UNIVERSITÀ DI PARMA

Puntatori a funzione

Memoria est thesaurus omnium rerum et custos

De Oratore, Cicerone, I sec a.C.

Sommario

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

Memoria

UNIVERSITÀ DI PARMA

- 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!

Indirizzo di una funzione

UNIVERSITÀ DI PARMA

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

Puntatore a funzione

UNIVERSITÀ DI PARMA

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

Puntatore a funzione

UNIVERSITÀ DI PARMA

- 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*);

Regole

- Sintassi complessa
- L'operatore array '[]' e l'operatore di funzione '()' hanno precedenza maggiore rispetto l'operatore '*'.
- Gli operatori '[]' e '()' vengono raggruppati da sinistra, mentre l'operatore '*' da destra.

Esempio intepretazione

UNIVERSITÀ DI PARMA

- int (*x[])() viene decifrata nel modo seguente:
  - x[] è un array;
  - (*x[]) è un array di puntatori;
  - (*x[])() è un array di puntatori a funzioni;
  - int (*x[])() è un array di puntatori a funzioni che restituiscono un intero.

Esempi

UNIVERSITÀ DI PARMA

- Non sempre puntatori a funzione
  - int *(*pap)[];
  - int *(*pfp)();
  - int (*fpa())[];
  - int (*fpf())();
  - int *(**ppf)();
  - int (*fpf())();

Uso puntatori a funzione

UNIVERSITÀ DI PARMA

- 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

qsort()

- 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

qsort()

- 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
    - ??

qsort()

UNIVERSITÀ DI PARMA

- 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 strcpy()
- Non solo
  - Ordinamento crescente o decrescente
- Soluzione, chi invoca la qsort() deve anche fornire la parte di codice che realizza il confronto

qsort()

UNIVERSITÀ DI PARMA

- 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 *));
```

bsearch()

- 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

Memoria est thesaurus omnium rerum et custos

De Oratore, Cicerone, I sec a.C.