

## ANSI C - Introduzione

Ok finalmente mi accingo a scrivere il primo di una serie di tutorial che senza dubbio saranno molto impegnativi, l'obiettivo finale non sarà tanto quello di riuscire a formare degli esperti programmatori, quanto di riuscire a fornire degli spunti per maggiori approfondimenti futuri.

Nella stesura di queste miniguide cercherò di seguire uno schema logico abbastanza rigido, partendo da una introduzione generale, dove verranno toccati alcuni concetti teorici, e arrivando alla parte relativa al codice, cercando di focalizzare l'attenzione sia sulla sintassi che sulla semantica.

Probabilmente all'interno di questi tutorial ci saranno vagonate di errori, quindi ogni commento, domanda, richiesta o critica è ben accetta. Ed ora si comincia!!!

Vi risparmio la storia del C dagli albori fino ad ora, vi basti sapere che il linguaggio C vede la sua genesi nei primi anni 70, ad opera di **Dennis Ritchie**. Il C nasce come evoluzione di un altro linguaggio che fondamentalmente era una interfaccia ad alto livello dell'assembler (non credo avrò mai il coraggio di fare tutorial in assembler, quindi spero non mi chiederete di farli :D ).

Grazie al linguaggio C appena nato fu possibile riscrivere il kernel del sistema operativo UNIX in brevissimo tempo ottenendo risultati inaspettati. Ovviamente il C si è evoluto nel tempo e grazie allo standard ANSI(American Standards Institute) è tutt'oggi il linguaggio più utilizzato e diffuso.

Ok finita la storia, vi starete chiedendo, perchè l'ANSI C, che cosa ha di particolare, perchè nelle scuole medie superiori dedicate alla formazione di periti informatici e nelle stesse università il linguaggio C viene snobbato e si studiano altri linguaggi come il JAVA. Immagino che molti di quelli che sfoglieranno questi tutorial avranno una buona esperienza in JAVA, ma veramente una scarsa conoscenza del C.

Ebbene quali sono i vantaggi del C? Innanzitutto non è portabile come del codice in JAVA.... si, in effetti non è un vantaggio, ma la cosa ci può condizionare relativamente, in quanto come già detto il linguaggio ANSI C è il più utilizzato al mondo ed esistono compilatori per ogni genere di sistema, quindi non sarà portabile come JAVA, ma con qualche accortezza è compilabile e compilabile quasi ovunque. Allora quali sono i veri vantaggi?

- **Efficienza** : ebbene si questo ritengo sia il punto più importante, quello che rende il linguaggio C unico, secondo solo a linguaggi che vanno ad interagire direttamente con i registri della macchina (ho detto che non lo faccio il tutorial in assembler!!!). Che cosa rende il linguaggio C così efficiente? Il linguaggio ci permette di andare a gestire a basso livello la memoria, dandoci un controllo totale sull'utilizzo anche in termini di spazio. (quello della memoria è un concetto fondamentale che richiede una capitolo a parte, quindi per ora evito di parlarne approfonditamente, mi va di sottolineare il fatto che a differenza di altri linguaggi come JAVA (parentesi nella parentesi, nominerò spesso JAVA in quanto io attualmente sviluppo in questo linguaggio, ed è uno dei linguaggi più conosciuti e usati, quindi ottimo per fare confronti e paragoni. Chiusa parentesi) non dispone del garbage collector, quindi bisogna prestare molta attenzione a come si usa la memoria, soprattutto se andate a scrivere codice per dispositivi portatili con scarsa quantità di memoria (chi ha detto iPhone alzi la mano).
- **Dimensioni ridotte** : sia in termini di scrittura del codice sorgente che spesso occupa solo pochi "k" di memoria, sia in termini di programma eseguibile, in quanto l'eseguibile essendo legato al sistema può essere fortemente ottimizzato, ovviamente nessuna portabilità per l'eseguibile.
- **Basso livello** : in realtà ANSI C è un linguaggio di alto livello, quindi con istruzioni che ricordano molto il linguaggio umano, ma al suo interno è possibile inserire delle istruzioni

Assembler (maledetto eccolo che torna), quindi è possibile anche gestire i registri e chiamare direttamente interrupt (se non sapete di cosa sto parlando non vi preoccupate, molto probabilmente non avrete mai a che fare con tutto questo a meno che non andrete a sviluppare applicazioni particolari, ma in quel caso non sarà certo la mia guida ad aiutarvi :D )

