

Informatica B - Esercizi di ricapitolazione

(con soluzioni)

Esercizio 1

Un sistema informatico per le prenotazioni in un albergo permette all'utente richiedente di inserire una o più prenotazioni. Prima dell'inserimento dei dettagli di ciascuna prenotazione, l'utente deve specificare il numero N di prenotazioni che intende inserire (con N minore di un certo valore MAX_PRENOTAZIONI_UTENTE). Per ogni prenotazione, le informazioni che l'utente deve specificare sono:

- Cognome (una stringa di 40 caratteri al max);
- Nome (una stringa di 40 caratteri al max);
- Numero di persone a cui si riferisce la singola prenotazione (ogni prenotazione, infatti, può riferirsi a una o più persone).
- Giorno di arrivo. Il giorno di arrivo è espresso come stringa avente il seguente formato: "aaaammgg", ossia 4 caratteri per l'anno, due caratteri per il mese, due caratteri per il giorno. Ad esempio, nel caso in cui l'utente vuole prenotare per il 24 aprile 2005, giorno di arrivo vale "20050424".
- Numero di giorni di permanenza nell'albergo.

Date le seguenti definizioni di costanti e tipi in linguaggio C:

```
#define MAX_LENGTH 40

#define MAX_PRENOTAZIONI_UTENTE 10

typedef Stringa char[MAX_LENGTH];

typedef struct {
    Stringa cognome;
    Stringa nome;
```

```

    int numPersone;

    Stringa giornoArrivo;

    int giorniPermanenza;
} Prenotazione;

/* una variabile di questo tipo contiene la singola prenotazione
richiesta da un utente */

typedef struct {

    Prenotazione vettPrenotazioni[MAX_PRENOTAZIONI_UTENTE];

    int numPrenotazioni;
} PrenotazioniUtente;

/* una variabile di questo tipo contiene tutte le prenotazioni
richieste dall'utente */

```

- 1) **Si scriva il codice del sottoprogramma void raccogliPrenotazione(PrenotazioniUtente *archPrenotazioni). Tale sottoprogramma deve acquisire da tastiera il numero N di prenotazioni che l'utente desidera inserire e, per ciascuna delle N prenotazioni, i valori di cognome, nome, numPersone, giornoArrivo e giorniPermanenza. Tali valori vengono memorizzati nella variabile del chiamante puntata da archPrenotazioni (si noti il passaggio di parametro per indirizzo).**
- 2) **Supponendo che il programma chiamante sia il main qui sotto, inserire al suo interno il codice della chiamata al sottoprogramma raccogliPrenotazione.**

```

void main() {

    PrenotazioniUtente prenotazioni;

    ...

    // chiamata al sottoprogramma raccogliPrenotazione

```

```
}
```

Soluzione

```
void raccogliPrenotazione(PrenotazioniUtente *archPrenotazioni)
{
    Prenotazione pren;
    int N, i;
    do {
        printf("Digita il numero di prenotazioni da inserire: ");
        scanf("%d", &N);
    } while(N > MAX_PRENOTAZIONI_UTENTE);
    for(i=0; i<N; i++)
    {
        printf("Inserisci i dati della prenotazione n. %d:", i);
        printf("Cognome: "); scanf("%s", pren.cognome);
        printf("Nome: "); scanf("%s", pren.nome);
        printf("Numero persone: "); scanf("%d", pren.numPersone);
        printf("Giorno di arrivo in formato ggmmaaaa: ");
        scanf("%s", pren.giornoArrivo);
        printf("Numero di giorni di permanenza: ");
        scanf("%d", pren.giorniPermanenza);
        archPrenotazioni->vettPrenotazioni[i] = pren;
    }
    archPrenotazioni->numPrenotazioni = N;
}
```

```

void main() {
    PrenotazioniUtente prenotazioni;

    // chiamata al sottoprogramma raccogliPrenotazione
    raccogliPrenotazione(&prenotazioni);
}

```

Esercizio 2

Si consideri il tipo `Prenotazione` definito nell'esercizio 1.

