Esercitazioni di Fondamenti di Informatica - Lez. 5

14/11/2019

Esercizi in C su puntatori, funzioni e passaggio parametri per valore e indirizzo.
*I codici e le soluzioni degli esercizi sono nella cartella **codice5***

----------**puntatori**

ricordiamo che:

ad una **variabile** --> corrisponde un nome, una locazione di memoria (un numero di byte corrispondente alla loro dimensione) e l'indirizzo della locazione di memoria.

Per esempio in C l'operatore **&**, che abbiamo utilizzato nella funzione *scanf*, ci **restituisce l'indirizzo di memoria di una varibile**.

I **puntatori** sono una speciale categoria di variabili abilitate a contenere l'indirizzo di altri elementi nel programma, come variabili, strutture, funzioni, array, file, ecc…

```
*pa  contiene un indirizzo,punta ad a
&a   ritorna il suo indirizzo in pa
```

Appena dichiarati i puntatori sono inizializzati al valore speciale NULL

Facciamo alcuni esempi di dichiarazione e uso delle variabili puntatore:

```
int *pa;   // puntatore ad una variabile intera

char *pc;  //puntatore ad una variabile char

float *pf; //puntatore ad una variabile float

void *p;   //puntatore a void

int *pa, a; // vicino ad altre variabili di tipo int

pa=&a      //il puntatore pa assume l'indirizzo di a

*pa = 10    // nell'indirizzo puntato metti valore 10
```

1. **Stampiamo l'indirizzo della variabile *a***

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

int main() {
   int *pa, a = 10;
   pa = &a; //  dereferenziazione di pa, cioè *pa indica  a

   printf("Indirizzo della variabile a: %p \n", pa);
   printf("Valore di a all'indirizzo puntato da *pa : %d \n", *pa);
   // si può assegnare il valora attraverso il puntatore

     *pa = 0;
      printf("Valore variabile: %d ", a);
   return 0;
}
```

Assegnazioni tra puntatori in C

Ora modifichiamo l'esempio di prima : **assegnamo** alla seconda variabile **puntatore pb** (*pb) il **contenuto di pa** e quindi **l'indirizzo** di memoria della variabile **a**.

In questo caso *pb=10 è equivalente a scrivere *pa=10 e a=10;

Una volta compilato il codice sotto, esso stamperà la variabile **a** vale 10.

```
int main(void) {
   int *pa, *pb, a;

   pa=&a;
   pb=pa;     // attenzione devono essere compatibili, cioè devono riferirsi
              //      allo stesso tipo di dati per avere il risultato desiderato
   *pb=10;

   printf("Il valore di a e': %d ", a);
      return 0;
}
```

Nell'esempio sopra abbiamo assegnato ad un puntatore un altro puntatore (pb=pa) ma se vogliamo trasformare un puntatore in modo che si riferisca ad un tipo di dati differente allora dovremmo utilizzare il **cast**.

```
char *pc;
int *pa;
pa=(int *) pc;    //in questo caso dunque il cast rende possibile l'assegnazione tra
                    puntatori
```

----------funzioni

2. **Scrivere i prototipi delle seguenti funzioni**.
   I. funzione ipotenusa riceve due argomenti in virgola mobile in doppia precisione e ritorna un risultato in doppia precisione.
   II. funzione minoreditre che riceve tre argomenti interi e ritorna il valore del minore.
   III. funzione istruzioni che non ha parametri in ingresso e non ha parametri in uscita.
   IV. funzione convertiFloat2Intero che riceve un argomento foat e ritorna in uscita il valore intero più vicino.

Soluzione:

```
0 double ipotenusa ( double lato1 , double lato2)
1 int minoreditre ( int a , int b , int c ) ;
2 void istruzioni( void ) ; /* anche void istruzioni () sarebbe valido */
3 int convertiFloat2Intero( float NumeroFloat ) ;
```

3. **Scrivere una funzione in grado di scambiare il contenuto di due variabili intere passate come parametri**

Se ho le variabili intere a = 1 e b = 2, chiamando una funzione ***scambiaInteri(a,b)*** le variabili avranno valore scambiato a = 2 e b = 1.

