Delegados

Un delegate es un tipo de referencia que puede utilizarse para encapsular un método con nombre o anónimo.

Imaginemos que podemos crear un método, almacenarlo en un objeto y pasarlo como parámetro de una función, pues en eso consiste.

```
delegate double Operacion(double x);
```

`Operacion` es compatible con cualquier método que retorne un double y tenga solo un parámetro de tipo double, como este:

```
static double Cuadrado(double x) {
    return x * x;
}
```

Al asignar un método a una variable de tipo `Operación` se crea una instancia del delegado:

```
Operación op = Cuadrado;
```

finalmente podemos usar esa instancia para invocar al método al que hace referencia:

```
double resultado = op(4); // 16
```

Como recibir por parámetro un Delegate

Para poder recibir un método encapsulado se usara la `Action<T1, T2>`

El motivo por el que se usan los delegados y los encapsulamientos de métodos es porque puede que necesitemos recibir un conjunto de instrucciones de código para ejecutarlas quizá de manera especial, por ejemplo:

La instrucción `Parallel.For(int, int, Action<T1>)`, Esta instrucción se encarga de generar un hilo por cada iteración del bucle, pero claro, hay que ejecutar una serie de contenido y no se sabe cual es, pues lo que hace es recibir un método encapsulado con un conjunto de instrucciones a ejecutar por cada iteración del bucle.

```
// Llamamos a pruebaDelegado mandándolo un método anónimo encapsulado
pruebaDelegado("HOLAAAAAAAAAAAAAA", (numero, mensaje) => {
    Console.WriteLine(mensaje);
});
```

```
public void pruebaDelegado (string mensaje, Action<int, string> metodoEncapsulado)
{
    // Usamos el invoke para llamar a las instrucciones encapsuladas
    metodoEncapsulado.Invoke(0, mensaje);
}
```

En la Lambda declaramos las variables para usarlas en las instrucciones que iran encapsuladas y lo recibimos como un objeto en el método, este realiza las instrucciones que tenga que realizar y cuando llegue el momento del Invoke, como no sabe que parámetro mandar al llamar al Invoke, el método recibirá un parámetro aparte del Action, que sera un string con el mensaje y después mandara ese mensaje como parámetro en el invoke que llegará a la variable de método "mensaje" declarada en la lambda