



UNIVERSITÀ DI PARMA

Puntatori a funzione

*Memoria est thesaurus omnium rerum et
custos*

De Oratore, Cicerone, I sec a.C.

- Puntatori a funzione
 - Sintassi
 - Definizione
 - Uso
- Funzioni
 - `qsort()`
 - `bsearch()`

SUMMARY



- Abbiamo definito le variabili come area di memoria con nome
 - Quindi hanno un indirizzo
- Ma tutto è in memoria
- Anche il codice
 - Quali porzioni di codice hanno un nome?
 - Le funzioni!

- Come lo ricavo?
- Nome della funzione senza ()
 - No operatore &

- Per le variabili:
 - Indirizzo
 - Tipo di dato puntato
- Per le funzioni
 - Indirizzo
 - Tipo di dato restituito
 - Elenco e tipi parametri formali

- Sintassi sempre basata su uso *
 - E con ()
- Esempi di puntatori:
 - `int (*a)(void);` // a funzione che restituisce int e non vuole argomenti
 - `int (*b)(int,int)` // a funzione che restituisce int e prende due int
 - `char (*c)(double*, float*);`

- Area di memoria non scrivibile né sensato leggerla
- Uso principale: callback ovvero codice che passo ad un altro pezzo di codice
 - Nel nostro caso → ad una funzione
- Esempi in C:
 - `qsort()` → ordinamento array generico
 - `bsearch()` → ricerca in array ordinati

- La funzione `qsort()` permette di ordinare un array generico
 - Quindi array di `int`, `float`, `double`, puntatori, `struct` di differenti tipi...
- Basi di tutti gli algoritmi di ordinamento:
 - Considero gli elementi dell'array a coppie
 - Confronto il loro contenuto
 - Nel caso li scambio di posizione

- Cosa serve per ordinare un array?
 - Per considerare gli elementi dell'array
 - Devo sapere di quanti elementi è composto
 - Passo alla funzione il numero di elementi dell'array
 - Per gli scambi
 - Devo sapere quanti byte occupa ciascun elemento
 - Passo alla funzione la dimensione in byte di un singolo elemento
 - Per il confronto
 - ??

- Se devo ordinare un array generico devo poter confrontare elementi di differenti tipologie
 - Ad esempio, il confronto di due scalari è differente dal confronto tra stringhe
 - Operatore ' $>$ ' vs `strcmp()`
- Non solo
 - Ordinamento crescente o decrescente
- Soluzione, chi invoca la `qsort()` deve anche fornire la parte di codice che realizza il confronto

- La qsort prende in ingresso:
 - Array da ordinare
 - Di quanti elementi è composto
 - Quanti byte occupa ciascun elemento
 - Un puntatore ad una callback che permette il confronto degli elementi

```
void qsort(void *base, size_t nmemb, size_t size,  
          int (*compar)(const void *, const void *));
```

- Binary Search ovvero ricerca binaria
- Ricerca all'interno di array ordinato
- Stessi parametri della qsort() + valore da ricercare
- Restituisce
 - Indirizzo elemento trovato
 - NULL se ricerca infruttuosa

```
void *bsearch(const void *key, const void *base,  
             size_t nmemb, size_t size,  
             int (*compar)(const void *, const void *));
```



UNIVERSITÀ DI PARMA

Puntatori a funzione



KEEP
CALM
IT'S
QUESTION
TIME

*Memoria est thesaurus omnium rerum et
custos*

De Oratore, Cicerone, I sec a.C.