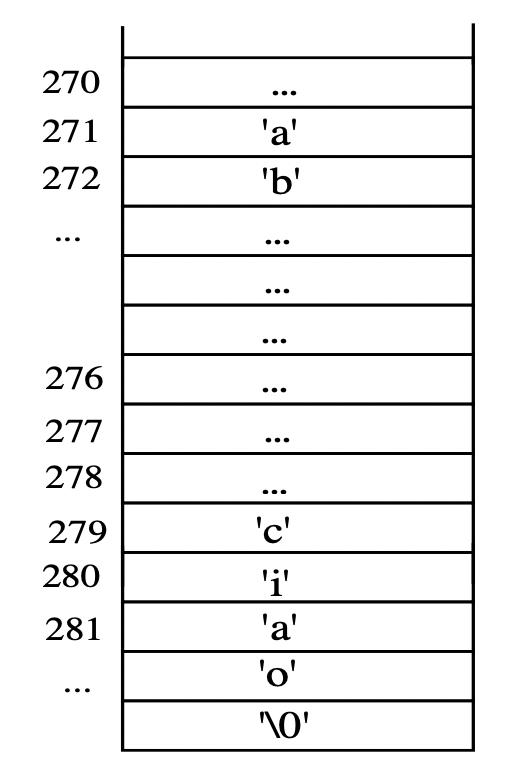
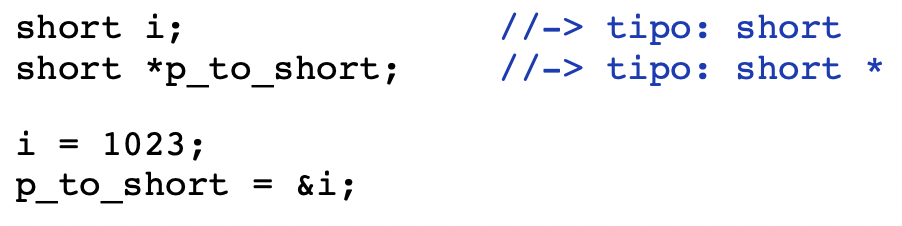
**Puntatori, Stringhe e Vettori**

****La grandezza di un pointer è indipendente dalla grandezza del tipo puntato. L’operatore ‘**&**’ fornisce l’indirizzo della variabile affiancata.

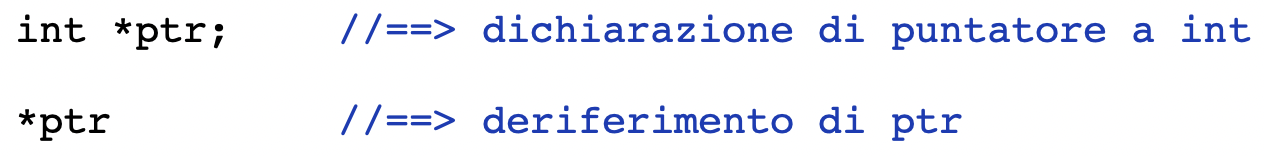
L'operatore ‘**&**’ può essere usato con variabili semplici o elementi di vettori.

*Esempio: char ch1 = ‘a’ char str[ ] = “ciao”  
 &ch1 == 271 &str[0] == str == 279*

Per le stringhe, l’indirizzo del primo elemento equivale al nome della stringa.

L’indirizzo di una sequenza di byte corrisponde all’indirizzo del primo byte.

In un array di stringhe con \*array[ i ] ottengo il primo carattere della i-esima stringa, con \*(array[ i ] + k) ottengo il k-esimo carattere della i-esima stringa

Quando uso l’asterisco in dichiarazione, serve per dichiarare una variabile di tipo puntatore, mentre se la uso a fianco a un puntatore già esistente significa che vado a recuperare il valore della variabile puntata dal puntatore.

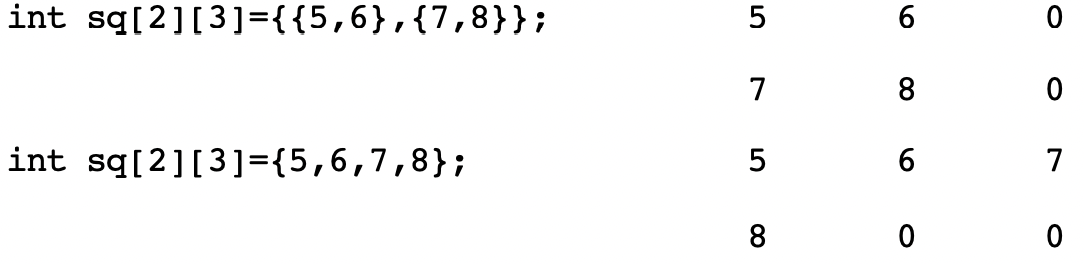
**Funzione di scambio con puntatori**

  
Se incremento un puntatore, passo all’elemento successivo dell’array.

Se dichiaro un puntatore **const \*ptr = &i ;** non permetto di modificare il valore di **i** anche se **i** non è costante.

