Lezione 4 - 08 03 18

Thursday, March 8, 2018 10:44

Cast di tipo:

```
#include <stdio.h>
#include <stint.h>

int main () {

    int16_t num = 1;
    if ( *(int8_t *) &num )
        printf("Little Endian");
    else
        printf("Big Endian");
    return 0;

} //main
```

Letto da destra a sinistra

Se scrivo

```
int x;
float y = 1.3f;

x = (int) y;
```

↑ è come se ci fosse

La conversione non modifica i bit del dato, ma il tipo di indirizzo!

Il tipo booleano non esiste in C, può essere qualsiasi tipo.

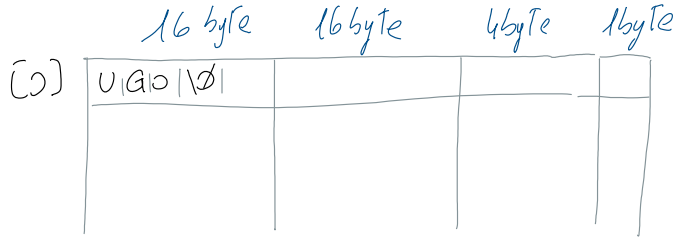
Se ha almeno un bit a 1 è vero, altrimenti falso.

Potevo scrivere:

```
printf(" L'architettura è : %s endian ", (*(int8_t *)&num ? "Little" : "Big"));
```

STRUTTURE: in C posso definire un nuovo tipo

Stringa: array NULL terminated (di caratteri)



Problema dell'allineamento: non posso mettere un tipo di 2 byte (int16_t) e in una cella dispari, il compilatore deve allineare.

↳ aggiungere un padding

Campo in un indirizzo non multiplo della dimensione.

Nei livelli di rete viene fatta eccezione a questa regola.

```c
#include <stdio.h>
#include <stdint.h>

struct anagrafica{

    char nome[16];
    char cognome[16];  //campi della struttura
    int16_t anno;
    char sesso;

};

//non ho ancora occupato un singolo byte in memoria
//ho solo detto al compilatore che esiste questo tipo di dato

struct anagrafica rubrica[100];  //creo una rubrica di 100 dati anagrafica

int main() {

    rubrica[0].nome[0]='U';
    rubrica[0].nome[1]='G';  //si arriva sempre alla scrittura o lettura di un tipo base
    rubrica[0].nome[2]='O';
    rubrica[0].nome[3]='\0';  //carattere terminatore di stringa
    rubrica[0].cognome[0]='C';
    rubrica[0].cognome[1]='A';
    rubrica[0].cognome[2]='L';
    rubrica[0].cognome[3]='\0';
    rubrica[0].anno=1990;
    rubrica[0].sesso='M';
    printf("riga 0: nome = %s\n", rubrica[0].nome);

    //come è disposta in memoria questa riga della struttura?

    unsigned char * c;
    c=(char *)rubrica;
    int i;
    for(i=0; i<40; i++)
        printf("%x %d: %d (%c) \n",&c[i],i,c[i],c[i]);

    return 0;
}
```