

# Clase 17

Viernes, 13 Octubre 2023

---

## [Indice](#)

### Funciones complejas de contenedores

- **FUNCION** le paso una lista de enteros y una posicion '*index*'. La funcion elimina el valor que se encuentra en la posicion '*index*'

```
,
    public static void RemoveElement(List<int> list, int index)
    {
        list.RemoveAt(index);
    }

    // Elimina el elemento que esta en la posicion 'index'

    public static void RemoveElement(List<int> list, int index)
    {
        list.Remove(index);
    }
    ,
    // Elimina el elemento cuyo valor es 'index'
```

- La función '**Remove()**' esta **PROHIBIDA**. Siempre usaremos '**RemoveAt()**'
- **Funcion** que le paso una lista y un valor que quiero borrar.

```
,
    public static void RemoveValue(List<int> list, int value)
    {
        if (list == null)
            return;

        for (int i = 0; i < list.Count; i++)
        {
            if (list[i] == value)
            {
                list.RemoveAt(i);
                i--;
            }
        }
    }
    ,
```

- Si le introduzco un punto de ruptura '**break**' inmediatamente sale del bucle donde está anidado.  
Un '**IF**' no es un bucle, solo '**WHILE**' Y '**FOR**'
- **Funcion** con un punto de ruptura '**BREAK**'

```
,
    public static void RemoveValue(List<int> list, int value)
    {
        for(int i = 0; i < list.Count; i++)
        {
            list.RemoveAt(i);
            i--;
            break;           // Rompe el bucle anidado 'FOR'
        }
    }
    ,
```

- **IMPORTANTE:** Si utilizamos **i++** ó **i--** como una **expresion** su funcionamiento es distinto al habitual:  
**i++**  
Primero la variable asignada es **IGUAL** a '*i*' y cuando se evalua toda la expresión se **INCREMENTA** la '*i*'.  
**++i**  
Primero la variable asignada es el **INCREMENTO** de '*i*' y cuando se evalua toda la expresión se **INCREMENTA** la '*i*'.

- **FUNCION:** Se le pasa una lista y queremos borrar de esta lista otra lista de valores que se le pasan.

•

```
,
public static void RemoveValues(List<int> list, List<int> listValues)
{
    for(int j = 0; j < listValues; j++)
    {
        for(int i = 0; i < list; i++)
        {
            if(list[i] == list[j])
                list.RemoveAt(i--);
        }
    }
}

public static void RemoveValues(List<int> list, List<int> listValues)
{
    for(int i = 0; i < listValues; i++)
    {
        for(int j = 0; j < list; j++)
        {
            if(list[i] == list[j])
                list.RemoveAt(i--);
        }
    }
}
,
```

#### Consejos y Sugerencias

- **CONSEJO:** No es adecuado contener un 'FOR' dentro de otro 'FOR'. Es mejor solución llamar a las funciones desde el interior del primer 'FOR'
- Si trabajas con 'Listas' es mejor no devolver nada. Por otro lado si trabajas con 'Arrays' lo normal es devolver un 'Array'
- **WARNING:** Cuando declaramos un tipo de dato, el compilador no espera que pueda devolver 'null'.  
Si añadimos ? al final del tipo de dato, le indicamos al compilador que este datos **SI PUEDE SER NULL**.  
int[]? --> PUEDE SER NULL  
int[] --> NO DEBERÍA SER NULL