Informatica B
Esercitazione 5 (Soluzioni)

3 novembre 2021

Matrici e Struct

5.1 Scrivere un **programma C** che permette di riempire una tabella 3×3 con numeri positivi e controlla che la somma dele righe sia costante.

Esempio: {{2,7,6},{9,5,1},{4,3,8}} (la somma delle righe è sempre 15).

```c
#include <stdio.h>

/*
    Es. 5.1
    Tabella
*/

#define DIM 3

int main()
{
    int tabella[DIM][DIM];
    int somme[DIM];
    int i,j;
    int flag = 0;

    //Acquisizione
    printf("Inserisci %d numeri\n", DIM*DIM);
    for(i=0;i<DIM;i++)
    {
        for(j=0;j<DIM;j++)
        {
            do
            {
                scanf("%d", &tabella[i][j]);
            } while(tabella[i][j]<=0 || tabella[i][j]>=100);

```

```
28          }
29      }
30
31
32      //Stampa tabella
33      printf("\n");
34      for(j=0;j<DIM;j++)
35      {
36          printf("  -- ");
37      }
38      printf("\n");
39      for(i=0;i<DIM;i++)
40      {
41          for(j=0;j<DIM;j++)
42          {
43              printf("| %2d ", tabella[i][j]);
44          }
45          printf("|\n");
46          for(j=0;j<DIM;j++)
47          {
48              printf("  -- ");
49          }
50          printf("\n");
51      }
52
53      //Calcola somme
54      for(i=0;i<DIM;i++)
55      {
56          somme[i] = 0;
57          for(j=0;j<DIM;j++)
58          {
59              somme[i]+=tabella[i][j];
60          }
61      }
62
63      //Verifica
64      for(i=1;i<DIM;i++)
65      {
66          if(somme[i]!=somme[0])
67          {
68              flag = 1;
69              break;
70          }
71      }
72
73      //Risultato
```

```
74      if(flag==1)
75      {
76          printf("\nSomme diverse!\n");
77      }
78      else{
79          printf("\nSomme uguali a %d\n", somme[0]);
80      }
81
82      return 0;
83  }
```

Bonus: Modificare il programma in modo tale che controlli che la tabella inserita sia un "quadrato magico":

- Tutti i numeri inseriti devono essere distinti;
- La somma di ciascuna riga, ciascuna colonna e delle due diagonali deve essere uguale.

L'esempio fornito è un quadrato magico.

```c
#include <stdio.h>

/*
    Es. 5.1 (bonus)
    Quadrato Magico
*/

#define DIM 3

int main()
{
    int tabella[DIM][DIM];
    int somma, cost_magica;
    int i,j;
    int flag = 0;

    //Acquisizione
    printf("Inserisci %d numeri\n", DIM*DIM);
    for(i=0;i<DIM;i++)
    {
        for(j=0;j<DIM;j++)
        {
            do
            {
                scanf("%d", &tabella[i][j]);
            } while(tabella[i][j]<=0 || tabella[i][j]>=100);

        }
    }


    //Stampa tabella
    printf("\n");
    for(j=0;j<DIM;j++)
    {
        printf("  -- ");
    }
    printf("\n");
```

```
39      for(i=0;i<DIM;i++)
40      {
41          for(j=0;j<DIM;j++)
42          {
43              printf("| %2d ", tabella[i][j]);
44          }
45          printf("|\n");
46          for(j=0;j<DIM;j++)
47          {
48              printf("  -- ");
49          }
50          printf("\n");
51      }
52
53
54      /*Verifica*/
55      //Somma della prima riga
56      somma = 0;
57      for(j=0;j<DIM;j++)
58          somma+=tabella[0][j];
59      cost_magica = somma;
60
61      //Controlla le altre righe
62      for(i=1;i<DIM && !flag;i++)
63      {
64          somma = 0;
65          for(j=0;j<DIM && somma<=cost_magica;j++)
66          {
67              somma+=tabella[i][j];
68          }
69          if(somma!=cost_magica)
70          {
71              flag = 1;
72              break;
73          }
74      }
75
76      //Controlla le colonne
77      for(j=0;j<DIM && !flag;j++)
78      {
79          somma = 0;
80          for(i=0;i<DIM && somma<=cost_magica;i++)
81          {
82              somma+=tabella[i][j];
83          }
84          if(somma!=cost_magica)
```

```
85              {
86                  flag = 1;
87                  break;
88              }
89          }
90
91          //Controlla la diagonale
92          somma = 0;
93          for(i=0;i<DIM && somma<=cost_magica && !flag;i++)
94          {
95              somma+=tabella[i][i];
96          }
97          if(somma!=cost_magica)
98          {
99              flag = 1;
100         }
101
102         //Controlla l'altra diagonale
103         somma = 0;
104         for(i=0;i<DIM && somma<=cost_magica && !flag;i++)
105         {
106             somma+=tabella[i][DIM-i-1];
107         }
108         if(somma!=cost_magica)
109         {
110             flag = 1;
111         }
112
113         //Risultato
114         if(flag==1)
115         {
116             printf("\nNon e' un quadrato magico!");
117         }
118         else{
119             printf("\nQuadrato magico! La costante magica e' %d\n", cost_magica);
120         }
121
122         return 0;
123 }
```

