Esercizi 13-11-22

Riccardo Gennaro

November 2023

NOTA

Tutti gli esercizi vanno svolti rispettando le buone pratiche di allocazione dinamica. Potete implementare su più file per esercitarvi.

Esercizio 1 (Coda a priorità)

Un messaggio viene rappresentato mediante una struct

```
struct messaggio {
   char testo[10000];
   int priorita;
}
```

dove priorità ha un range crescente da 1 a 10. Scrivere una struttura dati "coda a priorita" in cui e' possibile inserire messaggi, da cui vengono estratti in modo FIFO per classi di priorita': (prima quelli a priorita' 10, poi quelli a priorita' 9,...)

Esercizio 1 bis (Anagrafica)

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 (Correzione testo)

Scrivere un programma, nel file esercizio1.cc, che, presi come argomenti del main i nomi di due file, copi il primo file nel secondo correggendone la sintassi e generando in tal modo un testo "corretto" secondo le seguenti regole:

- la prima parola del testo deve iniziare con una lettera maiuscola;
- tutte le parole che seguono i seguenti caratteri: ".", "?" e "!", devono iniziare con una lettera maiuscola.

Se ad esempio l'eseguibile è a.out, il comando `./a.out testo testocorretto` creerà un nuovo file di nome `testocorretto` e vi copierà il contenuto del file dato testo, modificando le parole quando queste non verificano le regole descritte sopra. Nelle figure 1 e 2 un esempio di file `testo` e `testocorretto`.

Esempio di input:

```
filastrocca delle parole:
Fatevi avanti! chi ne vuole?
di parole ho la testa piena,
con dentro la 'luna' e la 'balena'.
ci sono parole per gli amici:
Buon giorno, Buon anno, Siate felici!
parole belle e parole buone;
parole per ogni sorta di persone.
di G. Rodari.
```

Relativo output:

```
Filastrocca delle parole:
Fatevi avanti! Chi ne vuole?
Di parole ho la testa piena,
con dentro la 'luna' e la 'balena'.
Ci sono parole per gli amici:
Buon giorno, Buon anno, Siate felici!
Parole belle e parole buone;
parole per ogni sorta di persone.
Di G. Rodari.
```

NOTA 1: Per semplicità si assuma che il testo contenuto nel primo file inizi con un carattere alfabetico, non contenga "..." e che ".", "?" e "!" siano sempre preceduti da una parola e seguiti da uno spazio.

NOTA 2: Per semplicità si assuma che ogni parola contenuta nel testo del primo file abbia al massimo lunghezza 30 caratteri.

NOTA 3: E' ammesso l'uso della funzione strlen della libreria `cstring`, non è ammesso l'uso di altre funzioni di libreria, in particolare della funzione toupper.

NOTA 4: il programma deve potenzialmente funzionare con ogni possibile codifica dei caratteri secondo le regole di tali codifiche viste a lezione (quindi non solo ASCII). Per realizzare la conversione da caratteri minuscoli in maiuscoli, è vietato l'uso di tabelle o di 26 if o switch-case, uno per ogni carattere.

Esercizio 3 (Matrici)

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\textunderscore 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.