Escape characters

La scorsa volta abbiamo visto come utilizzare la funzione printf() per stampare una stringa di testo formattato e come stampare integrate in esso i valori delle nostre variabili attraverso i place holder.
Ciò che non abbiamo visto, è come inserire alcuni caratteri speciali e eventualmente importanti, che non sono inseribili dalla tastiera.
Per esempio, immaginiamo di voler stampare a video (sulla riga di comando) delle ", non potremmo farlo, poiché, immettendo delle ", queste, durante la traduzione del programma, verrebbero riconosciute dal compilatore come la fine della stringa del testo da stampare!
Come facciamo?
Per risolvere questo problema ci vengono in aiuto gli escape characters:
Come vedete, alla nona riga, troviamo \", che è a tutti gli effetti UN carattere rappresentante le ", per questo, se volessimo usarle, dovremmo scrivere in questo modo:
printf("In questo testo è presente la parola \"testo\" :)");
Scrivendo in questo modo, il compilatore capirà che la nostra intenzione è quella di scrivere delle " e capirà che non deve terminare la stringa.

Ci sono molti altri escape characters, per esempio, il più usato, è sicuramente \n, che ci permette di andare a capo. Perché non premere invio?

Se usate un buon IDE (Integrated Develoment Environment), che per capirci può essere anche un blocco note, noterete che andando a capo a metà stringa, la colorazione di questa nella nuova riga cambierà, infatti, andare a capo risulta in un'interruzione sbagliata di una stringa che vi porterà ad un errore del compilatore!
Se avete intenzione di continuare a scrivere il vostro testo a capo, potete usare \Invio (ovviamente premento il tasto Invio).
Questo escape character vi permetterà di continuare a scrivere la vostra stringa a capo, ma non andrà a capo nella vostra stringa stampata!

Un altro che vale la pena conoscere è \\, usato per scrivere uno \

Se invece volete scrivere il simbolo % dovrete usare, diversamente dalle aspettative di alcuni, %%.

Ci tengo a precisare, che tutti questi escape characters, sono a tutti gli effetti un solo carattere e di conseguenza, se memorizzati, occupano sempre 1 byte di memoria!
Tant'è che è possibile assegnarli a variabile di tipo char
```
char a = '\n';
a='\\';
a='%'
/*notare che in questo caso ho inserito un solo %, è un carattere e non può rappresentare un place holder, di conseguenza % può significare solo il simbolo %*/
```

La scanf()

Indovinate un po'?

Abbiamo visto come stampare dati, ora vediamo come immetterli, dopo aver fatto partire il programma s'intende!

Della scanf() ci sarebbe da parlare per giorni, infatti iniziamo ora, ma non finiamo oggi, ogni tanto verrà sempre detto qualcosa, in ogni caso, cercheremo di essere abbastanza esaustivi a riguardo.

La scanf, si usa in questo modo:
int a; float b; char c;
scanf("%c%d%f", &c, &a, &b); /* notare bene le & che sono assolutamente necessarie, vedremo perché */
Cosa riconosciamo in questa funzione? Sicuramente, ci accorgiamo che non è poi così diversa dalla printf(), anzi, è quasi uguale! Ma come va usata esattamente
Ecco cosa succede quando chiamiamo la scanf scritta qui sopra.
Il programma si ferma e si mette in attesa che l'utente scriva qualcosa, vedrete il cursore che lampeggia a vuoto e avrete la possibilità di scrivere.
Potrete scrivere quello che volete, ma la scanf, quando analizzerà la stringa di ciò che avete scritto, aspetterà di trovarsi, in quest'ordine:
un carattere;
un intero;
un float.
La scanf, passerà alla variabile successiva, dopo essersi resa conto di aver finito di leggere la prima.
Immaginate di scrivere g 12 11.1 e premere invio. Una volta premuto invio, il sistema operativo che gestisce il cmd manderà questa stringa alla scanf(), che leggerà un carattere alla volta:
l: ok, è un carattere singolo, posso passare alla ricerca della prossima variabile
: è uno spazio, ma io mi aspetto un numero intero... continuiamo a cercare
1: Il carattere '1', nice! È, un numero, vediamo se è a più cifre
2: Il carattere '2', ok, lo aggiungo al numero di prima
: Il carattere ' ', perfetto, non devo più scrivere l'intero, passiamo alla prossia variabile che è un float
1: Stessa cosa, comincio a scrivere il float
1: continuo
.: continuo, il . può essere presente in un float
1: continuo
E poi? Come mi fermo?
\n: perché l'ho scritto? Perché quando premiamo invio, oltre a mandare il comando, inseriamo anche il carattere corrispondente, che viene salvato nel buffer di input, chiamato input stream.
Ogni volta che la scanf, o un'altra funzione di input legge un carattere, questo viene scartato dall'input stream, così, ad ogni nuova variabile, la scanf leggerà l'ultimo carattere non letto.
RICORDATEVI, l'ultimo carattere, quasi sempre \n, anche se viene letto NON viene rimosso dall'input stream, per questo, una seconda chiamata a scanf, troverà come primo carattere da leggere il \n, che non sarà un problema se dovrete leggere un'intero o comunque un numero, poiché verrà automaticamente scartata.
Ma lo sarà in caso dobbiate leggere un carattere o una stringa, poiché, l'ultimo \n dell'inserimento precedente verrà memorizzato nel char nel primo carattere della stringa in questione, terminando la sua lettura di conseguenza. (vedremo come funzionano le stringhe)

Per completezza, vi dico anche che
scanf("%ccaratteri%d", &a, &b);
Con questa scrittura, la scanf, dopo aver letto un primo carattere e averlo inserito in a, comincerà a cercare la serie di caratteri "caratteri", dopo la quale si aspetterà un intero.
scanf(%d%*c%*d%f, &a, &b);
Con questa scrittura invece, memorizziamo un intero in a, poi cerchiamo un carattere e lo scartiamo, cerchiamo un'intero e lo scartiamo, poi cerchiamo un float e lo salviamo in b
In ogni caso raramente dovrete utilizzare tecniche simili, anche perché non sapete mai cosa può inserire l'utente.
Vedremo in futuro come sarà più sicuro prendere l'intera stringa e trattarla noi stessi per estrarne le informazioni che vogliamo :)