7 Secondo riepilogo

Questa dispensa propone esercizi riepilogativi sui concetti visti finora.

7.1 Esercizi

Esercizio 7.1

Scrivere un programma che inizializzi una matrice 10×10 con valori crescenti, in modo tale che $m[i][j] > m[i-1][j-1]$.

Dopodiché, il programma dovrà stampare la matrice seguendo un percorso a spirale in senso orario, a partire dalla cella $m[0][0]$.

Per semplicità si può assumere la matrice quadrata.

Esercizio 7.2

Scrivere un programma che inizializzi una matrice m di dimensione `DIM` $\times$ `DIM` fissata nel programma, `DIM`, con valori crescenti, in modo tale che $m[i][j] > m[i-1][j-1]$.

Dopodiché, il programma dovrà acquisire una matrice s di dimensione `DIMS` $\times$ `DIMS` fissate nel programma, `DIMS` $<$ `DIM`.

1. Scrivere un'opportuno frammento di codice che determini se s è una sottomatrice di m. Il codice deve essere parametrico rispetto a `DIM` e `DIMS`; ovvero cambiando i valori di `DIM` e `DIMS` il codice deve rimanere invariato.
2. (TODO) stampare la posizione i, j dove la sottomatrice viene trovata.

Esercizio 7.3

(TdE November 2012) Si considerino le seguenti dichiarazioni di tipi e variabili che definiscono le strutture dati per rappresentare informazioni relative alle tessere fedeltà dei clienti di una compagnia aerea:

```
#define MAXVIAGGI 100 typedef char stringa[15];

typedef struct {
  stringa aeroportoPartenza;
  stringa aeroportoArrivo;
  float distanza; /* distanza in chilometri (lunghezza del volo) */
} viaggio;

typedef struct{
  char codiceTessera[10];
  stringa nome;
  stringa cognome;
  stringa nazionalita;
  int numViaggiEffettuati;
  viaggio ElencoViaggi[MAXVIAGGI];
} cliente;
```

1. Definire, usando il linguaggio C, un'appropriata variabile per memorizzare le informazioni relative a 50 clienti. Si chiami tale variabile `ElencoClienti`.

2. Scrivere in linguaggio C, aggiungendo eventualmente opportune dichiarazioni di variabili, un frammento di codice che permetta di visualizzare a video, per ogni cliente che ha effettuato almeno 10 viaggi, il nome, cognome, numero totale di chilometri percorsi e la lunghezza media dei voli. Si supponga che l'elenco dei clienti sia memorizzato nella variabile `ElencoClienti` definita al punto 1 e che essa sia già stata inizializzata con le informazioni relative a 50 clienti.

Esercizio 7.4

Si considerino le seguenti dichiarazioni di tipi e variabili che definiscono le strutture dati per rappresentare i messaggi scambiati attraverso Facebook e ai profili degli utenti:

```
typedef struct {
  char nome[24];
  char cognome[24];
  char email[64];
} utente;

typedef struct {
  char contenuto[256];
  utente mittente;
  int numDestinatari;
  utente destinatari[256];
} messaggio;
```

1. Definire, usando il linguaggio C, un'appropriata variabile per memorizzare le informazioni relative a 10 utenti e 10 messaggi. Si chiamino tali variabili `Utenti` e `Messaggi`.

2. Scrivere in linguaggio C, aggiungendo eventualmente opportune dichiarazioni di variabili, un frammento di codice che permetta di visualizzare a video l'indirizzo email di tutti gli utenti che hanno spedito almeno un messaggio e che non siano tra i destinatari di tale messaggio. Si presupponga che la varibile `Messaggi` sia inizializzata correttamente con le informazioni relative a 10 messaggi. Si presupponga inoltre che l'indirizzo email sia identificatore univoco di un utente: pertanto si può utilizzare l'indirizzo email per verificare se due utenti sono lo stesso utente.