Quando passo ad una funzione un **array** è come se lo passassi per indirizzo, ecco perché i valori vengono modificati anche all’esterno.   
Infatti: **array = &array[0]**Gli elementi del vettore sono \*(ptr+0), \*(ptr+1), \*(ptr+2), … —> \*(ptr+i) 0 <= i < N

*int ( \*pm )[2] —> ptr a un array di due interi — int \*pm[2] —> array di ptr*

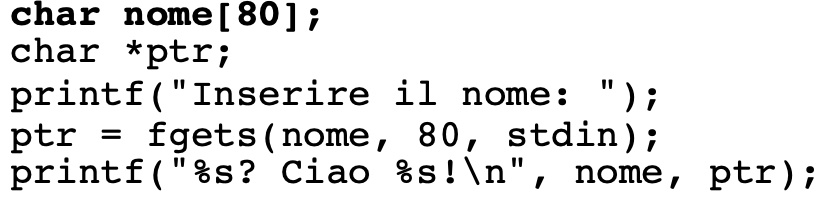
**sizeof( )**La funzione sizeof(var) restituisce la dimensione in byte dell’elemento var richiesto.  
Esempio: int array[4] ; sizeof(array) == 16 sizeof(int) == 4 n = sizeof(array)/sizeof(int) = 16/4 = 4

**Dichiarazione alternativa di puntatore di stringa**

char \*str = malloc(20 \* sizeof(char));

**Array multidimensionali (matrici)**

*Esempio inizializzazione:   
m*atrice = matrice[0] = &matrice[0][0]  
matrice[1] = &matrice[1][0]  
Nei parametri delle funzioni -> funz(int mat[ ][ncol]) oppure “int (\*mat)[ncol]”

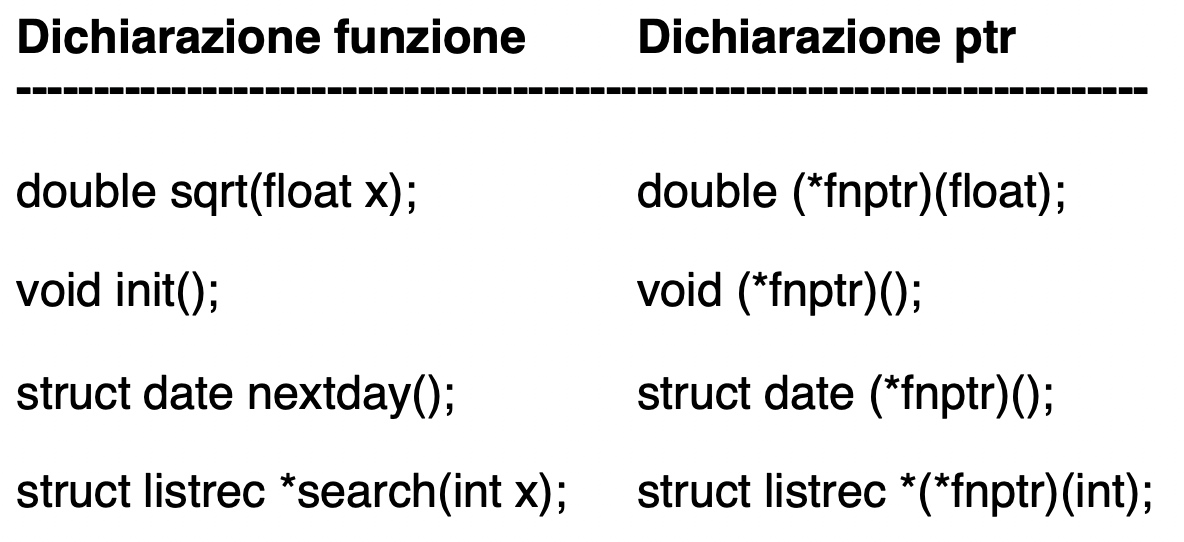
**fgets(var, length, stdin)**

Ritorna **null** se non riesce a prendere l’input

C’è anche **puts(var)** che stampa una stringa (mandata con puntatori, variabile già esistente o direttamente con costante tipo “ciao”)

**Argomenti a linea di comando**

argv[ i ] —> i-esimo argomento inserito al momento dell’esecuzione (partono da 1, 0 è il comando/file !)  
argc —> num argomenti inseriti e quindi dimensione di argv main(int argc, char \*argv[ ])  
Gli argomenti sono di tipo stringa!

**Puntatori a funzioni**

Utili perché non è possibile passare una funzione come argomento di un’altra funzione, ma con i pointer è possibile.

Coi pointer si possono anche registrare le funzioni in array.

*Esempio:*

*Funzione  
double cube(double y) { return … ; }*

*Funzione che la utilizza  
void show( double (\*pf) (double), …, …) { printf(“%f”, pf(3)) ; }*

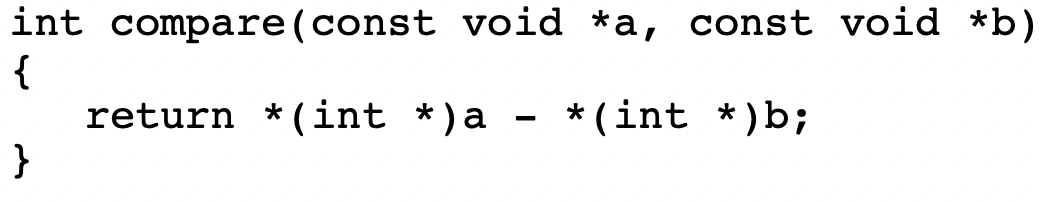
*Chiamata alla funzione  
show(cube, val1, val2, … );*

il puntatore può anche essere di tipo void.

