REF & OUT

CORSO DI GAME PROGRAMMING
1° ANNO

Docente **Davide Caio**

ACCADEMIA ITALIANA VIDEOGIOCHI


Tipi di Valore
Tipi di Riferimento
Ref
Out

TIPI DI VALORE

In C# i tipi possono essere categorizzati in due categorie: i tipi valore e i tipi riferimento.

Una variabile di un **tipo di valore** contiene un'istanza del tipo.

Cosa significa? Significa che all'interno della cella di memoria rappresentata da quella variabile troviamo il **valore effettivo** del tipo.

Tutti i tipi predefiniti di C# (quindi quelli visti fino ad ora) sono tipi di valore.

TIPI DI RIFERIMENTO

Una variabile di un **tipo di riferimento** contiene un **riferimento** all'istanza del tipo.

Cosa significa? Significa che all'interno della cella di memoria rappresentata da quella variabile troviamo un riferimento (un indirizzo di memoria) dove è contenuto il valore effettivo a tutti gli effetti.

Come si definisce un tipo di tipo riferimento? Lo vedremo più avanti, ma ogni tipo definito tramite una *classe* sarà di tipo di riferimento.

TIPI DI VALORE - ESEMPIO

Cerchiamo di capire meglio cosa significa.
Ricapitoliamo: tipo di valore significa che nella variabile è contenuto a tutti gli effetti il valore di quel tipo.

Esempio:
int i = 4;

All'interno di i (e cioè della cella di memoria a cui noi possiamo accedere attraverso il nome i) è contenuto a tutti gli effetti il valore 4.

TIPI DI VALORE - ASSEGNAMENTO

Ci sono una serie di effetti collaterali che bisogna ricordare quando si tratta con variabili di tipo di valore. Partiamo dall **assegnamento**.

```
int i = 4;
int j;
```

Cosa succede nel momento in cui noi assegniamo a j la variabile i? j = i;

Succede che adesso anche la variabile j contiene il valore 4.

Se adesso a i assegno il valore 5, qual'è il valore di j? i = 5;
Console.WriteLine (j); ??????

Il valore di j è rimasto invariato, appunto perché è un tipo di valore. Nella cella di memoria indicata con l'etichetta j c'è ancora il valore 4, mentre nella cella di memoria indicata con l'etichetta i contiene il valore 5.

TIPI DI VALORE - PASSAGGIO DI PARAMETRI

Se passiamo un tipo di valore come **parametro** di un metodo cosa succede?

```
static void Main () {
        int i = 5;
        MyMethod (i);
        Console.WriteLine (i);
}

static void MyMethod (int parametro) {
        parametro = 10;
}
```

Il valore di i rimane invariato. Il WriteLine stamperà a video ancora il numero 5. Perché? Perché in questo modo il parametro viene passato per valore, e quindi essendo di tipo di valore, è come se venisse assegnato a parametro i, quindi parametro = i. Essendo int un tipo di riferimento, *ogni cambiamento che verrà fatto alla variabile parametro non avrà nessun effetto collaterale sulla variabile i.*

PROBLEMA

Ma se io volessi creare un metodo che mi consenta di cambiare il valore di alcuni tipi di valore che io passo come parametri?

Esempio: se volessi creare un metodo che mi ritorna un interno chiesto all'utente come input e il numero di volte che il metodo ha dovuto leggere un input dell'utente prima che questo input potesse essere parsato come un interno?

REF

La keyword **Ref** serve per passare una variabile di tipo di valore come **riferimento** in un metodo.

Questo vuol dire che all'interno del metodo, ogni cambiamento fatto al valore di questo parametro all'interno del metodo *comporterà un cambiamento al valore della variabile passata* come riferimento al metodo stesso.

```
static void Main (string[] args) {
        int i = 5;
        RefParameter (ref i);
        Console.WriteLine (i);
}

static void RefParameter (ref int parameter) {
        parameter = 12;
}
```

OUT

La keyword **Out** serve per passare una variabile di tipo di valore come riferimento in un metodo.

Questo vuol dire che all'interno del metodo, *ogni cambiamento fatto al valore di questo parametro all'interno del metodo comporterà un cambiamento al valore della variabile passata* come riferimento al metodo stesso.

```
static void Main (string[] args) {
        int i = 5;
        OutParameter (out i);
        Console.WriteLine (i);
}

static void OutParameter (out int parameter) {
        parameter = 12;
}
```

REF VS OUT

Se avete notato, la definizione della parola chiave ref e out che ho dato è la stessa.

Ma quindi cosa cambia?

Cambia che utilizzando la parola **out** stiamo dicendo che di quella variabile *non ci interessa la inizializzazione* e sarà solamente assegnata. Questo significa che se un metodo di un parametro è definito come out non potrà essere utilizzato all'interno del parametro stesso. **Ref** invece è *una variabile che sicuramente ha un valore prima che venga passata come parametro* di un metodo e quindi potrà essere utilizzata all'interno del metodo stesso.

REF VS OUT

```
static void RefParameter (ref int parameter) {
        Console.WriteLine (parameter);
        parameter = 12;
}
static void OutParameter (out int parameter) {
        ~~Console.WriteLine (parameter);~~  → errore di compilazione
        parameter = 12;
}
```

Per riassumere quindi, la keyword ref e out ci possono aiutare a capire, guardando la firma del metodo, quel parametro per cosa sarà utilizzato:

- solamente come contenitore per un risultato e non ci interessa una inizializzazione→ out
- potrebbe essere utilizzato anche all'interno del metodo prima di essere sovrascritto e quindi è importante che sia correttamente inizializzato → ref
- Inoltre se un parametro è passato come ref allora può anche non essere assegnato nel metodo. Un parametro inserito come out deve per forza essere assegnato prima della fine del metodo.

ESERCIZIO 1

Scrivere un metodo che ritorni un valore intero inserito dall'utente e il numero di input necessari all'utente prima che l'inserimento possa essere effettivamente parsato come int.

ESERCIZIO 2

Creare un metodo che incrementi una variabile di tipo intero.

ESERCIZIO 3

Creare un metodo che calcoli le equazioni di primo grado utilizzando out (e quindi con tipo di ritorno void).

Ricordo che una equazione di primo grado è del tipo ax + b = 0. E che quindi x = -b/a.

ESERCIZIO 4

Creare un metodo che calcoli la somma, la differenza e la media di due numeri reali (non solo interi) inseriti come parametro. I valori della somma della differenza e della media devono poter essere tutti e 3 ritornati al chiamante.

Creare il metodo con l'utilizzo di ref e poi con l'utilizzo di out, cercando di mettere in evidenza la differenza di ref e out.