Esercizi Aggiuntivi su Struct

Credits Prof. Campi

Esercizio 1

Definire tipi di dato per un PRA:

- Il tipo dati Motoveicolo rappresenta i dati di un motoveicolo. Questi dati si compongono di:
  - targa del motoveicolo (7 lettere)
  - marca del motoveicolo (massimo 15 caratteri),
  - modello (massimo 20 caratteri),
  - cilindrata (in cc),
  - potenza (in kW),
  - categoria (massimo 16 caratteri).

Soluzione

```
typedef struct {
  char targa[7] ;
  char marca[15] ;
  char modello[20] ;
  int cilindrata;
  float potenza;
  char categoria[16];
} Motoveicolo;
```

Esercizio (cont)

- Supponiamo un char occupi 1 byte, un int 2 byte e un float 4 byte
- Quanto occupano le strutture dati in tutto?

- 7+15+20+2+4+16=64

Esercizio (cont)

- Il tipo dati Proprietario rappresenta i dati di una persona (il proprietario del motoveicolo):
  - nome (massimo 30 caratteri),
  - cognome (massimo 40 caratteri),
  - codice fiscale (16 caratteri).
- Il tipo dati VocePRA rappresenta una singola voce nel registro automobilistico; una voce si compone di 2 elementi: i dati del proprietario del motoveicolo ed i dati del motoveicolo stesso.

Soluzione

```
typedef struct {
  char nome[30];
  char cognome[40];
  char codice_fiscale[16];
} Proprietario;

typedef struct {
  Motoveicolo motoveicolo;
  Proprietario proprietario;
} VocePRA;
```

Esercizio (cont)

- Il tipo dati PRA rappresenta un tipo adatto a contenere i dati di un PRA. Questo tipo di dati è un elenco di voci del PRA (si suppone che un PRA non possa contenere più di 10.000 elementi), più un contatore che dice quante voci sono effettivamente presenti nel PRA.

Soluzione

```
typedef struct {
  VocePRA elementi[10000];
  int n_elementi;
} PRA;
```

Esecizio (cont)

Scrivere un programma che estrare l’automobilista con l’auto con cilindrata maggiore

Soluzione

```
PRA p;int maxcc=0;int i;
Proprietario pr;
.....

for(i=0;i<p.n_elementi;i++){
    if(p.elementi[i].motoveicolo.cilindrata>maxcc){
      maxcc=p.elementi[i].motoveicolo.cilindrata;
      pr=p.elementi[i].proprietario;
}
```

Esercizio (cont)

Scrivere un programma che estrare l'automobilista con la somma delle cilindrate delle sue auto maggiore

Soluzione

```
PRA p;int maxcc=0;int i; Proprietario pr;
  ..................
for(i=0;i<p.n_elementi;i++){
 currentcc=0;
 for(j=0;j<p.n_elementi;j++){
   //if(p.elementi[i].proprietario==p.elementi[j].proprietario)
   loStesso=1;
   for(k=0;k<16 && loStesso==1;k++) {
     if(p.elementi[i].proprietario.codice_fiscale[k]!=p.elementi[j].proprietario.codice_fiscale[k]) {
       loStesso=0;
     }
   }
   if(loStesso==1) // non è mai diventato 0, quindi nessuna cifra del codice fiscale è diversa
      currentcc+=p.elementi[j].motoveicolo.cilindrata;
 }
 if(currentcc>maxcc){
     maxcc=currentcc;
     pr = p.elementi[i].proprietario;
 }
}
```

Esercizio 2

- Scrivere un programma che svolga le seguenti operazioni:
  - Acquisisca informazioni relative a caratteristiche fisiche di 10 persone.
  - Ogni informazione è composta da peso, altezza ed età.
  - Terminata la fase di acquisizione, stampi sullo schermo le informazioni relative a tutte le persone per le quali il valore dell'età è <= 20.

```
#include <stdio.h>
#define MAX_PERS 10
#define MAX_ETA 20
typedef struct { int peso; int alt; int eta; } dati;
int main() {
    dati arch[MAX_PERS];
    int i;
    for(i=0;i<MAX_PERS;i++)  {
        printf("Altezza: "); scanf("%d", &(arch[i].alt));
        printf("\nPeso: "); scanf("%d", &(arch[i].peso));
        printf("\nEta': "); scanf("%d", &(arch[i].eta));
    }
    printf("\n persone con eta' <= %d", MAX_ETA");
    for (i=0; i<MAX_PERS; i++)
       if(arch[i].eta <= MAX_ETA)
            printf("\n %d, %d, %d", arch[i].alt,arch[i].peso,arch[i].eta);
    return 0;
}
```

Esercizio (cont.)

