# Lezione 3 - 02 03 18

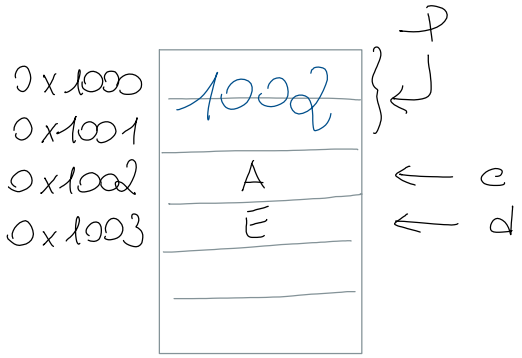
Friday, March 2, 2018 8:34

```c
#include <stdio.h>

int main ( ){

    char *p;
    char c,d;
    c = 'A';
    printf("%d",c); //65
    p = &c;
    printf("%c"); // 'A'
    printf("%x",p); // 81
    p = p+1;
    *p = 'E';
    printf("%c",*p); // 'E'

}
```




Il tipo in C determina la dimensione di un dato (dove il dato inizia, spazio occupa)

Si può chiedere la dimensione attraverso:

printf("%d", sizeof(char)) → 1 byte

Esistono tipi di dati

short int

long int

long long int

int16_t

int32_t

int64_t

→ bisogna fare #include <stdint.h>;

↳ esplicitano nel nome la dimensione

PS: i tipi di dati sono implementati in hardware

↳ 1 sola operazione per quelle elementari

int e basta è ambiguo, dipende dalla macchina così.

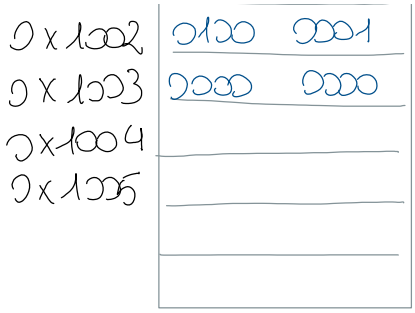
```c
#include <stdio.h>
#include <stdint.h>

int main ( ){
```

```c
int main ( ){

    short int *p;
    short int c,d;
    c = 'A';
    printf("%d",c); //65
    p = &c;
    printf("%c"); // 'A'
    printf("%x",p); // 81
    p = p+1;
    *p = 'E';
    printf("%c",*p); // 'E'

}
```




perché
comunq
indiz
però i
infatti
p=p+
+1 * dimensione

65 in binario è 0000 0000 | 0100 0001

Big endian: inizia con i bit meno significativi.

Little endian: inizia con i bit più significativi.

} è una scelta hardware

Esercizio per casa: scrivere un programma che capisce se l'architettura viene eseguito è little o big endian.

Il formato network order è big endian.

Non ha senso parlare di ordinamento di singoli bit.

(*) p = p+n; su puntatore è in realtà p = p + n x sizeof(*p);

short int *p

due modi di lettura in C

(short int *) p

short int (*p)

quando scrivo è uno

ARRAY:

*(p+n) ≡ p[n]

Quindi posso scrivere:

p = p+1;
*p = 'E';

⟹

p[1] = 'E';

Le quadre sono un operatore, non qualcosa di proprio degli array.

p[n]

1) prendo p
2) sommo n x dimensione del tipo puntato
3) legge il valore all'indirizzo ottenuto

} *(p+n)

Il puntatore si comporta come un array.

cioè ↳ leggo la memoria come

Allora, l'array può essere pensato come un puntatore? Sì!

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

int main ( ){

    short int a[3];
    a[0]=10;
    a[1]=20;
    a[2]=30;
    printf("%d",*(a+2));

}
```

=> a corrisponde all'indirizzo del primo spazio di

a = &a[0]

Differenza tra array e puntatore:

1) Array: ha la sua zona di memoria pari a quella allocata, associata al

↳ però è vincolato all'indirizzo statico dato dal compilatore

2) Puntatore: ha la sua area di memoria ma usa quella degli altri.

↳ non ha vincoli di proprio indirizzo.

=> NON posso scrivere a = a+1; perché a è sempre costante

Scaricare PuTTY per usare il cloud (ci darà server, username e passw

⇓

88.80.187.84

Funzioni:

nano nomefile.c editor

cat " mostra ciò scritto

gcc nomefile.c -o nomefile compila

./nomefile esegue