6 Riepilogo

Questa dispensa propone esercizi riepilogativi sui concetti visti finora ovvero:

- costrutti condizionali (`if`, `switch`);
- costrutti iterativi (`for`, `while`);
- dichiarazione di vettori e matrici;
- dichiarazione di dati strutturati (`struct`);
- dichiarazione di nuovi tipi (`typedef`, `enum`);

6.1 Esercizi

Esercizio 6.1

(TdE Novembre 2007) Trasformare il frammento di codice in un equivalente frammento che contenga istruzioni `while` invece che `for`. Specificare, infine, qual è il valore stampato quando il codice viene eseguito, giustificando adeguatamente la risposta.

```
int i, j, s;
s = 1;

for (i = 3; i >= 0; i--)
  for (j = 3; j >= 0; j--)
    if ((i+j) % 2 != 0 )
      s = s * 2;

printf("%d", s);
```

Esercizio 6.2

La catena cinematografica *TROVO Cinemas* gestisce cinema multisala in tutto il territorio nazionale. Ogni cinema della catena dispone di un certo numero di sale, non superiore a 10. Nell'arco della giornata, in ciascuna sala viene proiettata una sequenza

di film. Un film in programmazione è caratterizzato da un titolo, il nome del regista e da un genere fra i seguenti: drammatico, commedia, tarantino.

Per specifica politica aziendale, tutti i cinema della catena vogliono garantire ai clienti un'offerta *maratona*. Con questa offerta un cliente, particolarmente appassionato di una dato genere, puó sedersi in una sala ed assistere ad una sequenza di film del genere prediletto senza mai doversi spostare in un'altra sala.

A. Definire, in linguaggio C, le strutture dati *Cinema*, *Sala*, *Film* (piú eventuali tipi di supporto).

B. Assumendo di aver dichiarato la variabile: *cinema*, e che questa variabile sia stata opportunamente riempita in tutti i suoi campi, scrivere un frammento di codice per verificare che in ciascuna sala la programmazione sia omogenea (ovvero, tutti i film proiettati in una sala sono dello stesso genere).

C. Scrivere un frammento di codice per verificare che non ci sono sale che proiettano due volte lo stesso film.

Esercizio 6.3

(TdE Novembre 2006) Le seguenti dichiarazioni definiscono un tipo di dato che rappresenta una matrice quadrata di dimensione `DIM` (non indicata nel codice proposto, ma che deve essere precisata per ottenere del codice compilabile) e una variabile `m` di quel tipo.

```
#define DIM ... /* dimensione della matrice, da precisare */

typedef int MatriceQuadrata [DIM][DIM];
MatriceQuadrata m;
int s,p;
```

Scrivere un frammento di codice che permetta di calcolare nelle variabili:

- `s` la somma dei prodotti degli elementi delle due diagonali della matrice, presi ordinatamente;
- `p` il prodotto delle somme degli elementi delle due diagonali della matrice, presi ordinatamente;

Per esempio, se `DIM` avesse valore 5 e la matrice `m` fosse la seguente:

$$\begin{array}{ccccc} 3 & 2 & 1 & 5 & 8 \\ 2 & 5 & 1 & 6 & 4 \\ 12 & 4 & 2 & 6 & 7 \\ 5 & 2 & 13 & 6 & 8 \\ 7 & 3 & 1 & 4 & 1 \end{array}$$

allora:

$\texttt{s} = \; 3 \cdot 8 + 5 \cdot 6 + 2 \cdot 2 + 6 \cdot 2 + 1 \cdot 7$

$\texttt{p} = \; (3 + 8) \cdot (5 + 6) \cdot (2 + 2) \cdot (6 + 2) \cdot (1 + 7)$

Esercizio 6.4

Scrivere un programma che inizializzi una matrice m di dimensione `DIM` $\times$ `DIM` fissata nel programma, `DIM`, con valori della tavola pitagorica.

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. Stampare la posizione i, j dove la sottomatrice viene trovata.

Esercizio 6.5

(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, nome, cognome, numero totale di chilometri percorsi e 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 6.6

Scrivere un programma che, letta 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 6.7

Si consideri la seguente definizione di tipi di dato per rappresentare alcune nazioni, visitatori e padiglioni di EXPO2015:

```
#define MAX_CARAT 50 /* Dimensione del tipo Stringa */
#define MAX_PREF 3 /* Numero di padiglioni preferiti */
#define MAX_VIS 500 /* Numero massimo di visitatori in coda a un
   padiglione */
#define MAX_PAD 6 /* Numero di padiglioni */

typedef enum{italia, giappone, nordcorea, emiratiarabi, francia, cile
   } Nazione;

typedef char Stringa[MAX_CARAT];

typedef struct {
        Stringa nome, cognome;
        Nazione preferiti[MAX_PREF];
        double prezzo;
        } Visitatore;

typedef struct {
        Stringa nome;
        Nazione ident;
        int ore_visita, minuti_visita;
```

```
    int len_coda;
    Visitatore coda[MAX_VIS]; // array dei visitatori attualmente
        in coda
    } Padiglione;
```

