Expresiones Lambda

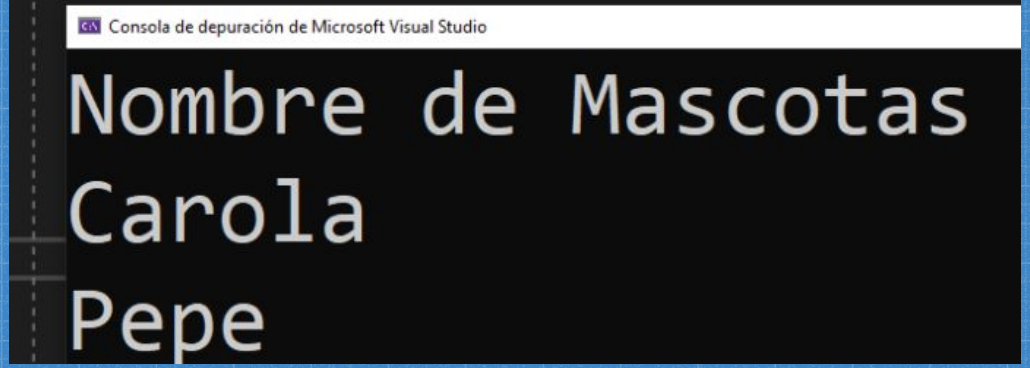
Las expresiones lambda se utilizan para crear métodos anónimos

Normalmente, se utilizan como argumentos de métodos que tienen delegados como parámetros de entrada

El operador lambda => se utiliza para separar la lista de parámetros del cuerpo del método anónimo

Las expresiones lambda pueden ser convertidas a un tipo delegado. El tipo delegado deberá coincidir con los parámetros y el tipo de retorno de la expresión

```
static void Main(string[] args)
{

    Action<string> MostrarPorPantalla;

    MostrarPorPantalla = (x) => Console.WriteLine(x);

    MostrarPorPantalla("Nombre de Mascotas");
    MostrarPorPantalla("Carola");
    MostrarPorPantalla("Pepe");
```



Si la expresión lambda no devuelve un valor puede convertirse a un delegado de tipo Action

Si, por el contrario, tiene un valor de retorno puede convertirse a un delegado de tipo Func

```
1 referencia
enum EPais{Argentina, Polonia,Brasil}
0 referencias
static void Main(string[] args)
{
    // No devuelve nada
    Action<string> Mostrar;
    Mostrar = (x) => Console.WriteLine(x);

    Mostrar("Hola Mundo");

    //Devuelve un string
    Func<Enum,string> InfoPais;
    InfoPais = (p) => $"El pais es: {p}";

    Console.WriteLine(InfoPais(EPais.Polonia));
```

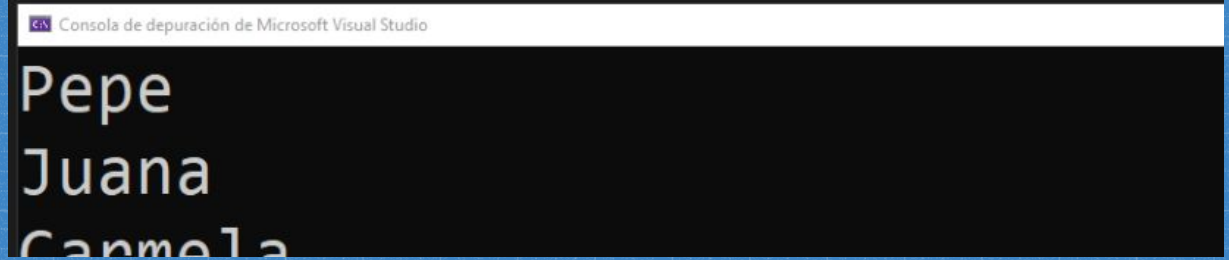
Expresiones lambda en listas

```
1 referencia
public class Persona
{
    int id;
    string nombre;
    0 referencias
    public Persona(int id, string nombre)...

    0 referencias
    public int Id { get { return id; } set { id = value; } }
    0 referencias
    public string Nombre { get { return nombre; } set { nombre = value; } }
}
```

```
List<Persona> personitas = new List<Persona>()
{
    new Persona(1,"Pepe"),
    new Persona(34,"Juana"),
    new Persona(32,"Carmela"),
    new Persona(5,"Jazmin"),
    new Persona(17,"Carola"),
    new Persona(20,"Romeo")
};
```