Si supponga, inoltre, di avere a disposizione la funzione `float costoPrenotazione(Prenotazione p)`. Tale funzione restituisce il costo della prenotazione `p` passata come parametro di ingresso. Tale costo prende in considerazione il numero delle persone associate alla prenotazione, i giorni di permanenza di tali persone, il costo per persona.

- 1) **Si scriva il sottoprogramma `memorizzaEstrattoSuFile(Stringa nomeFile, Lista l)` che, ricevuto in ingresso il nome di un file testuale e una lista dinamica `l` di prenotazioni tutte relative allo stesso utente (ogni prenotazione nella lista è associata allo stesso cognome e nome per ipotesi), memorizzi su un file il cognome, il nome e il costo totale delle prenotazioni contenute nella lista stessa. Per il calcolo del costo totale è possibile usare (senza implementarla) la funzione `costoPrenotazione`, discussa in precedenza.**

Soluzione

```

void memorizzaEstrattoSuFile(Stringa nomeFile, Lista l)
{
    float costoTot=0;

    Lista tmp;

    FILE *f;

    if(l != NULL)
        /* la lista non e` vuota */
        {

```

```

f = fopen(nomeFile, "a");
if(f!=NULL)
{
    fprintf(f, "%s\n", l->info.cognome);
    fprintf(f, "%s\n", l->info.nome);
    while(l!=NULL)
    {
        costoTot = costoTot + costoPrenotazione(l->info);
        l = l->next;
    }
    fprintf(f, "%f\n", costoTot);
    fclose(f);
}
}
}

```

Esercizio 3

In presenza delle seguenti, usuali dichiarazioni di tipo che definiscono una lista dinamica di elementi collegati a puntatori:

```

typedef struct  EL      {      int  info;
                        struct EL      *prox;
                        } elemLista;
typedef elemLista      *listaDiElem;

```

è definita la seguente funzione ricorsiva, che riceve come parametro una lista e restituisce un valore intero.

```

int cheFa (listaDiElem lis) {
    int ris;
    if (lis == NULL)
        return 0;
    else

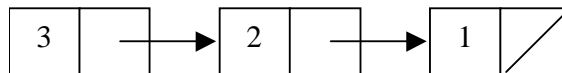
```

```

{
    ris = cheFa (lis -> prox) + 1;
    return ris;
}
}

```

a) Immaginando che tale funzione venga invocata passando come parametro attuale un puntatore al primo elemento della lista mostrata in figura



si indichi il valore restituito da tale invocazione; per giustificare la risposta si rappresenti graficamente lo stato dell'esecuzione della funzione quando l'ultima invocazione è attiva.

Si indichi, in generale, qual è la funzione matematica calcolata dall'invocazione `cheFa(lista)` quando l'argomento della chiamata è una generica lista collegata a puntatori del tipo sopra descritto.

b) Si indichi motivando la risposta (se utile, si faccia riferimento all'esempio di lista mostrato sopra), qual è la funzione calcolata dalle seguenti due varianti della funzione `cheFa`

<pre> int cheFa1 (listaDiElem lis) { int ris; if (lis == NULL) return 1; else { ris = cheFa1(lis->prox)*lis->info; return ris; } } </pre>	<pre> int cheFa2 (listaDiElem lis) { int ris; if (lis == NULL) return 1; else { ris=cheFa2(lis->prox)+cheFa2(lis->prox); return ris; } } </pre>
---	---

Soluzione

La funzione `cheFa` calcola la lunghezza della lista passata come parametro.

`cheFa1` calcola la produttoria dei numeri interi contenuti negli elementi della lista

cheFa2 calcola il valore 2^{lun} , dove lun è la lunghezza (numero degli elementi) della lista ricevuta come parametro.

