Introduzione al C

Corso di Fondamenti di Informatica

Ingegneria Clinica

Esercitazione 8

DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Domenico Daniele Bloisi
Raffaele Nicolussi

Menù di oggi

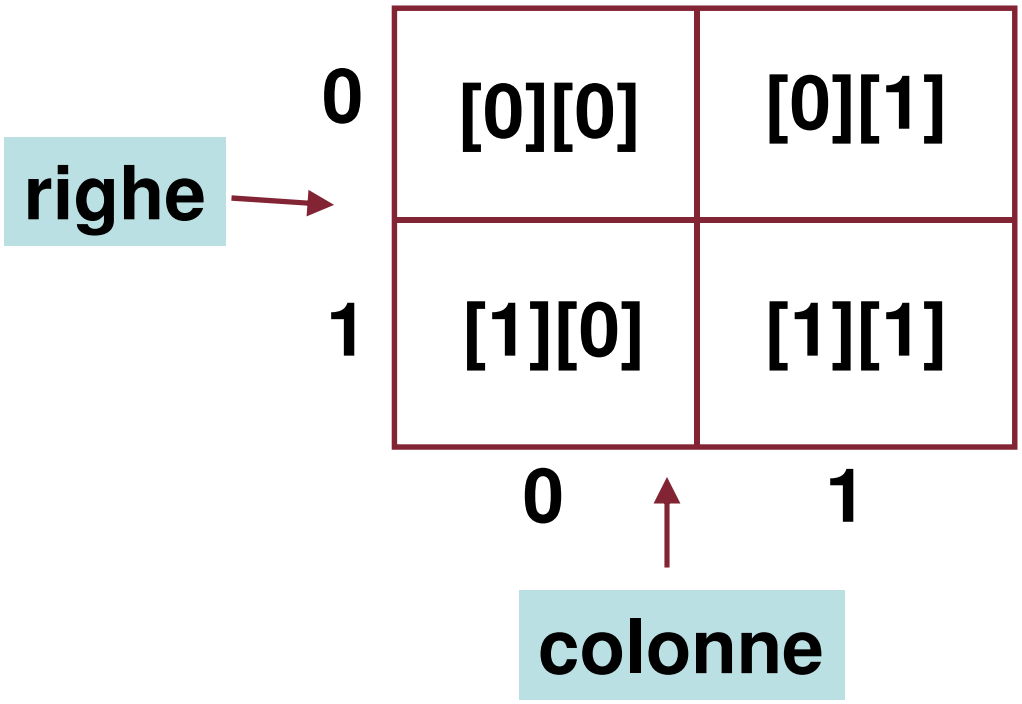
Matrici

Funzioni

Stringhe

Le matrici

Le matrici non sono altro che array multidimensionali



Notazione C:
matrice[righe][colonne]

Gli Array multidimensionali

Gli array multidimensionali sono cosi definiti:

```
int tabella_numeri[50][50]
```

(per due dimensioni)

```
int big_D[20][30][10][40]
```

(per più di due dimensioni)

Indicizzazione degli Array multidimensionali

si accede agli elementi di un array multidimensionale nel seguente modo:

```
numero = tabella_numeri[5][32];

tabella_numeri[1][23] = 100;
```

Esempio Matrici

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

int main()
{
    int matrice[2][2], i, j;
    matrice[0][0] = 1;
    matrice[0][1] = 2;
    matrice[1][0] = 3;
    matrice[1][1] = 4;
        for(i = 0; i < 2; i++)
          for(j = 0; j < 2; j++)
                printf("matrice[%d][%d] = %d\n", i, j, matrice[ i ][ j ]);
   system("PAUSE");
}
```

Output Matrici

```
matrice[0][0] = 1
matrice[0][1] = 2
matrice[1][0] = 3
matrice[1][1] = 4
Premere un tasto per continuare . . .
```

Le stringhe

In C le stringhe sono definite come array di caratteri.

Ad esempio, la seguente istruzione definisce una stringa di 50 caratteri:

```
char name[50];
```

Gestione delle stringhe

Il C non ha un sistema maneggevole per costruire le stringhe, così le seguenti assegnazioni NON SONO VALIDE:

```
char firstname[50], lastname[50], fullname[50];
firstname = "Mario" /* illegale */
lastname = "Rossi" /* illegale */
fullname = "Sig."+firstname+lastname /* illegale */
```

Esiste pero' una libreria di routines per il trattamento delle stringhe ("< string.h >").

Stampare una stringa

Per stampare una stringa si usa printf() con lo speciale carattere di controllo %s:

```
printf("%s", nome);
```

Nota: è sufficiente avere il nome della stringa.

Esempio stampa di una stringa

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

main()
{
   char string_nome[80];
   printf("inserisci il tuo nome:\n");
   scanf("%s", string_nome);
   printf("NOME: %s\n", string_nome);
   //ulteriore modo di stampare il nome
   printf("NOME IN ALTRO MODO: %s\n", &string_nome[0]);
   system("PAUSE");
}
```

Output stampa di una stringa

inserisci il tuo nome:
antonio
NOME: antonio
NOME IN ALTRO MODO: antonio
Premere un tasto per continuare . . .

il carattere ‘\0’

Al fine di permettere l'utilizzo di stringhe con lunghezza variabile, il carattere \0 viene utilizzato per indicare la fine di una stringa.
In questo modo, se abbiamo una stringa dichiarata di 50 caratteri (`char name[50];`), e la utilizziamo per memorizzare il nome "Dave", il suo contenuto (a partire da sinistra) sarà la parola Dave immediatamente seguita dal segno di fine stringa \0, e quindi tutti gli altri caratteri (fino ad arrivare alla lunghezza di 50) risulteranno vuoti.

Esercizio Stringhe

Dichiarare un array di 6 elementi di tipo char.

Inserire in tale array la stringa “DINO”

Stampare a video il contenuto dell’array di 6 elementi dichiarato in precedenza.

Soluzione esercizio Stringhe

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

int main()
{
    char nome[6];
    nome[0] = 'D';
    nome[1] = 'I';
    nome[2] = 'N';
    nome[3] = 'O';
    printf("prova stampa DINO = %s\n", nome);
    nome[4] = '\0';
    printf("RI-prova stampa DINO = %s\n", nome);
    system("PAUSE");
}
```

Output esercizio Stringhe

prova stampa DINO = DINO$³¨☻CÂÇ|¿ #
RI-prova stampa DINO = DINO
Premere un tasto per continuare . . .

Spiegazione esercizio Stringhe

char nome[6];

nome[0] = 'D';
nome[1] = 'I';
nome[2] = 'N';
nome[3] = 'O';

nome[4] = ‘\0';

Senza ‘\0’ la printf non può conoscere la fine della stringa che si vuole stampare, quindi vengono stampati anche valori ignoti (tipo $³¨☻CÂÇ|¿ #)

Funzioni ed array

Possono essere passati alle funzioni come parametri anche array singoli o multidimensionali.
Gli array monodimensionali possono essere passati nel seguente modo:

```
float trovamedia(int size,float list[]){
     int i;
     float sum = 0.0;
     for (i = 0; i < size; i++)
          sum += list[i];
     return(sum/size);
}
```

Spiegazione

Nella precedente funzione “trovamedia” la dichiarazione "float list[]" dichiara al C che "list" è un array di float.

Non viene specificata la dimensione di un array quando è un parametro di una funzione.

Esempio d'uso per "trovamedia"

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

//dichiarazione di funzione
float trovamedia(int size, float list[]);

//main
main(){
	float numeri[2];
	numeri[0] = 1.2; numeri[1] = 3.6;
	float media = trovamedia(2, numeri);
	printf("la media e' %f\n", media);
	system("PAUSE");
}

//codice della funzione
float trovamedia(int size,float list[]){
		int i;
		float sum = 0.0;
		for (i = 0; i < size; i++)
			sum += list[i];
		return(sum/size);
}
```

