TECNICHE DI PROGRAMMAZIONE, A.A. 2020/2021

Esercitazione di Laboratorio 6

Valutazione: **l'esercizio 1** sarà oggetto di valutazione.
Scadenza: caricamento di quanto valutato - entro le 23:59 del 6/5/2021: andranno caricati insieme i laboratori 4, 5 e 6.

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. (Esercizio da consegnare per il bonus-laboratorio)

Competenze: selezione/filtro dati mediante ricerca in tabelle di nomi/stringhe, tipi enumerativi
Categoria: problemi di selezione (Dal problema al programma: 3.4.2 e 4.5.2)

Azienda di trasporti

Un'azienda di trasporto urbano traccia i propri automezzi in un file di log (file testuale di nome `log.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>
```

`<codice_tratta>`, `<partenza>` e `<destinazione>` sono stringhe lunghe al massimo `30` caratteri che rappresentano rispettivamente il codice, la partenza e la destinazione della tratta;
`<data>` è la data della tratta nel formato `yyyy/mm/dd`;
`<ora_partenza>`, `<ora_arrivo>` indicano l'ora di partenza e arrivo della tratta nel formato `hh:mm:ss`;
`<ritardo>` è un numero intero `>=0` che rappresenta i minuti di ritardo accumulati dalla corsa (Si assuma che il ritardo non influisca sulla data della tratta).

Esempio del file `log.txt`:

```
6
GTT001 Braccini Porta_Nuova 2018/10/10 18:50:00 19:07:25 1
GTT001 Braccini Porta_Nuova 2018/12/10 19:50:00 20:06:00 1
GTT002 Politecnico XVIII_Dicembre 2018/10/10 10:01:23 10:12:08 4
```

```
GTT003 Einaudi Cso_Trapani 2018/09/10 14:11:23 14:38:23 2
GTT004 Marmolada Sebastopoli 2018/11/10 00:01:02 00:12:00 3
GTT002 Politecnico Piazza_Statuto 2018/11/10 23:11:59 23:20:07 0
```

Dopo aver letto il file e acquisito il contenuto in una opportuna struttura dati, si scriva un programma in C che fornisca un menu all'utente per scegliere tra una delle seguenti operazioni:

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 specifica 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
6. terminare il programma

Per selezionare una voce del menu l'utente deve inserire uno specifico comando da tastiera (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. Per esempio, "`date`" deve essere seguito da due date nel formato `yyyy/mm/dd`, "`partenza`" deve essere seguito dal nome della fermata di partenza, etc.

Per gestire la selezione da menu si utilizzi la strategia di codifica dei comandi simile a quella riportata nella slide 30 in C3 Capitolo 4 – Problem solving con uso di vettori – parte 2. Si definisca un nuovo tipo `enum comando_e`, contenente i simboli `r_date`, `r_partenza`, `r_capolinea`, `r_ritardo`, `r_ritardo_tot`, `r_fine`. Si codifichi ogni comando nel corrispondente simbolo `comando_e` che consente facilmente di gestire menu basati su `switch-case`.

Si consiglia di:

- realizzare una funzione `leggiComando` che acquisisca la prima parola del comando da tastiera e ritorni il corrispondente valore di tipo `comando_e`
- realizzare una funzione `menuParola` che riceva come parametri la struttura dati dove sono memorizzate le tratte ed il numero di tratte memorizzate e gestisca mediante menu l'acquisizione dei comandi (la prima parola e le informazioni aggiuntive necessarie) così come la chiamata di un'opportuna funzione di selezione e stampa dei dati selezionati.

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 un numero intero che rappresenta il numero totale di sequenze, al più `20`, mentre 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` (posizione 1)
`non` (posizione 18)
`nonostante` (posizione 25)
`non` (posizione 31)
`turno` (posizione 40)
…
…