Fondamenti di Informatica L-A

Esercitazione 6

Passaggio dei Parametri nelle Funzioni

Ordinamento di Vettori

Paolo Torroni

Università degli Studi di Bologna
Laurea in Ingegneria Elettronica e delle Telecomunicazioni

Anno Accademico 2007/2008

Ordinamento di un Vettore con il Naïve Sort

- Esistono molti algoritmi per ordinare un vettore.
- Un algoritmo molto semplice—ma anche relativamente poco efficiente—si chiama **naïve sort**.
- Funziona in questo modo (per un ordinamento crescente):

1. posiziona un indice i sul primo elemento del vettore;
2. fintantoché ci sono **almeno due elementi** da i alla fine del vettore, ripeti:
   - 2.1 **cerca il minimo** elemento tra $i + 1$ e la fine del vettore
     $\rightarrow$ ricorda l'**indice** j dell'elemento minimo trovato.
   - 2.2 se il minimo trovato (j-esimo elemento) è minore dell'i-esimo elemento, **scambia** i due elementi
   - 2.3 incrementa i
3. termina (una volta arrivati a un solo elemento, si ha un vettore "banalmente" ordinato);

△ 1. Ordinamento di un Vettore di Interi con Naïve Sort▽

- Si realizzi tramite funzioni, in C, un programma che ordina un vettore di interi usando l'algoritmo del naïve sort.
- Le funzioni da implementare sono:
  - una funzione per l'**inizializzazione** di un vettore V tramite lettura da input di **al più** N interi (N = dimensione fisica del vettore);
  - una funzione per la **visualizzazione** di un vettore V di M interi (M = dimensione logica);
  - una funzione per l'**ordinamento** di un vettore V di interi. Tale funzione faccia uso a sua volta di:
    - una funzione per **cercare il minimo** in un vettore;
    - una funzione per **scambiare due elementi** di V tra loro;

△ 2. Ordinamento di un Vettore di Interi con `qsort` ▽

- Nella libreria `stdlib.h` del C è già definita una funzione per l'ordinamento di un vettore
  - È la funzione `qsort`, che implementa il *quick sort*.
  - *Quick sort* è un algoritmo **molto più efficiente** di naïve sort.
  - Quindi, cerchiamo di usare `qsort` per l'ordinamento!
- `qsort` è in realtà una procedura (restituisce `void`):

```
void qsort ( void *V, int dim, int size,
             int ( * comp ) ( void *, void * ) );
```

  che riceve in input:
  - l'indirizzo di partenza del vettore da ordinare (`V`);
  - il numero di elementi del vettore da ordinare (`dim`);
  - la dimensione del singolo elemento (`size`);
  - il **nome di una funzione** (`comp`), da definire, che:
    - deve ricevere in input i **puntatori** a due elementi;
    - deve restituire zero/un numero negativo/un numero positivo a seconda che il il primo elemento sia $=/</>$ del secondo.

△ 2. Ordinamento di un Vettore di Interi con qsort▽

- Vediamo un esempio di utilizzo della procedura `qsort`.
  - Consideriamo un vettore di caratteri, `char K[5];`
  - Definiamo una funzione `compare` con l'interfaccia richiesta:
    `int compare( void *, void * );`
  - Ad esempio:

```
int compare( void* A, void *B ) {
    return *( char* )A-*( char* )B;
}
```

  - Si noti l'uso del **cast esplicito**:
    - serve a convertire A e B da `(void *)` a `(char *)`.
    - Infatti, gli elementi di `K` sono caratteri, ma `qsort` funziona con vettori i cui elementi possono essere di qualsiasi tipo.
  - A questo punto possiamo utilizzare `qsort` per ordinare `K`:
    `qsort( K, 5, sizeof( char ), compare );`
- Per esercizio, si modifichi il programma sviluppato al punto 1, in modo che il vettore venga ordinato con la funzione `qsort`.

△ 3. Ordinamento di un Vettore di Stringhe ▽

- Si modifichi il programma realizzato nel punto precedente, in modo che l'ordinamento venga svolto su un vettore di stringhe.
- L'ordinamento deve essere svolto in base alla tabella ASCII (ordinamento "lessicografico").
- Suggerimento: usare le funzioni della libreria `string.h`, in particolare:
  - `int strcmp( char *S1, char *S2 )`, che resitituisce un numero negativo se `S1` "è minore di" `S2` secondo l'ordine lessicografico.

△ 4. Ordinamento di un Vettore di Strutture ▽

- Si modifichi il programma realizzato nel punto precedente, in modo che l'ordinamento venga svolto su una tabella (vettore di record).
- I record definiscono degli individui:
  - Nome (stringa di 20 caratteri);
  - Cognome (stringa di 20 caratteri);
  - Anno_Nascita (intero).
- A seconda della scelta dell'utente, il programma deve essere in grado di visualizzare i record ordinati
  - secondo il campo Nome, il campo Cognome, o secondo il campo Anno_Nascita, e
  - in senso crescente o decrescente.
- Le scelte sono ortogonali, ovvero sono possibili tutte le combinazioni: nome/crescente, anno/decrescente, ...