Esercizi 23-11-06

Riccardo Gennaro

November 2023

Esercizio 1

Scrivere un programma che gestisca l'anagrafica di un set di utenti.

In particolare, il programma deve poter:

- aggiungere un l'anagrafica di un utente;
- stampare le anagrafiche ordinate per nome;
- stampare le anagrafiche ordinate per cognome;
- cercare un anagrafica per nome (corrispondenza esatta);
- cercare un anagrafica per cognome (corrispondenza esatta);

Con anagrafica si intende:

- nome;
- cognome;
- indirizzo, composto da:
  - via;
  - civico;
  - comune;
  - CAP;
  - provincia;

Descrivere l'anagrafica tramite *struct*.
Il programma prevede delle dimensioni massime per gli array.

L'utente deve poter scegliere una delle opzioni sopra. Il programma termina quando l'utente inserisce la stringa '`exit`' nel menù. Scrivere il programma rispettando i principi della programmazione su file multipli.

Esercizio 2

Scrivere un programma che, dato un file di input contente delle parole, generi un secondo file in output che contenga le stesse parole, ma in ordine inverso rispetto al file iniziale. Per essere copiata, una parola deve essere composta da un numero di caratteri **pari**.

Il programma dovrà accettare due parametri da riga di comando: il nome del file in input e il nome del file su cui effettuare l'output, in questo ordine.

Il programma dovrà anche implementare dei controlli sul numero di argomenti passati da riga di comando e sull'apertura dei file (in caso di un file di input non esistente).

Procediamo ora con un esempio. Dato in input il file input_A, contenente le seguenti parole:

```
Impara a risolvere tutti i problemi che sono stati risolti.  Richard
Feynman
```

se l'eseguibile è a.out, allora il comando `./a.out input_A output` genererà un file chiamato `output` che conterrà le seguenti parole:

```
risolti.  sono problemi Impara
```

Note

- Per semplicità, considerate come una *parola* qualsiasi stringa di caratteri compresa tra due spazi bianchi (e/o separatori di tabulazione, nuova linea e fine file). Quindi, sono da considerarsi parole anche stringhe come `"andato:"` o `"!esempio"`.
- Il file in input può contenere al massimo 10000 parole e ognuna di queste parole è formata da al massimo 100 caratteri.
- Non è consentito l'utilizzo di librerie diverse da `<iostream>` ed `<fstream>` pena **l'annullamento dell'esercizio**.
- E' consentito definire ed implementare funzioni ausiliarie che possano aiutarvi nella soluzione del problema.

Esercizio 3

Scrivere un programma che calcoli il *prodotto matriciale* di due matrici di numeri interi. Se le matrici non sono compatibili per il prodotto stampare un messaggio di errore.
La dimensione delle matrici deve essere specificata dall'utente. Il prodotto è calcolato con la funzione seguente:

```
1    int** compute_prod(int** m1, int r1, int c1, int** m2, int c2);
```

Dove

- `m1` è la prima matrice;
- `r1` è il numero di righe di m1;
- `c1` è il numero di colonne di m1;
- `m2` è la seconda matrice;
- `c2` è il numero di colonne di m2;

NOTA: la complessità di `compute_prod(...)` deve essere pari o inferiore a $O(n^3)$. Ciò significa che potete implementare la funzione con l'algoritmo che si basa sulla definizione del prodotto fra matrici, oppure, se siete dei pazzi maniaci, potete implementare l'algoritmo di Strassen.