- Modificare ora il programma precedente facendo in modo che gli elementi dell'array vengano ordinati in base all'età in ordine crescente

```
#include <stdio.h>
#define MAX_PERS 10
#define MAX_ETA 20
typedef struct { int peso; int alt; int eta; } dati;
int main() {
    dati arch[MAX_PERS],temp; int i, j;
    for(i=0;i<MAX_PERS;i++)  {
        printf("Altezza: "); scanf("%d", &(arch[i].alt));
        printf("\nPeso: "); scanf("%d", &(arch[i].peso));
        printf("\nEta': "); scanf("%d", &(arch[i].eta));
    }
    for(i=0;i<MAX_PERS-1;i++) {
        for(j=0;j<MAX_PERS-1-i;j++) {
            if(arch[j].eta > arch[j+1].eta)
            { temp=arch[j+1]; arch[j+1]=arch[j]; arch[j]=temp }
        }
    }
    printf("\n persone con eta' <= %d", MAX_ETA);
    for (i=0; i<MAX_PERS; i++)
        if(arch[i].eta <= MAX_ETA)
            printf("\n %d, %d, %d", arch[i].alt,arch[i].peso,arch[i].eta);
    return 0;
}
```

Esercizio 3

- Si definisca il tipo **Corso**, che permetta di rappresentare informazioni relative a corsi di lingue. In particolare, per ogni corso è necessario rappresentare i seguenti dati:
  - **Lingua**: una sequenza di 10 caratteri;
  - **Livello**: un intero;
  - **NumeroIscritti**: un intero;
  - **NomeInsegnante**: una sequenza di 15 caratteri;
  - L'insieme di **Studenti** (al massimo 10). Ogni studente è un dato a sua volta strutturato, composto da:
    - **NomeStudente** (sequenza di 10 caratteri)
    - **CognomeStudente** (sequenza di 10 caratteri)
    - **Età** (numero intero)
- Si dichiari inoltre la variabile **ScuolaLingue**, in grado di rappresentare l'insieme dei corsi di una certa scuola. Una scuola raggruppa al massimo 30 corsi.

```
#define maxclassi 30
#define maxstud 10
typedef struct { char nomestud[100];
                 char cognstud[100];
                 int eta;
} Studente;

typedef struct { char lingua[10] ;
                 int liv;
                 int numiscritti;
                 char nomeinsegn[15];
                 Studente alunni[maxstud];
} Classe ;
Classe ScuolaLingue[maxclassi];
```

Esercizio (cont.)

- Scrivere in linguaggio C la parte di algoritmo per calcolare l'età media degli studenti iscritti ai corsi di lingua inglese (supponendo di aver inizializzato la variabile **ScuolaLingue** con le informazioni relative a 30 corsi).

```
#include <stdio.h>
#define maxclassi 4 /* invece di 30 */
#define maxstud 2 /* invece di 10 */
typedef struct { char nomestud[10];
                 char cognstud[10];
                 int eta;
}Studente;
typedef struct { char lingua[10] ;
                 int liv;
                 int numiscritti;
                 char nomeinsegn[15];
                 Studente alunno [maxstud];
} Classe ;
```

```
/* inserimento dati di 4 classi */
Classe ScuolaLingue[maxclassi]=
{{"inglese",1,2,"maestr1","alice","ferrari",8,"luca",
   "bianchi",8,},
{"inglese",2,2,"maestr2","guzzi","tommi",18,"gialli",
   "fede",18},
{"inglese",1,2,"maestr3","innoi","ldddd",10,"sssss",
   "ddddd",10},
{"spagnolo",2,2,"maestr4","ducai","ddddd",5,
   "wwwww","qqqqq",5}};
```

```
int main () {
    int i,j,cont=0;
    float somma=0;
    char linIng[]="inglese";
    /* stampo tutti gli alunni di tutte le classi con eta' */
    for (i=0;i<maxclassi;i++)
          for(j=0;j<maxstud;j++)
                printf("\n %s %s eta':%d",
                      ScuolaLingue[i].alunno[j].nomestud,
                      ScuolaLingue[i].alunno[j].cognstud,
                      ScuolaLingue[i].alunno[j].eta);
```

```
    for (i=0;i<maxclassi;i++) {
        if(strcmp(linIng,ScuolaLingue[i].lingua)==0){
            for(j=0;j<ScuolaLingue[i].numiscritti;j++) {
                cont++;
                somma+=ScuolaLingue[i].alunno[j].eta;
            }
        }
    }

    if (cont>0)
        printf("\nl'eta' media e':%f", somma/cont);
    else
        printf("\nnessun corso di %s",linIng);
    return 0;
}
```