- **Tipizzazione Loose** : altro argomento molto importate quando si tratta un linguaggio di programmazione è quello dei tipi di dati, anche qui è necessario un capitolo a parte(forse più), quindi verranno rivisti in futuro, vi basti sapere che in C la flessibilità è un lusso che si paga a caro prezzo, spesso commettere un errore di tipizzazione può portare a conseguenze molto gravi (attacchi di buffer overflow ad esempio), e vi assicuro che commettere errori è molto molto semplice. Allora vi chiederete come mai lo metto tra i vantaggi, perchè proprio questa flessibilità estrema mi consente di raggiungere la tanto agognata efficienza. A già... ovviamente JAVA ha una tipizzazione di tipo Strong, speravate che avessero almeno qualche cosa in comune :D

Dopo aver parlato quindi dei vantaggi, e fondamentalmente dopo aver mostrato alcune differenze con altri linguaggi, credo sia giusto dedicare qualche riga ad un concetto abbastanza importante, ovvero la differenza tra codice compilato, e codice interpretato. Non entrerò nel dettaglio sul funzionamento di interpreti e compilatori, altrimenti sarebbe necessario scrivere una "bibbia" dell'informatica e non un tutorial, ma se siete interessati ad approfondire la cosa inviatemi pure una mail e potrò consigliarvi qualche buon testo dal quale prendere spunto, soprattutto il mondo dei compilatori è davvero affascinante, ma preparatevi a farvi un'abbuffata di teoria e di matematica.

Ogni programma, viene implementato scrivendo del codice in uno qualsiasi dei linguaggi di programmazione desiderati, quindi sia che io decida di programmare in JAVA, in ANSI C, in Visual Basic, in Objective-C, in C++, C#, Delphi, Pascal, PHP, etc... devo comunque scrivere una sequenza di righe di codice che seguono una specifica sintassi. Alla fine della scrittura avrò un insieme di file che conterranno il codice sorgente del mio programma finito. Una volta scritto il codice vogliamo vedere il risultato del nostro lavoro. Quindi in base al linguaggio scelto abbiamo la possibilità di trovarci davanti o un **compilatore**, o un **interprete**, oppure **entrambi**.

Come già detto in precedenza il linguaggio ANSI C fa parte della famiglia dei linguaggi compilati, in pratica si scrive il codice sorgente, lo si da in pasto ad un compilatore che lo controlla, lo elabora e trasforma ogni singola riga di codice nel linguaggio macchina corrispondente, che può così essere eseguita direttamente sul processore.

Nei linguaggi interpretati, invece, il codice viene interpretato al volo senza bisogno di trasformarlo in un linguaggio direttamente comprensibile dalla macchina (ad esempio del codice in PHP viene interpretato al volo e ci viene restituita una pagina in html pura che può essere visualizzata dal browser) quindi il codice non è più strettamente legato alla macchina sulla quale è stato scritto (in quanto non esegue istruzioni macchina comprensibili solo a quell'hardware), ma può essere portato su qualsiasi sistema dispone di quell'interprete.

Naturalmente il solito JAVA deve differire da tutto (giuro che mi sta simpatico, ma quando si parla di C il linguaggio della Sun diventa il mio bersaglio preferito). Esistono quindi anche dei linguaggi "ibridi", ovvero che vengono prima compilati in un formato intermedio, successivamente questo formato viene interpretato su un interprete che lo esegue al volo, il vantaggio di tutto questo è che si ha un codice controllato e in parte ottimizzato ma che allo stesso tempo, grazie alla macchina virtuale che lo interpreta, è fortemente portabile.

Non spaventatevi se questi concetti non vi sono chiari, vedrete che dai prossimi tutorial la cosa sarà molto più complicata. :D Quindi alla prossima

Manuel (aka Jekrom)

per segnalazioni : [manuel.montini@gmail.com](mailto:manuel.montini@gmail.com)

<http://iphonecodes.wordpress.com/>

#### LICENZA:

**Questo tutorial è rilasciato sotto licenza Creative Commons : “Attribuzione-Non commerciale-Non opere derivate 2.5 Italia”,**

<http://creativecommons.org/licenses/by-nc-nd/2.5/it/>