

# C# II

**Instructor:** Ing. Gabriel Roa  
**Email:** gabrielrb14a@gmail.com  
**Teléfono:** 0424-7592768  
**Fecha:** Enero del 2022

# AGENDA

- ¿Qué son Funciones Lambda?
- Función Lambda vs Expresión Lambda.
- Sintaxis.
- Usos comunes.
- Enumerable.
- Métodos de Enumerable.
- Ejercicios propuestos.
- Ejercicio práctico – SQL Server.

# Funciones Lambda

- ¿Cómo instanciamos una función en C#?

```
public int ElevarAlCuadrado(int numero){  
    return numero * numero;  
}
```

# Funciones Lambda

- Esta no es la única manera, puesto que existen formas más cortas y rápidas de definir funciones en caso de que estas sean de pequeña complejidad.
- Esta forma se ve representada por las **funciones Lambda**.
- Debe su nombre al Cálculo Lambda, un sistema matemático para expresar computación en máquinas de Turing.



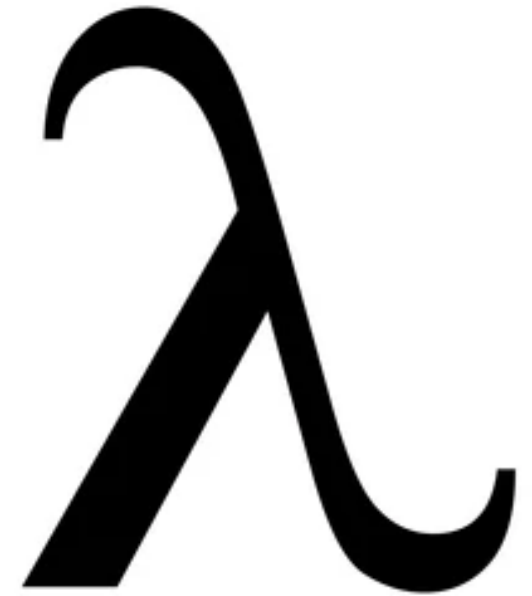
# Sintaxis



```
(parametro) => operación
```

# Características

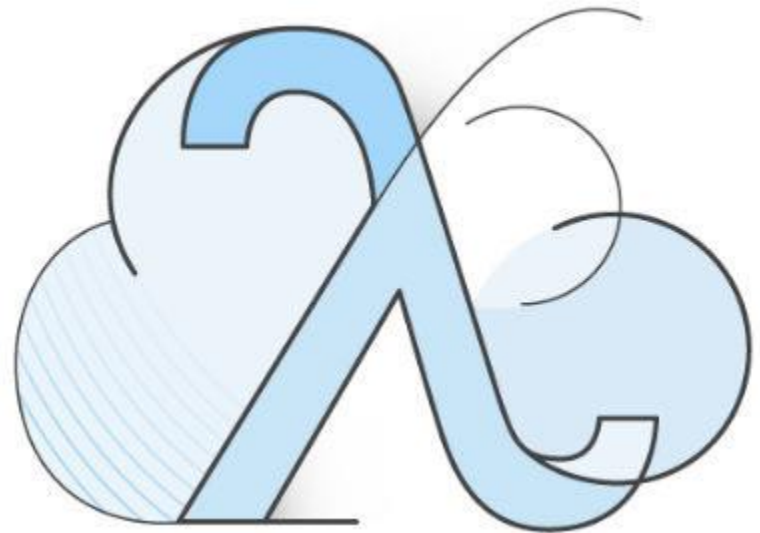
- Son funciones cortas y de baja complejidad.
- Son funciones *desechables*, que necesitamos sólo en un contexto local.
- Son funciones anónimas (aunque pueden *no* serlo), en el sentido que no tienen nombre en el momento de instanciarse.
- Expresión: Sólo una línea de código. Función: Más de una línea de código.



shutterstock.com • 437401495

# Expresiones vs Funciones Lambda

- Una expresión Lambda es una función que sigue esta sintaxis y sólo contiene una línea de código.
- Esta línea de código define una operación que representa el valor de retorno de la función.
- No hace falta la palabra return ni hace falta indicar el tipo de dato del valor de retorno o de los parámetros.



