Esercizi di programmazione in linguaggio C – Ordinamento

Nota: memorizzare i programmi in cartelle separate. Ogni cartella deve contenere, oltre al codice sorgente e al file eseguibile, gli eventuali file di input indicati nel testo e file di output prodotti dal programma. I file di input devono essere generati con un apposito editor prima della scrittura del codice.

1. **ordina-float** – Scrivere un programma che inizializzi staticamente un vettore di 10 elementi *float* e lo ordini in modo crescente memorizzando il risultato in un altro vettore.

2. **ordina-char** – Scrivere un programma che inizializzi staticamente un vettore di 10 elementi char e lo ordini in modo **decrescente** memorizzando il risultato in un altro vettore.

3. **ordina-file** – Il file di testo `numeri.txt` contiene un certo numero di valori interi senza segno separati da *newline*. Il numero di valori non è noto a priori, ma non può essere maggiore di 1000. Scrivere un programma che, dopo aver memorizzato il contenuto del file in un apposito vettore, visualizzi i numeri in ordine crescente (un numero per riga).

4. **ordina-file2** – Ripetere l'esercizio precedente riportando i numeri ordinati nel file di testo `ordinati.txt` (su righe distinte) anziché su monitor.

5. **temperature** – Il file di testo `temperature.txt` contiene un certo numero di temperature rilevate in una data località. Le temperature sono approssimate a una cifra decimale e memorizzate in un'unica riga. Scrivere un programma che, dopo aver letto le temperature dal file, visualizzi le seguenti informazioni: elenco delle temperature secondo l'ordine con cui sono registrate nel file; elenco delle temperature ordinate in modo crescente; valori delle temperature minima e massima (suggerimento: si usi il vettore ordinato per determinare rapidamente tali valori).

6. **ordina-stringhe** ★ Scrivere un programma che ordini in modo crescente e con un algoritmo *in-place* un array di cinque stringhe inizializzato staticamente, visualizzando il risultato sullo schermo. Si assuma che ogni stringa abbia una lunghezza non superiore a 8 caratteri.

7. **ordina-date** – Dopo aver definito una struttura adatta a memorizzare una data composta da giorno, mese (numerico) e anno, scrivere un programma che memorizzi un elenco di date in un apposito array (mediante inizializzazione statica) e stampi l'elenco in ordine cronologico.

8. **ordina-lettere** – Scrivere un programma che richieda l'inserimento da tastiera di una stringa e mostri le lettere che la compongono in ordine alfabetico. Per esempio, inserendo la stringa `CERCHIO`, il programma deve riportare la sequenza `CCEHIOR`.

9. **stringa-casuale** ★ Scrivere una funzione `str_casuale` (di tipo `void`) che, ricevuti in ingresso una stringa S e la sua dimensione fisica, assegni a ogni elemento di S un carattere alfabetico maiuscolo scelto casualmente, a eccezione dell'ultimo elemento che deve essere obbligatoriamente impostato a `'\0'`. Utilizzare la funzione per generare e stampare sullo schermo due stringhe casuali di lunghezza 15 e 24 caratteri.

10. **stringhe-casuali** – Scrivere un programma che, utilizzando la funzione definita nell'esercizio precedente, memorizzi in un opportuno array N stringhe di 10 caratteri ciascuna aventi contenuto casuale, essendo N un numero inserito da tastiera (si supponga 5 <= N <= 20, è richiesta la validazione del dato). L'array di stringhe deve essere successivamente ordinato in modo decrescente. Terminato l'ordinamento, dovranno essere prodotti

Nelle pagine successive sono indicate le soluzioni degli esercizi che riportano il simbolo ★.

l'elenco delle stringhe generate prima dell'ordinamento (elenco iniziale) e quello successivo all'ordinamento (elenco finale). Non è ammessa la stampa dell'elenco iniziale prima dell'esecuzione dell'algoritmo di ordinamento.

11. **maratona** – Il file di testo `maratona.txt` (di cui si può trovare un esempio alla fine di questo documento) contiene i tempi impiegati dagli atleti (maschi o femmine) per completare la maratona di New York. Il tracciato record del file è il seguente:

    ```
    cognome  nome  tipo  tempo
    ```

    I campi `cognome` e `nome` sono privi di spazi e hanno ciascuno una lunghezza massima di 35 caratteri; `tipo` può essere M oppure F; `tempo` è una stringa che riporta il tempo impiegato nel formato fisso `hh:mm:ss`. Gli atleti sono registrati in ordine sparso. Scrivere un programma che acquisisca i dati degli atleti dal file e li memorizzi in modo ordinato in due tabelle distinte (una per la gara maschile, una per quella femminile) utilizzando l'algoritmo di *insertion-sort*. Successivamente, e per ciascuna tabella, dovranno essere stampati gli ordini di arrivo degli atleti, da determinare sulla base tempo impiegato. Infine, i nomi e i tempi degli atleti che hanno vinto le rispettive gare dovranno essere memorizzati nel file `vincitori.txt`.