```c
#include<stdio.h>

void scambiaInteri(int *p1, int *p2);  // dichiaro le funzioni che userò

int main(){

  int n1, n2;

  /* Inserisco i due valori */
  scanf("%d",&n1);
  scanf("%d",&n2);

  /* e li stampo */
  printf("n1=%d n2=%d\n",n1,n2);

  /* chiamo la funzione scambiaInteri
     e scambio i valori
     N.B. i valori di n1 ed n2 sono  passati per indirizzo
  */
  scambiaInteri(&n1,&n2);

  /* stampo di nuovo i contenuti di n1 e n2 */
  printf("n1=%d n2=%d\n",n1,n2);

}
// implemento la funzione
void scambiaInteri(int *p1, int *p2){
  int tmp; /* variabile temporanea*/
  tmp = *p1;
  *p1=*p2;
  *p2=tmp;
}
```

4. **Trova gli errori nel seguente codice per il calcolo del massimo di una sequenza di interi.**

```c
#include<stdio.h>

void calcolaMassimo(int numeri[],int LUN);

int main(){

  int LUN = 10;
  int numeri[10] = {2,3,4,5,6,412,23,5,1,4};
  int massimo;

  massimo = calcolaMassimo(numeri,LUN);
}

void calcolaMassimo(int numeri[],int LUN){

  int massimo;
  int i;

  massimo = numeri[0];
  for(i=0;i<LUN;i++){
        if (massimo == numeri[i]){
              massimo = numeri[i];
        }
  }
}
```

Soluzione:

Oltre alla **condizione sbagliata** per l'aggiornamento del valore massimo, la funzione **non ritorna il valore** massimo ritornato. Per questa secondo errore ci sono due possibili soluzioni: dovendo ritornare un solo valore intero, posso modicare il valore di ritorno da void a int e ritornare il massimo con return(massimo). Il codice risultante sarebbe:

soluzione 4.1

```c
#include<stdio.h>
int calcolaMassimo(int numeri[],int LUN);

int main(){

  int LUN = 10;
  int numeri[10] = {2,3,4,5,6,412,23,5,1,4};
  int massimo;

  massimo = calcolaMassimo(numeri,LUN);
  printf("\nmassimo-> %d\n",massimo);
}

int calcolaMassimo(int numeri[],int LUN){

  int massimo;
  int i;
  massimo = numeri[0];
  for(i=0;i<LUN;i++){
        if (massimo < numeri[i]){    //  <---
              massimo = numeri[i];
        }
  }
  return massimo;
}
```

soluzione 4.2

```c
#include<stdio.h>
// la funzione riceve il riferimento alla variabile,

void calcolaMassimo(int numeri[],int LUN,int *massimo);

int main(){

int LUN = 10;
int numeri[10] = {2,3,4,5,6,412,23,5,1,4};
int massimo;

calcolaMassimo(numeri,LUN,&massimo);//  passo l'indirizzo e la modifica si riperquote
printf("\nmassimo-> %d\n",massimo);
}

void calcolaMassimo(int numeri[],int LUN,int *massimo){
    int i;
    int temp;
    *massimo = numeri[0];
    for(i=0;i<LUN;i++){
           if (*massimo < numeri[i]){
                  *massimo = numeri[i];
           }
    }
}
```

5. **Scrivere il codice che, data una sequenza di numeri interi, calcoli la media dei valori attraverso una chiamata di funzione.**

```c
#include<stdio.h>
#define MAX 5

float calcolaMedia(int *sequenza,int n);


int main(){

  int a[MAX];

  printf("\nInserisci 5 numeri interi e premi invio: \n");
  scanf("%d %d %d %d %d",&a[0],&a[1],&a[2],&a[3],&a[4]);

  float media = calcolaMedia(a,MAX);
  printf("\nmedia = %f\n",media);
}

float calcolaMedia(int *sequenza,int n){

/* gli array sono sempre copiati per riferimento,
   devo quindi passare anche il numero di elementi della sequenza */

  float media = 0;
  for (int i=0;i<n;i++){
         media += sequenza[i];
  }
  media = media/n;
  return media;
}
```

6. *Scrivere una funzione che, **data una sequenza di numeri interi** e un numero intero A, ritorni due elementi: un booleano che **indichi se A è presente** nella sequenza di numeri **e l'indice** in cui A si trova nella sequenza (o il valore -1 se non è nella sequenza).*

   Soluzione:

```c
#include<stdio.h>
#define MAXLEN 5

/* definiamo il tipo booleano bool */

typedef enum{FALSE=0,TRUE=1} bool;

/* la funzione cercaIntero vuole in input:
  -array di interi in cui cercare
  -lunghezza dell'array di interi
  -il numero da cercare
  -la variabile dove salvare l'indice
*/

bool cercaIntero(int *sequenza, int len, int n, int *indice_n);

/* scrivo anche una procedura utile per stampare il risultato della funzione
cercaIntero  *
/
void printCercaIntero(int *sequenza,int len, int n);

int main(){

  int sequenza[MAXLEN] ={1,2,3,4,5};

  int A=2;
  printCercaIntero(sequenza,MAXLEN,A);
  A=6;
  printCercaIntero(sequenza,MAXLEN,A);
}

bool cercaIntero(int *sequenza, int len, int n, int *indice_n){

  bool trovato = FALSE;
  *indice_n = -1;

  for (int i=0; i < len; i++){
        if(sequenza[i]==n){
              *indice_n = i;
              trovato = TRUE;
              break;
        }
  }
  return trovato;
}

void printCercaIntero(int * sequenza, int len, int n){

  int indice;
  if(cercaIntero(sequenza,len,n,&indice)){
        printf("Il valore %d si trova nella sequenza in indice %d\n",
              n, indice);
  }
```

```
  else{
        printf("Il valore %d non si trova nella sequenza\n",n);
  }
}
```

7. **Dato un oggetto di tipo Libreria defnito attraverso la seguente struttura dati:**

```
 typedef struct {
 char nome[STRLEN] ;
 double costo ;
 int giacenza ; /* numero di copie presenti */
  Libro ;
}
 typedef Libro Libreria [NUMLIBRO] ;
```

Scrivere una funzione che, dato il nome di un libro: controlla se nella libreria ci sono abbastanza copie e se ci sono ancora copie presenti, vende il libro diminuendo il numero di copie presenti e aggiornando un valore di cassa con il prezzo del libro.

Suggerimento:
per controllare se il libro è presente in libreria tramite il nome, si devono compare le due stringhe dei nomi: questo è possibile velocemente tramite la funzione strcomp nella libreria <string.h>.

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

#define NUMLIBRO 3
#define STRLEN 30

typedef struct{
  char nome[STRLEN];
  double costo;
  int giacenza;
  } Libro;

typedef Libro Libreria[NUMLIBRO];
typedef enum {FALSE=0,TRUE=1} bool;

Libreria libreria ={
  {"Informatica\0",10.0,3},
  {"Matematica\0",20.0,5},
  {"Statistica\0",5.0,2}
};

double cassa = 0.0;
void librocomprato(char*nome);

int main(){

char nome[STRLEN] = "Informatica\0";

librocomprato(nome);
librocomprato(nome);
librocomprato(nome);
librocomprato(nome);

printf("In cassa hai %f euro\n",cassa);
return 0;
}
```

```
void librocomprato(char*nome){

  /* variabile che indica se  il libro e' nella libreria
  inizializzata a FALSE*/

  bool trovato = FALSE;

  for(int i=0; i<NUMLIBRO; i++){

  /* compariamo i nomi dei libri */
        if(strcmp(libreria[i].nome,nome)==0){
              trovato = TRUE;

              /* guardiamo se il libro e' disponibile*/
              if(libreria[i].giacenza > 0){
                    printf("Hai venduto %s con prezzo %f\n",
                          nome,libreria[i].costo);
                    cassa += libreria[i].costo;
                    libreria[i].giacenza--;
              }
              else{
                    printf("Libro non in giacenza\n");
              }
              break;
        }
  }
  if(trovato == FALSE){
        printf("Libro non trovato.\n");
  }

}
```

