Prova scritta del 27/2/2012

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

1) Sia `S` il tipo di una struttura dati (`struct`) costituita da due campi, `val` e `occ`, entrambi di tipo intero. **(a)** Realizzare una funzione `void` di nome `aggiorna` che, presi come suoi parametri un array `R` di elementi di tipo `S`, il numero `m` di elementi in `R`, e un intero `x`, controlla se `R` contiene già un elemento il cui campo `val` coincide con `x`; in caso affermativo, incrementa di `1` il campo `occ` dell'elemento trovato; altrimenti, aggiunge un nuovo elemento, con `val` uguale a `x` e `occ` uguale a `1`, come ultimo elemento di `R`. In quest'ultimo caso, la funzione restituisce in `m` il numero aggiornato di elementi di `R`. N.B. La funzione deve essere necessariamente di tipo `void`.
Es.: dati `R={{7,2},{5,3},{4,2}}` e `m=3`, con `aggiorna(R,m,5)` si ottiene `R={{7,2},{5,4},{4,2}}`, `m=3`, mentre con `aggiorna(R,m,8)` si ottiene `R={{7,2},{5,3},{4,2},{8,1}}`, `m=4`.
(b) Descrivere la funzione `aggiorna` anche tramite diagramma di flusso.

2) Scrivere un programma principale che legge da un file, il cui nome è fornito dall'utente tramite std input, una sequenza di numeri interi e crea un array `dati` di elementi di tipo `S` (max. *1000*), in cui il campo `occ` di ciascun elemento indica quante volte il numero nel campo `val` di quell'elemento compare nel file (= numero di "occorrenze"). L'array `dati` deve essere riempito utilizzando obbligatoriamente la funzione `aggiorna`. Al termine della lettura dal file, il programma provvede a memorizzare l'array `dati` su un file il cui nome è ottenuto dal file di input preceduto dalla stringa `"statistiche_per_"`. Il programma quindi chiede all'utente se vuol continuare e, in caso affermativo, riprende dall'inizio, richiedendo all'utente il nome di un nuovo file di input.
N.B. Il nome del file di input può avere lunghezza massima di *80* caratteri e può contenere "spazi". Utilizzare soltanto stringhe *"tipo C"*. Controllare l'apertura del file e la situazione in cui l'array `dati` risulti "pieno". Controllare anche che i dati del file di input siano letti correttamente (in caso di errore terminare il ciclo di input e dare opportuno messaggio). Es.:

File di input `prova1.txt`**:**

```
7 5 7 4 5 5 4 5 8
```

File di output `statistiche_per_prova1.txt`:

```
7  2
5  4
4  2
8  1
```

Interazione con l'utente (input sottolineato):

```
Dai il nome del file: prova1.txt
Risultati scritti su file statistiche_per_prova1.txt
Vuoi continuare? ('n' per smettere): s
Dai il nome del file: prova2.txt
Risultati scritti su file statistiche_per_prova2.txt
Vuoi continuare? ('n' per smettere): n
```

3) Realizzare una funzione `void` di nome `ordina` che, presi come suoi parametri un array `A` di elementi di tipo `S` e il numero `n` di elementi in `A`, ordina `A` in senso decrescente rispetto ai valori del campo `occ`. Mostrare anche dove e come inserire la chiamata alla funzione `ordina` nel programma realizzato al punto 2 in modo che i risultati memorizzati sul file di output risultino ordinati. Es.:

File di output `statistiche_per_prova1.txt` ***ordinato***:

```
5  4
7  2
4  2
8  1
```