Laboratorio di programmazione

24 gennaio 2005

Torri di Hanoi

Il gioco delle torri di Hanoi (anche detto delle torri di Brahma) è stato inventato nel 1883 dal matematico francese Edouard Lucas. L'obiettivo è spostare da un paletto ad un altro un certo numero di dischi forati di dimensione crescente infilati sul paletto e appoggiati l'uno sull'altro. Le regole del gioco sono che si può spostare solo il disco che è in cima ad una pila e non si deve mai appoggiare un disco di dimensione più grande sopra uno più piccolo; per aiutarsi, è possibile usare un terzo paletto come appoggio ausiliario. Secondo una leggenda (forse inventata dal matematico stesso) alcuni monaci di un tempio Indù sono costantemente impegnati a spostare sessantaquattro dischi secondo le regole del gioco; la leggenda dice che quando i monaci completeranno il lavoro, il mondo finirà!

Obiettivo dell'esercizio è scrivere un programma in grado di giocare al gioco delle torri di Hanoi, ossia di specificare la sequenza di mosse da effettuare per risolvere il rompicapo data l'altezza $n > 0$ della pila. Potete assumere che i tre paletti siano numerati da 0 a 2 e che gli n dischi siano inizialmente impilati dal più piccolo (in cima alla pila) al più grande (sotto tutta la pila) sul paletto 0 e vadano spostati al paletto 2. Per semplicità, le mosse sono date semplicemente dall'indicazione del paletto da e verso cui si deve muovere il disco.

Ad esempio, una soluzione per una pila di altezza 3 è data dalla seguente sequenza di mosse:

```
0 -> 2
0 -> 1
2 -> 1
0 -> 2
1 -> 0
1 -> 2
0 -> 2
```

Questo vuol dire che il disco più piccolo va spostato dal paletto 0 al paletto 2, quindi il disco mediano, ora in cima al paletto 0, va spostato al paletto 1; a questo punto il disco più piccolo (rimasto sul paletto 2) va rimesso sopra il mediano (ora sul paletto 1) e, finalmente, il disco più grande va spostato dal paletto 0 al paletto 2, nella sua posizione finale. Le restanti tre mosse, spotsano i due dischi rimanenti dal paletto 1 al paletto 2.

Scrivete una funzione `hanoi( int n, int from, int temp, int to );` che stampi le mosse per spostare `n` dischi dal paletto `from` al paletto `to` aiutandosi, se necessario, con il paletto ausiliario `temp`. Osservate che tale funzione, posto che siano state stampate le mosse per spostare $n-1$ dischi da `from` a `temp` (usando eventualmente `to` come paletto ausiliario) può stampare la mossa `from -> to` e quindi stampare le rimanenti mosse necessarie a spostare gli $n-1$ dischi da `temp` a `to` (usando eventualmente `from` come paletto ausiliario). Questa osservazione dovrebbe suggerirvi immediatamente una implementazione ricorsiva della funzione `hanoi`.

La fine del mondo. Modificate la funzione precedente perché restituisca soltanto il numero di mosse effettuate (invece che stamparle). Come cresce tale numero al crescere del numero di dischi? Per rendervi meglio conto del tasso di crescita, provate a considerare il logaritmo del numero di mosse, come cresce?

Vi sembra realistico che per spostare 64 dischi ci voglia un tempo pari alla durata del mondo?

Le torri in dettaglio. Supponendo ora di chiamare i dischi dal più grande al più piccolo con le lettere `A`, `B`, `C`, ..., scrivete ora una versione più dettagliata della funzione appena sviluppata che stampi ad ogni passo del gioco il contenuto di ogni paletto (usando una linea per ogni passo e separando il contenuto dei tre paletti con una virgola). Ad esempio, se la pila iniziale ha alteza 3, ed è quindi data da `ABC`, la nuova funzione deve scrivere le seguenti mosse

```
ABC, ,
AB, , C
A, B, C
A, BC,
, BC, A
C, B, A
C, , AB
, , ABC
```

Utilità che manipolano file

1. Scrivere un programma che, specificati i nomi di due file sulla linea di comando, determini se il loro contenuto è uguale, oppure no. Se il programma si chiamasse `uguali`, un esempio di esecuzione potrebbe essere il seguente:

   $./uguali uguali.c uguali.c
   `Il file uguali.c è identico a uguali.c`
   $./uguali uguali.c /etc/passwd
   `Il file uguali.c è diverso da /etc/passwd`

2. Scrivere un programma che, specificato il nome di un file sulla linea di comando, stampi il suo contenuto riga per riga, stampando ogni riga da sinistra a destra. Ad esempio, supponendo che un file contenga il testo

   ```
   ciao mamma
   come va
   diciamocelo!
   ```

   l'invocazione del programma su tale file produce l'output

   ```
   ammam oaic
   av emoc
   !olecomaicid
   ```

3. Scrivere un programma che, specificato il nome di un file sulla linea di comando, conti quanti caratteri contiene per ogni tipo tra quelli classificati dalle funzioni `is...` contenute nel file di intestazione `ctype.h`.

4. Scrivere un programma che, specificato il nome di un file e una stringa sulla linea di comando, stampi tutte e sole le linee del file che contengono la stringa specificata. Ad esempio, se invocato sul file dell'esercizio 2 e specificando la stringa pari a `cia`, produce l'output

   ```
   ciao mamma
   diciamocelo!
   ```

5. Scrivere un programma che, specificato il nome di un file sulla linea di comando, stampi il suo contenuto riga per riga, permutando a caso i caratteri di ciascuna riga. Ad esempio, se invocato sul file dell'esercizio 2 produce l'output

   ```
   ciam oaamm
   cvem oa
   diciaco!moel
   ```

6. Scrivere un programma che, specificato il nome di un file sulla linea di comando, stampi una permutazione casuale delle sue linee. Ad esempio, se invocato sul file dell'esercizio 2 produce l'output

   ```
   come va
   ciao mamma
   diciamocelo!
   ```