8. Tratto dall' ESAME 10/09/2012: **Scrivere in C una funzione, senza parametri e che non restituisce nulla, legge una sequenza di numeri terminata dallo 0. La funzione deve considerare solo i valori dei numeri letti che sono multipli di 3, calcolarne la mediana (non la media) e stamparne a video il valore.**
   Si ricorda che la mediana di una sequenza ordinata di numeri interi è il valore centrale, se la sequenza ha cardinalità dispari, oppure la media tra i due numeri centrali, se la cardinalità è pari.

```
#include<stdio.h>
#define DIM 100

void ordina(int a[], int d);
void StampaMediana();

int main(){
   StampaMediana();
}

/* ordina un array dal minore al maggiore */
void ordina(int a[], int d){

   int i,j,t;

   for(i=0; i<d; i++){
         for(j=i+1;j<d;j++){
               if(a[i]>a[j]){
                     t=a[j];
```

```
                        a[j]=a[i];
                        a[i] = t;
                }
        }
   }
}

void StampaMediana(){

   int n=1,i=0,d;
   int a[DIM];
   float t1,t2,med;

   while(n!=0 && i!=(DIM-1)){
          printf("Inserisci un numero:\n");
          scanf("%d",&n);

          if((n!=0) && (n%3)==0 ){
                 a[d++]=n;
          }
   }

   ordina(a,d);

   t1 = a[(d/2)-1];
   t2 = a[d/2];

   if (d%2 == 0){
          med = (t1 +t2)/2;
   }
   else{
          med = t2;
   }

   printf("La mediana e'%f\n",med);
}
```

----------accesso a struct per valore e puntatore

- l'operatore **punto** accede ai valoi dei campi della struttura
- l'operatore freccia **->** sembra fare sostanzialmente la stessa cosa ma viene usato quando si accede alla struttura con il puntatore (con riferimento all'indirizzo e non per valore diretto)

se ho una funzione che inserisce il nume di un giocatore nella struttura Squadra

void inserisciPersona(Squadra *squ , persona p)

per valorizzare i dati di ***squ*** nel corpo della funzione accedo usando il puntatore ai singoli campi

```
squ->giocatori[squ->persona].cognome = p.cognome;
```

```
typedef struct {
    char cognome[20];
    char nome[20];
  } persona;

typedef struct{
  char squadra[20];
  int punteggio;
  struct rec_Allenatore{
    char cognome[20];
    char nome[20];
    int titoli_vinti;
  } allenatore;
  persona giocatori[10];
} Squadra;
```

```
//quando nel codice creo un array di Squadra
Squadra fantacalcio[MAX];

// accedo ai valori delle singole variabili della structcon il punto

char samp[20], cogn[20]
//nome della squadra i-esima
cogn = fantacalcio[i].squadra;

int class = fantacalcio[i].punteggio;
// cognome allenatore squdra i-esima
cogn = fantacalcio[i].allenatore.cognome;
// nome del giocaore j della squadra i-esima
cogn = fantacalcio[i].giocatori[j].nome;
....
```

```
//quando  ho una funzione e passo i parametri per indirizzo
//  nel codice della funzione uso la notazione ->

Squadra fantacalcio[MAX];
....
void inserisciPersona(Squadra *squ , persona p, char nomes[20] )
{
.....
// vado all'indirizzo dei campi squ

 squ->squadra = nomes ;
 squ-> allenatore.cognome  = "Paolo";
// inserisco in coda a giocatori[]
 squ->giocatori[squ->persona].cognome = p.cognome;

 squ-> punteggio = squ->punteggio+1;
```

```
#include <stdio.h>

/* dichiaro la struttura  */
typedef struct student {
                        char srollno[10];
                        char sclass[10];
                        char name[25];
                        char fname[25];
        }Student;

void dot_access(Student stu);    /* prototipo */

int main(void)
{
    Student a = {"3514", "filosofia", "Cristina", "Porta"};

    printf("Informazioni su stu :\n");
    dot_access(a);    /* viene passata l'intera struttura  'a'  */
    return 0;
}

/* l'intera struttura Student 'a'     è copiata in  Student 'stu' come valore non come
puntatore  */

 void dot_access(Student stu)
{
    /* Let's access members of 'a' using dot operator */

    printf("matricola.: %s\n", stu.srollno);
    printf("facoltà: %s\n", stu.sclass);
    printf("nome: %s\n", stu.name);
    printf("cognome: %s\n", stu.fname);
}
```

