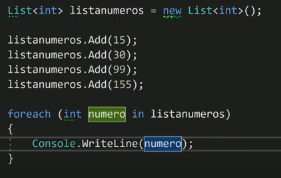
**COLECCIONES**

**¿Qué son?**

* **Son clases genéricas que pertenecen al namespace System.Collections.Generic.**
* **Estas clases permiten almacenar elementos**
* **No tienen las limitaciones de los arrays**
* **Permiten:**
  + **Ordenar**
  + **Añadir**
  + **Eliminar**
  + **Buscar**
  + **Etc.**

**Colecciones más frecuentes:**

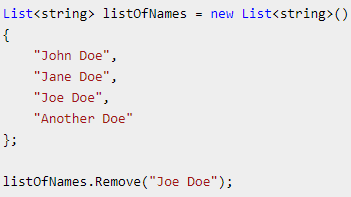
**Ejemplos prácticos:**

**Cabe recalcar que se ejemplificara con listas porque son las colecciones más utilizadas.**

**Se puede ver a la lista de tipo de int que pertenece a las colecciones agregándole nuevos elementos y con el foreach se recorre la misma lista para imprimir los elementos en consola.**

**Actualmente hay tres métodos a su disposición cuando desea eliminar uno o varios elementos de una lista: *Remove()*, *RemoveAt()* y *RemoveAll()*.**

**El método Remove() toma solo un parámetro: el elemento que desea eliminar.**



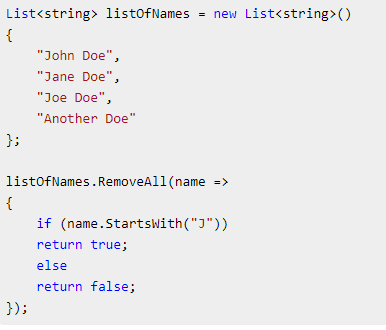
**El método RemoveAt() aprovecha el hecho de que la lista genérica se basa en índices al permitirle eliminar un elemento en función de su índice / posición en la lista. Por ejemplo, podría eliminar el primer elemento de la lista así:**



**O el último elemento de la lista así:**



**RemoveAll() toma un delegado a un método como parámetro y este método decide si un elemento debe eliminarse o no devolviendo verdadero o falso.**



**Para ordenar se puede utilizar el método Sort() y Reverse() el primer método se encargara en caso de string de ordenarlo alfabéticamente y reverse viceversa. Sin embargo para objetos específicos se pueden ordenar implementado la interfaz IComparable o usando LINQ.**



**EXPRESIONES LAMBDA**

**¿Qué son?**

**Una expresión lambda es un método anónimo, que sustituye a una instancia de delegado. El compilador transforma una expresión lambda en un delegado.**