# Expresiones vs Funciones Lambda

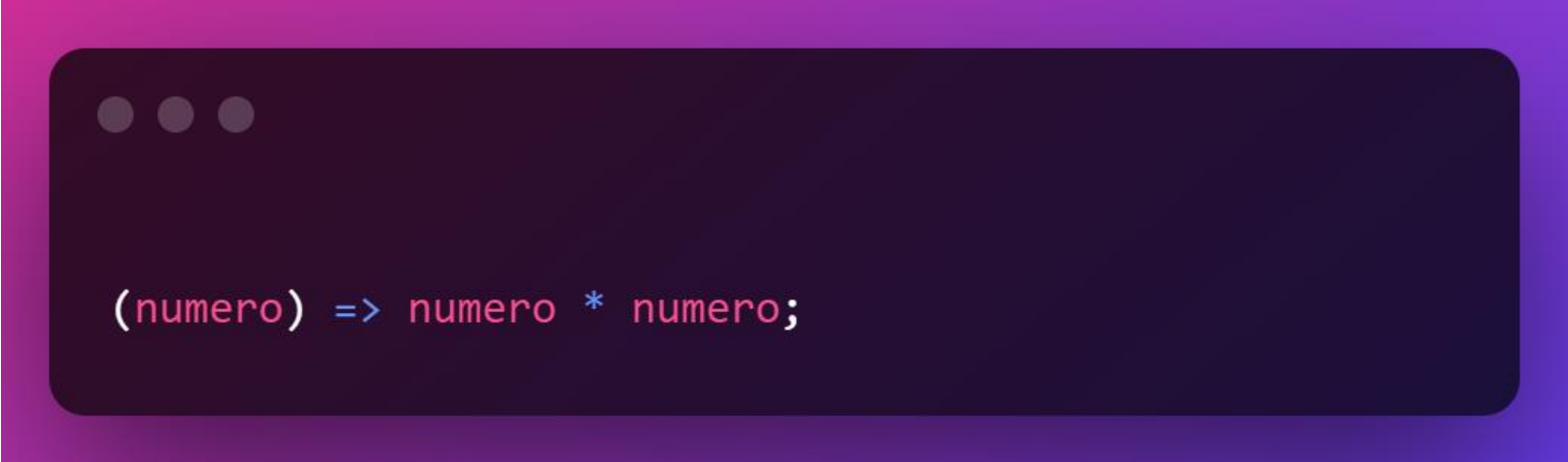
- Una función Lambda es una expresión lambda que contiene más de una línea de código.
- Estas líneas definen las operaciones a ejecutar, y al final se debe retornar un valor explícitamente.
- No se debería utilizar funciones lambda de más de 3 o 4 líneas de código. En ese caso, es más recomendable definir una función tradicional.





# Sintaxis


- Está comprendida por los parámetros, una flecha ( $\Rightarrow$ ) y una o más expresiones a ejecutar.



```
(numero) => numero * numero;
```

# Sintaxis

- Pueden recibir tantos parámetros como desee, se separan con una coma.



```
(nro1, nro2) => nro1 * nro2;
```

# Sintaxis

- Si la función o expresión tiene un solo parámetro, se puede prescindir de los paréntesis.



```
numero => numero * numero;
```

# Sintaxis


- Puedo definir una función lambda sin parámetros, al utilizar sólo paréntesis vacíos.



```
() => Console.WriteLine();
```

