qsort

La funzione qsort (stdlib.h)

Il linguaggio C possiede una sua implementazione del quicksort per ordinare un array di elementi generici

```
void qsort(void *buf, size_t num, size_t size,
           int (*compare)(const void *, const void *));
```

`void* buf`
: Puntatore all'array di elementi che si vuole ordinare

`size_t num`
: Numero di elementi dell'array

`size_t size`
: Dimensione (in byte) di un singolo elemento

`int(*compare)(const void*, const void *)`
: Funzione di comparazione tra elementi dell'array (questa funzione serve a fornire un criterio per l'ordinamento).

La funzione compare

qsort necessita di una funzione di comparazione tra elementi dell'array:

```
int compare (const void * a, const void * b)
```

che dati due puntatori a due elementi restituisce:

< 0	se il primo elemento è più piccolo del secondo
= 0	se sono uguali
> 0	se il primo elemento è più grande del secondo

Una funzione di questo tipo deve essere implementata ogni volta a seconda della tipologia degli elementi che vogliamo ordinare.

Esempio: interi

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

int compare (const void * a, const void * b){
    return ( *(int*)a - *(int*)b );
}

int main(){
    int i;
    int values[] = {10, 21, 1 , 7 , 24 , 9};
    int n=6;
    qsort(values,n,sizeof(int),compare);
    for(i=0;i<n;i++)
        printf("%d ", &values[i]);
}
```

OUTPUT: 1 7 9 10 21 24

Esempio: stringhe

```
#include<string.h>
#include<stdlib.h>
#include<stdio.h>
int compare(const void* a,const void* b){
      char **a1= (char **)a;
      char **b1= (char **)b;
      return strcmp(*a1,*b1);
}
int main(){
      int i;
      int n=10;
      char ** values = malloc(n*sizeof(char *));
      for(i=0; i < n; i++) {
            values[i] = malloc(101*sizeof(char));
            scanf(“%s”, values[i]);
      }
      qsort(values,n,sizeof(char *),compare);
      for(i=0;i<n;i++)
            printf("%s ",values[i]);
}
```