9. ESAME 28/11/2013:

Dichiarare in C una struttura per memorizzare la carriera universitaria di uno studente. Ovvero, la struttura deve consentire la memorizzazione di

*nome e cognome dello studente, la matricola, i voti riportati nei 30 esami da sostenere per conseguire la laurea e la media pesata, aggiornata dopo l'aggiunta di ogni esame***.**

La media è pesata perche ogni esame puo` valere 3, 5, 8, 10 o 12 crediti e la media deve tenere conto del "peso" del singolo esame.

Scrivere anche una funzione che prende come parametro la struttura dati relativa ad uno studente, consente l'inserimento dei dati relativi ad un esame, aggiorna la media e restituisce la struttura dati aggiornata.

Soluzione:

```c
#include<stdio.h>
#define DIM 20 /* lunghezza campi testuali*/
#define MATR 6 /* lunghezza max matricola */
#define ESAMI 30 /* numero massimo esami */

/* definiamo il campo stringa */
typedef char Stringa[DIM];

/* definiamo il campo matricola */
typedef char Matricola[MATR];

typedef struct{
  int voto;
  int crediti;
}Esame;

/* struttura per uno studente */

typedef struct{
  Stringa nome;
  Stringa cognome;
  Matricola matricola;
  Esame voti[ESAMI];
  int esami;
  float media;
} Studente;
void inserisciEsame(Studente *s , Esame e);

int main(){

// la struttura è costruita inizialmente con soli 3 campi  gli altri non vengono valorizzati

  Studente studente = {"Mario","Rossi","123456"};
  Esame esami[3] = { {25,5},
                     {30,10},
                     {23,10}  };

 inserisciEsame(&studente,esami[0]);
 inserisciEsame(&studente,esami[1]);

printf("La media esami di %s %s e'%f\n",
        studente.nome,studente.cognome,studente.media);
}

void inserisciEsame(Studente *s , Esame e){

  int i, c = 0;
  float v = 0.0;
// per accedere al puntatore usiamo la notazione freccia (arrow notation)

  /* aggiornamento nuovo esame */
   (s->voti[s->esami]).voto = e.voto;
  (s->voti[s->esami]).crediti = e.crediti;
  (s->esami)++;
  /* aggiornamento media studente */

  for(i=0; i < s->esami; i++){
        v+=  ((s->voti)[i].voto * (s->voti)[i].crediti);
```

```
        c+= (s->voti)[i].crediti;
  }
  s->media = v/c;
}
```

10. ESAME 31/08/2017**: Scrivere in C una funzione ed un programma principale:**

La funzione prende *in ingresso una stringa* **e** *restituisce***, nel modo che si ritiene più opportuno, tre valori diversi: la** *lunghezza della stringa, il numero di vocali contenute e la somma della codifca ASCII dei suoi caratteri***.**

Il programma principale deve chiedere all'utente di inserire una stringa, invocare la funzione defnita al punto precedente, usando la stringa inserita, e stampare i risultati restituiti in modo appropriato.

Soluzione 1:

NOTIAMO CHE `AnalisiStringa AnalizzaStringa(Stringa stringa)`

- riceve il passaggio del parametro per riferimento e non per valore: anche se il valore di 'stringa' non verrà modificato (per definizione di Stringa che è un array passa il puntatore)

- il risultato della funzione viene assegnato ad una variabile dichiarata nel main()
- nella Soluzione 2 è chiaro invece il passaggio di parametri per puntatore alle variabili che devono mantenere la modifica al termine della funzione