**qsort( )**

La funzione accetta qualsiasi tipo di array ma vanno specificati dimensioni e dati:

qsort(array, numero elementi, sizeof(tipo vettore), funzioneCompare)

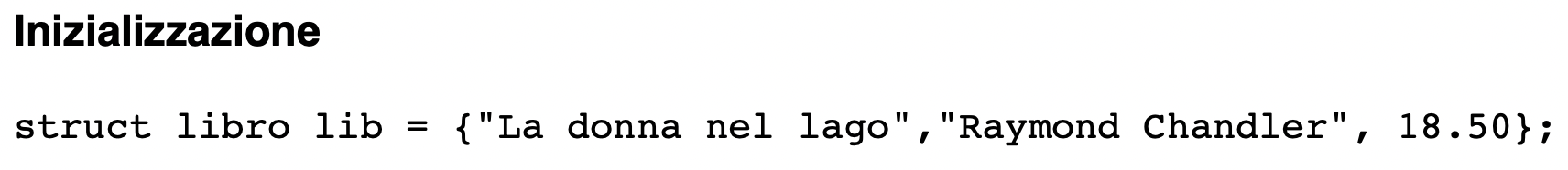
La funzione compare è la funzione che dobbiamo scrivere che userà il qsort per ordinare i numeri:

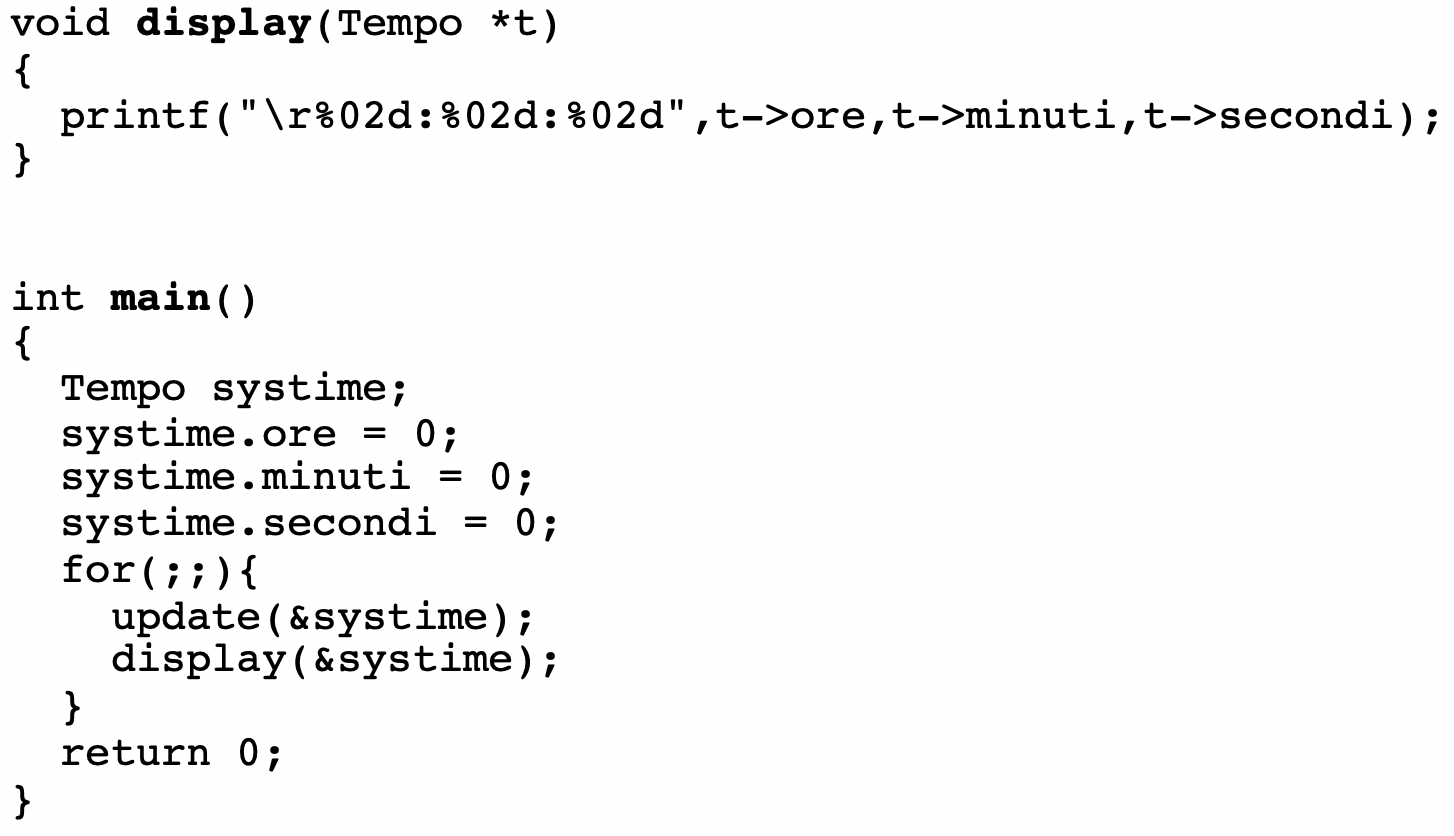
In pratica ritorna -1 se a è più piccolo di b: quindi ordina il vettore in modo crescente.

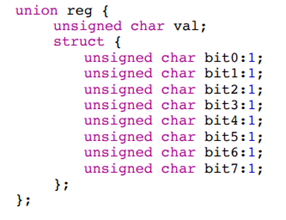
Se scelgo una dimensione più piccola di quanto sia lungo l’array, ordina solo i primi n valori.

