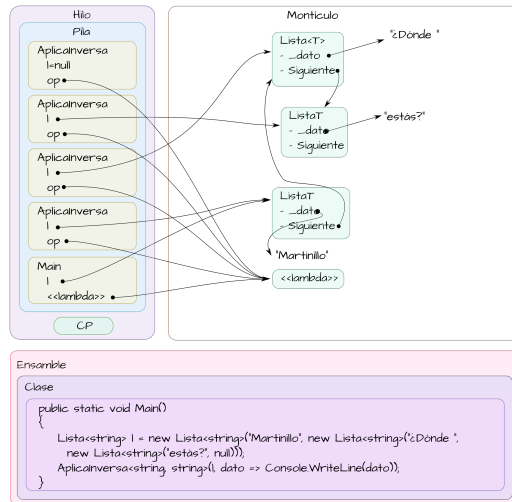
Temas

- 2 Inversión
 - Recursión invertida
 - Ejecución en memoria

```
1 public static void AplicaInversa<T, S>  
2   (Lista<T>? l, Action<S> op) where T : S  
3 {  
4     if(l != null)  
5     {  
6       Aplica(l.Siguiente, op);  
7       op(l.Dato());  
8     }  
9 }
```

Código: Ejecución

```
1 estás?  
2 ¿Dónde  
3 Martinillo
```



Otro ejemplo

- 1 Código recursivo
- 2 Inversión
- 3 Otro ejemplo
- 4 Bibliografía

Reducción

Código: Reduce suma y multiplicación

```
1 public class Lista<E>
2 {
3     public static T Reduce<T>(Lista<T>? l, Func<T, T, T> op, T valorInicial)
4     {
5         if (l == null) return valorInicial;
6         else return op(l.Dato, Reduce(l.Siguiente, op, valorInicial));
7     }
8 }
9 public class UsoLista
10 {
11     public static void Main(String[] args)
12     {
13         Action<object> impresora = dato => Console.WriteLine(dato);
14         Console.WriteLine("Sea la lista:");
15         Lista<int> lint = new Lista<int>(-8, new Lista<int>(9, new Lista<int>(4, null)));
16         Lista.Lista<string>.Aplica(lint, impresora);
17
18         int res = Lista.Lista<string>.Reduce(lint, (n1, n2) => n1 + n2, 0);
19         Console.WriteLine("La suma de sus elementos es: " + res);
20         int resm = Lista.Lista<string>.Reduce(lint, (n1, n2) => n1 * n2, 1);
21         Console.WriteLine("La multiplicación de sus elementos es: " + resm);
22     }
23 }
```


Código: Ejecución

```
1 Sea la lista:
2 -8
3 9
4 4
5 La suma de sus elementos es: 5
6 La multiplicación de sus elementos es: -288
```

Bibliografía

- 1 Código recursivo
- 2 Inversión
- 3 Otro ejemplo
- 4 Bibliografía

Bibliografía I

 *System Delegates* (s.f.). URL: <https://learn.microsoft.com/en-us/dotnet/api/system?view=net-7.0#delegates>.

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual

