# Generics & delegate kata

## Indice generale

# 1 Ricerca in lista generica

Aggiungere un metodo di ricerca alla lista generica che restituisca la posizione dell'elemento cercato, oppure -1 se non esiste. (vedi tutorial **Generics**)

Poiché il tipo degli elementi è generico, il metodo deve definire un parametro *delegate* che restituisca `true` se un elemento soddisfa la condizione di ricerca. Il metodo applicherà il delegate ad ogni elemento, terminando con il primo che soddisfa la condizione.

Ad esempio, dato il segue tipo:

```
class Pet
{
    public string Nome;
    public int Età;
}
```

Un metodo simile consentirebbe di eseguire il seguente codice:

```
Lista<Pet> lista = new Lista<Pet>();

lista.Add(new Pet { Nome = "Fido", Età = 4 });
lista.Add(new Pet { Nome = "Pulce", Età = 2 });
lista.Add(new Pet { Nome = "Thor", Età = 3 });

int indice = lista.FindIndex(v => v.Nome == "Fido");
//->0
```

# 2 Ordinamento in lista generica

Aggiungere un metodo di ordinamento alla lista generica. (vedi tutorial **Generics**)

Poiché il tipo degli elementi è generico, il metodo deve definire un parametro *delegate* che stabilisca il criterio di ordinamento tra due elementi. Tale criterio, compatibilmente al metodo `CompareTo()` definito da tutti i tipi "confrontabili", deve produrre un valore intero <0, 0, >0 in base al fatto che il primo elemento sia minore, uguale o maggiore del secondo.

Un metodo simile consentirebbe di eseguire il seguente codice:

```
Lista<Pet> lista = new Lista<Pet>();

lista.Add(new Pet { Nome = "Fido", Età = 4 });
lista.Add(new Pet { Nome = "Pulce", Età = 2 });
lista.Add(new Pet { Nome = "Thor", Età = 3 });

lista.Sort((e1, e2) => e1.Età.CompareTo(e2.Età));
//->lista ordinata in modo crescente per età
```

# 3 Concatenazione di valori in una stringa

Realizza un metodo generico che, dato un vettore generico e un carattere di concatenazione, restituisca una stringa contenente i valori del vettore separati dal carattere di concatenazione.

(Un metodo simile esiste già ed è il metodo statico `Join()` del tipo `string`)

```
int[] dati = {10, 34, 12};
char ch = '|';
// -> "10|34|12"
```

### *Suggerimenti*

Inizia con un metodo non generico che accetti un vettore di interi.

### *Variazioni*

Implementa una versione del metodo che invece di un carattere di concatenazione accetta una stringa.

```
int[] dati = {10, 34, 12};
string s = " | ";
// -> "10 | 34 | 12"
```

# 4 Concatenazione “programmabile”

Implementa una nuova versione del metodo di concatenazione, in modo che sia utilizzabile anche con i tipi di dati che non forniscono una rappresentazione stringa (non implementano `ToString()`).

Per ottenere questo, il metodo deve definire un parametro delegate che consenta al codice chiamante di specificare il metodo di formattazione in stringa degli elementi del vettore. (Cioè: dato un elemento, ottenere la stringa corrispondente.)

Un metodo simile consentirebbe di scrivere il seguente codice:

```
Pet[] cuccioli = new Pet[]
{
    new Pet { Nome = "Fido", Età = 4},
    new Pet { Nome = "Pulce", Età = 2},
    new Pet { Nome = "Thor", Età = 3},
};

string s = Concatena(cuccioli, " | ", c => $"{c.Nome}:{c.Età} anni");
//->"Fido: 4 anni | Pulce: 2 anni | Thor: 3 anni"
```