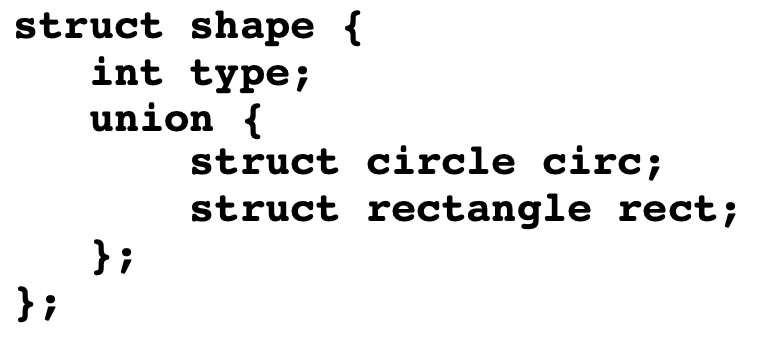
Se la funzione compare riceve puntatori a puntatori, allora raddoppio tutti gli \* nel return.

**Struct**

**Immagine**Con il typedef posso evitare di scrivere tutte le volte “struct libro” e scrivere solo “Libro”. Aggiungo tra la graffa e il “;” il nome che voglio dare (tipo “Libro”) e prima di “struct libro” scrivo “typedef”.

Se dichiaro un puntatore a struct allora accederò ai dati usando la freccia invece del “.”. Vi accedo col punto se la dichiaro come mostrato nell’esempio “Inizializzazione”.   
*Esempio: Clock \*clk1; clk1->seconds++;*

**Struct union**

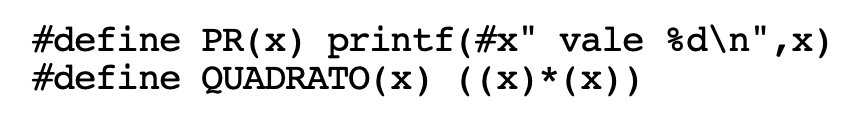
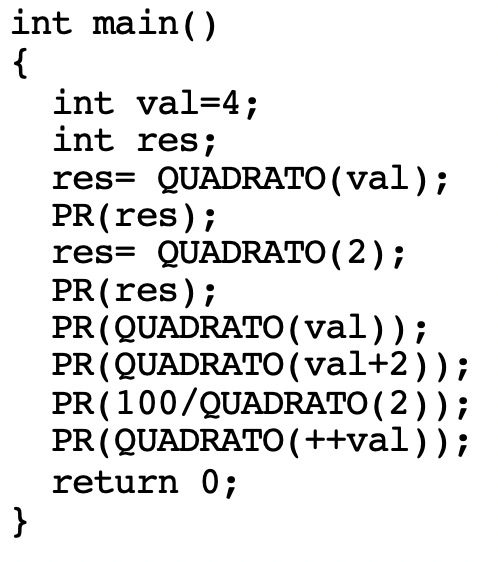
****

Le union allocano lo spazio del tipo più grande, i dati più piccolo avranno il valore troncato alla loro dimensione massima. È possibile dichiarare struct anonime all’interno della union.

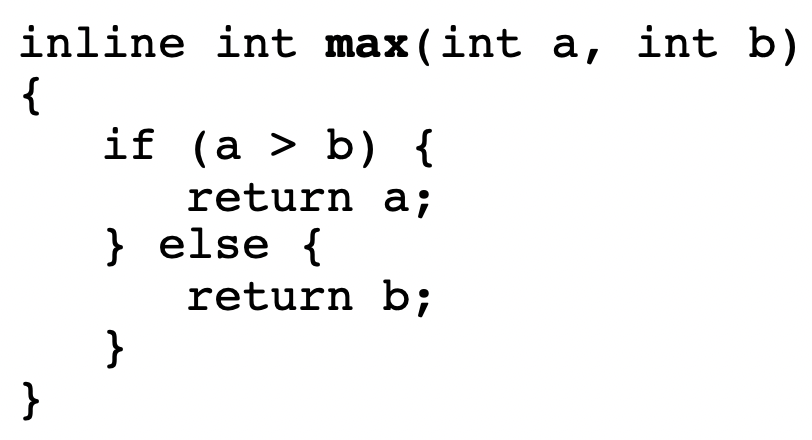
Esempio: nella union reg il valore in val sarà distribuito sul tipo anonimo e ogni bit avrà il corrispondete valore nel numero contenuto in val.

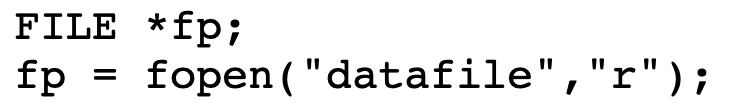
**Strutture Packed**Posso specificare quanti bit utilizza un dato se ad esempio ho un flag e voglio che usi solo un bit, il flag nella struct lo dichiaro del tipo: *unsigned int flag:1;*

**Macro Inline function**

****Quando il compilatore scorre attraverso il programma, va a sostituire le chiamate alla funzione con direttamente il loro codice;

