ESPRESSIONI, OPERATORI E FUNZIONI – Linguaggio C#

Gli Operatori

Le *Espressioni Matematiche* in C# devono essere scritte secondo regole *leggermente diverse* da quelle note dallo studio della matematica.

Le *Operazioni Aritmetiche* devono essere sempre indicate con gli appositi **Operatori Aritmetici** e non possono essere in nessun caso "sottintese".

☞ Ad esempio, se in matematica sei abituato a scrivere l'espressione "2a+3b", in C# dovrai "esplicitare" la *moltiplicazione* utilizzando *l'operatore asterisco* e dovrai quindi scrivere "**2 * a + 3 * b**"

Gli *Operatori Aritmetici* principali sono i seguenti:

+ Addizione	**- Sottrazione**	*** Moltiplicazione**	**/ Divisione**	**% Resto di una Divisione**

Gli Operatori su indicati sono tutti **Operatori Binari**, ossia necessitano di *Due Operandi*, uno a sinistra e uno a destra secondo la seguente sintassi: *<espressione> <operatore> <espressione>*

☞ Nell'espressione "**A * B**", l'*asterisco* (*Operatore*) ha due *operandi* (*Espressioni*), uno a sinistra (A) e uno a destra (B).

✋ In C# non puoi usare la "*linea di frazione*", bensì devi usare l'**operatore / (slash)** per la Divisione. Ad esempio: "**A / B**".

✋ L'operatore "-" può essere usato anche con un solo operando a destra (es. – A): in questo caso ha l'effetto di "**cambiare il segno**" e *viene calcolato prima di tutti gli altri operatori*.

L'**Operatore % (Modulo)** consente di ottenere il **Resto di una Divisione**.

Quando l'**Operatore "/" (Divisione) è applicato a due operandi INTERI**, restituisce la Divisione Intera, ossia la sola "*parte intera*" del risultato della divisione stessa.

☞ Ad esempio, l'espressione "**10 % 3**" dà come risultato il **valore 1** che è il *Resto della Divisione* 10 / 3.
L'espressione "**10 / 3**" dà come risultato il **valore 3** che è la *Divisione Intera* fra 10 e 3 (la sola *parte intera* di 3,33333…).

Gli Operatori presenti in una espressione vengono elaborati secondo un *preciso ordine* chiamato **Priorità degli Operatori**, simile a quello noto dalla matematica:

1° - operatori *** / %** 2° - operatori **+ -**

☞ Ad esempio, nell'espressione "**3 - 5 * 4 / 10 + 8 * 6**", C# anzitutto calcola *moltiplicazioni e divisioni* nell'ordine in cui si presentano, ottenendo "3 - **20** / 10 + 8 * 6", quindi "3 - **2** + 8 * 6" e poi "3 - 2 + **48**"; Infine calcola, nell'ordine, *addizioni e sottrazioni*: prima ottiene "**1** + 48" e infine "**49**".

Le Parentesi

L'uso delle **Parentesi (Tonde)** consente di **Forzare l'Ordine** in cui gli Operatori vengono valutati.

☞ Nell'espressione "**3 - 5 * 4 / 10 + 8 * 6**" è possibile *forzare* l'ordine di calcolo degli operatori e far si che C# calcoli anzitutto la somma 10+8 semplicemente usando delle parentesi tonde: "3 - 5 * 4 / **(10 + 8)** * 6"

E' ovviamente possibile utilizzare anche più **Livelli di Parentesi** (*parentesi* dentro altre *parentesi*), ma sempre utilizzando *parentesi Tonde*: non si devono quindi usare *parentesi Quadre o Graffe* che in C# sono destinate ad altri usi.