Esercizio 4

- **Le seguenti dichiarazioni definiscono tipi di dati che descrivono i candidati per l'assunzione in un'azienda.**

```
typedef char lingua[20];
typedef char stringa[20];
typedef struct {
        lingua l;
        int livello; /* numero da 1 a 5 con 1=basso 5=alto */
} linguaParlata;
typedef struct {
        stringa nome, cognome, diploma, laurea;
        linguaParlata lingue[5];
        int nLauree;
        int anniEsperienza;
} persona;

/* definizioni delle variabili */
persona persone[40];
persona personeScelte[40];
```

- Si scriva un frammento di codice, che includa eventualmente anche le dichiarazioni di ulteriori variabili e tipi, che copi nella parte iniziale del vettore personeScelte (senza lasciare buchi) le persone che parlano inglese con un livello superiore a 3 oppure soddisfano entrambi i seguenti requisiti: hanno non meno di una laurea e un numero di anni di esperienza non inferiore a tre.

```
int iPersone, iPersoneScel, iLingue;
iPersoneScel=0;
for(iPersone=0; iPersone<40; iPersone++) {
    if (persone[iPersone].nLauree>=1 && persone[iPersone].anniEsperienza>=3){
        personeScelte[iPersoneScel]=persone[iPersone];
        iPersoneScel++;
    } else
        for (iLingue=0; iLingue<5; iLingue++)
            if(strcmp(persone[iPersone].lingue[iLingue].l,"inglese")==0 &&
                persone[iPersone].lingue[iLingue].livello >3) {
                personeScelte[iPersoneScel]=persone[iPersone];
                iPersoneScel++;
            }
}
```

Esercizio 5

- Si scriva un programma C che legga due serie di dati e li memorizzi in due vettori di strutture. Nel primo vettore S (di dimensione 3) vengono memorizzati dati del tipo: <matricola, nome, cognome>. Si noti che la matricola identifica univocamente uno studente e che non ci sono due strutture che contengono lo stesso numero di matricola.
- Nel secondo vettore E (di dimensione 7) vengono memorizzati dati del tipo: <matricola, esame, voto>. Possono esserci più record con lo stesso numero di matricola che denotano diversi esami fatti dallo stesso studente.
- Si scriva un programma che tramite opportune procedure legga i dati in ingresso e li inserisca nei due vettori. Successivamente per ogni studente con matricola X contenuto nel vettore S sommi tutti i suoi voti ottenuti negli esami contenuti nel vettore E.

```
#include <stdio.h>

#define DIMS 3
#define DIME 7

typedef struct {
   int matricola;
   char Nome[20];
   char Cognome[20];
} studente;

typedef struct {
   int matricola;
   char esame[20];
   int voto;
} esami;
```

```
main(){
   int i, j, S=0;
   studente S[DIMS];
   esami E[DIME];
   //leggi gli studenti
   for(i=0;i<DIMS;i++){
      printf("Inserisci Matricola, Nome, Cognome\n");
      scanf("%d",&S[i].matricola);
      for(j=0;j<i;j++)
            trovato=0;
            if(S[i].matricola==S[j].matricola) {
                        printf(“Matricola doppia”);
                        i--; trovato=1;}
       if{trovato==0)
            scanf("%s",S[i].Nome);
            scanf("%s",S[i].Cognome); }
   }
   //leggi esami
   for(i=0;i<DIME;i++){
      printf("Inserisci Matricola, Esame, Voto\n");
      scanf("%d %s %d ",&E[i].matricola,E[i].esame,&E[i].voto);
   }
```

```
   //somma
   for(i=0;i < DIME;i++)
      if (E[i].matricola == X)
         S = S + E[i].voto;

   // cerco dati studente e stampo
   for(i=0; i< DIMS;i++){
      if (S[i].matricola == X) {
         printf("Studente %s %s\n", S[i].Cognome, S[i].Nome);
         printf("Somma voti: %d\n",S);
      }
   }
}
```

