Corso di laurea in fisica
Anno Accademico 2020/2021
Laboratorio di informatica 1

Esercitazione 6

Nota 1 La libreria *standard* del *C* comprende diverse funzioni per il trattamento di sequenze di caratteri (comunemente dette *stringhe*). Il programma contenuto nel file `semparole.c` esemplifica l'impiego di alcune di queste funzioni, qui brevemente descritte. Ciascuna assume che le stringhe siano memorizzate in un vettore (o in un'area di memoria allocata dinamicamente), e terminate dal carattere speciale '\0'. A eccezione delle prime due, il loro impiego richiede l'inclusione del file `string.h`.

- `int atoi(char * s)` – converte la stringa `s` nel corrispondente valore intero.
- `double atof(char * s)` – converte la stringa `s` nel corrispondente valore decimale.
- `size_t strlen(char * s)` – restituisce la lunghezza della stringa `s` (potete considerare il tipo `size_t` come sostanzialmente equivalente al tipo dei numeri interi non negativi).
- `char * strcpy(char * s, char * t)` – copia la stringa `t` in `s`; `s`, considerato come puntatore, deve puntare a un'area di memoria già allocata, e sufficiente a contenere l'intera stringa `t`, compreso il segnale di fine stringa, '\0'. Per il momento, non preoccupatevi del valore restituito dalla funzione.
- `char * strcat(char * s, char * t)` – concatena la stringa `t` in coda alla stringa `s`. Valgono le stesse considerazioni fatte per `strcpy`.
- `int strcmp(char * s, char * t)` – confronta le due stringhe `s` e `t` secondo l'ordine alfabetico; restituisce un valore negativo se `s` precede `t`, zero se le due stringhe sono uguali, e un valore positivo se `t` precede `s`.

Nota 2 Argomenti sulla riga di comando: si vedano il file `parole.c` e gli *Appunti su puntatori, stringhe e altro*.

Nota 3 Finora i programmi che abbiamo scritto hanno acquisito i loro dati dallo *standard input* e hanno scritto i risultati sullo *standard output*, con la possibilità di deviare questi due flussi di dati su *file*.

A volte, però, un programma deve operare su diversi *file*, e in questi casi il meccanismo di deviazione non è sufficiente.

La libreria standard del C mette a disposizione diverse funzioni utili in questi casi, e un tipo di dati specifico per rappresentare un *file*. Questo tipo è di fatto una struttura (`struct`), sui dettagli della quale non ci soffermiamo qui. Un programma che deve manipolare un *file*, e le funzioni della libreria standard, accedono a questa struttura attraverso puntatori. L'uso semplice è illustrato nel programma `opfile.c`.

1 Un *file* di dati è composto di righe, ciascuna delle quali contiene (1) una lettera, (2) un numero intero, (3) un numero reale, separati fra loro da un singolo spazio. Le righe sono disposte in modo che i numeri interi siano

ordinati in senso crescente. I valori nei tre campi possono essere ripetuti in righe diverse.

Dovete scrivere un programma che acquisisce due argomenti di tipo intero, K e n, dalla riga di comando; legge e memorizza le prime K righe del *file* di dati; stabilisce se il numero n è presente nella seconda posizione di una riga; stampa l'intero contenuto di quella riga in caso di risposta positiva, e un messaggio in caso di risposta negativa.

2 Scrivete un programma che legge dallo *standard input* una sequenza di caratteri terminata dal segnale `EOF` e la riproduce sullo *standard output* conservando solo i caratteri 'parentesi aperta' e 'parentesi chiusa', di qualsiasi tipo (tonde, graffe e quadre). Se, ad esempio, la sequenza in ingresso fosse

```
{
  while ( v[i] < h[i-1] ) {
      v[f(i)] = sqrt(h[i]);
      i++;
  }
}
```

il programma dovrebbe stampare la sequenza `{([][]){[()]([])}}`. Il programma deve anche determinare il massimo livello di profondità delle parentesi, misurato come segue: una parentesi al livello esterno ha profondità 0; una parentesi all'interno di una parentesi di profondità n ha profondità $n+1$. La sequenza dell'esempio ha profondità 3.

3 Modificate il programma dell'esercizio 2 in modo che legga la sequenza da un file di nome `testo.c`, e stampi la sequenza risultante in un file di nome `p-testo.c`.

Modificate ulteriormente il programma in modo che possa ricevere i nomi dei due file sulla riga di comando. Il programma deve leggere dallo *standard input* e stampare sullo *standard output* se, all'avvio dell'esecuzione, non sono presenti tutti e due gli argomenti.