

Architettura degli elaboratori

Laboratorio



Linguaggio C – Ripasso su aritmetica dei puntatori e stringhe

Dr. Luca Dariz <luca.dariz@unife.it>
Prof. Matteo Manzali <matteo.manzali@unife.it>

Array e puntatori

- L'aritmetica dei puntatori dipende dal tipo di dato puntato, la cui dimensione (e segno), per i tipi standard, può variare in base a
 - Architettura
 - su ARM char è uint8_t
 - Alcuni DSP hanno char=uint16_t
 - Ambiente di esecuzione
 - Unix x86_64 long=int64_t
 - Unix mips_32 long=int32_t
 - windows64 long=int32_t

cfr. <https://wiki.debian.org/ArchitectureSpecificsMemo> e <https://software.intel.com/en-us/articles/size-of-long-integer-type-on-different-architecture-and-os>
- Quando la dimensione di una variabile è importante, usare l'header *stdint.h* che definisce tipi di dato standard (C99)
 - uint8_t, int8_t, uint16_t, int16_t, etc

Array e puntatori

- Scrivere un breve programma C contenente due array `uint16_t a1[N]` e `uint16_t a2[M]`. Usando come unità di misura il byte, quindi convertendo tutti i puntatori in (`uint8_t*`):
 - 1) Stampare l'indirizzo di alcuni elementi di `a1`
 - 2) Sommare una costante all' dell'elemento `i`-esimo dell'array; successivamente sottrarre l'*indirizzo* dell'elemento `i`-esimo all'indirizzo dell'elemento `(i+1)`-esimo dello stesso array. Ripetere usando un puntatore ad un tipo di dato di dimensione diversa (es `uint32_t`)

Array e puntatori (2)

- Scrivere un breve programma C contenente due array *uint16_t a1[N]* e *uint16_t a2[M]*. Usando come unità di misura il byte, quindi convertendo tutti i puntatori in (*uint8_t**):
 - 3) Verificare cosa succede usando puntatori ad elementi di array diversi (suggerimento: non si dovrebbe mai fare perché ...).
 - 4) Scrivere una funzione che stampi tutti gli elementi di un array senza usare la notazione *a[i]* ma usando l'aritmetica dei puntatori
 - 5) Ripetere i punti precedenti usando una array di strutture.

Stringhe e puntatori

- Implementare i seguenti punti usando l'aritmetica dei puntatori invece della notazione *a[i]*. Per i primi 4 punti scrivere almeno due funzioni di test. Usare come unità di misura il byte, quindi convertire i puntatori a (uint8_t*).
- 1) Implementare la funzione *strchr()*
 - 2) Implementare la funzione *strlen()*
 - 3) Implementare la funzione *strncpy()*
 - 4) Implementare la funzione *strncmp()*
 - 5) Usando le funzioni dei punti precedenti e i file *travellers.txt* e *paths.txt* verificare che ogni commesso viaggiatore abbia completato il suo viaggio (ignorare le linee che iniziano con #)

Matrici e puntatori

- Implementare i seguenti punti usando l'aritmetica dei puntatori invece della notazione $a[i][j]$. Scrivere almeno due funzioni di test per ogni punto.

1) Implementare la funzione il cui prototipo è

```
void moltiplica_MxV(int *matrice, size_t dim_m,  
                    int *vettore, size_t dim_v,  
                    int *risultato);
```

2) Implementare la funzione il cui prototipo è

```
void moltiplica_MxM(int *matrice1, size_t dim_m1,  
                    int *matrice2, size_t dim_m2,  
                    size_t dim_comune,  
                    int *risultato);
```