OUTPUT

```
la media e' 2.400000
Premere un tasto per continuare . . .
```

Funzioni ed array multidimensionali

Array multidimensionali possono essere passati alle funzioni nel seguente modo:

```
void stampatabella(int xsize, int ysize,
                   float tabella[][5])
{
      int x,y;
      for (x = 0; x < xsize; x++) {
            for (y = 0; y < ysize; y++)
                  printf("\t%f", tabella[x][y]);
            printf("\n");
      }
}
```

Spiegazione

Nella precedente funzione “stampatabella”, la parte di codice "`float tabella[][5]`" dichiara al C che tabella è un array di float di dimensioni nx5.

E' importante notare che dobbiamo specificare la seconda dimensione (e le successive) del vettore, ma non la prima dimensione.

Riepilogo: array e funzioni

riepilogando,

nel caso di array singoli non e' necessario specificare la dimensione dell'array nella definizione come parametro della funzione,

mentre nel caso di array multidimensionali si può non specificare solo la prima dimensione.

Esercizio: array e funzioni

Si scriva un programma che prenda in ingresso da tastiera 10 interi, li memorizzi in un array, ordini tale array e stampi la lista dei numeri inseriti dall'utente e la stessa lista ordinata.

Esempio:
Input: 1 3 2 23 43 521 98 43 9 10
Output: 1 2 3 9 10 23 43 43 98 521

Vecchi esercizi da completare

Esercizio 4

Scrivere un programma che, dopo avere letto da standard input 10 caratteri da memorizzare in un array, verifichi se la sequenza di caratteri letta è palindroma (è la stessa letta da destra o da sinistra). Stampare un messaggio con il risultato.

Esempio:

se l'input è ramo44omar il messaggio stampato sarà "la sequenza è palindroma"

Soluzione

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

#define NUM_CHAR 10

int main( )
{
    char vetcar[ NUM_CHAR ], carattere;
    int i, palindroma = 1;

    printf( "Inserire %d caratteri: ", NUM_CHAR ) ;
    for ( i = 0; i < NUM_CHAR; i++)
        scanf("%c", &vetcar[i]);
    for ( i = 0; i < NUM_CHAR/2; i++)
        if (vetcar[i] != vetcar[NUM_CHAR -1 -i])
            palindroma = 0;
```

Soluzione

```
if (palindroma == 1 )
        printf ("La sequenza di caratteri e' palindroma\n");
    else
        printf ("La sequenza di caratteri non e' palindroma\n");

    system( "PAUSE" ) ;
    return 0 ;
}
```

Esercizio 5

Progettare una funzione che, ricevuti:

1. un array A di interi
2. un array B di interi
3. un intero d indicante la dimensione di A e B
4. un intero I da cercare in A,

restituisca un valore intero pari al numero di occorrenze di I in A ed inserisca in B, a partire dalla 1a posizione, le posizioni in cui I compare in A

ESEMPI:

Se d=8, A = [0, 1, 0, 10, 1, 1, 1, 0], I=0, sarà
B = [0, 2, 7, -, -, -, -, -] e verrà restituito 3

Se d=8, A = [0, 1, 0, 10, 1, 1, 1, 0], I=-1, sarà
B = [-, -, -, -, -, -, -, -] e verrà restituito 0

Inserire la funzione all'interno di un programma contenente quanto necessario per verificare se il comportamento della funzione è corretto

Soluzione

- La soluzione proposta definisce 25 come massima dimensione array.
- E' possibile configurare la dimensione degli array A e B ed il contenuto dell'array A.
- Dopo l'acquisizione (funzione "**arrayscan**") dei valori contenuti nell'array A, e l'inizializzazione degli elementi dell'array B al valore -1, si acquisisce il valore da ricercare in A.
- Vengono visualizzati alla fine: il numero di occorrenze nell'array A del valore cercato, il contenuto degli array A e B (funzione "arrayprint")
- L'inizializzazione dell'array B ad un valore negativo e la visualizzazione finale consentono di verificare le operazioni eseguite su questo dalla funzione "search"

Soluzione