Le Funzioni e la Libreria Math

Nelle Espressioni è possibile anche utilizzare le **Funzioni della libreria Math**, che consentono calcoli speciali come la *radice quadrata*, l'*arrotondamento*, e altri calcoli matematici. Esse devono essere scritte secondo la seguente sintassi:

Regola per scrivere una Funzione della libreria Math: **Math .** *<funzione>* **(** *<espressione>* **)**

In particolare, per calcolare la **Radice Quadrata** in una espressione, è necessario utilizzare la **Funzione Sqrt** *(Square Root)* e, secondo la regola vista sopra, scrivere: **Math.Sqrt (**<*espressione*>**)**

☞ Ad esempio, per calcolare la *Radice Quadrata* di (A+5), in C# dovrai scrivere l'espressione: *Math.Sqrt(A+5)*.

☞ Un tipico "esempio riassuntivo" è il calcolo delle soluzioni di una equazione di 2° grado del tipo $\mathbf{ax^2 + bx + c = 0}$.
Dovrai scrivere due *Istruzioni di Assegnazione* con le *Espressioni* indicate di seguito:

```
x1 = ( -  b - Math.Sqrt (b*b - 4*a*c) ) / (2 * a)
x2 = ( -  b + Math.Sqrt (b*b - 4*a*c) ) / (2 * a)
```

Altre Funzioni utili sono:

Math.Truncate (<*espressione*>**)** restituisce la **Parte Intera** del risultato dell'espressione indicata fra parentesi
Math.Round (<*espressione*>**)** restituisce il risultato dell'espressione ma **Arrotondato all'intero più vicino.**

☞ Esempi: *Math.Truncate(3.85) restituisce **3*** *Math.Round(3.55) restituisce **4***

La *libreria Math* offre molte altre Funzioni Matematiche (logaritmi, funzioni trigonometriche, ecc.) e consente anche di ottenere il valore di alcune importanti *costanti matematiche.*

In particolare scrivendo **Math.PI** si ottiene il **valore di Pi Greco**.

☞ Ad esempio, l'Istruzione di Assegnazione per calcolare l'Area di un cerchio di Raggio *r* è: **Area = Math.PI * r ^ 2**

Le funzioni della libreria Math, nella maggior parte dei casi, come argomento (fra parentesi), **accettano espressioni di tipo Double e restituiscono valori di tipo Double** (anche *Truncate* e *Round*, nonostante sembra restituiscano interi).

Le Costanti

Si chiamano **Costanti** i **Valori che Restano Invariati** sia durante una stessa l'esecuzione del programma, sia da una esecuzione all'altra.

☞ Un esempio di *Costante* è il voto minimo che determina la promozione di uno studente: questo valore è sempre 6.

E' possibile **Dichiarare una Costante** ossia *specificarne il Valore*, *attribuirvi un Nome* e, eventualmente, anche *indicarne il Tipo*.

✋ Il vantaggio di *Dichiarare una Costante* è che, nel programma, si potrà *indicare in Nome anziché il Valore*, migliorando la leggibilità del programma. Inoltre sarà più facile una eventuale modifica del valore della costante perché sarà sufficiente modificare la sola dichiarazione e non tutti i punti del programma ove la costante è usata.

Regola per Dichiarare una Costante: **const <***tipo***>** *<nome>* **=** *<espressione>*

☞ Se vuoi "dichiarare" una costante per il voto minimo, dovrai scrivere come segue:

```
const int VotoMinimo = 6
```

Dopo questa dichiarazione, se nel programma utilizzi l'identificatore *VotoMinimo* esso assumerà il valore 6.

Dichiarazione e Assegnazione insieme

In C#, è possibile attribuire una valore ad una Variabile, *già al momento della sua dichiarazione*:

Regola per Dichiarare e Inizializzare una Variabile: *<tipo> <nome>* **=** *<espressione>*

☞ Se vuoi "dichiarare" una variabile intera *Anno* e attribuirvi inizialmente il valore *2000*, dovrai scrivere come segue:

```
int Anno = 2000
```

In effetti si tratta di una *Dichiarazione* e di una *Assegnazione*, effettuate in un'unica istruzione.

Quando si attribuisce ad una variabile un valore iniziale si effettua l'**Inizializzazione** della variabile.