Esercizio 4

Scrivere un programma che legga da un file binario chiamato componenti.dat dei record con dati relativi a prodotti di un magazzino di materiale elettronico. I record sono definiti dal tipo:

```
typedef struct {
    char sigla_componente[20];
    char produttore[20];
    char tipo[20];
    int costo;
} materiale;
```

Il tipo può essere: condensatore, resistenza, diodo o circuito_integrato. Il programma deve visualizzare la sigla del componente e il costo del componente di tipo condensatore avente costo minimo.

Soluzione

```
#include <stdio.h>
#include <string.h>
#define FILE_NAME "componenti.dat"

typedef struct {
    char sigla_componente[20];
    char produttore[20];
    char tipo[20];
    int costo;
} materiale;

typedef enum {falso, vero} boolean;
```

```

void main()
{
    FILE *f;

    float costoMin;

    char siglaMin[20];

    boolean primoCondensatore = falso;

    materiale mat;

    f = fopen(FILE_NAME, "rb");

    if (f!=NULL)
    {
        /* vado alla ricerca del primo condensatore nel file */
        while (primoCondensatore == falso &&
                fread(&mat, sizeof(materiale), 1, f) > 0)
            if(strcmp(mat.tipo, "condensatore")==0)
            {
                primoCondensatore = vero;

                costoMin = mat.costo;

                strcpy(siglaMin, mat.sigla_componente);
            }

        if(primoCondensatore == vero)
        {
            /*vado ad identificare l'elemento di costo minimo */
            while(fread(&mat, sizeof(materiale), 1, f) > 0)
                if(strcmp(mat.tipo, "condensatore")==0 && mat.costo<costoMin)
                {
                    /* l'elemento in mat e` candidato ad essere quello di costo minimo */

```

```

        costoMin = mat.costo;

        strcpy(siglaMin, mat.sigla_componente);
    }

    printf("costo e sigla del condensatore a costo minimo: %f, %s",
           costoMin, siglaMin);

}

else printf("non esistono condensatori disponibili\n");

fclose(f);

}

else printf("Errore di apertura del file %s\n", FILE_NAME);
}

/* il programma che segue consente di creare un file che possa essere utilizzato
dal programma che risolve questo esercizio */

#include <stdio.h>

#include <string.h>

#define FILE_NAME "componenti.dat"

typedef struct {

    char sigla_componente[20];

    char produttore[20];

    char tipo[20];

    int costo;

} materiale;

typedef enum {falso, vero} boolean;

void main()

```

```

{
FILE *f;

materiale mat;

int i;

f = fopen(FILE_NAME, "wb");

for(i=0;i<5;i++)
{
printf("Sigla componente: "); scanf("%s", mat.sigla_componente);
printf("Produttore: "); scanf("%s", mat.produttore);
printf("Tipo: "); scanf("%s", mat.tipo);
printf("Costo: "); scanf("%d", &mat.costo);
fwrite(&mat, sizeof(materiale), 1, f);
}
fclose(f);
}

```

Esercizio 5

Si sviluppi in C un sottoprogramma che prende come parametri una lista dinamica contenente interi, una matrice quadrata di interi ed il suo rango massimo (rangoMax. Il rango di una matrice quadrata corrisponde al suo numero di righe e colonne). Il sottoprogramma inserisce nella matrice i valori interi contenuti nella lista, prestando attenzione a distribuirli opportunamente in modo che la matrice risulti quadrata ed il suo rango effettivo sia quello minimo rispetto al numero di elementi contenuti nella lista. Nel caso in cui il numero di elementi della lista sia superiore a rangoMax x rangoMax, si trascurino gli ultimi elementi della lista. Nel caso in cui non esiste un rango t.c. $0 \leq \text{rango} \leq \text{rangoMax}$ e $\text{rango} \times \text{rango} = \text{numero degli elementi della matrice}$, si individui un valore per il rango t.c.