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

#define MAX_L 30 /* Lunghezza massima stringa */

typedef enum{FALSE,TRUE} bool;
typedef char Stringa[MAX_L];

typedef struct{
        int lunghezza;
        int numeroVocali;
        int sommaAscii;
        } AnalisiStringa;

bool carattereVocale(char a); /* restituisce TRUE se il carattere e' una vocale*/

// dichiaro la funzione usata nel main
AnalisiStringa AnalizzaStringa(Stringa stringa);

int main(){

 Stringa puccio;
 AnalisiStringa analisiStringa;

printf("Inserisci una stringa (max %d caratteri):\n",MAX_L);
scanf("%s",puccio);

// USO LA FUNZIONE E ASSEGNO IL VALORE ALLA STRUTTURA analisiStringa
analisiStringa = AnalizzaStringa(puccio);
```

```
printf("Lunghezza della stringa: %d \nNumero di vocali: %d\nSomma ASCII %d\n",
  analisiStringa.lunghezza,analisiStringa.numeroVocali,
   analisiStringa.sommaAscii);
}

 // fine main

bool carattereVocale(char a){

        if (a =='A' || a == 'E' || a== 'I'|| a == 'O' || a=='U'||
      a == 'a' || a=='e'|| a=='i' || a =='o' || a == 'u'){

               return TRUE;
        }

        else{
                return FALSE;
        }
} // fine carattereVocale

//
AnalisiStringa AnalizzaStringa(Stringa stringa){

     AnalisiStringa analisiStringa;

     int len =0, numeroVocali=0, sommaASCII = 0;

      len = strlen(stringa);
   //    altro modo di calcolare la lunghezza delle stringa

   //    while(stringa[len]!='\0')  { len++;      }


      for(int i =0;i<len;i++){

        sommaASCII += stringa[i];

        if((stringa[i]>='a' && stringa[i]<='z') ||
          (stringa[i]>='A' && stringa[i]<='Z')) {

           if (carattereVocale(stringa[i])) { numeroVocali ++; }
         }
  }
  analisiStringa.lunghezza = len;
  analisiStringa.numeroVocali = numeroVocali;
  analisiStringa.sommaAscii = sommaASCII;

  return analisiStringa;
}
```

Soluzione 2:
In alternativa l'esercizio poteva essere svolto anche senza utilizzare una struttura, ma dichiarando nel main() *: lunghezza, numeroVocali, sommaAscii* come interi e **passandoli per riferimento alla funzione** AnalizzaStringa.

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

#define MAX_L 30 /* Lunghezza massima stringa */

typedef enum{FALSE,TRUE} bool;
typedef char Stringa[MAX_L];

// void AnalizzaStringa(Stringa stringa); soluzione 1

void AnalizzaStringa(Stringa stringa, int * len, int * numeroVocali, int *
sommaAscii);

bool carattereVocale(char a); /* restituisce TRUE se il carattere e' una vocale */

int main(){

   Stringa stringa;
   int lunghezza, numeroVocali, sommaAscii;

   printf("Inserisci una stringa (max %d caratteri):\n",MAX_L);
   scanf("%s",stringa);

   AnalizzaStringa(stringa,&lunghezza,&numeroVocali,&sommaAscii);

   printf("Lunghezza della stringa: %d \nNumero di vocali: %d\nSomma ASCII %d\n",
   lunghezza,numeroVocali,sommaAscii);
}

bool carattereVocale(char a){

        if (a =='A' || a == 'E' || a== 'I'|| a == 'O' || a=='U'||
       a == 'a' || a=='e'|| a=='i' || a =='o' || a == 'u')
        {
          return TRUE;
        }

        else{
          return FALSE;
        }
}

void AnalizzaStringa(Stringa stringa, int * len, int * numeroVocali, int * sommaAscii){

   *len = strlen(stringa);
   /*
   while(stringa[*len]!='\0'){ * len++;}
   */

   for(int i =0;i<(*len);i++){

          *sommaAscii+= stringa[i];

          if((stringa[i]>='a' && stringa[i]<='z') ||
              (stringa[i]>='A' && stringa[i]<='Z')) {
```

```
                if (carattereVocale(stringa[i])){
                        (*numeroVocali)++;
                }
        }
   }

}
```