Laboratorio di programmazione

9 febbraio 2005

Rubrica del telefono

Scopo di questo esercizio è implementare una rudimentale rubrica del telefono.

Il programma deve inizialmente leggere da un file di testo (indicato come argomento sulla linea di comando) un elenco di coppie cognome/numero di telefono e in seguito, dato un cognome (letto dallo standard input), riportare il numero di telefono corrispondente.

Il formato del file di testo è esemplificato dalla seguente rubrica

```
3
Santini 0250316259
Rossi 023344567
BusH 0015552349876
```

come notate, sulla prima riga c'è un numero intero, che rappresenta il numero di righe seguenti, ciascuna delle quali contiene un cognome, uno spazio e quindi un numero di telefono. Osservate che non c'è un limite prefissato al numero di righe della rubrica, ovviamente se tale numero fosse troppo grande per la memoria a disposizione, il programma terminerebbe segnalando un errore. Potete invece fare affidamento sul fatto che cognomi e numeri di telefono non siano mai più lunghi di 256 caratteri.

Supponendo che tale rubrica sia nel file `rubrica.txt` e che il programma che avete sviluppato si chiami `rubrica`, un esempio di esecuzione è il seguente

```
$./rubrica rubrica.txt
santini
Il numero di Santini è 0250316259
Paperino
Il numero di Paperino non è presente nella rubrica
Bush
il numero di BusH è 0015552349876
```

osservate che il programma deve memorizzare i cognomi esattamente come scritti nella rubrica, ma effettuare le ricerche a meno di maisucole/minuscole.

Suggerimenti. Rappresentate le coppie cognome/numero di telefono con una struttura con due membri stringa. Scrivete una funzione che legga il file della rubrica in un array di tali strutture, attenzione: dovrete usare `malloc` per allocare in modo dinamico la memoria necessaria, a senconda del numero di righe indicato nel file della rubrica. Dopo aver letto la rubrica, ordinatela alfabeticamente con la funzione `qsort` e quindi usate `bsearch` per effettuare la ricerca del cognome. Per ottenere che maiuscole/minuscole non siano rilevanti, nella chiamata alle precedenti funzioni, potete passare (invece di `strcmp`) la seguente funzione per il confronto tra stringhe:

```
int compara( char *x, char *y )
{
        while ( *x && *y && toupper( *x ) == toupper( *y ) ) { x++; y++; }
        return toupper( *x ) - toupper( *y );
}
```

Da CSV a HTML

Scopo dell'esercizio è scrivere un programma che trasformi un file in formato CSV (*comma separated values*) in una tabella HTML.

Per prima cosa, dovete scrivere una libreria (collezione di funzioni, racchiuse in un unico file sorgente) per la lettura di un file in formato CSV. Per semplicità, potete assumere che un file in tale formato sia dato da una sequenza di linee separate dal carattere `\n` contenenti ciascuna dei "campi" separati dal carattere `,` (virgola); il numero massimo di caratteri per linea è 1024 e nei campi non compare mai il carattere `,`. Il file `esempio.csv` seguente costituisce un esempio di tale formato:

```
qui va tutto, molto, bene
ciao, 13, mondo!
```

La libreria deve comprendere le tre funzioni:

1. `char *cvsgetline( FILE *in )`, che legga la prossima linea del file corrispondente a `in` e la suddivida nei campi che la compongono (restituendo un puntatore alla linea appena letta);
2. la funzione `char *cvsfield( int n )`, che restituisca un puntatore alla stringa corrispondente all'`n`-esimo campo dell'ultima linea letta (i campi sono numerati da 0);
3. la funzione `int cvsnfields( void )`, che restituisca il numero complessivo di campi dell'ultima linea letta.

Ad esempio, se invocata sul file precedente, dopo la seconda chiamata di `cvsgetline`, la chiamata `cvsfiled( 2 )` deve restituire un puntatore alla stringa `mondo!`.

Utilizzando la libreria appena realizzata, scrivete un programma che, dato come parametro sulla linea di comando il nome di un file in fomato CSV, a meno dell'estensione `.csv`, generi un file in formato HTML, con lo stesso nome e estensione `.html`, contenente una tabella (elemento `TABLE`) avente una riga (elemento `TR`) per ogni linea del file CSV e, su ogni riga, una colonna (elemento `TD`) per ogni campo della linea.

Ad esempio, se invocato con argomento `esempio`, il programma leggerà il file `esempio.csv` qui sopra descritto e scriverà il file `esempio.html` con il seguente contenuto:

```
<HTML>
<HEAD><TITLE>esempio</TITLE></HEAD>
<BODY>
<TABLE>
<TR><TD>qui va tutto</TD><TD>molto</TD><TD>bene</TD></TR>
<TR><TD>ciao</TD><TD>13</TD><TD>mondo!</TD></TR>
</TABLE>
</BODY>
</HTML>
```

Suggerimento. Per la lettura del file CSV e la suddivisione in campi potete usare la funzione di libreria `strtok`.