7 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`);

7.1 Esercizi

Esercizio 7.1

(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:

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

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 7.2

(TdE November 2007) Le seguenti dichiarazioni definiscono tipi di dati relativi alla categoria degli impiegati di un'azienda (gli impiegati possono essere di prima, seconda, ..., quinta categoria), agli uffici occupati da tali impiegati, all'edificio che ospita tali uffici (edificio diviso in 20 piani ognuno con 40 uffici).

```
/* definizioni dei tipi */
```

```
typedef struct {
  char nome[20], cognome[20];
  int cat; // contiene valori tra 1 e 5
  int stipendio;
} Impiegato;

typedef enum{
  nord, nordEst, est, sudEst, sud, sudOvest, ovest, nordOvest
} Esposizione;

typedef struct {
  int superficie; /* in m^2 */
  Esposizione esp;
  Impiegato occupante;
} Ufficio;

/* definizioni delle variabili */
Ufficio torre[20][40];

/* rappresenta un edificio di 20 piani con 40 uffici per piano */
```

1. Si scriva un frammento di codice (che includa eventualmente anche le dichiarazioni di ulteriori variabili) che stampi il cognome, lo stipendio e la categoria di tutte e sole le persone che occupano un ufficio orientato a sud oppure a sudEst e avente una superficie compresa tra 20 e 30 metri quadri;

2. Visualizzi a schermo i piani che non hanno neanche un ufficio esposto a nord;

3. Stampi lo stipendio medio degli impiegati in questi piani che si chiamano "Giacomo" di nome.

Esercizio 7.3

Assumendo che `c1` e `c2` siano due variabili di tipo char, che memorizzano rispettivamente i valori `'e'` ed `'m'` indicare, per ognuna delle espressioni logiche:

1. se l'espressione è vera o falsa (per i valori delle variabili sopra indicati);

2. se è sempre vera per qualsiasi valore che le due variabili possono assumere;

3. se è sempre falsa per qualsiasi valore che le due variabili possono assumere.

Si fornisca una giustificazione per ogni risposta inserita nella tabella (risposte senza giustificazione potranno essere considerate nulle).

1. `((c1 != 'e') && (c2 == 'm')) || ((c1 != 'h') && (c2 == 'm'))`

2. `(c1 < 'g') || !((c1 <= 'g') && (c1 != 'g'))`
3. `(c1 <= 'm') || ((c2 > 'm') || !(c2 > c1))`

Esercizio 7.4

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 7.5

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 7.6

(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 7.7

1. Si definiscano le seguenti variabili che specificano le strutture dati per rappresentare i messaggi scambiati attraverso Facebook e ai profili degli utenti:

   - una struttura che definisca un'utente, con nome, cognome e e-mail, tutti composti da caratteri alfanumerici;

   - una struttura che definisca un messaggio, con un contenuto alfanumerico, un mittente scelto tra gli utenti, un numero fissato di destinatari e i destinatari, anch'essi scelti tra gli utenti (fino ad un massimo di 256).

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

3. 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 variabile `messaggi_inviati` 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 7.8

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 7.9

(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 7.10

(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 7.11

(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 7.12

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
```

Soluzioni

Soluzione dell'esercizio 7.1

```
/* prima variante */

s = 0;
p = 1;

for(i = 0; i < DIM ; i++) {
    s += m[i][i] * m[i][DIM - i -1];
    p *= m[i][i] + m[i][DIM - i -1];
 }

/* seconda variante con (due indici) */
s = 0;
p = 1;

for(i = 0, j = DIM-1; i < DIM, j > 0; i++, j--) {
    s += m[i][k] * m[i][j];
    p *= m[i][j] + m[i][j];
 }
```

Soluzione dell'esercizio 7.2

