

## ANSI C – IDE e Compilatori

Con questo tutorial cercherò di trattare un argomento piuttosto noioso e per la maggior parte di voi totalmente inutile, ma dato che ho deciso di partire dal presupposto che anche mia nonna possa voler leggere questi tutorial e mettersi a scrivere programmi in C, sono costretto a dedicare uno spazio anche agli IDE di sviluppo (o agli editor) e al compilatore.

Cercherò di rendere il tutto il meno noioso possibile, se ritenete di saperne abbastanza e che leggere queste paginette sia una inutile perdita di tempo, potete saltarle senza troppi problemi.

Ora mi rivolgo a tutti quei lettori che hanno deciso di soffermarsi su questo documento, in questo tutorial cercheremo di spiegare a grandi linee le differenze tra un comune editor e un'IDE di sviluppo complessa e completa. Cercheremo di spiegare che cos'è un compilatore e a cosa serve, nonché quali sono i principali tools che ci possono aiutare nello sviluppo del codice (lungi da me dilungarmi in inutili dettagli, non ho il tempo materiale di scrivere libri e libri di roba). Infine vedremo la differenza tra file con estensione “.c” e “.h” e che cosa avviene appena compiliamo del codice sorgente, con un piccolo accenno tra errori a “tempo di compilazione” e a “runtime”. Ok se la cosa ancora non vi interessa saltate pure il tutorial, altrimenti si comincia.

### *IDE Vs Editor*

Per toccare questo argomento faccio un salto nel passato, ero poco più che un bambino quando i miei genitori mi regalarono il primo computer, era un bellissimo commodore 64 fiammante, ricordo che per caricare un gioco ci metteva una vita, i giochi erano su una cassetta e per giocare era necessario riavvolgerla fino al punto giusto. Quanto si è evoluto il mondo dell'informatica in poco più di 15 anni. Comunque ricordo che mio zio (poco più grande di me) aveva un libricino sulla programmazione in basic, un linguaggio che mi ha segnato (ragazzi nulla di più bello che scrivere codice usando i goto :D ).

Chiesi a mio zio se poteva prestarmi quel libricino, tanto a lui serviva per tenere ferme le gambe del tavolo, e iniziai a sfogliarlo nel buio della mia cameretta (ai tempi ancora non sfogliavo i playboy). Sinceramente non ci capivo assolutamente nulla, allora decisi di aprire una pagina a caso. Trovai una bellissima immagine di tante linee disegnate a caso su un monitor, al giorno d'oggi una cosa del genere può sembrare una stupidata, ma se pensate che ai tempi la grafica tridimensionale era lontana anni luce, non ricordo nemmeno se il mio monitor fosse a colori ma non credo. Comunque nella pagina accanto era presente del codice sorgente scritto in basic, decisi di copiarlo riga per riga sulla shell e diedi il comando “run”, fantastico... le linee si disegnavano e si muovevano a caso sullo schermo.

Perché tutto questo? Non solo per rivangare il passato che è sempre un piacere, ma per farvi capire che per scrivere un programma avete bisogno solamente di due cose, un editor dove scrivere il vostro codice e un qualche cosa che ce lo manda in esecuzione.

Ma che cos'è un editor? La risposta è molto semplice, un editor di testo è un software che ci permette di creare, visualizzare e modificare dei testi in formato digitale. Quindi un editor è un programmino (che può essere più o meno complesso) che ci permette di scrivere il nostro codice e di farcelo salvare all'interno di un file. Tutti abbiamo almeno un semplice editor di testo sul nostro computer, quindi tutti siamo in grado di scrivere del codice sorgente di un programma.