Esercizio (tde 12-11-2010)

- Si considerino le seguenti dichiarazioni di tipi e variabili, che definiscono le strutture dati per rappresentare dei rilevamenti meteorologici effettuati per 30 comuni lombardi nell'intero anno 2009
- Scrivere un frammento di codice, definendo eventuali variabili aggiuntive, che trovi tutte le città che nel mese di dicembre hanno registrato almeno un rilievo in cui il livello di pioggia caduta è superiore a quello memorizzato nella variabile sogliaP. Per ognuna di tali città si visualizzi su un'unica riga il nome e la media della quantità di pioggia caduta calcolata sull'intero anno.

```
typedef struct {
      unsigned int giorno; /* tra 1 e 31 */
      unsigned int mese; /* tra 1 e 12 */
      float livelloP; /* pioggia caduta */
} rilievo;

typedef struct      {
      char comune[20]; /*nome del comune*/
      rilievo rilievi[365]; /* max 365 rilievi, non è detto che i
                        /* rilievi abbiano cadenza giornaliera */
       int nRilievi; /* numero di rilievi effettivamente
                  /* registrati in rilievi */
} rilieviComune; /* dati monitorati nell'anno per un singolo comune */

typedef rilieviComune rilieviGlobali[30]; /* dati monitorati nell'anno
                                  /* per tutti i 30 comuni */
rilieviGlobali dg;
float sogliaP;  /* Si assuma che le variabili sogliaP e dg siano state inizializzate
                                                      attraverso istruzioni qui non mostrate */
```

```
int i,j,n,trovato;
float totaleP, media;

for (i=0; i<30; i++) {
  n=dg[i].nRilievi;
  trovato=0;
  totaleP=0.0;
  for (j=0; j<n; j++) {
    totaleP += dg[i].rilievi[j].livelloP;
    if ((dg[i].rilievi[j].mese==12) && (dg[i].rilievi[j].livelloP > sogliaP))
      trovato=1;
  }
  if (trovato) {
    media = totaleP/n;
    printf("Il comune di %s nel 2009 ha registrato un livello medio di
            precipitazioni di %f", dg[i].comune, media);
  }
}
```

Esercizio (tde 13-11-2009)

- Si scriva un programma C per la gestione di una scuola di ballo. Ogni classe consiste di un insieme di persone e di due istruttori. Ogni persona ha un nome, un cognome, un sesso e un'età
- Si definiscano le strutture dati necessarie allo sviluppo del programma (strutture, array, ecc.)
- Si implementi un frammento di codice che partendo da una struttura di tipo "classe" stampa l'età media degli studenti di sesso maschile

```
#define N 100

typedef struct {
    char nome[N], cognome[N];
    char s;
    int eta;
} persona;

typedef struct {
    persona studenti[N];
    persona istruttori[2];
    int nIscritti;
} classe
```

```
classe c;
…
float media=0.0;
int i=0;tot=0, quanti=0;
for(i=0;i<c.nIscritti;i++) {
    if(c.studenti[i].s==‘m’)
        tot+= c.studenti[i].eta;
        quanti++;
}

if(quanti==0)
    return -1.0;

media=(float)tot/quanti;

printf(“%d”,media);
```

Esercizio

- Si scriva un programma C gestisce i dati di cantanti rappresentati con questa struttura dati:

```
typedef struct {
        char cognome[N];
        char nome[N];
        int dischi;
} Tabella;
```

- Il programma deve gestire il menù

Che operazione vuoi eseguire?