Ogni visitatore è caratterizzato da un nome, un cognome, tre padiglioni preferiti e un prezzo pagato per il biglietto di ingresso a EXPO2015. Ogni padiglione è caratterizzato da un nome, dalla nazione di appartenenza, da un tempo di visita (in ore e minuti), dal numero di visitatori in coda (ovvero dalla lunghezza della coda) e dall'elenco di tutti i visitatori in coda.

1. Si dichiari una variabile `EXPO` per contenere al massimo 6 padiglioni e si inizializzi il primo come padiglione con nome "Italia", caratterizzato da un tempo di visita di 3h e 5min.

2. Si dichiari una variabile `primo` per contenere i dati del primo visitatore e si scriva una porzione di codice per richiedere all'utente i dati di `primo`.

3. Si inserisca quindi `primo` nella coda del padiglione "Italia", aggiornando opportunamente i campi del padiglione.

Si assuma che la variabile `EXPO` sia stata popolata con `n_pad` padiglioni (numero intero, positivo e minore di `MAX_PAD`) e che i padiglioni contengano anche la lista dei visitatori in coda. Si scrivano porzioni di codice per risolvere le seguenti richieste, ricordandosi di dichiarare tutte le variabili che si ritiene necessario utilizzare.

1. Si scriva una porzione di codice per individuare tutti i visitatori che sono in coda a un padiglione tra i loro preferiti.

2. Inserire i dati di questi visitatori in un vettore `visitatori_felici` da dichiarare opportunamente.

3. Si calcoli quindi il prezzo totale pagato dai visitatori così selezionati.

4. Si modifichi la dichiarazione del tipo Padiglione per associarvi anche il ristorante del padiglione. In particolare, si definisca un tipo Ristorante per contenere le seguenti informazioni: nome, costo del coperto, prezzo medio di un pasto, numero di coperti totale, lista dei visitatori che vi stanno mangiando (massimo 50) e numero di posti attualmente occupati.

Esercizio 6.8

(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 ulteriori 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 6.9

(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.

Esercizio 6.10

(TdE Settembre 2011) Si considerino le seguenti dichiarazioni:

```
typedef struct {
    float x;
    float y;
} Punto;

typedef struct {
  Punto a;
  Punto b;
} Segmento;

Segmento dati[100];
Segmento s;
Segmento ris[100];
int num_coincidenti;
```

dove il tipo `punto` rappresenta un punto nel piano cartesiano (x, y), il tipo `segmento` rappresenta un segmento con i punti `a` e `b` come estremi, e l'array `dati` contiene le informazioni relative a `100` segmenti nel piano cartesiano.

Si assuma che l'array `dati` sia opportunamente inizializzato. Scrivere un frammento di codice in linguaggio C che:

1. acquisisca da tastiera e memorizzi in `s` le informazioni relative ad un segmento;

2. inserisca nell'array `ris` tutti i segmenti presenti in `dati` che sono coincidenti con quello appena letto e scritto in `s`; si ricorda che due segmenti si dicono coincidenti quando entrambi i loro punti estremi, indipendentemente dall'ordine, hanno le stesse coordinate cartesiane;

3. assegni alla variabile `num_coincidenti` il numero di segmenti coincidenti trovati.

4. (bonus) cercare in `dati` tutte le coppie di segmenti adiacenti che risultano formare delle spezzate e stampare a video i punti in sequenza.

Esercizio 6.11

Scrivere un programma che calcoli elabori una matrice di interi di almeno 3x3 elementi, nel seguente modo:

```
      0 1 2 3 4 5

  0   a b c d e f
  1   g h i j k l
  2   m n o p q r
  3   s t u u v x
```

nell'elemento 0,0 (in questo caso esemplificato con 'a' per brevità) il programma dovrà scrivere la somma degli elementi della sottomatrice 3x3 "centrata" in 0,0. Quando questa sottomatrice non è completamente definita, come nel caso di 0,0, il programma dovra sommare solo gli elementi esistenti.

Ad esempio, al posto di 'a' il programma scriverà $a + b + h + g$;

```
+-----+
|     |
|  a b| c d e f
|  g h| i j k l
+-----+
   m n  o p q r
   s t  u u v x
```

al posto di 'i' il programma scriverà b+c+d+h+i+j+n+o+p.

```
     +-----+
   a |b c d| e f
   g |h i j| k l
   m |n o p| q r
     +-----+
   s  t u u  v x
```