**Settimana prossima, tutoraggio il Lunedì e lezione il Venerdì. Settimana prossima ancora, lezione tutti e quattro i giorni, Lunedì incluso.**

Printf non conosce la dimensione della variabile, e nemmeno il tipo. Dobbiamo essere noi a dirglielo. Fare attenzione alla dimensione delle variabili che gli andiamo a passare. Passare 16-bit in un comando ad 8 ok, in quanto va a prendere i primi 8 bit significativi. Però nel caso diverso, cioè passare 8 bit in un comando che ne richiede 16, non è corretto. Come nell’esempio fatto in classe potrei avere un risultato corretto, ma non è sempre detta come cosa perché dipende dall’interpretazione del calcolatore della funzione printf.

Se do un numero, e lo interpreto come “Carattere”, lui mi stampa il valore in ASCII corrispondente al numero, mentre se lo stampo come intero ottengo il numero esatto. Bisogna ricordare che il dato in memoria è lo stesso, quello che cambia è solamente l’interpretazione che gli viene data

**Qualificatori C – Tipo e Dimensione**

**Tipo**

c carattere

u intero senza segno

d intero con segno

x intero in esadecimale

f, g virgola mobile

**Dimensione**

hh char

h short

l long

ll long long

L long double

%s è un segnaposto per l’argomento che andiamo a passare dopo

**Sizeof**

Stampa la dimensione in memoria della variabile che gli vado a passare in byte (RIVEDERE A CASA)

**Do while**, che non c’è in Python ma ci sta qua

Una qualsiasi espressione in C è caratterizzata da un effetto ed un valore. Non è presente in molti linguaggi. Un espressione che ha un valore diverso da 0 viene sempre valutata vera, mentre una con valore 0 viene sempre valutata falsa.

**Operatori: precedenza ed associatività**

**Shift** mi permette di spostare i bit in una determinata direzione, ogni Shift può essere visto come una moltiplicazione o divisione per 2, in quanto un Bit corrisponde al valore 2