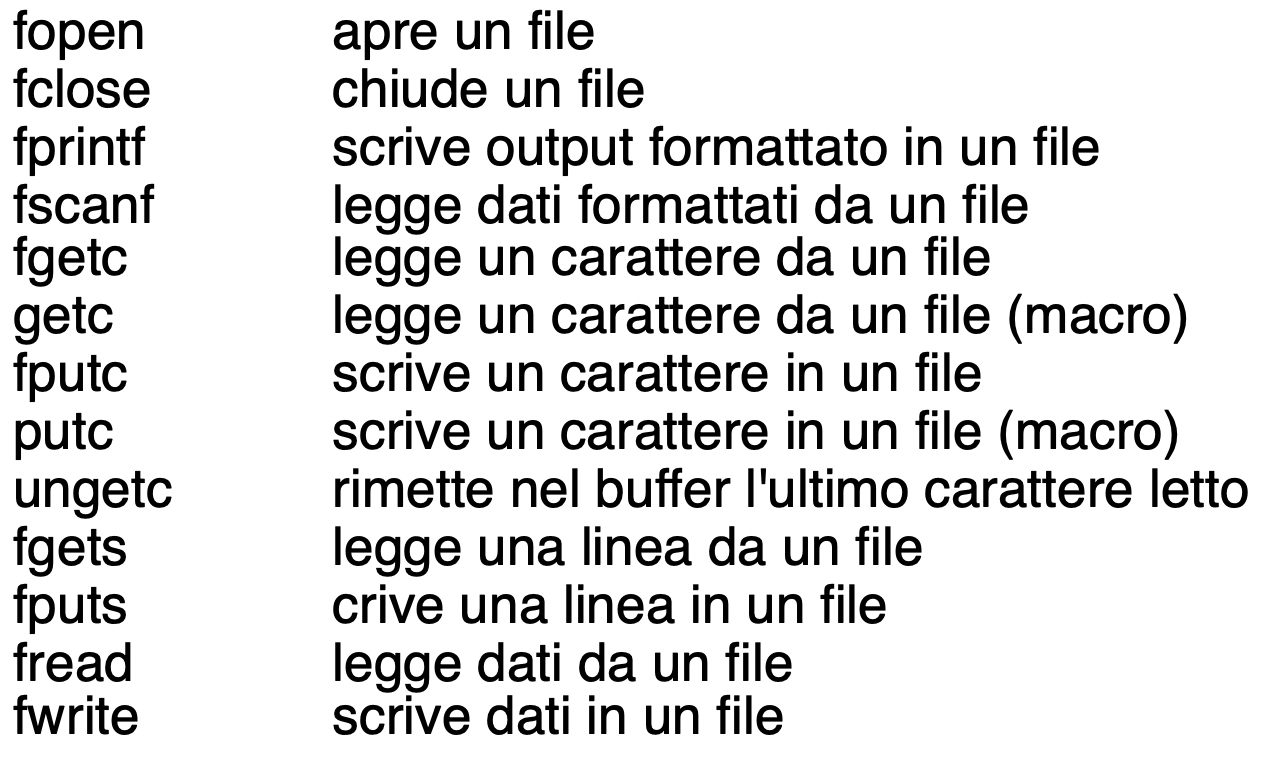
sostituisce il corpo della funzione alla chiamata (tipo le macro)

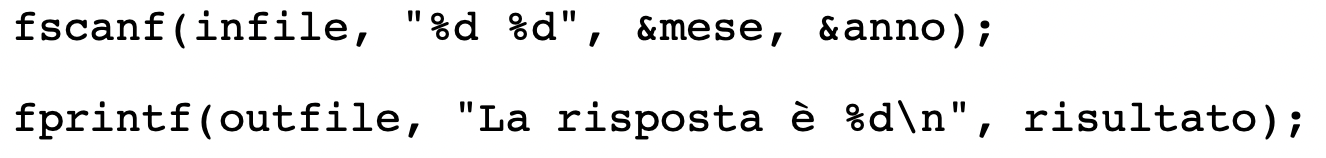


**File**

r = read, w = write, r+/w+ = read&write, a = append (anche a+)

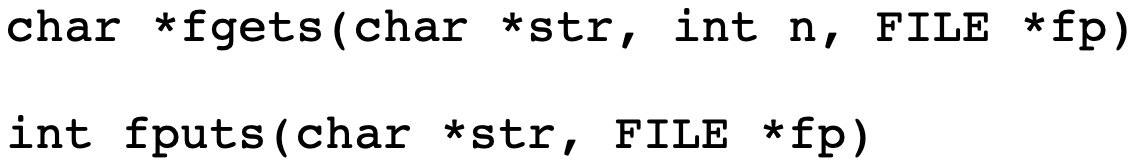
se aggiungo ‘b’ dopo, è con file binari.

**Devo chiudere il file alla fine!**



*Esempio scrittura lettura:*  
posso usare anche fprintf con ‘stdout’ al posto del file e sarebbe il printf. Posso usare fscanf con ‘stdin’ al posto del file e sarebbe scanf.

Per le stringhe:



fRead()  
viene restituito n, il numero di “cose” riuscite a leggere. Cosa leggere è indicato dal “sizeof(type)”.

“Buf” in questo caso è un ‘char buf[1024]’