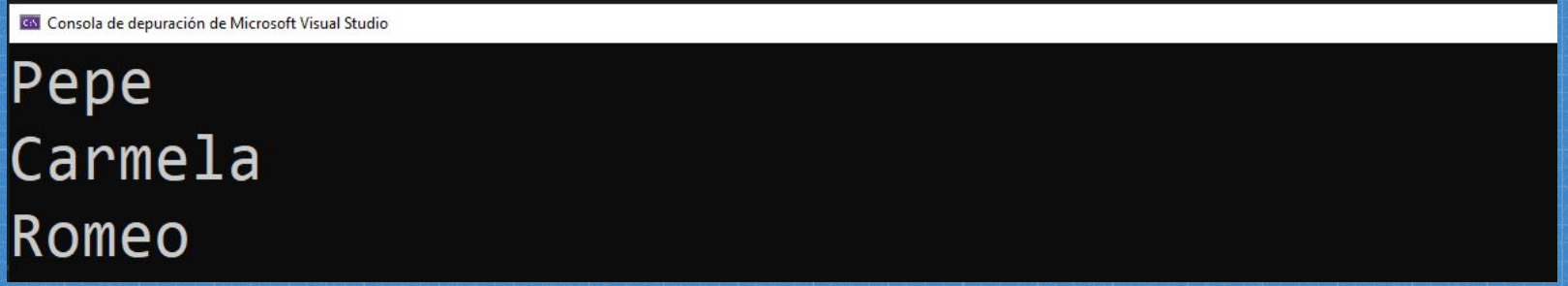
ForEach

```
List<Persona> personitas = new List<Persona>()
{
    new Persona(2,"Pepe"),
    new Persona(34,"Juana"),
    new Persona(8,"Carmela"),
    new Persona(73,"Jazmin"),
    new Persona(22,"Carola"),
    new Persona(13,"Romeo")
};

personitas.ForEach((p) => Console.WriteLine(p.Nombre));
```



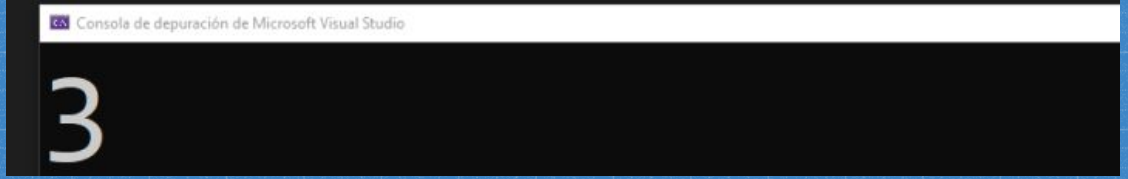
FindAll

```
List<Persona> listaAux = personitas.FindAll((x) => x.Nombre.Contains("e"));

listaAux.ForEach((p) => Console.WriteLine(p.Nombre));
```



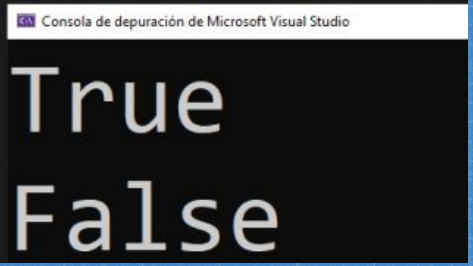
FindIndex

```
List<Persona> personitas = new List<Persona>()
{
    new Persona(2,"Pepe"),     // 0
    new Persona(34,"Juana"),   // 1
    new Persona(8,"Carmela"),  // 2
    new Persona(73,"Jazmin"),  // 3
    new Persona(22,"Carola"),  // 4
    new Persona(13,"Romeo")    // 5
};

int posicion = personitas.FindIndex(
                (x) => x.Nombre.ToLower() == "jazmin");

Console.WriteLine(posicion);
```



Exists

```
List<Persona> personitas = new List<Persona>()
{
    new Persona(1,"Pepe"),
    new Persona(34,"Juana"),
    new Persona(8,"Carmela"),
    new Persona(73,"Jazmin"),
    new Persona(4,"Carola"),
    new Persona(13,"Romeo")
};

bool resultado1 = personitas.Exists((x) => x.ID == 4);

bool resultado2 = personitas.Exists((x) => x.ID == 99);

Console.WriteLine(resultado1);
Console.WriteLine(resultado2);
```



TrueForAll

```
List<Persona> personitas = new List<Persona>()
{
    new Persona(1,"Pepe"),
    new Persona(34,"Juana"),
    new Persona(8,"Carmela"),
    new Persona(73,"Jazmin"),
    new Persona(4,"Carola"),
    new Persona(13,"Romeo")
};

bool resultado1 = personitas.TrueForAll((x) => x.ID < 75);

bool resultado2 = personitas.TrueForAll((x) => x.ID < 40);

Console.WriteLine(resultado1);
Console.WriteLine(resultado2);
```