Fondamenti di Informatica - Ing. Civile/Edile - Dott. Penzo
Soluzione compito 17/06/2002

Esercizio 1

```
Per i passaggi intermedi vedere le dispense.

22    in base 2: 00010110
0.75  in base 2: .11000....

22.75 in base 2: 10110.11000....
normalizzando:  .10110110 mantissa e 00000101 esponente
rappresentaz.   :  .00110110 mantissa e 00000101 esponente

13    in base 2: 00001101
9     in base 2: 00001001
-9    in complemento a 2: 11110111

13 - 9 = 00000100 = 4 in base 10
4     in floating point normalizzato: .10000000 mantissa e 00000011 esponente
4     in rappresentazione interna:    .00000000 mantissa e 00000011 esponente

4     incolonnato all'esponente maggiore:
      .00100000 mantissa e 00000101 esponente

differenza 22.75 - 4:  .10110110
                       .00100000
                       ------------
                       .10010110  mantissa  e 00000101 esponente

ovvero: 10010.110 che in decimale vale  16 + 2 + 0.5 + 0.25 = 18.75
```

Esercizio 2

```
Il programma inizializza una matrice M di N righe e N colonne (5x5 elementi)
contenente numeri interi. Viene eseguito quindi un ciclo (while) per i valori di
k che variano fra 1 e 4, all'interno del quale viene chiamata la funzione F con
parametri la matrice M e il valore k come argomento dei rimanenti parametri.
Al termine del ciclo, il programma stampa il contenuto della matrice modificata
dalla funzione F. La funzione F prende come parametro lo stesso valore k per
indicare rispettivamente un indice di riga e un indice di colonna per
individuare un elemento della matrice M da modificare. In particolare saranno
modificati i soli valori sulla diagonale principale.
Iterazioni:
k = 1. M[1][1] = M[1][1] + M[1][0]*M[0][1] = 3 + 2*2 = 7
k = 2. M[2][2] = M[2][2] + M[2][1]*M[1][2] = 5 + 4*4 = 21
k = 3. M[3][3] = M[3][3] + M[3][2]*M[2][3] = 2 + 1*1 = 3
k = 4. M[4][4] = M[4][4] + M[4][3]*M[3][4] = 4 + 3*3 = 13

La matrice stampata sarà la seguente:

M[0][0] = 1     M[0][1] = 2     M[0][2] = 3     M[0][3] = 4     M[0][4] = 5
M[1][0] = 2     M[1][1] = 7     M[1][2] = 4     M[1][3] = 5     M[1][4] = 1
M[2][0] = 3     M[2][1] = 4     M[2][2] = 21    M[2][3] = 1     M[2][4] = 2
M[3][0] = 4     M[3][1] = 5     M[3][2] = 1     M[3][3] = 3     M[3][4] = 3
M[4][0] = 5     M[4][1] = 1     M[4][2] = 2     M[4][3] = 3     M[4][4] = 13
```

Esercizio 3

```
#include<stdio.h>
#define MAXPRODOTTI 100
#define MAXCLIENTI 1000

typedef struct{
    int codice;
    char descrizione[20];
    int prezzoUnitario;
} Prodotto;

typedef struct{
    char nome[20];
    char cognome[20];
    int tesseraSocio;
    Prodotto carrello[MAXPRODOTTI];
} Cliente;


void InserisciCliente(Cliente S[], int *numelem, Cliente c);
int CalcolaSpesaCliente(Cliente S[], int numClienti, char nome[20], char
cognome[20]);

/* Questa funzione serve solo per la fase di test.
 * Non e` richiesta dal testo */
Cliente *trovaCliente(Cliente S[], int numClienti, char nome[20], char
cognome[20]);

main(){

Cliente Supermercato[MAXCLIENTI], C, *clienteCorrente;
int numClienti = 0, fine = 0, spesaTotale = 0,i, numProdotti;
char scelta, nome[20], cognome[20];
Prodotto p;

do{
printf("Inserisci i dati del cliente \n");
printf("Inserire nome e cognome: ");
scanf("%s%s",C.nome,C.cognome);
C.tesseraSocio = 0;

/*  Inizializzazione del vettore. Un valore di codice
 *  pari a zero indica assenza di prodotto nella locazione.
 *  All'inizio il carrello è vuoto quindi tutte le locazioni
 *  dell'array hanno prodotti con codice pari a zero */

for(i=0; i<MAXPRODOTTI; i++) C.carrello[i].codice = 0;

InserisciCliente(Supermercato,&numClienti,C);

printf("Vuoi inserire un altro cliente? (s/n): ");
scanf("%c",&scelta); /* per pulire il buffer di input */
scanf("%c",&scelta);
if (scelta == 'n' || scelta == 'N') fine = 1;

} while(!fine & (numClienti < MAXCLIENTI));

/* fase di inserimento prodotti non richiesta dal testo */
```

```
printf("Inserisci i dati del cliente per il quale inserire i prodotti\n");
printf("Inserire nome e cognome: ");
scanf("%s%s",nome,cognome);

/* riportiamo in clienteCorrente il riferimento
 * al cliente nel supermercato per il quale inserire i
 * prodotti. Fase non richiesta nel testo */

clienteCorrente = trovaCliente(Supermercato,numClienti,nome,cognome);

printf("Inserisci i prodotti del cliente \n");
fine = 0;
numProdotti = 0;
do{
            printf("Inserisci il codice del prodotto: ");
            scanf("%d",&(p.codice));

            printf("Inserisci la descrizione del prodotto: ");
            scanf("%s",p.descrizione);

            printf("Inserisci il prezzo unitario del prodotto: ");
            scanf("%d",&(p.prezzoUnitario));

            (*clienteCorrente).carrello[numProdotti++] = p;

            printf("Vuoi inserire un altro prodotto? (s/n): ");
            scanf("%c",&scelta); /* per pulire il buffer di input */
            scanf("%c",&scelta);
            if (scelta == 'n' || scelta == 'N') fine = 1;

} while(!fine & (numProdotti < MAXPRODOTTI));

spesaTotale = CalcolaSpesaCliente(Supermercato,numClienti,nome,cognome);

if (spesaTotale > 0)
    printf("La spesa totale di %s %s è: %d\n", nome, cognome, spesaTotale);

}


void InserisciCliente(Cliente S[], int *numelem, Cliente c)
{
if (*numelem < MAXCLIENTI) S[(*numelem)++] = c;
}


int CalcolaSpesaCliente(Cliente S[], int numClienti, char nome[20], char
cognome[20])
{
int totale = 0, trovato = 0, i = 0, j = 0, finito = 0;

while (i < numClienti && !trovato){
    if (!strcmp(S[i].nome,nome) && !strcmp(S[i].cognome,cognome) &&
S[i].tesseraSocio == 0)
    {
        trovato = 1;
        while ((j < MAXPRODOTTI) && !finito){
            if ((S[i].carrello[j]).codice == 0) finito = 1;
            else{
                totale += S[i].carrello[j].prezzoUnitario;
                j++;
            }
        }
```

```
    }
    else i++;
}
return(totale);
}

/* Funzione non richiesta dal testo. Si riporta per la fase di test */
Cliente *trovaCliente(Cliente S[], int numClienti, char nome[20], char
cognome[20])
{
            int i, trovato = 0;
            Cliente *clienteTrovato;

            for(i=0;i < numClienti && !trovato; i++){
                        if (!strcmp(S[i].nome,nome) &&
!strcmp(S[i].cognome,cognome)){
                                    trovato = 1;
                                    clienteTrovato = &(S[i]);
                        }
            }
return(clienteTrovato);
}
```