Prova scritta del 17/6/2009

Strutturare adeguatamente i programmi ed evidenziarne la strutturazione mediante indentazione. Inserire anche adeguati commenti

1) (a) Sia `d` il tipo di una struttura dati `struct` costituita da due campi, `alfa` di tipo stringa (di lunghezza massima *32*), e `val` di tipo reale. Realizzare una funzione di nome `aggiungi_o_aggiorna` che, presi come suoi parametri un array `A` di strutture di tipo `d`, il numero `n` di elementi in `A`, ed una struttura `x` di tipo `d`, ricerca un elemento di `A` il cui campo `alfa` sia uguale a quello di `x`; se lo trova, aggiorna il corrispondente campo `val`, sommando ad esso il valore del campo `val` di `x`; altrimenti, aggiunge `x` come ultimo elemento di `A`. La funzione restituisce come suo risultato la nuova dimensione di `A`.

(b) Descrivere la funzione `aggiungi_o_aggiorna` anche tramite diagramma di flusso.

2) Sia `"config.txt"` il nome di un file di testo contenente una sequenza di nomi di altri file, contenenti ciascuno una o più coppie di valori, in cui il primo valore è di tipo stringa ed il secondo (separato dal primo da almeno uno spazio) è di tipo numero reale. Scrivere un programma principale che legge tutte le coppie presenti in tutti i file elencati in `"config.txt"`, e per ogni coppia letta, provvede ad inserirla in un array `dati` di strutture di tipo `d`, utilizzando (obbligatoriamente) la funzione `aggiungi_o_aggiorna`, a cui viene passato, oltre all'array `dati`, la coppia letta dal file opportunamente memorizzata in una struttura di tipo `d`. Terminata la lettura dei file il programma memorizza l'intero array `dati` su un nuovo file, di nome `"risultati.txt"`.
N.B. Si assuma che i nomi nel file `"config.txt"` siano separati tra loro da (esattamente) un "a capo" e che abbiano lunghezza massima di *128* caratteri. Inoltre, si assuma che il programma possa leggere al massimo *1000* elementi nell'array dati: nel caso si superi tale numero il programma interrompe la lettura da file e continua con le altre operazioni. Infine, nel caso un file elencato in `"config.txt"` non risulti presente il programma dà opportuno messaggio e passa a leggere il file successivo.

3) Scrivere un programma principale che: (1) legge da standard input una sequenza di *100* numeri interi e li memorizza, uno alla volta, in una matrice `M` di *10x10* elementi; (2) chiede all'utente di fornire un numero intero `k`; (3) controlla se in `M` c'è (almeno) una riga contenente una sottosequenza di (almeno) `k` zero consecutivi e, in caso affermativo, termina stampando le coordinate del primo elemento della sottosequenza trovata. Il programma quindi ripete dal punto (2) finché non viene fornito un valore di $k \leq 0$. N.B. Nel caso in cui `k` risulti > 10 il programma chiede l'immissione di un nuovo valore di `k`, dando opportuno messaggio all'utente.
SUGG.: Si definisca una variabile contatore, inizializzata a 0, che viene incrementata di 1 ogni volta che l'elemento della matrice è 0 e riportata a 0 ogni volta che l'elemento della matrice è diverso da 0 o quando si arriva al termine di una riga.

N.B. Negli esercizi (1) e (2) si utilizzino soltanto **stringhe tipo C** (= array di caratteri terminati da `'\0'`)..