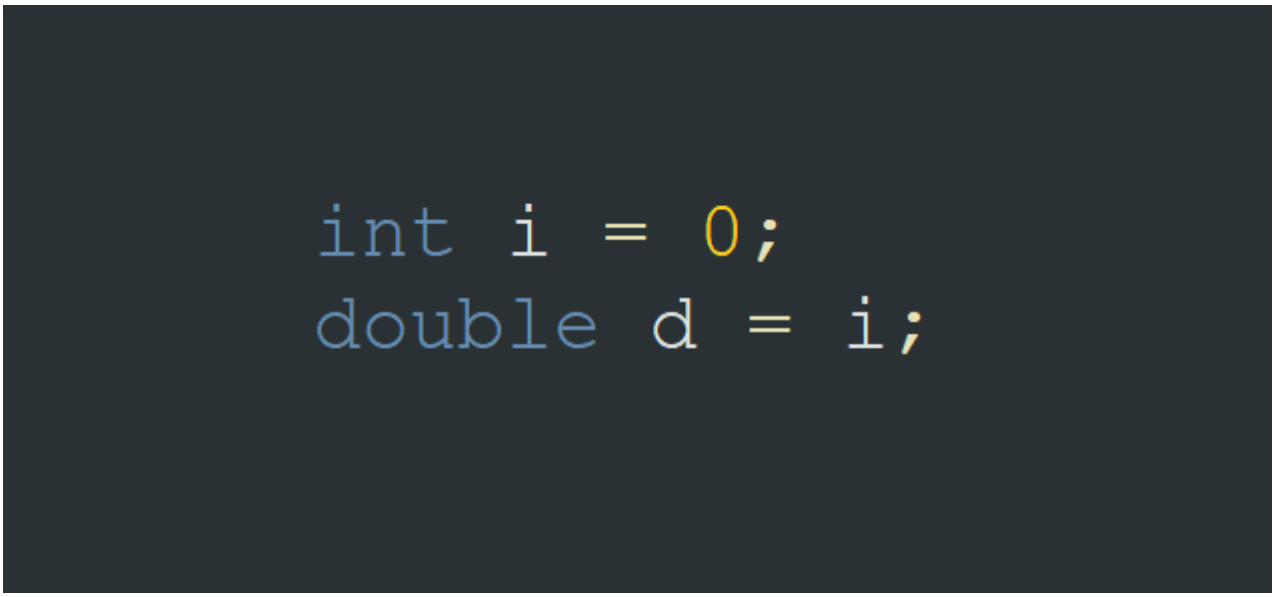
L’operatore implicit



Assumendo una classe chiamata MyClass, chiunque abbia usato almeno una volta nella propria vita il C#, senza ombra ha familiarità con la sintassi var myObj = new MyClass().
Niente di nuovo, giusto? Con un piccolo sforzo mentale, verrebbe spontaneo pensare che tutti i tipi complessi vengano inizializzati tramite l’operatore new; ma è davvero così?

Il tipo string

Il titolo del paragrafo già risponde alla domanda posta poc’anzi, ma nel caso non fosse ancora chiaro, voglio ricordarvi un aspetto estremamente importante delle stringhe, talmente ovvio che non ci si bada nemmeno: nessuno sano di mente ha mai costruito una stringa facendo new string(char[]).
Difatti, quello che si fa in genere, è inizializzare una *stringa* nel modo seguente:

```
var myString = "Italian Coders";
```

Non ci sono operatori new, giusto? Eppure, stiamo **inizializzando** un tipo complesso.
Come è mai possibile? Si tratta di una peculiarità delle stringhe? Oppure il C# mette a disposizione uno strumento per rendere qualsiasi nostra classe inizializzabile senza

l’utilizzo dell’operatore new?
Ovviamente la risposta corretta è la seconda, altrimenti questo articolo non avrebbe senso di esistere dal principio (anche se sarebbe stata un’idea davvero simpatica, se fosse stato il primo aprile!).

L’operatore implicit

Ogni qual volta è nostra intenzione fare una *conversione implicita* tra due tipi, è l’operatore implicit che vogliamo usare (incredibile, non è vero?)!

A titolo assolutamente d’esempio immaginiamo una classe per rappresentare una temperatura, come la seguente:

```
public class Temperature
{ public double Value { get; } public char MeasureUnit { get; }; public Temperature(double value, char measureUnit) { Value = value; MeasureUnit = measureUnit; }
}
```

Niente di speciale, vero? Ma ora immaginiamo che questa classe venga consumata principalmente in ambiti accademici dove esiste un’unica religione: il sistema internazionale di misura. Tutti sanno che in SI (ovviamente, no?), la temperatura si misura solo e soltanto in Kelvin, quindi perché non immaginare che la nostra classe per la temperatura abbia una conversione implicita da double a Temperature? Ha perfettamente senso, no?

E allora eccoci accontentati:

```
public class Temperature
{ [...] public static implicit operator Temperature(double lhs) => new Temperature(lhs, 'K'); public static implicit operator double(Temperature lhs) => lhs.Value;
}
```

Fico, vero?

Ora mostriamo come funziona la cosa:

```
var t0 = new Temperature(10d, 'K');
Temperature t1 = 10d; double v1 = t0.Value;
double v2 = t0;
```

Riuscite a vedere la differenza?

Conclusioni

Abbiamo davvero bisogno dell'operatore implicit? La risposta è no. Possiamo vivere senza e magari pensare di fare un costruttore con parametri di default. Il risultato sarebbe perfettamente identico a quello da me illustrato.
Tuttavia, chi ha detto che la **leggibilità** non conti? Se non contasse nulla non staremmo usando C# ma assembly, no?
Il potere dei linguaggi ad alto livello sta proprio in questi simpatici **synctactic sugar** che rendono tutto più bello e vicino al modo di pensare degli umani (che poi gli informatici pensino davvero come gli umani, è un discorso a parte).

Alla fine, l'operatore implicit è l'operatore che meritiamo, e quello di cui abbiamo bisogno!

https://italiancoders.it/loperatore-implicit/