# Sintaxis

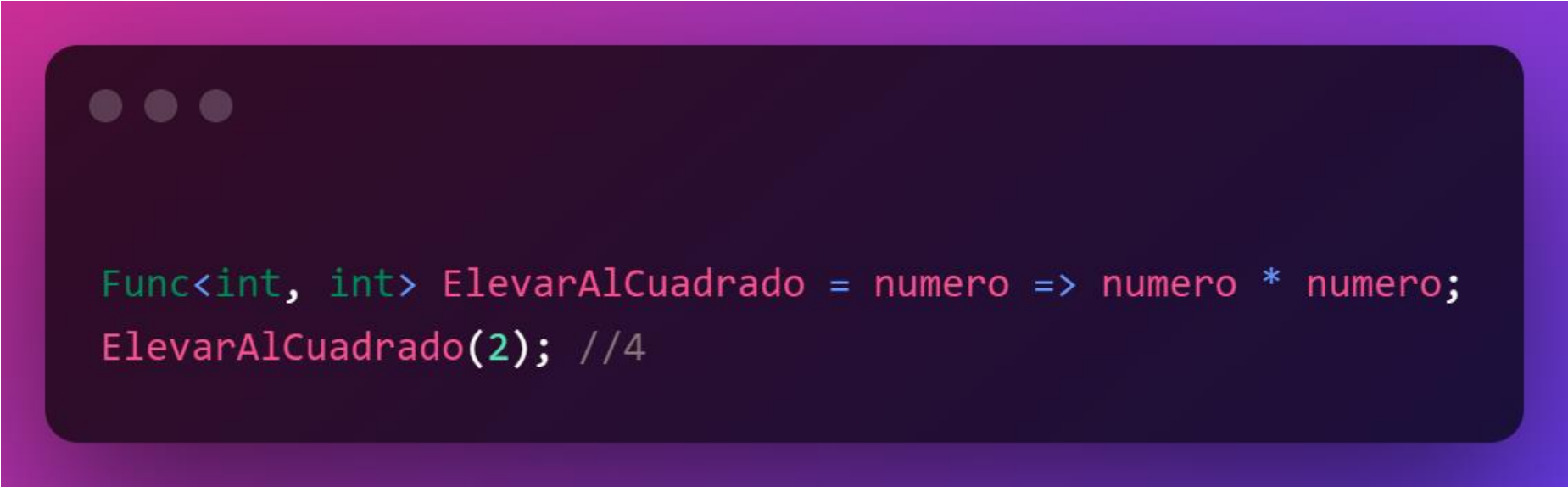
- La función Lambda encierra sus líneas de código entre llaves.



```
(numero) => {  
    return numero * numero;  
}
```

# Sintaxis

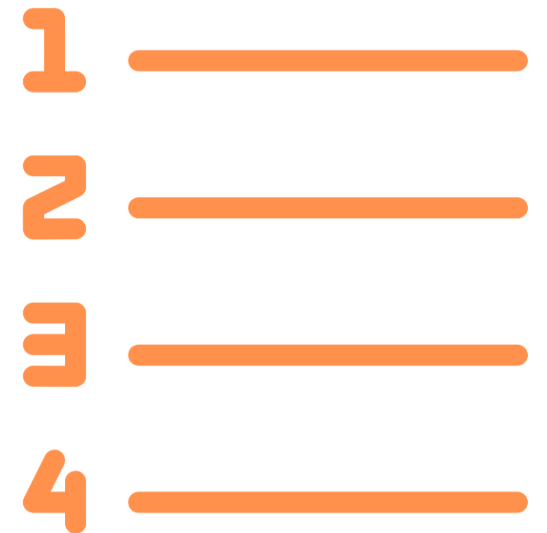
- Puedo nombrar una función o expresión Lambda al utilizar Func<>.



```
Func<int, int> ElevarAlCuadrado = numero => numero * numero;  
ElevarAlCuadrado(2); //4
```

# Enumerable

- Es una clase ofrecida por Microsoft en su paquete System.Linq que nos permite escribir código para ejecutar sentencias similares a Linq utilizando métodos.
- Estas sentencias las escribo en objetos que implementen la interfaz IEnumerable<>



# Enumerable

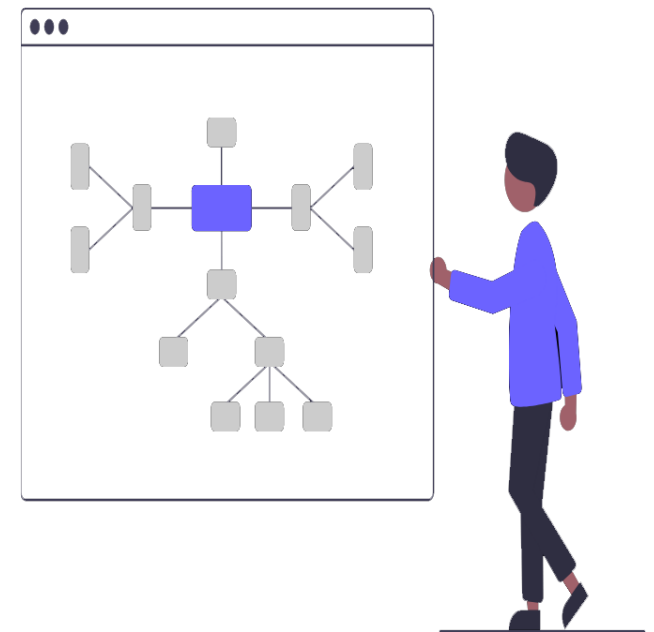
- LINQ (Language Integrated Query) es un componente de .NET que permite consultar datos de manera nativa en cualquier colección de elementos.
- Se puede utilizar con una sintaxis explícita, similar a SQL, o a través de los métodos de Enumerable.