**Sintaxis:**

* **Expresión lambda**

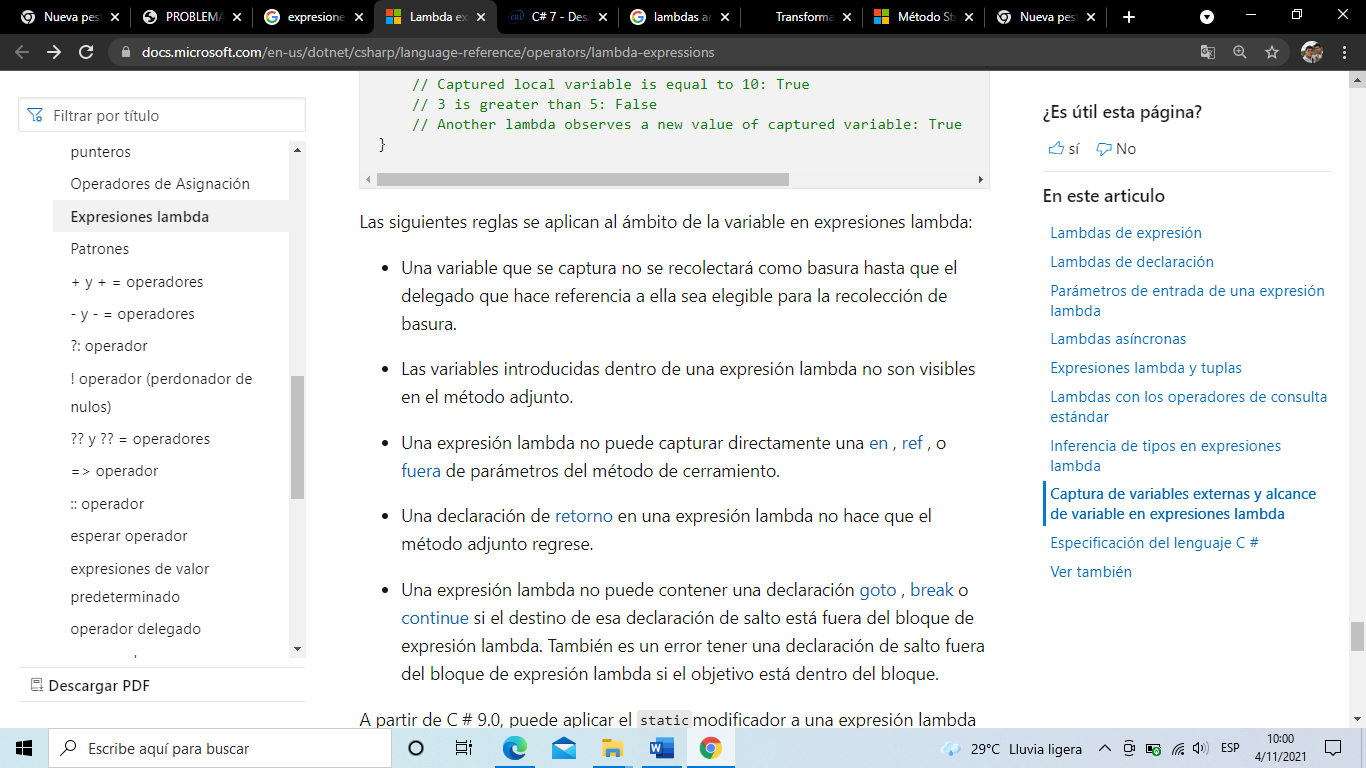
(input-parameters) => expression

**En caso, que solo se contiene un parámetro se puede omitir los paréntesis.**

* **Instrucción Lambda**

**Este tiene un bloque de instrucción como cuerpo:**

(input-parameters) => {<sequence-of-statements>}



* **Lambda Asincrónica**

**Puede crear fácilmente expresiones y declaraciones lambda que incorporen procesamiento asincrónico mediante el uso de las palabras clave “async” y “await”.**

**Ejemplos prácticos:**

Func<int, int> square = x => x \* x;

Console.WriteLine(square(5));

// Output:

// 25

**En este sencillo ejemplo, se observa una sintaxis de expresión lambda, donde tiene una entrada y salida de tipo entero y como parámetro esta la “x” y este al ser único, es opcional utilizar los paréntesis y al final este devuelve el valor de x al cuadrado, y en este se asignó una variable de tipo delegado.(Func)**

**Cabe recalcar al utilizar una expresión lambda, se requiere de una instancia de tipos delegados o arboles de expresión.**

**También una expresión lambda se pueden convertir a los tipos de árboles de expresión.**

System.Linq.Expressions.Expression<Func<int, int>> e = x => x \* x;

Console.WriteLine(e);

// Output:

// x => (x \* x)

int[] numbers = { 2, 3, 4, 5 };

var squaredNumbers = numbers.Select(x => x \* x);

Console.WriteLine(string.Join(" ", squaredNumbers));

// Output:

// 4 9 16 25

**Este se trabaja con el arreglo números, y se escribe la palabra “select” para transformar los datos en nuevas variantes, también se tiene como parámetro a la x y este va devolver el cuadrado de cada numero del conjunto, y se imprime con la palabra “Join” para concatenar el resultado de los elementos del arreglo.**

**LINQ**

**¿Qué son?**

**Es un conjunto de extensiones integradas en el lenguaje C#, que nos permite trabajar de manera cómoda y rápida con colecciones de datos. Es decir, podemos llevar a cabo inserciones, selecciones y borrados, así como operaciones sobre sus elementos.**

**Todas estas operaciones las vamos a conseguir muy fácilmente gracias a los métodos de extensión para colecciones que nos ofrece el espacio de nombres "System.Linq" y a las expresiones lambda.**

**Las Operaciones que podemos llevar a cabo con LinQ son las siguientes:**

* **Select**

**Nos va a permitir hacer una selección sobre la colección de datos, ya sea seleccionándolos todos, solo una parte o transformándolos:**

var nombresAlumnos = alumnos.Select(x => x.Nombre).ToList();

* **Where**

**Nos permite seleccionar una colección a partir de otra con los objetos que cumplan las condiciones especificadas:**

var alumnosAprobados = alumnos.Where(x => x.Nota >= 5).ToList();

* **First/Last**

**Esta extensión nos va a permitir obtener respectivamente el primer y el último objeto de la colección. Esto es especialmente útil si la colección está ordenada.**

var primero = alumnos.First();

var ultimo = alumnos.Last();

* OrderBy

Gracias a este método, vamos a poder ordenar la colección en base a un criterio de ordenación que le indicamos mediante una expresión lambda. Análogamente, también existe OrderByDescending, el cual va a ordenar la colección de manera inversa según el criterio:

var ordenadoMenorAMayor = alumnos.OrderBy(x => x.Nota).ToList();

var ordenadoMayorAMenos = alumnos.OrderByDescending(x => x.Nota).ToList();

* Sum

Como hemos visto más arriba, nos va a permitir sumar la colección:

var sumaNotas = alumnos.Sum(x => x.Nota);

* Max/Min

Gracias a esta extensión, vamos a poder obtener los valores máximo y mínimo de la colección:

var notaMaxima = alumnos.Max(x => x.Nota);

var notaMinima = alumnos.Min(x => x.Nota);

* Average

Este método nos va a devolver la media aritmética de los valores (numéricos) de los elementos que le indiquemos de la colección:

var media = alumnos.Average(x => x.Nota);

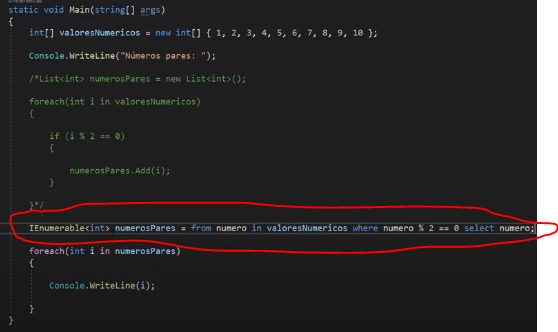
* All/Any

Con este último operador, vamos a poder comprobar si todos o alguno de los valores de la colección cumplen el criterio que le indiquemos:

var todosAprobados = alumnos.All(x => x.Nota >= 5);

var algunAprobado = alumnos.Any(x => x.Nota >= 5);

**Ejemplos prácticos:**



Este ejemplo es acerca de como con una expresión LINQ implementando la interfaz IEnumerable almacenamos los numeros pares de los elementos de un array sin necesidad de utilizar un for y un if.