12. **classifica** – Il file di testo `serie-a.txt` (di cui si può trovare un esempio alla fine di questo documento) contiene il numero totale di vittorie (V), pareggi (P) e sconfitte (S) delle squadre di calcio di serie A al termine della stagione. Le squadre sono memorizzate in ordine alfabetico (una riga per squadra) secondo il seguente formato:

    ```
    nome_squadra  V  P  S
    ```

    Scrivere un programma che memorizzi il contenuto del file in un'apposita tabella determinando il punteggio totale di ciascuna squadra (si ricorda che ogni vittoria vale 3 punti, ogni pareggio 1 punto, ogni sconfitta 0 punti) e visualizzando la classifica finale secondo il seguente schema:

    ```
    Pos. Squadra           V   P   S   Punti
    ========================================
     1.  NOME_SQUADRA     21  14   3     77
    ...
    ```

13. Modificare l'applicazione "**Fiumi d'Italia**" indicata sul sito web aggiungendo e implementando i seguenti comandi:

    - Ordina per nome (crescente)
    - Ordina per lunghezza del fiume (decrescente)

Soluzioni degli esercizi

Esercizio n. 6 (ordina-stringhe)

```
/* ordina-stringhe.c
 *
 * Ordinamento in-place di un array di stringhe inizializzato staticamente.
 */

#include <stdio.h>
#include <string.h>

#define NUM_RIGHE 5
#define MAX_LUNGHEZZA 8

void selection_sort(char a[][MAX_LUNGHEZZA + 1], int dim) {
    for(int i = 0; i < dim; i++) {
        int i_min = i;
        for(int j = i+1; j < dim; j++) {
            if (strcmp(a[j], a[i_min]) < 0)   // a[j] < a[i_min]
                i_min = j;
        }

        if (i != i_min) {
            char temp[MAX_LUNGHEZZA + 1];       // Scambia gli elementi usando
            strcpy(temp, a[i]);                 // una stringa temporanea
            strcpy(a[i], a[i_min]);
            strcpy(a[i_min], temp);
        }
    }

}
int main( void ) {
    char stringhe[NUM_RIGHE][MAX_LUNGHEZZA + 1] =
         {"Paolo", "Carlo", "Ada", "Stefano", "Riccardo"};

    selection_sort(stringhe, NUM_RIGHE);

    for(int i=0; i < NUM_RIGHE; i++)
        printf("%s\n", stringhe[i]);

    return 0;
}
```

Esercizio n. 9 (stringa-casuale)

```
/* stringa-casuale.c
 *
 * Funzione generatrice di stringhe casuali.
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define LUNGHEZZA_A 15
#define LUNGHEZZA_B 24

void str_casuale(char s[], int dim) {
    for(int i = 0; i < dim-1; i++)
        s[i] = 'A' + rand() % ('Z' - 'A' + 1);

    s[dim-1] = '\0';
}

int main( void ) {
    char sa[LUNGHEZZA_A + 1], sb[LUNGHEZZA_B + 1];

    srand(time(NULL));

    str_casuale(sa, LUNGHEZZA_A + 1);
    str_casuale(sb, LUNGHEZZA_B + 1);

    printf("Stringa A = %s\n", sa);
    printf("Stringa B = %s\n", sb);

    return 0;
}
```

Contenuti di alcuni file di testo

Esercizio n. 11 (maratona) – File “maratona.txt” (estratto dell’edizione 2016)

```
Abdirahman     Abdi        M    02:11:23
Mergia         Aselefech   F    02:33:28
Hailemaryam    Ayantu      F    02:37:07
Payne          Ben         M    02:15:46
Nukuri         Diane       F    02:33:04
Ghebreslassie  Ghirmay     M    02:07:51
Yamamoto       Hiroyuki    M    02:11:49
Chepkirui      Joyce       F    02:29:08
Marchant       Lanni       F    02:33:50
Rotich         Lucas       M    02:08:53
Keitany        Mary        F    02:24:26
Huddle         Molly       F    02:28:13
Kipsiro        Moses       M    02:14:18
Gracey         Neely       F    02:34:55
Smyth          Patrick     M    02:16:34
Kipyego        Sally       F    02:28:01
Hall           Sara        F    02:36:12
Biwott         Shadrack    M    02:12:01
Dabi           Tadesse     M    02:13:06
Pennel         Tyler       M    02:15:09
```

Esercizio n. 12 (classifica) – File “serie-a.txt” (relativo alla stagione 2011/2012)

```
Atalanta    13    13    12
Bologna     13    12    13
Cagliari    10    13    15
Catania     11    15    12
Cesena      4     10    24
Chievo      12    13    13
Fiorentina  11    13    14
Genoa       11    9     18
Inter       17    7     14
Juventus    23    15    0
Lazio       18    8     12
Lecce       8     12    18
Milan       24    8     6
Napoli      16    13    9
Novara      7     11    20
Palermo     11    10    17
Parma       15    11    12
Roma        16    8     14
Siena       11    11    16
Udinese     18    10    10
```