1. **Inserisci nuovo record**
2. **Ordina record**
3. **Visualizza record**
4. **Esci**

```
#include <stdio.h>
#include <stdlib.h>
#define RIGHE 5
#define N 15

//definizione di un nuovo tipo
typedef struct {
  char cognome[N];
  char nome[N];
  int dischi;
} Tabella;

int main() {
  int i, j, record_inseriti=0;
  Tabella tabella[RIGHE], temp;//temp usato per ordinare
```

```
do {
 printf("Che operazione vuoi eseguire? ");
 printf("\n\t 1. Inserisci nuovo record\n\t 2. Ordina record");
 printf("\n\t 3. Visualizza record\n\t 4. Esci\n\t ");
 scanf("%d", &operazione);
 getchar(); //scarta il carattere di a capo

 switch(operazione) {
   case 1:
     if(record_inseriti == RIGHE)
         printf("Hai inserito il numero massimo di record all'interno della tabella.");
     else {
         printf("\nInserisci il cognome dell'artista: ");
         scanf("%s",tabella[record_inseriti].cognome);
         printf("Inserisci il nome dell'artista: ");
         scanf("%s",tabella[record_inseriti].nome);
         printf("Inserisci il numero di dischi venduti: ");
         scanf("%d", &tabella[record_inseriti].dischi);
         getchar();//consuma l'invio
        record_inseriti++;
     }
     break;
```

```
case 2:
   for(i = 0; i < record_inseriti - 1; i++) {
      for(j = i+1; j < record_inseriti; j++) {
         if(tabella[i].dischi < tabella[j].dischi) {
            temp = tabella[i];
            tabella[i] = tabella[j];
            tabella[j] = temp;
         }
      }
   }
   printf("\nVettore ordinato secondo ordinamento decrescente dei dischi");
   break;
 case 3:
   if(!record_inseriti)
      printf("\nNon sono presenti record all'interno della tabella.");
   else {
      printf("\n%s %s %s", "Nome", "Cognome", "Dischi");
      for(i = 0; i < record_inseriti; i++) {
         printf("\n%s %s %d“,tabella[i].cognome,tabella[i].nome,tabella[i].dischi);
   }
   break;
```

```
   case 4:
         break;
      default:
         printf("\nHai effettuato una scelta non valida");
    }

    if(operazione != uscita) {
      printf("\n\nPremi INVIO per continuare.");
      getchar();
    }

  } while(operazione != uscita);

  printf("\nPremi INVIO per uscire.");
  getchar();
  return 0;
}
```

Esercizio (tde 7-2-2012)

- Un archivio di offerte di lavoro è organizzato nel seguente modo: le offerte sono disposte in un array e sono ordinate per codice. Le strutture dati utilizzate sono le seguenti:

```
typedef struct d { int giorno, mese, anno; } Data;
typedef struct job { char codice[N], descrizione[N], titoloStudioRichiesto[N];
                     int stipendio;
                     Data dataInserimento;
                     int valido; } OffertaDiLavoro;
typedef OffertaDiLavoro Offerte[10000];
```

- Il campo “valido” serve a dire se la casella dell’array contiene contenuto valido (nel caso l’attributo ha valore 1) o è da considerarsi vuota (nel caso l’attributo ha valore 0).
- Si codifichi in C un frammento di codice che invalida dalla lista delle offerte quelle offerte di cui ne esiste una più recente con identica descrizione.

Esercizio (tde 5-2-2013)

- Un archivio di spettacoli teatrali è organizzato nel seguente modo: gli spettacoli sono disposti in un array e sono ordinati per codice. Le strutture dati utilizzate sono le seguenti:

```
typedef struct d { int giorno, mese, anno; } Data;
typedef struct { char codice[N], titolo[N] , descrizione[N];
                 int costoBiglietto;
                 Data data;
                 int valido; } Spettacolo;
typedef Spettacolo Spettacoli[10000];
```

- Il campo “valido” serve a dire se la casella dell’array contiene contenuto valido (nel caso l’attributo ha valore 1) o è da considerarsi vuota (nel caso ha valore 0).
- Si codifichi in C un frammento di codice che riceve l’array *spe* che contiene tutti gli spettacoli e copia (senza lasciare buchi) nell’array *speEconomici* gli spettacoli con un costo del biglietto inferiore a 20 euro con data posteriore alla data di oggi (presente in una variabile *oggi* di tipo *Data*).