Esercitazioni di Tecniche di Programmazione

1. Prima esercitazione autoguidata

Tre avvertenze:

1) Le soluzioni agli esercizi, le versioni di programmi dati nel testo delle esercitazioni e quant'altro sono raggiungibili tramite la pagina web del corso. Nel titolo di ogni sezione è specificato tra parentesi il nome del (o dei) file in cui è proposta una soluzione (se disponibile ...).
2) I programmi che scriveremo dovranno essere in accordo con la definizione standard **ANSI C** del linguaggio C; perciò, prima di cominciare vogliamo assicurarci che l'ambiente di programmazione che usiamo "usi" anche lui la medesima definizione.
    a. SE si usa il Dev C++, nella versione 4.9.9.2 (lingua inglese) bisogna andare nel menu' "Tools", selezionare "Compiler Options", scegliere "Settings" e poi "C Compiler" (selezionare almeno "Support all ANSI Standard C Programs")
    b. Se si usa il vecchio ed eroico Turbo C++, per essere sicuri di star usando ANSI C, bisogna assegnare opportunamente una certa opzione: aprire il menù OPTIONS, selezionare *Compiler* e poi *Source*. Nella finestra di scelta che appare, selezionate ANSI C.

1.1. ESERCIZI INTRODUTTIVI (PRIMA.C).

Scrivere (e provare) un programma che usa le seguenti funzioni (da definire opportunamente):

- funzione `leggiArray` che riceve un array di interi e ne legge e assegna gli elementi
- funzione `stampaArray` che riceve un array di interi e ne stampa gli elementi
- funzione `sommaPrimiM` che riceve un array di interi `arr` e un valore `m` e restituisce la somma dei primi m elementi dell'array (o di tutti gli elementi se m>N)
- funzione `trovaElemento` che riceve un array di interi `arr` un valore `val` e restituisce il puntatore ad un elemento val presente nell'array (o `NULL` se non c'e')
- funzione `trovaEconta` che riceve un array di interi `arr`, un valore `val` e restituisce il puntatore ad un elemento val presente nell'array (o `NULL` se non c'e') **e inoltre** fornisce tramite un parametro di output **`q`** il numero di occorrenze di `val` nell'array

(Per ***parametro di output***, qui, intendiamo che il parametro formale `q` sarà un puntatore e il parametro attuale corrispondente a q sarà l'indirizzo di una locazione di memoria che verrà modificata con il valore calcolato per q.

Il programma che fa uso delle funzioni sopra dichiarate e definite potrebbe eseguire le seguenti operazioni:

- si legge un array di N interi
- si chiede e legge da input un valore m e si stampa la somma dei primi m elementi dell'array (controllando che ci siano m elementi)
- si chiedono e leggono m e k e si stampa la somma degli m elementi dell'array che cominciano dal k-esimo (supponendo che ci siano)
- si chiede e legge un valore da trovare nell'array e si stampa se quel valore c'e' o no nell'array
- si chiede e legge un valore da trovare nell'array e si stampa se quel valore c'e' o no nell'array, stampando anche il numero di occorrenze del valore nell'array

1.2. ESERCIZI DI INTRODUTTIVI (STRINGHE1.C).

Scrivere e provare un programma che usi tre stringhe dimensionate staticamente (al più di 100 caratteri). Il programma deve leggere due stringhe (la prima priva di spazi bianchi) e concatenarle in una terza stringa, che poi viene stampata. Definire ed usare le funzioni `stringlung` (che calcola la lunghezza di una stringa ricevuta come argomento) e `concatena` (che riceve tre stringhe e riempie la terza con la concatenazione delle prime due. Soluzione proposta in STRINGHE1.C.

1.3. ESERCIZI DI INTRODUTTIVI (SEQUENZA.C).

Scrivere e provare un programma che legga una sequenza di numeri interi , terminata da 0, stampi il massimo tra tali valori, la loro media e stampi anche i valori che superano la media.
I valori dati in input non saranno piu' di N (ad esempio con N=10).

Si usera' un array di 10 locazioni intere.
I valori dati in input verranno memorizzati negli elementi dell'array, a partire da quello di indice 0. Dato che i numeri sono al massimo N, molto spesso solo una parte dell'array sara' occupata dai valori letti, mentre altre componenti saranno non significative.
Per distinguere tra gli elementi significativi e quelli non, utilizziamo una variabile addizionale `numeroDati`, intera, che contiene il numero di dati che sono stati effettivamente inseriti da input prima dello 0.
Dopo la lettura dei dati, gli elementi significativi dell'array saranno quelli che vanno da indice 0 a indice `numeroDati-1`.

Soluzione proposta in SEQUENZA.C.

1.4. ALLOCAZIONE DINAMICA (ARDOUBL0.C, ARRDOUBL.C)

Scrivere e provare un programma che chieda il nome di un file testo contenente numeri reali e memorizzi tali numeri in un array di double allocato dinamicamente in base al numero effettivo di dati da memorizzarvi.