```
int p, u; /* indice di piano nelledificio e di ufficio nel piano */

for (p = 0; p < 20; p++)
    for (u = 0; u < 40; u++)
        if ((torre[p][u].esp == sudEst || torre[p][u].esp == sud) &&
            (torre[p][u].superficie >=20 && torre[p][u].superficie<=30)) {
            printf("\n il Signor %s è impiegato di categoria %d",
                   torre[p][u].occupante.cognome,
                   torre[p][u].occupante.cat);
            printf("e ha uno stipendio pari a %d euro \n",
              torre[p][u].occupante.stipendio);
        }
```

```
int uffNord; /* uffNord fa da flag*/

for (p = 0; p < 20; p++) {
    uffNord = 0; //per ogni piano assumo 0
    for (u = 0; u < 40 && !uffNord; u++)
        if(torre[p][u].esposizione == nord)
            uffNord = 1;

    /* se qui vale ancora 0 vuol dire che non ci sono uffici a nord*/
    if (!uffNord)
        printf("il piano %d  non ha edifici esposti a nord", p);
 }

/* è corretto anche senza !uffNord nel for(), anche se non è efficiente. */
```

```
for(i = 0; i < 20; i++) { //scorre i piani
    noUfficiNord = 1; //versione con flag inversa
```

```
    for(j = 0; j < 40; j++) //senza flag arriva fino a 39
        if(torre[i][j].esposizione == nord)
            noUfficiNord = 0;

    if(noUfficiNord) {
        printf("il piano %d non ha uffici a nord", i);
        stipendioMedio = 0;
        cnt = 0;
        for(j = 0; j < 40; j++)
            if(strcmp(torre[i][j].occupante.nome, "Giacomo") == 0){
                stipendioMedio += torre[i][j].occupante.stipendio;
                cnt++;
            }
        printf("Lo stipendio medio dei Giacomo nel "
               "piano è %f ", stipendioMedio / cnt);
    }
}
```

Soluzione dell'esercizio 7.3

1. È vera perché è vero il secondo disgiunto (di cui è vero sia il primo congiunto, in quanto `c1` vale `'e'`, che il secondo congiunto, in quanto `c2` vale `'m'`). L'espressione, essendo vera, ovviamente non è sempre falsa. Inoltre non è sempre vera, per esempio sarebbe falsa se `c2` valesse `'k'` o qualsiasi altro valore diverso da `'m'`.

2. È sempre vera perché, in base alla legge di De Morgan, è equivalente a

   ```
   (c1 <'g') || (c1 > 'g') || (c1 == 'g')
   ```

   quindi la formula non è sempre falsa.

3. È sempre vera perché equivalente alla formula

   ```
   (c1 <= 'm') || (c2 > 'm') || (c2 <= c1)
   ```

   che è identicamente vera: infatti sarebbe falsa solo se fossero falsi tutti e tre i suoi disgiunti, cosa che non può essere, perché se sono falsi i primi due (cioè se `c1 > 'm'` e `c2 <= 'm'`) allora `c2 < c1`, quindi il terzo è vero. Quindi la formula non è sempre falsa.

Soluzione dell'esercizio 7.4

```
#define DIM 10

#include <stdio.h>

int main() {
    int i, j, inf, sup;

    //definizione tipo matrice
    typedef int matrice_t[DIM][DIM];
```

```
    //una matrice
    matrice_t m;

    //inizializzazione della matrice con valori crescenti
    for (i = 0; i < DIM; i++) {

        for (j = 0; j < DIM; j++) {
            m[i][j] = i*j;
            printf("%2d ", m[i][j]);
        }

        printf("\n");
    }

    //stampa a spirale

    //limite inferiore (e superiore)
    for (inf = 0; inf < DIM/2; inf++) {
        sup = DIM - inf - 1;

        printf("\n");

        //da sinistra a destra
        for (j = inf; j < sup; j++)
            printf("%2d ", m[inf][j]);

        printf("\n");

        //dall'alto verso il basso
        for (i = inf; i <= sup; i++)
            printf("%2d ", m[i][sup]);

        printf("\n");

        //da destra a sinistra
        for (j = sup-1; j >= inf; j--)
            printf("%2d ", m[sup][j]);

        printf("\n");