Nel 1983 la Borland mise sul mercato il primo IDE commerciale basato sul linguaggio Pascal, ovvero il Turbo Pascal (anche se già dai primi anni 70 si trovavano IDE che comprendevano un sistema operativo integrato). Ora la domanda che vi state facendo dall'inizio è: ok, ma che

diavolo è un IDE? Un **integrated development environment (IDE)**, come dice l'acronimo stesso, un ambiente di sviluppo integrato. Ovvero è un software che aiuta il programmatore nello sviluppo delle applicazioni. Normalmente oltre ad un editor di testo offre anche degli strumenti per la compilazione e per il debugging del codice (ovvero la correzione degli errori). Alcuni IDE dedicati a linguaggi di programmazione orientata agli oggetti offrono anche la possibilità di navigare all'interno delle classi e di analizzare gli oggetti.

Come già detto prima per sviluppare del codice è necessario avere un editor, ma ovviamente utilizzare questi ambienti integrati può agevolare notevolmente il lavoro di un programmatore. Tra gli IDE più conosciuti spuntano NetBeans, Anjuta, Eclipse, Visual Studio .NET, Xcode, Dev-C++, JBuilder, KDevelop, e molti altri. Siete liberi di cercare su internet le varie caratteristiche di ogni IDE e di scegliere quella che preferite.

## *Il Compilatore*

Una volta che abbiamo scritto il nostro codice in ANSI C, questo deve essere tradotto in un linguaggio che la macchina riesce a comprendere ed eseguire. Fondamentalmente un calcolatore è solo uno stupido strumento in grado di interpretare degli impulsi di corrente facendoli passare attraverso circuiti integrati. Quindi il vostro calcolatore capisce solo ed esclusivamente due cose, o passa corrente, o non passa corrente. Vi ricorda qualche cosa? Mai sentito parlare di numeri binari?

Dobbiamo quindi utilizzare un software che macina il nostro codice e ne crea un pappone da far digerire alla nostra macchina, questo software è il compilatore. Il compilatore prende in ingresso un programma scritto in un determinato linguaggio (ovviamente per quel che ci riguarda è il C), esegue su di esso una serie di operazioni e se in assenza di errori lo "compila" (l'operazione di trasformazione del codice si chiama compilazione). Il risultato è un codice oggetto che può essere eseguito sulla macchina.

I compilatori dividono quindi la fase di compilazione in due macrofasi, una prima nella quale il sorgente viene preso e trasformato in un linguaggio intermedio. In questa fase viene fatta una analisi di tipo lessicale, un'analisi di tipo sintattico e un'analisi di tipo semantico. Evito di entrare nei dettagli, perchè questa parte richiede una teoria molto complessa.

A questo punto abbiamo un codice intermedio, questo passa alla seconda macrofase dove viene ottimizzato e viene creato il codice oggetto finale.

Spesso i compilatori sono in grado di riconoscere alcune classi di errori e di suggerire all'utente come correggerli.

Il codice sorgente, però, subisce una ulteriore modifica prima di giungere al compilatore, questo viene dato in pasto ad un altro software chiamato preprocessore. Il preprocessore effettua delle sostituzioni testuali al codice sorgente in base alle direttive fatte dal programmatore, più avanti nei tutorial vedrete ad esempio la direttiva al preprocessore `#include`.

Ci sarebbero moltissimi altri argomenti da trattare, ma per ora limitiamoci a questi, magari il resto verrà ripreso nei tutorial futuri. Ora cerchiamo di capire dove memorizzare il codice sorgente che abbiamo scritto.

## *Estensione ".h" e ".c"*

Abbiamo detto che il codice sorgente deve essere salvato all'interno di alcuni file e che questi poi vengono compilati per ottenere un programma eseguibile dal nostro calcolatore. Esistono due tipi diversi di file che dobbiamo utilizzare per memorizzare il nostro programma, il file che hanno come estensione ".h" e quelli che hanno estensione ".c". Vedrete nei prossimi capitoli a cosa

servono entrambi i tipi di file e quale sia il loro utilizzo, per ora vi basti sapere che nei file “.c” sarà contenuto il codice vero e proprio delle funzionalità del nostro programma, mentre negli header file ( da header appunto viene l'estensione .h) sono contenute informazioni utili per il nostro programma.

Nel prossimo tutorial analizzeremo il nostro primo programma. Buona lettura.

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/>