Un suggerimento per la soluzione potrebbe essere:
- il programma legge il nome del file in una variabile nomefile;
- apre un file di testo di nome nomefile
- conta i numeri ivi contenuti
- alloca un blocco di tanti double quanti ne servono, puntato dalla variabile `arrayDouble`
- rilegge i numeri dal file, memorizzandoli nell'array appena allocato
- e poi stampa i valori contenuti nell'array, se non si tratta di un array vuoto (NULL)

1.5. ARRAY DI STRINGHE "ESATTE" (PRESENZE.C)

Scrivere e provare un programma che legge da input N stringhe prive di spazi bianchi e di lunghezza al massimo MAXLUNG, memorizzandole in un array di stringhe. La memorizzazione delle stringhe deve essere "esatta" (cioè ciascuna stringa deve occupare solo il numero di caratteri ad essa necessario). (N=5, MAXLUNG=50 ad esempio).
Il programma deve poi chiedere in input una stringa e dire se essa è presente nell'array (usando una funzione `presente` appositamente definita, che riceve un array di stringhe e una stringa e restituisce 1 se la stringa è presente nell'array e 0 altrimenti). Se la stringa cercata non e' presente nell'array, la si ristampa rivoltata.
Soluzione proposta in PRESENZE.C.

Poi, nei file PRESERR1.C e PRESERR2.C c'e' il contenuto di PRESENZE.C in cui sono stati inseriti due errori: provare i programmi in questi file e trovare gli errori.

1.6. ANCORA ARRAY DI STRINGHE "ESATTE" (PRESENZ2.C)

Ripetere l'esercizio precedente, ma stavolta bisogna poter eseguire numerose ricerche: l'utente viene invitato a chiedere diverse stringhe da cercare nell'array di stringhe; quando viene inserita la stringa "STOP" il programma termina.

1.7. ANCORA ARRAY DI STRINGHE "ESATTE" (PRESENZ3.C)

Ripetere l'esercizio precedente, ma stavolta bisogna usare una funzione `duplicato` per assegnare gli elementi dell'array di stringhe: la funzione duplicato riceve una stringa, ne alloca un duplicato (allocando solo il numero di caratteri necessario) e restituisce tale duplicato (come risultato della funzione).

1.8. ARRAY DI STRINGHE USATO PARZIALMENTE (STRMENU.C)

Scrivere un programma che soddisfi i seguenti requisiti:

- il programma deve gestire un array di stringhe nel quale possono essere memorizzate da 0 a N stringhe (es. N=40);
- il programma mostra un menù di scelte possibili (aggiunta di una stringa nell'array, sostituzione di una stringa presente nell'array con un altra, stampa delle stringhe contenute nell'array) e chiede all'utente di scegliere tra le varie opzioni;
- in base alla scelta, il programma attiva l'opportuna sua funzione;
- il programma usa le funzioni
    - `aggiunta` (che, ricevendo un array di stringhe, il numero di stringhe presenti nell'array e una stringa, aggiunge la stringa nella prima posizione libera dell'array e aggiorna il numero di stringhe presenti);
    - `ricerca` (che, ricevendo un array di stringhe e una stringa, restituisce l'indice dell'eleemto dell'array in cui c'e'e la stringa (oppure –1));
    - `sostituzione` (che, ricevendo un array di stringhe e due stringhe, s1, s2, cerca la prima occorrenza di s1 nell'array e la sostituisce con s2);
    - `stampa` (che, ricevendo un array di stringhe, stampa tali stringhe sul monitor).

per gestire l'array di stringhe, oltre all'array vero e proprio di puntatori a carattere (come `char * stringhe[N]`) sarà necessario usare una variabile supplementare intera `numeroStringhe`, che contenga il numero di stringhe che sono state effettivamente aggiunte nell'array in un dato momento dell'esecuzione del programma.

La funzione `aggiungi` riceve l'indirizzo della variabile `numeroStringhe`, in modo da poter usare il suo valore e poterlo modificare. questa funzione deve usare quel valore per memorizzare nel primo elemento libero dell'array la nuova stringa; inoltre deve poterlo modificare perche' dopo l'aggiunta il numero complessivo di stringhe nell'array e' cresciuto di 1.

Le altre funzioni dovranno ricevere anche `numeroStringhe` come argomento, in modo da poter limitare la scansione dell'array alle sole componenti effettivamente occupate.

1.9. CALCOLI SU UN ARRAY ESATTO

Scrivere un programma che legge da input un valore n e poi n valori di tipo double, memorizzandoli in un array di esattamente n componenti. Su tale array vanno calcolati (per poi visualizzarli sullo schermo, il valore medio del contenuto, il valore massimo e il valore minimo.

1.10. CALCOLI SU UN ARRAY ESATTO - 2

Verificare che l'esercizio precedente poteva essere risolto senza usare l'array (calcolando i valori richiesti durante le operazioni di input).

1.11. CALCOLI SU UN ARRAY ESATTO - 3

Se non gia' fatto nell'esercizio 1.9), scriver una seconda versione di quel programma, in cui si calcolano tutti i valori richiesti con l'esecuzione di un unico ciclo di scansione dell'array dinamico impiegato.

1.12. CALCOLI SU UN ARRAY ESATTO - 3

Se non gia' fatto nell'esercizio 1.9), scrivere una seconda versione di quel programma, in cui si calcolano tutti i valori richiesti mediante l'uso di funzioni apposite per il calcolo della media, del minimo e del massimo.