        //dal basso verso l'alto
        for (i = sup-1; i > inf; i--)
            printf("%2d ", m[i][inf]);

        printf("\n");
    }
}
```

Soluzione dell'esercizio 7.5

```
#define DIM 10
#define DIMS 3

#include <stdio.h>

int main() {
    int i, j, si, sj, uguali;

    //una matrice
```

```
    int m[DIM][DIM];

    //una sottomatrice
    int s[DIMS][DIMS];

    //inizializzazione della matrice con valori crescenti
    for (i = 0; i < DIM; i++) {
        for (j = 0; j < DIM; j++) {
            m[i][j] = i*j;
            printf("%2d ", m[i][j]);
        }

        printf("\n");
    }

    //inizializzazione della sottomatrice
    /*
      Test 1: 36 42 48 42 49 56 48 56 64
      Test 2: 1  2  3 2  4  6 3  6  9
      Test 3: 36 42 48 42 49 56 48 56 65
     */
    for (i = 0; i < DIMS; i++) {
        for (j = 0; j < DIMS; j++) {
            scanf("%d", &s[i][j]);
            printf("%2d ", s[i][j]);
        }

        printf("\n");
    }

    //indici per scorrere la sottomatrice
    si = sj = 0;
    uguali = 0;

    for (i = 0; i < DIM-DIMS && !uguali; i++) {
        for (j = 0; j < DIM-DIMS && !uguali; j++) {
            uguali = 1;

            for (si = 0; si < DIMS && uguali; si++)
                for (sj = 0; sj < DIMS && uguali; sj++) {
                    uguali = (m[i+si][j+sj] == s[si][sj]);
                    printf("%d %d %d %d\n", i, j, si, sj);
                }
        }
    }

    //se ha completato i 2 cicli interni e 'uguali == 1'
    if (!uguali)
        printf("NON ");
    printf("trovata");
}
```

Soluzione dell'esercizio 7.6

1. Definizione: `Clienti elencoClienti[50]`
2. Nome, cognome e chilometri totali percorsi e lunghezza media dei voli per i clienti che hanno effettuato almeno 10 viaggi:

```
void main () {
  float somma_distanze;
  int i, j;
  Cliente elencoClienti[50];

  //inizializzazione [...]

  for (i = 0; i < 50; i++) {
    if (elencoClienti[i].numViaggiEffettuati >= 10) {
      somma_distanze = 0.0;
      for (j = 0; j < elencoClienti[i].numViaggiEffettuati; j++)
        somma_distanze += elencoClienti[i].elencoViaggi[j].distanza;

      printf("%s %s %f %f",
             elencoClienti[i].nome,
             elencoClienti[i].cognome,
             somma_distanze,
             somma_distanze/elencoClienti[i].numViaggiEffettuati);
    }
}
```

Soluzione dell'esercizio 7.7

1.
```
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;
```

2.
```
Utente utenti_facebook[10];
Messaggio messaggi_inviati[10];
```

3. Gli utenti che hanno spedito almeno un messaggio sono necessariamente presenti come mittenti nella variabile `messaggi_inviati`:

```
void main () {
  int i, j;
  Utente utenti_facebook[10];
  Messaggio messaggi_inviati[10];

  for (i = 0; i < 10; i++) {
    j = 0;

    //ricerca destinatario.email != mittente.email con strcmp()
    while (j < Messaggi[i].numDestinatari &&
              strcmp(Messaggi[i].mittente.email,
                     Messaggi[i].destinatari[j].email) != 0) {
      j++;
    }
```

```
    if (j == Messaggi[i].numDestinatari) //non trovato
      printf("%s", Messaggi[i].mittente.email);
}
```

Soluzione dell'esercizio 7.8

Per ordinare una stringa di due caratteri si scambiano tali caratteri di posizione solo se questi non sono già ordinati. In una stringa di lunghezza maggiore di due caratteri si procede a scambiare tutti i caratteri adiacenti fino a che non ci sono più scambi da effettuare. Arrivati a tale condizione la stringa è ordinata.