$(\text{rango}-1) \times (\text{rango}-1) < \text{numero degli elementi della matrice} < \text{rango} \times \text{rango}$

Inoltre, si inserisca il valore 0 in tutti gli elementi della matrice che altrimenti risulterebbero vuoti. Per esempio, nel caso in cui la lista contenga i valori: {1, 2, 3, 4, 5, 6, 7}, la matrice risultante sarebbe la seguente:

1	2	3
4	5	6
7	0	0

Il sottoprogramma deve restituire al chiamante il rango effettivo della matrice così costruita.

a) Si definisca l'intestazione del sottoprogramma e si sviluppi il suo corpo. Nello sviluppo del sottoprogramma si faccia uso delle funzioni:

```
/* restituisce il numero di elementi contenuti nella lista passata  
come parametro */
```

```
int contaElem(Lista l);
```

```
/* restituisce il rango della matrice che può contenere il numero di  
elementi numElem a patto che questo sia inferiore a rangoMax *  
rangoMax. In caso contrario, restituisce rangoMax */
```

```
int trovaRango(int numElem, int rangoMax);
```

b) Si implementi la funzione trovaRango.

Soluzione

La soluzione proposta implementa anche alcuni sottoprogrammi non richiesti dalla traccia. Si tratta di un programma autocontenuto che potete eseguire. Le stampe nei vari sottoprogrammi hanno il solo scopo di mostrare l'avanzamento del programma.

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

```
#include <stdlib.h>
```

```
#include <math.h>
```

```
#define RANGO_MAX 10
```

```
typedef int matrice[RANGO_MAX][RANGO_MAX];
```

```
typedef struct El {
    int info;
    struct El *next;
} ElemLista;
```

```
typedef ElemLista *Lista;
```

```
void inserisciInTesta(Lista *li, int a)
```

```
{
    Lista temp;
    printf("Sono in InserisciInTesta\n");
    temp = malloc(sizeof(ElemLista));
    temp->info = a;
    temp->next = *li;
    *li = temp;
}
```

```
/* restituisce il numero di elementi contenuti nella lista passata come parametro
*/
```

```
int contaElem(Lista l)
```

```
{
    int cont = 0;
    printf("Sono in contaElem\n");
    while(l!=NULL)
    {
        cont++;
        l = l->next;
    }
}
```

```
return cont;
```

```
}
```

```
/* restituisce il rango della matrice che può contenere il numero di elementi  
numElem a patto che questo sia inferiore a rangoMax * rangoMax. In caso contrario,  
restituisce rangoMax */
```

```
int trovaRango(int numElem, int rangoMax)
```

```
{
```

```
int ris;
```

```
printf("Sono in trovaRango\n");
```

```
ris = sqrt(numElem);
```

```
printf("rango: %d\n", ris);
```

```
if(ris <=rangoMax)
```

```
    if(ris * ris == numElem)
```

```
        return ris;
```

```
    else return ris+1;
```

```
else return rangoMax;
```

```
}
```

```
int copiaInMatrice(Lista l, matrice m, int RangoMax)
```

```
{
```

```
int numElem, rango, i, j;
```

```
numElem = contaElem(l);
```

```
printf("numero elementi: %d\n", numElem);
```

```
rango = trovaRango(numElem, RangoMax);
```

```
printf("rango: %d\n", rango);
```

```
for(i=0; i<rango; i++)
```

```
    for(j=0; j<rango; j++)
```

```
        if(l!=NULL) {m[i][j]=l->info; l = l->next;}
        else m[i][j] = 0;
    return rango;
}
```

```
Lista creaLista() {return NULL;}
```

```
void main()
{
    Lista l;
    int i, j, rango;
    matrice m;

    l= creaLista();
    for(i=0; i<8;i++)
        inserisciInTesta(&l, i+1);
    rango = copiaInMatrice(l, m, RANGO_MAX);
    for(i=0; i<rango; i++)
    {
        for(j=0; j<rango; j++)
            printf("%d ", m[i][j]);
        printf("\n");
    }
}
```