7 Secondo riepilogo

Soluzioni

Soluzione dell'esercizio 7.1

```c
#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 7.2

```
#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 7.3

1. Definizione: `client 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 tmp;
  int i, j;
  cliente ElencoClienti[50];

  //inizializzazione [...]

  for (i = 0; i < 50; i++) {
    if (ElencoClienti[i].numViaggiEffettuati >= 10) {
      tmp = 0.0;
      for (j = 0; j < ElencoClienti[i].numViaggiEffettuati; j++)
        tmp += ElencoClienti[i].ElencoViaggi[j].distanza;

      printf("%s %s %f %f",
             ElencoClient[i].nome,
             ElencoClient[i].cognome,
             tmp,
             tmp/ElencoClienti[i].numViaggiEffettuati);
    }
}
```

Soluzione dell'esercizio 7.4

1. Definizioni: `utente Utenti[10]; messaggio Messaggi[10];`
2. Gli utenti che hanno spedito almeno un messaggio sono necessariamente presenti come mittenti nella variabile `Messaggi`:

```
void main () {
  int i, j;
  utente Utenti[10];     //non indispensabile in questo punto
  messaggio Messaggi[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) //trovato
      printf("%s", Messaggi[i].mittente.email);
}
```

Soluzione dell'esercizio 7.5

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*, è il più semplice che si possa progettare. Per una stringa di lunghezza n il bubble sort richiede, nel caso medio e peggiore, n^2 passi, nel caso migliore (stringa in ingresso ordinata), n passi.

Soluzione dell'esercizio 7.6

```
#define N 5       // dimensione matrice
#define RANGE 10  // da 0 a 9
```

```c
#include <stdio.h>

int main () {
    int m[N][N];
    int v1[N*N], v2[N*N];
    int freq[RANGE];
    int fmax = 0;
    int valmax;

    int u, z;

    int i, j;

    //inizializzo matrice con valori crescenti (non necessario
    // se non per evitare di inserire a mano i valori)
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            m[i][j] = i * j;
            printf("%2d ", m[i][j]);
        }
        printf("\n");
    }

    //inizializzo vettore frequenze
    for (i = 0; i < RANGE; i++)
        freq[i] = 0;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            freq[m[i][j]]++;
        }
    }

    //ricerco valore piu frequente
    for (i = 0; i < RANGE; i++)
        if (i == 0 || freq[i] > fmax) {
            valmax = i;    //valore
            fmax = freq[i]; //frequenza
        }

    //spostamento valori
    u = z = 0;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            if (m[i][j] < valmax) {
                v1[u] = m[i][j];
                u++;
            }

            if (m[i][j] > valmax) {
                v2[z] = m[i][j];
                z++;
            }
        }
    }

    //stampa valori
    printf("fmax = %d, valmax = %d\n", fmax, valmax);
    for (i = 0; i < u; i++)
        printf("%d ", v1[i]);
```

```
    printf(" (%d elementi)\n", u);

    for (i = 0; i < z; i++)
        printf("%d ", v2[i]);

    printf(" (%d elementi)\n", z);

    //controllo se v2 e' monotono
    i = 0;
    while (i < z-1 && v2[i] <= v2[i+1])
        i++;

    //uscita prematura dal ciclo?
    if (i < z-1)
        printf("Vettore NON monotono crescente.");
    else
        printf("Vettore monotono crescente.");
}
```

Soluzione dell'esercizio 7.7

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 7.8

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 7.9

```
#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].a.x, dati[i+1].a.y);

return 0;
```

```
}
```

Soluzione dell'esercizio 7.10

```
#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;
}
```