```
#include <stdio.h>

#define LEN 256

int main () {
    char str[LEN+1];
    char tmp;
    int passi = 0;
    int scambi;
    int i;

    printf("Inserire una stringa: ");
    gets(str);

    do {
        scambi = 0; //non ho scambiato
        for (i = 0; str[i+1] != '\0'; i++) {
            if (str[i] > str[i+1]) { //se non ordinati
                //scambio
                tmp = str[i];
                str[i] = str[i+1];
                str[i+1] = tmp;
                scambi = 1; //ho dovuto scambiare
                printf("  %d: %s\n", passi, str);
            }
        }

        printf("%d: %s\n", passi, str);

        passi++;
    } while (scambi); //stop quando scambi non vale 1

    return 0;
}
```

Questo algoritmo di ordinamento, noto anche con il nome di *bubble sort*. Alternativamente si può usare un algoritmo meno efficiente che cerchi iterativamente l'elemento da mettere all'inizio della stringa ordinata:

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

#define LEN 256
```

```
void main () {
char str[LEN+1];
char ordered_str[LEN+1];

char min_char;
int ind_min_char, n;
int i,j;

printf("Inserire una stringa: ");
gets(str);
n = strlen(str);

for (i = 0; i < n; i++) {
    //Find first letter
    min_char = str[0];
    ind_min_char = 0;
    for (j = 1; j < n - i; j++) {
        if (str[j] < min_char){
            min_char = str[j];
            ind_min_char = j;
        }
    }

    //Insert letter in ordered_str
    ordered_str[i] = min_char;

    //Shift letter after the minimum
    for (j = ind_min_char; j < n - i; j++)
        str[j] = str[j+1];


}

ordered_str[n] = '\0';

printf("La parola ordinata e': %s",ordered_str);

}
```

Soluzione dell'esercizio 7.9

1. Si scorre il vettore `s` fino a `nStud` e si stampano i dati solo se `pres1` XOR `pres2` hanno valori diversi da zero.

```
int i;

for (i = 0; i < r.nStud; i++)
  if (!(r.s[i].pres1 && r.s[i].pres2) &&
      (r.s[i].pres1 || r.s[i].pres1)) {
          printf("%s ", r.s[i].stud.m);

          if (r.s[i].pres1)
              printf("%d\n", r.s[i].voto1);
          else
              printf("%d\n", r.s[i].voto2);
      }
```

2. Al posto della stampa effettuo una copia valore per valore e aggiorno il contatore `nStud`:

```
int i;
neg.nStud = 0;

for (i = 0; i < r.nStud; i++)
  if (!(r.s[i].pres1 && r.s[i].pres2) &&    // non entrambe
      (r.s[i].pres1 || r.s[i].pres1)) {     // una delle due
          //neg[neg.nStud].m = r.s[i].stud.m; ERRATO!
          strcpy(neg[neg.nStud].m, r.s[i].stud.m);

          if (r.s[i].pres1)
              neg[nStud].punti = r.s[i].stud.voto1;
          else
              neg[nStud].punti = r.s[i].stud.voto2;

          neg.nStud++;
      }
```

Soluzione dell'esercizio 7.10

1. Letto il numero, se è dispari, si decrementa di 1 e si ottiene un numero pari; altrimenti il numero stesso era già pari (`p1`). L'altro numero (`p2`) si ottiene decrementando `p1` di 2.

```
//...

int n;

scanf("%d", &n);

if (n > 3) {
  if (n % 2 != 0) //dispari
      p.p1 = n - 1; //pari
  else
      p.p1 = n;     //pari

  p.p2 = p.p1 - 2;  //pari
}
```

2. Si procede ciclicamente da 0.

```
#define MAX 100

#include <stdio.h>

int main () {
    typedef struct {
        int p1, p2;
    } pari;

    int i, n;
    pari arrayCoppiePari[MAX];

    do {
        scanf("%d", &n);
```

```
    } while(n <= 0 || n > MAX);

    //primo numero pari == 2
    arrayCoppiePari[0].p1 = 2;

    for (i = 0; i < n; i++) {
        arrayCoppiePari[i+1].p1 = arrayCoppiePari[i].p2 = arrayCoppiePari[i].p1 +
             2;
        printf("<p1: %d, p2: %d>\n", arrayCoppiePari[i].p1, arrayCoppiePari[i].p2
            );
    }

    return 0;
}
```