5.2 Scrivere un **programma C** data una matrice 4x4, A, calcola la somma della matrice A con la sua matrice trasposta A'. La matrice trasposta si ottiene scambiando le righe con le colonne.
Esempio: dato
A = {{1,2,3,4},{4,5,6,7},{7,8,9,10},{10,11,12,13}},
A' = {{1,4,7,10},{2,5,8,11},{3,6,9,12},{4,7,10,13}}
A+A' = {{2,6,10,14},{6,10,14,18},{10,14,18,22},{14,18,22,26}}.

```c
#include <stdio.h>

/*
    Es. 5.2
    Somma Trasposta
*/

#define DIM 4

int main()
{
    int A[DIM][DIM];
    int somma[DIM][DIM];
    int i,j;

    //Acquisizione
    printf("Inserisci %d numeri\n", DIM*DIM);
    for(i=0;i<DIM;i++)
    {
        for(j=0;j<DIM;j++)
        {
            scanf("%d", &A[i][j]);

        }
    }


    //Stampa tabella
    printf("La matrice originale!\n");
    for(j=0;j<DIM;j++)
    {
        printf("  -- ");
    }
    printf("\n");
    for(i=0;i<DIM;i++)
    {
        for(j=0;j<DIM;j++)
        {
```

```
39              printf("| %2d ", A[i][j]);
40          }
41          printf("|\n");
42          for(j=0;j<DIM;j++)
43          {
44              printf("  -- ");
45          }
46          printf("\n");
47      }
48
49
50      // Calcoliamo somma
51      // Stampiamo allo stesso tempo
52      printf("La somma con la trasposta!\n");
53      for(i=0;i<DIM;i++)
54      {
55          for(j=0;j<DIM;j++)
56          {
57              // Non ci serve calcolare apparte la matrice A'
58              // Basta scambiare gli indici
59              somma[i][j] = A[i][j] + A[j][i];
60              printf("| %2d ", somma[i][j]);
61          }
62          printf("|\n");
63          for(j=0;j<DIM;j++)
64          {
65              printf("  -- ");
66          }
67          printf("\n");
68      }
69      return 0;
70  }
```

5.3 Scrivere un **programma C** data una matrice nxn (n dato dall'utente), A, stampa solo il gli elementi sotto la diagonale principale (inclusa la diagonale).

```c
#include <stdio.h>

/*
    Es. 5.3
    Triangolo minore
*/

#define DIM 100

int main()
{
    int A[DIM][DIM];
    int n, i,j;

    do{
        printf("Inserire la dimensione della matrice quadrata");
        scanf("%d", &n);
    }
    while(n < 1 || n > DIM);

    //Acquisizione
    printf("Inserisci %d numeri\n", n*n);
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d", &A[i][j]);

        }
    }

    //Stampa tabella
    printf("La matrice originale!\n");
    for(j=0;j<n;j++)
    {
        printf("  -- ");
    }
    printf("\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            printf("| %2d ", A[i][j]);
```

```
44          }
45          printf("|\n");
46          for(j=0;j<n;j++)
47          {
48              printf("  -- ");
49          }
50          printf("\n");
51      }
52
53      printf("\n Il triangolo minore!\n");
54      printf("  -- \n");
55      for(i=0;i<n;i++)
56      {
57          for(j=0;j<n;j++)
58          {
59              if(i >= j)
60                  printf("| %2d ", A[i][j]);
61          }
62          printf("|\n");
63          for(j=0;j<= i + 1 && j < n;j++)
64          {
65              printf("  -- ");
66          }
67          printf("\n");
68      }
69      return 0;
70  }
```

5.4 Si scriva un **programma C** per gestire una rubrica telefonica. Un contatto della rubrica è composto da un nome di persona e dal rispettivo numero di telefono. Il programma permette di inserire quattro contatti. Una volta che l'inserimento è terminato, il programma permette di cercare un contatto per nome. Se il nome inserito da tastiera è memorizzato nella rubrica, stampa il numero corrispondente, altrimenti stampa "Non trovato". In ogni caso, permette di cercare un altro contatto.

In questa soluzione, dichiariamo le variabili struct direttamente nel main. Potremmo anche usare `typedef` per definire un tipo `stringa_t` e un tipo `contatto_t`. Altri possibili miglioramenti (lasciati per esercizio) sono:

- Usare una stringa anche per il numero di telefono (per gestire numeri molto lunghi e zeri iniziali).
- Permettere l'inserimento di un numero arbitrario di contatti (fissando solo il massimo).
- Permettere all'utente di passare dalla fase di ricerca a quella di inserimento, e viceversa, ogni volta che lo desidera.
- Permettere all'utente di eliminare dei contatti.
- Gestire nomi ripetuti.