# Enumerable

- Nos ofrece un sinfín de métodos que podemos utilizar para manipular colecciones de datos.
- En su mayoría, estos métodos reciben funciones lambda como parámetros, las cuales definen los procedimientos a seguir para la ejecución de los mismos.



# Count

- Puede devolver el conteo normal de una colección, o puedo especificar una condición (mediante lambda) y saber la cantidad de elementos que la cumplen.

```
int cantidadDueñosMax3Mascotas = dueños.Count(  
    d => d.CantidadMaximaMascotas == 3);
```

# First

- Devuelve el primer elemento de una colección; o el primer elemento que cumpla con una condición dada.

```
Dueño dosMascotas = dueños.First(d => d.Mascotas.Count == 2);
```

# Last

- Análogo a First, pero con el último elemento.

```
Dueño dosMascotas = dueños.Last(d => d.Mascotas.Count == 2);
```

# Single

- Devuelve un único elemento de una colección que cumpla con una condición dada.

```
Dueño dosMascotas = dueños.Single(d => d.Cedula == "1234567");
```

# OrderBy

- Ordena los elementos de manera ascendente dada una propiedad.  
Esta propiedad se selecciona mediante lambda.

```
var dueñosPorFechaNacimiento = dueños.OrderBy(d => d.FechaNacimiento);
```

# Where

- Análogo al Where de SQL, devuelve una subcolección con todos los elementos que satisfagan una condición dada.

```
var dueñosPorJ = dueños.Where(d => d.Nombre.StartsWith("J"));
```

# Sum

- Acumula una propiedad de un objeto dado en una colección de elementos.

```
int cantidadTotalMascotas = dueños.Sum(d => d.CantidadMaximaMascotas);
```



# Average

- Devuelve el promedio de una propiedad dada una colección de elementos.

```
double cantidadTotalMascotas = dueños.Average(d => d.CantidadMaximaMascotas);
```

# OrDefault

- Valida para que, en caso de que ocurra cualquier tipo de excepción al ejecutar una sentencia, esta devuelva un objeto *default* en lugar de la excepción en sí.
- Se puede utilizar con First, Single o Last.



# Combinación de métodos

- Todos estos métodos pueden combinarse entre sí, cual sentencia SQL.
- Se concatenan al hacer llamados sucesivos de los mismos.

```
int cantidadDueñosConGatos = dueños.Where  
    (d => d.Mascotas.Count(m => m.Especie == "Gato") != 0)  
    .Count();
```

## Ejercicios propuestos

- Dado el proyecto Mascotas, ubicado en el aula virtual, escriba el código que le permita extraer:
  - Cuántos dueños hay con 3 mascotas exactamente.
  - Mascotas con más de 5 kilos de peso, cuyo nombre comience por J.
  - El primer dueño que haya nacido el 13 de marzo del 84.
  - Todas las mascotas, ordenadas por peso y luego por fecha de nacimiento.
  - Promedio de edad de las mascotas cuyos dueños son menores de edad.

# Referencias Bibliográficas

- Farrell, J. (2018). *Microsoft Visual C# 2017: An Introduction to Object-Oriented Programming*. Boston: Cengage Learning.
- Griffiths, I. (2019). *Programming C# 8.0*. Sebastopol: O'Reilly Media, Inc.
- Microsoft. (28 de 01 de 2021). *C# documentation*. Obtenido de Microsoft Docs: <https://docs.microsoft.com/en-us/dotnet/csharp/>