C# basics

* Qual è la differenza fra un metodo, una proprietà e un campo come definizione nel codice?

Il metodo è un blocco di codice che contiene una serie di istruzioni, una proprietà è un membro che fornisce un meccanismo flessibile per leggere, scrivere o calcolare il valore di un campo privato. Sono metodi speciali chiamati funzioni di accesso che consentono di accedere facilmente ai dati e di alzare il livello di sicurezza e flessibilità dei metodi. Un campo è una variabile di qualsiasi tipo che viene dichiarata direttamente in una classe o una struct. Una classe o struct può includere campi d’istanza o statici o entrambi.

* Cosa sono i namespace e a cosa servono? Come faccio ad usare classi in altri namespace?

Namespace indica un ambito che contiene un set di oggetti correlati. È possibile usare uno spazio dei nomi per organizzare gli elementi di codice e creare tipi univoci globali.

* Quali sono i modificatori di accesso e cosa significano?

I modificatori sono parole chiave usate per specificare l’accesso a un membro o a un tipo. Sono quattro :

public, protect, internal e private.

In public l'accesso è limitato alla classe o ai tipi derivati dalla classe che li contiene.

In internal  l'accesso è limitato all'assembly corrente.(In fase di esecuzione compila).

In private l’accesso è limitato al tipo di contenitore.

* Cosa significa static

Usare il modificatore static per dichiarare un membro statico, che appartiene allo stesso tipo invece che a un oggetto specifico. Il modificatore static può essere usato con classi, campi, metodi, proprietà, operatori, eventi e costruttori. Una classe [statica](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/static) corrisponde fondamentalmente a una classe non statica, ma c'è una differenza: di una classe statica non è possibile creare un'istanza. Non è possibile usare l’operatore new per creare una variabile di tipo classe.

* Come accedo a un metodo/proprietà/campo statico?

Senza istanziare l’oggetto della classe corrispondente.

* Cosa significa const?

E’ una variabile che deve essere istanziata con un valore che non può più essere modificato.

* Cosa significa readonly? Quando viene utilizzato in genere?

In una [dichiarazione di campo](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/readonly#readonly-field-example) readonly indica che l'assegnazione al campo può avvenire solo come parte della dichiarazione del campo o in un costruttore della stessa classe. Un campo readonly può essere assegnato e riassegnato più volte nella dichiarazione del campo e nel costruttore ( diversamente da const).

* In una definizione [readonly struct](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/readonly#readonly-struct-example), readonly indica che non è possibile modificare struct.
* In una [definizione di membroreadonly](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/readonly#readonly-member-examples)readonly indica che un membro di un struct non modifica lo stato interno dello struct.
* In un [ref readonly metodo restituito](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/readonly#ref-readonly-return-example), il modificatore di readonly indica che il metodo restituisce un riferimento e le Scritture non sono consentite a tale riferimento.
* Che cos'è l'interfaccia IEnumerable?

Espone un enumeratore, che supporta un'iterazione semplice su una raccolta non generica.

* Qual è la differenza fra while, do… while, for e foreach?

While si basa su un controllo anticipato, mentre do while il controllo è posticipato al ciclo, entrambi svolgono una iterazione condizionata al verificarsi di un determinato evento. L'istruzione for esegue un'istruzione o un blocco di istruzioni mentre un'espressione booleana specificata restituisce true. L'istruzione foreach esegue un'istruzione o un blocco di istruzioni per ogni elemento in un'istanza del tipo che implementa l’interfaccia [System.Collections.IEnumerable](https://docs.microsoft.com/it-it/dotnet/api/system.collections.ienumerable) o [System.Collections.Generic.IEnumerable<T>](https://docs.microsoft.com/it-it/dotnet/api/system.collections.generic.ienumerable-1). L'istruzione foreach non è limitata a questi tipi e può essere applicata a un'istanza di qualsiasi tipo che soddisfa le condizioni seguenti:

* include il metodo GetEnumerator pubblico senza parametri con tipo restituito classe, struct o interfaccia,
* il tipo restituito del metodo GetEnumerator include la proprietà Current pubblica e il metodo MoveNext pubblico senza parametri con tipo restituito [Boolean](https://docs.microsoft.com/it-it/dotnet/api/system.boolean).

In un punto qualsiasi all'interno del blocco dell'istruzione foreach o for è possibile uscire dal ciclo usando l'istruzione [break](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/break) o passare all'iterazione successiva nel ciclo con l'istruzione [continue](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/continue).