Immagine

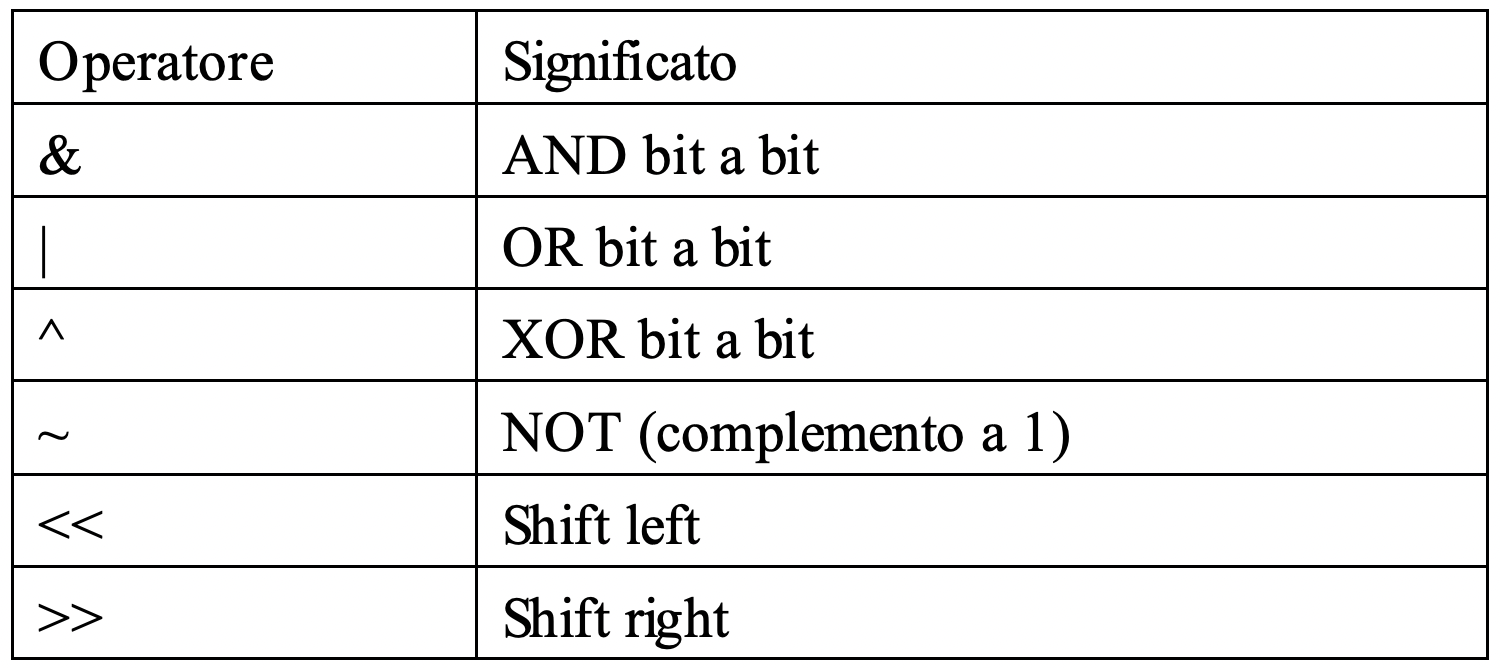
***strtock(input,set per separare):*** mantiene lo stato interno (variabile static), va inviato l’input solo alla prima chiamata . la funzione separa la stringa in input in una sotto stringa in base ai caratteri presenti nel set(parametro 2).

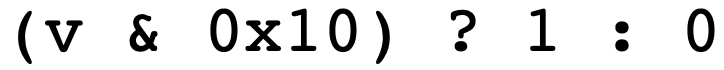
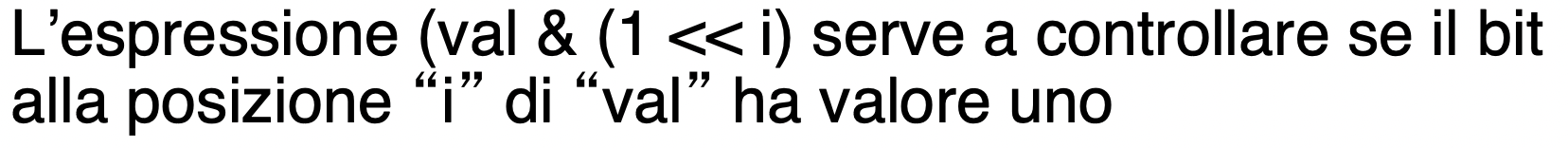
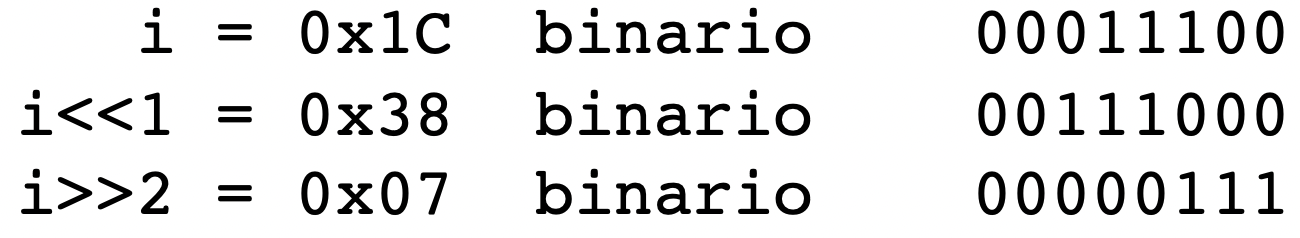
**Fseek rewind ftell**

* fseek(file, offset, mode): sposta il cursore di offset in base a mode:
  + SEEK\_SET: parte dal inizio del file e si muove all’indice offset (valore offset positivo)
  + SEEK\_CUR: sposto il cursore di offset dalla posizione attuale (valore offset positivo e anche negativo)
  + SEEK\_END: è come SEEK\_SET, ma parte dal fondo (valore offset negativo)

Tutte le operazioni vengono fatte in nella posizione successiva alla posizione del puntatore.

* rewind(file): riporta il cursore a 0
* ftell(file):legge il valore del cursore nel file

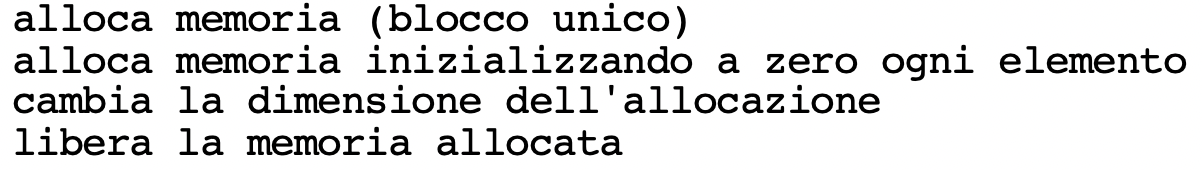
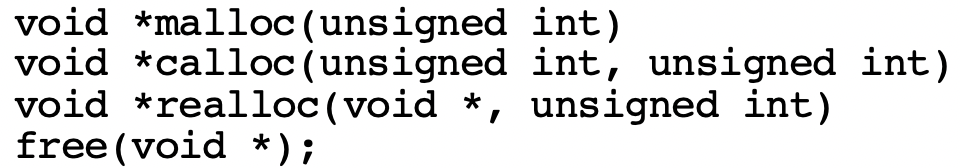
**Operatori bit a bit** *Esempio: pari/dispari —> (x & 1) == 0 (pari)*