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


/*
    Es.5.4
    Rubrica telefonica
*/

#define NUM_CONTATTI 4
#define MAX_LEN 20

int main()
{

    struct {
    char nome[MAX_LEN]; //il primo campo e' una stringa
    long int numero; //il secondo campo e' un numero
    } contatti[NUM_CONTATTI]; //array di struct

    int i, trovato;
    char nome_in[MAX_LEN];

```

```
24
25      printf("INSERIMENTO\n");
26      for(i=0;i<NUM_CONTATTI;i++)
27      {
28          printf("\nInserisci nome\n");
29          gets(contatti[i].nome);
30          printf("Insersci numero\n");
31          scanf("%ld", &(contatti[i].numero));
32          scanf("%*c"); //consuma newline (per gets successiva)
33      }
34
35      printf("\nRICERCA\n");
36      // Ciclo Infinito
37      while(1)
38      {
39          printf("\nInserisci nome da cercare\n");
40          gets(nome_in);
41
42          trovato=0;
43          for(i=0;i<NUM_CONTATTI;i++)
44          {
45              if(strcmp(nome_in, contatti[i].nome)==0)
46              {
47                  printf("Il suo numero e' %ld\n", contatti[i].numero);
48                  trovato = 1;
49                  break;
50              }
51          }
52          if(!trovato)
53          {
54              printf("Non trovato!\n");
55          }
56      }
57  }
```

5.5 Si scriva un **programma C** per gestire dei rettangoli in uno spazio cartesiano 2D. I rettangoli sono rapresentati dai due angoli, l'angolo in basso a sinitra, e l'angolo in alto a destra. Ognuni dei angoli e descritto dalle coordinate cartesiane x e y. Il programma chiede all'utente le coordinate di due rettangoli, e calcola la superficie dell'intersezione dei rettangoli. Nel caso l'intersezione sia nulla stampa un messaggio di errore.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
/*
    Es.5.5
    Rettangoli
*/
// definire la strutture per rapresentare un
// punto in uno spazio 2D
typedef struct {
    float x, y;
} punto_t;

// definire la strutture per rapresentare un
// rettangolo tramite le coordinate dei due
// angoli
typedef struct {
    punto_t angolo_sinistro, angolo_destro;
} rettangolo_t;


int main()
{
    rettangolo_t ret_a, ret_b, intersezione;
    float lunghezza, altezza;

    printf("Inserire le coordinate del primo rettangolo!\n");
    printf("Inserire x e y dell'angolo sinitro in basso!\n");
    scanf("%f %f", &ret_a.angolo_sinistro.x, &ret_a.angolo_sinistro.y);
    printf("Inserire x e y dell'angolo destro in alto!\n");
    scanf("%f %f", &ret_a.angolo_destro.x, &ret_a.angolo_destro.y);

    printf("Inserire le coordinate del secondo rettangolo!\n");
    printf("Inserire x e y dell'angolo sinitro in basso!\n");
    scanf("%f %f", &ret_b.angolo_sinistro.x, &ret_b.angolo_sinistro.y);
    printf("Inserire x e y dell'angolo destro in alto!\n");
    scanf("%f %f", &ret_b.angolo_destro.x, &ret_b.angolo_destro.y);

    // prendere x e y, massimi tra gli angoli sinistri
```

```
40      if (ret_a.angolo_sinistro.x > ret_b.angolo_sinistro.x)
41          intersezione.angolo_sinistro.x = ret_a.angolo_sinistro.x;
42      else
43          intersezione.angolo_sinistro.x = ret_b.angolo_sinistro.x;
44
45      if (ret_a.angolo_sinistro.y > ret_b.angolo_sinistro.y)
46          intersezione.angolo_sinistro.y = ret_a.angolo_sinistro.y;
47      else
48          intersezione.angolo_sinistro.y = ret_b.angolo_sinistro.y;
49
50      // prendere x e y, minimi tra gli angoli destri
51      if(ret_a.angolo_destro.x < ret_b.angolo_destro.x)
52          intersezione.angolo_destro.x = ret_a.angolo_destro.x;
53      else
54          intersezione.angolo_destro.x = ret_b.angolo_destro.x;
55
56      if(ret_a.angolo_destro.y < ret_b.angolo_destro.y)
57          intersezione.angolo_destro.y = ret_a.angolo_destro.y;
58      else
59          intersezione.angolo_destro.y = ret_b.angolo_destro.y;
60
61
62      lunghezza = intersezione.angolo_destro.x - intersezione.angolo_sinistro.x;
63      altezza = intersezione.angolo_destro.y - intersezione.angolo_sinistro.y;
64
65      if (lunghezza <= 0 || altezza <= 0)
66      {
67          printf("\n I due rettangoli non hanno un'intersezione!\n");
68      }
69      else
70      {
71          printf("L'intersezione dei rettangoli ha superficie %.2f", lunghezza * a
72      }
73      return 0;
74  }
```