* Cos’è il costrutto <condition> ? <value1> : <value2>?

int x = 1;

int y = 0;

var z = x == y ? false : true;

* Cos’è il costrutto <object> ? ?<value> ?

Book book = new Book();

\_ = book ?? book;

* Cos’è il costrutto <object> ? ?.<mathodOrProperty> ?

book?.Name();

* Nel costrutto if (<cond1> AND <cond2>) cond2 viene sempre eseguita? Se no, quando non viene eseguita?

if (true && true) Console.WriteLine("yes");

* Nel costrutto if (<cond1> OR <cond2>) cond2 viene sempre eseguita? Se no, quando non viene eseguita?

if (true || true) Console.WriteLine("yes");

* Qual è la differenza fra un campo definito const e uno definito static readonly?

Const è un campo che non si può modificare con il costruttore, i get o i set.

* Che cos’è un array?

Fornisce metodi per creare, manipolare, ordinare ed eseguire ricerche all'interno di matrici, diventando così la classe base per tutte le matrici in Common Language Runtime.

* A cosa servono ref e out?

La parola chiave ref indica un valore che viene passato per riferimento. Viene usata in quattro contesti diversi:

* Nella firma di un metodo e in una chiamata al metodo, per passare un argomento a un metodo per riferimento. Per altre informazioni, vedere [Passaggio di un argomento per riferimento](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/ref#passing-an-argument-by-reference).
* Nella firma di un metodo, per restituire un valore al chiamante per riferimento. Per altre informazioni, vedere [Valori di riferimento restituiti](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/ref#reference-return-values).
* Nel corpo di un membro, per indicare che un valore restituito di riferimento è archiviato in locale come un riferimento che il chiamante intende modificare o, in generale, che una variabile locale accede a un altro valore per riferimento. Per altre informazioni, vedere [Variabili locali ref](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/ref#ref-locals).
* In una dichiarazione struct per dichiarare ref struct o readonly ref struct. Per altre informazioni, vedere [Tipi ref struct](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/ref#ref-struct-types).

La parola chiave out fa sì che gli argomenti vengono passati per riferimento. Imposta il parametro formale come alias dell'argomento, che deve essere una variabile. In altre parole, qualsiasi operazione sul parametro viene eseguita sull'argomento. È come la parola chiave [ref](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/ref), con la differenza che ref richiede l'inizializzazione della variabile prima di essere passato. È anche come la parola chiave [in](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/in-parameter-modifier), con la differenza che in non consente al metodo chiamato di modificare il valore dell'argomento. Per usare un parametro out, la definizione del metodo e il metodo chiamante devono usare in modo esplicito la parola chiave out.

* Qual è la differenza tra passare una variabile per valore e per riferimento? Qual è un esempio di parametro che viene passato per valore e uno per riferimento?

Esistono due generi di tipo in C#: tipi di riferimento e tipi di valore. Le variabili dei tipi di riferimento archiviano i riferimenti ai relativi dati (oggetti), mentre le variabili dei tipi di valore contengono direttamente i dati. Con i tipi di riferimento, due variabili possono fare riferimento allo stesso oggetto. Di conseguenza le operazioni su una variabile possono influire sull'oggetto a cui fa riferimento l'altra variabile. Con i tipi valore, ogni variabile ha una propria copia dei dati e non è possibile che le operazioni su una variabile influiscano sull'altra (tranne nel caso delle variabili dei parametri in, ref e out, vedere Modificatore del parametro [in](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/in-parameter-modifier), [ref](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/ref) e [out](https://docs.microsoft.com/it-it/dotnet/csharp/language-reference/keywords/out-parameter-modifier)).

Quando viene usata nell'elenco di parametri di un metodo, la parola chiave ref indica che un argomento viene passato per riferimento, non per valore. La parola chiave ref imposta il parametro formale come alias dell'argomento, che deve essere una variabile. In altre parole, qualsiasi operazione sul parametro viene eseguita sull'argomento. Ad esempio, se il chiamante passa un'espressione variabile locale o un'espressione di accesso dell'elemento della matrice e il metodo chiamato sostituisce l'oggetto a cui fa riferimento il parametro ref, l'elemento della matrice o la variabile locale del chiamante fa riferimento al nuovo oggetto quando viene restituito il risultato del metodo.