Soluzione dell'esercizio 7.11

```
#include <stdio.h>

int main () {
    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;
    int i;

    //[...] inizializzazione della variabile dati

    //1
    printf("Inserire coordinata x del primo punto: ");
    scanf("%f", &s.a.x);

    printf("\nInserire coordinata y del primo punto: ");
    scanf("%f", &s.a.y);

    printf("Inserire coordinata x del secondo punto: ");
    scanf("%f", &s.b.x);

    printf("\nInserire coordinata y del secondo punto: ");
    scanf("%f", &s.b.y);

    //2
    num_coincidenti = 0;
    for (i = 0; i < 100; i++)
        if (dati[i].a.x == s.a.x &&
            dati[i].a.y == s.a.y &&
            dati[i].b.x == s.b.x &&
            dati[i].b.y == s.b.y ||

            dati[i].b.x == s.a.x &&
```

```
            dati[i].b.y == s.a.y &&
            dati[i].a.x == s.b.x &&
            dati[i].a.y == s.b.y) {

            ris[num_coincidenti].a.x = s.a.x;
            ris[num_coincidenti].a.y = s.a.y;
            ris[num_coincidenti].b.x = s.b.x;
            ris[num_coincidenti].b.y = s.b.y;

            num_coincidenti++; //3
        }

    //4
    for (i = 0; i < 100-1; i++)
        if (dati[i].b.x == dati[i+1].a.x &&
            dati[i].b.y == dati[i+1].a.y)
            printf("(%f, %f)--(%f, %f)--(%f, %f)\n",
                   dati[i].a.x, dati[i].a.y,
                   dati[i].b.x, dati[i].b.y,
                   dati[i+1].b.x, dati[i+1].b.y);

    return 0;
}
```

Soluzione dell'esercizio 7.12

```
#define DIM 10

#include <stdio.h>

int main(void)
{
    //definisco il tipo matrix_t, matrice di interi
    typedef int matrix_t[DIM][DIM];

    //definisco due matrici
    matrix_t m,
             n;

    //variabili ausiliarie
    int i,                      //indice delle righe
        j,                      //indice delle colonne
        x,                      //indice delle righe della sottomatrice
        y,                      //indice delle colonne della sottomatrice
        Imin, Imax,             //limiti delle righe della sottomatrice
        Jmin, Jmax;             //limiti della colonne della sottomatrice

    //inizializzo la matrice con valori crescenti
    for (i = 0; i < DIM; i++) {
        for (j = 0; j < DIM; j++) {
            m[i][j] = i * j;

            printf("%2d ", m[i][j]);
        }

        printf("\n");
    }

    for (i = 0; i < DIM; i++) {
```

```
        for (j = 0; j < DIM; j++) {
            /* limiti della sottomatrice:
             *
             * Imin = min(i-1,0),
             * Jmin = min(0,j-1)
             *
             * Imax = min(i+1,DIM)
             * Jmax = min(j+1,DIM)
             */

            //calcolo Imin
            if (i-1 < 0)
                Imin = 0;
            else
                Imin = i-1;

            //calcolo Imax
            if (i+1 > DIM)
                Imax = DIM;
            else
                Imax = i+1;

            //calcolo Jmin
            if (j-1 < 0)
                Jmin = 0;
            else
                Jmin = j-1;

            //calcolo Jmax
            if (j+1 > DIM)
                Jmax = DIM;
            else
                Jmax = j+1;

            //inizializzo a zero
            n[i][j] = 0;

            //scansione della sottomatrice
            for (x = Imin; x < Imax; x++)
                for (y = Jmin; y < Jmax; y++)
                    n[i][j] = n[i][j] + m[x][y];

            printf("%3d ", n[i][j]);
        }

        printf("\n");
    }

    return 0;
}
```