***Shift* Mask****

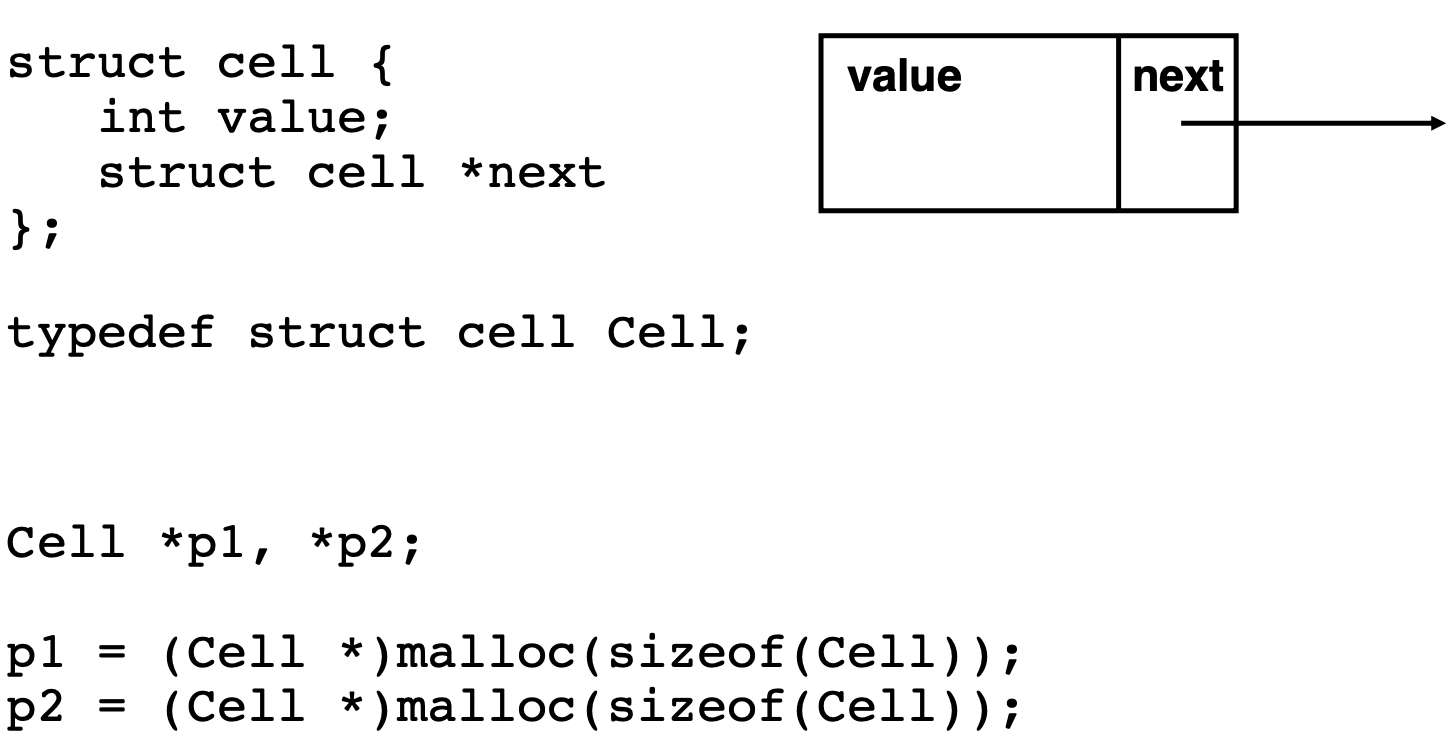
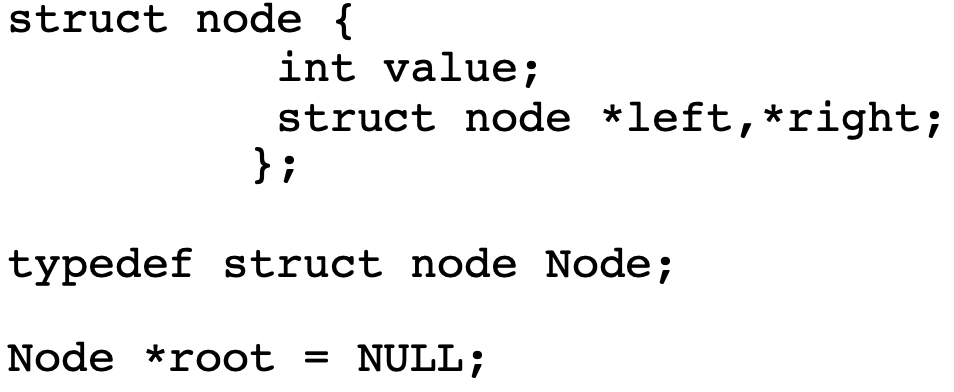
Stampare valori in esadecimale, ottale

