TECNICHE DI PROGRAMMAZIONE
Esercitazione di Laboratorio 7

Obiettivi
- Risolvere problemi di elaborazione testi e di verifica/selezione, iterativi, utilizzando vettori e matrici (*Dal problema al programma: Cap. 4*),

Contenuti tecnici
- Basi di Input Output
- Utilizzo di funzioni
- Costrutti condizionali e iterativi
- Manipolazioni elementari di vettori e matrici (di int e float)
- Funzioni per gestione di stringhe

Da risolvere durante il laboratorio oppure prima/dopo il laboratorio stesso

Esercizio 1.

Competenze: selezione/filtro dati mediante ricerca in tabelle di nomi/stringhe, tipi enumerativi
Categoria: problemi di selezione (Dal problema al programma: 4.4.2 e 5.5.2)

Azienda di trasporti

Un'azienda di trasporti urbani, traccia i propri automezzi in un file di log (file testuale di nome `corse.txt`).
Il file è organizzato come segue:
- sulla prima riga, un intero positivo indica il numero di successive righe del file stesso (al più `1000`)
- le righe successive riportano le informazioni sulle tratte, una per riga, con formato:

```
<codice_tratta><partenza><destinazione><data><ora_partenza><ora_arrivo><ritardo>
```

Tutte le stringhe sono lunghe al massimo `30` caratteri. Il ritardo è un numero intero, eventualmente nullo, che rappresenta i minuti di ritardo accumulati dalla corsa.
Si scriva un programma C che, letto il file e acquisitone il contenuto in una oppurtuna struttura dati, risponda alle seguenti interrogazioni (presentate mediante un menu):
1. elencare tutte le corse partite in un certo intervallo di date
2. elencare tutti le corse partite da una certa fermata (partenza)
3. elencare tutti le corse aventi una data destinazione (capolinea)
4. elencare tutte le corse che hanno raggiunto la destinazione in ritardo, in un certo intervallo di date
5. elencare il ritardo complessivo accumulato dalle corse identificate da un certo codice di tratta

Le interrogazioni di cui sopra siano gestite mediante menu di comandi (si veda il paragrafo 4.4.1, Dal problema al programma). Ogni comando consiste di una parola tra "`date`", "`partenza`", "`capolinea`", "`ritardo`", "`ritardo_tot`" e "`fine`", eventualmente seguita sulla stessa riga da altre informazioni, ad esempio due date per "`date`", una fermata di partenza per "`partenza`", etc.

Si utilizzi la strategia di codifica dei comandi mediante tipo `enum comando_e`, contenente i simboli `r_date`, `r_partenza`, `r_capolinea`, `r_ritardo`, `r_ritardo_tot`, `r_fine`, che consente menu basati su `switch-case`.
Si consiglia di:

- realizzare una funzione `leggiComando` che, acquisito in modo opportuno il comando, ritorni il corrispondente valore di tipo `comando_e`
- realizzare una funzione `menuParola` che, ricevuti tra i parametri la tabella, la dimensione (cioè il numero effettivo di dati) della tabella e il tipo di comando, gestisca mediante menu l'acquisizione delle informazioni aggiuntive necessarie per quel comando e la chiamata di un'opportuna funzione di selezione e stampa dei dati selezionati.

Si noti che la data di una corsa non subisce variazioni a causa del ritardo.

Esercizio 2.
Competenze: lettura/scrittura di file, manipolazioni di testi, ricerca in tabelle di nomi/stringhe
Categoria: problemi di elaborazione testi mediante stringhe (Dal problema al programma: 4.4.3)

Occorrenze di parole

Si scriva un programma in grado di localizzare, all'interno di un generico testo, le occorrenze di ogni parola che contiene una certa sequenza di caratteri. Le "parole" (nel testo) contengono unicamente caratteri alfanumerici (identificati dalla funzione `isalnum` di `ctype.h`), più parole sono separate da spazi o segni di punteggiatura (identificati dalle funzioni `isspace` e `ispunct`).

Il programma riceve in input:

- il file `sequenze.txt`: sulla prima riga contiene il numero totale di sequenze, al più `20`, sulle righe successive, una per riga, le sequenze da ricercare. La lunghezza delle singole sequenze è limitata a massimo `5` caratteri. Si trascuri la differenza tra maiuscole e minuscole
- il file `testo.txt`: contiene il testo. Il numero di righe non è noto a priori. Si assuma che la lunghezza di ogni riga sia al più pari a `200` caratteri. Si assuma inoltre che nessuna parola del testo sia più lunga di `25` caratteri.

Il programma deve visualizzare, per ognuna delle "sequenze", quali parole la contengono e dove si trovano nel file. La posizione delle parole viene data contando le parole stesse a partire dall'inizio del testo (si usi 1 per la prima parole, 2 per la seconda e così via). Ai fini dell'esercizio ci si limiti a identificare e visualizzare solamente le prime `10` occorrenze per ogni sequenza.

Esempio
file `sequenze.txt`:

```
4
no
Al
per
s
```

file `testo.txt`:

```
Non sempre si capisce un esercizio alla prima lettura, ma prestando
attenzione al testo e all'esempio non dovrebbe essere impossibile
```

scrivere codice funzionante nonostante i dubbi iniziali. Se ancora non si capisce, allora basta chiedere all'esercitatore di turno.

La sequenza no è contenuta in Non (parola in posizione 1 nel testo), non (posizione 18), nonostante (posizione 25), non (posizione 31) e turno (posizione 40).