Esercizio 7.5

Scrivere un programma che, legga una stringa inserita da tastiera, di lunghezza massima 256. Dopo la lettura il programma deve ordinare i caratteri presenti nella stringa in ordine lessicografico (e.g., `'a' < 'b' < ... < 'z'`) secondo la tabella ASCII.

Suggerimento: pensare al caso limite di una stringa composta da due soli caratteri oltre al terminatore (e.g., `"zc\0"`).

Esercizio 7.6

(TdE Febbraio 2012, variante) Data m una matrice di dimensione $n \times n$ (costante simbolica) di numeri interi nell'intervallo $[0, 9]$:

1. si scriva un frammento di programma in linguaggio C (con le relative dichiarazioni di variabili necessarie al corretto funzionamento), che trovi il numero più frequente (si ipotizzi che tale numero sia unico). Il programma deve memorizzare in un array opportunamente dimensionato, senza buchi, tutti e soli i numeri presenti nella matrice che hanno valore strettamente minore al numero più frequente, ed in un altro vettore quelli strettamente maggiori.

2. Aggiungere al programma di cui sopra un frammento di codice che legga tutti e soli i valori memorizzati nel secondo dei due vettori (che potrebbero essere meno della lunghezza massima del vettore) e stampi a video se sono o meno monotoni crescenti, ovvero se tutti gli elementi adiacenti sono ordinati $a_i \geq a_{i+1}$.

Esercizio 7.7

(TdE Novembre 2010) Si considerino le seguenti dichiarazioni

```
typedef char stringa[30];

typedef char matricola[10];

typedef struct {
  stringa cognome, nome;
  matricola m;
} datiStudente;

typedef struct {
  datiStudente stud;

  /* presenza e voto delle 2 prove intermedie */
  int pres1, pres2; //0 se non presente, !=0 altrimenti
  int voto1, voto2;
} datiProveStudente;

typedef struct {
  datiProveStudente s[300];
  int nStud; //numero studenti effettivamente inclusi nel registro
} registroProveInt;

registroproveInt r;
```

Durante ogni corso sono previste due prove scritte in itinere non obbligatorie: gli studenti possono partecipare, a loro scelta, a una o a entrambe. Se entrambe le prove sono valide e se la somma dei punteggi è almeno 18, lo studente ha superato l'esame del

corso senza dover sostenere altre prove. Ogni prova in itinere assegna al massimo 17 punti, e la prova in itinere è valida solo se il voto è di almeno 8 punti.

1. Assumendo che la variabile `r` sia inizializzata, si scriva un frammento di codice, dichiarando eventuali variabili aggiuntive, che stampi a schermo la matricola e i punti ottenuti dagli studenti che hanno presenziato a una sola delle due prove in itinere, ma non ad entrambe.

2. Con riferimento alle uteriori dichiarazioni di seguito:

```
typedef struct {
    matricola m[300];
    int punti[300];
    int nStud; //come sopra, studenti effettivamente in elenco
} registroEsiti;

registroEsiti neg;
```

   si scriva una variante del codice precedente che, invece di stampare matricole e punteggi, li inserisca nella variabile `neg` senza lasciare buchi e aggiornando opportunamente il valore di `nStud`.

Esercizio 7.8

(TdE Luglio 2011) Si considerino le seguenti dichiarazioni

```
typedef struct {
    int p1, p2;
} pari;

pari p;
```

1. Si scriva un frammento che legga da tastiera un numero intero ed inserisca in `p1` e `p2` i due numeri pari più vicini a quello letto da tastiera e minori di esso. Se ad esempio l'utente inserisce 15, la variabile `p` deve contenere `p.p1 = 14` e `p.p2 = 12`.

2. Data la seguente ulteriore dichiarazione

```
pari arrayCoppiePari[100];
```

   si scriva un nuovo frammento di codice che, letto da tastiera un numero `n > 0`, trovi, a partire da 0, le prime `n` coppie di numeri pari e le memorizzi nell'array. Il frammento di codice deve verificare anche che il valore `n` sia positivo e compatibile con le dimensioni dell'array dichiarato.