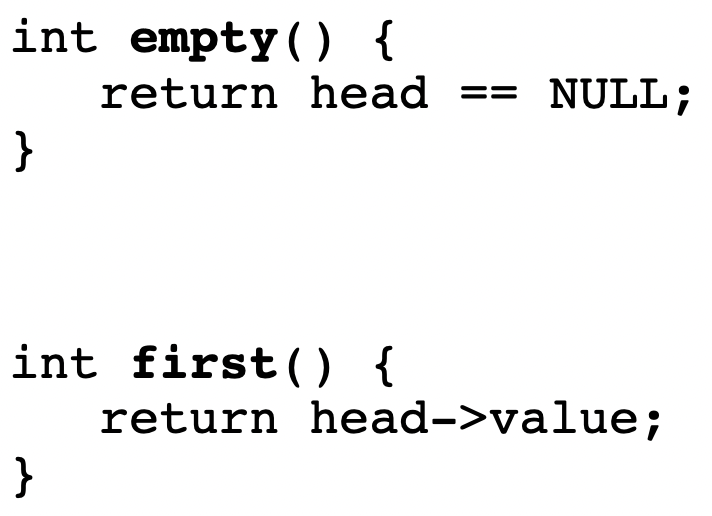
*printf("%#x esadecimale\n", val); //# serve a mostrare il tipo*

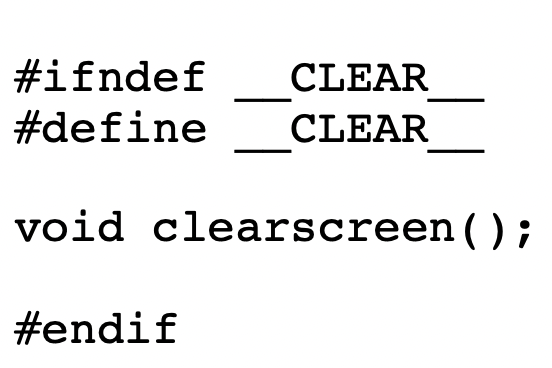
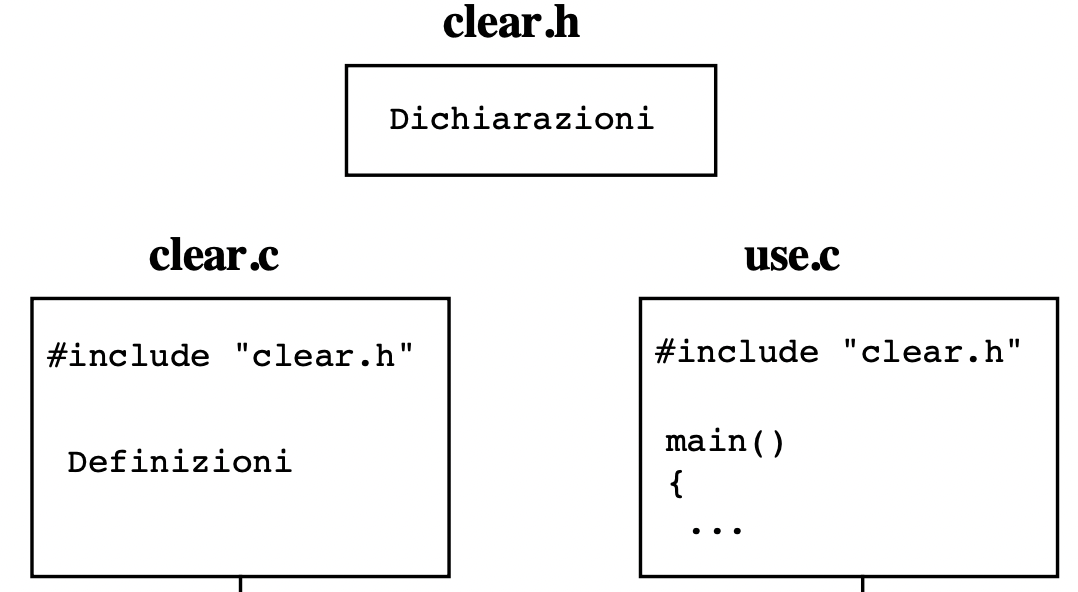
*printf("%#o ottale\n", val);*

*printf("%d decimale\n", val);****Memoria dinamica*** *Esempio stringa di 20 caratteri: char \*str = malloc(20 \* sizeof(char));  
Esempio tabella variabile: righe definite da i.   
Ciclo con: tab[ i ] = (char \*)malloc(strlen(buffer)) e. strcpy(tabella[ i ], buffer) (per inserire i dati)  
dove buffer —> char buffer[maxwidth];*

*Creare matrice su heap e usare la gestione classica con array, quindi con m[i][j]  
int (\*newMatrice)[righe] = malloc(sizeof(colonne \* righe \* sizeof(int)));*

***Liste dinamiche*** *Inizializzazione lista vuota* ***Alberi binariImmagine***

******

**Modularizzazione File.h Variabili**

Una variabile dichiarata **static** limita la visibilità e l’accesso alla memoria in questione al solo modulo in cui essa è dichiarata.  
Se viene dichiarata **extern** si va a prendere l’accesso da una variabile dichiarata in un altro modulo (non static!)

Definire una libreria.h

#ifndef \_\_NOME\_\_

#define \_\_NOME\_\_

….

#endif

Per avere più istanze di un modulo bisogna creare una funzione che restituisce ed alloca una struttura(struct)e ne restituisce il puntatore. Sfruttando un tipo opaco, non definire la struct con typedef nel file c, ma farlo nel file .h esempio in file .h

typedef struct stack Stack; //nel file c c’è una struct che si chiama stack -> struct stack{…};

**make file**

esempio

prog1 : main.o sub.o

<TAB>cc -o prog1 main.o sub.o