```
/* definizione funzione arrayscan */
void arrayscan( int v[ ] , unsigned int size )
{
    unsigned int i ;
    /* ciclo acquisizione elementi array */
    for( i = 0 ; i < size ; i++ )
    {
        printf( "Inserire elemento int di indice %u\n" , i ) ;
        scanf( "%d" , &v[ i ] ) ;
    }
}
```

Soluzione

```
/* definizione funzione arrayprint*/
void arrayprint( int v[ ] , unsigned int size )
{
    unsigned int i ;

    /* ciclo stampa elementi array */
    for( i = 0 ; i < size ; i++ )
       printf( "%u\t%d\n" , i , v[ i ] ) ;
}
```

Soluzione

```
/* definizione funzione ricerca */
unsigned int search( int v1[] , int v2[] , unsigned int size , int
    val )
{
  unsigned int cont = 0 , i ;
  /* cont = valore restituito */

  for ( i = 0 ; i < size ; i++ )
    if ( v1[ i ] == val )
      {
        v2[ cont ] = i ;
        cont++ ;
      }
  return cont ;
}
```

Soluzione

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

#define DIM1 25


/* prototipo funzione ricerca */
unsigned int search( int v1[] , int v2[] , unsigned int size , int val ) ;

/* prototipo funzione arrayscan */
void arrayscan( int v[ ] , unsigned int size ) ;

/* prototipo funzione arrayprint */
void arrayprint( int v[ ] , unsigned int size ) ;
```

Soluzione

```
int main( )
{
   int vA[ DIM1 ] , vB[ DIM1 ] , dim , val , i ;

   printf( "\nInserire dimensione comune (max %u) array A e B:\n" , DIM1 )
     ;
   scanf( "%u" , &dim ) ;

   printf( "\nAcquisizione elementi array A:\n" ) ;
   arrayscan( vA , dim ) ;

   /* inizializzazione array B */
   for ( i = 0; i < dim; i++)
        vB[i] = -1;

   printf( "\nInserire valore intero da cercare in array A:\n" ) ;
   scanf( "%d" , &val ) ;
```

Soluzione

```
  printf( "\nNumero di occorrenze di %d in array A e': %u\n" ,
          val , search( vA , vB , dim , val ) ) ;

  printf( "\nStampa elementi array A:\n" ) ;
  arrayprint( vA , dim ) ;

  printf( "\nStampa elementi array B:\n" ) ;
  arrayprint( vB , dim ) ;

  system("PAUSE") ;

  return 0 ;
}
```

Esercizio :: DNA

- L'informazione genetica del DNA, é codificata nella sequenza di basi (adenina, guanina, citosina e timina) che lo formano.
- Per convenzione, sequenze di DNA sono rappresentate come liste di lettere 'A','G','C', 'T'.
- Vogliamo analizza sequenze di questo tipo, di lunghezza fissata DIM, che rappresentiamo come array di caratteri 'A','G','C', 'T'

Inizializzare l'array nel main

A[]={'A' ,'G', 'T', 'A', 'C', 'A', 'T', 'G', 'T', 'A'}

int DIM ***= 10***

DNA

1. Scrivere un programma che stampa quante volte ciascun carattere è presente
2. Scrivere un programma che, dato un array di caratteri ‘A’,’G’,’C’, ‘T’, elimina dall’array *la prima occorrenza di ‘A’, e stampa l’array* risultante.
3. Scrivere un programma che, dato un array di caratteri ‘A’,’G’,’C’, ‘T’, elimina dall’array *tutte le occorrenze di ‘A’, e stampa l’array* risultante.
4. Trasformare i programmi in funzioni

DNA

- **Da pensare**: cosa significa eliminare dall'array?
- **Attenzione**: non vogliamo lasciare "vuoti"...
- **Sugg**: la dimensione dell'array é fissa, ma possiamo tener conto, in una variabile dedicata, del numero di elementi significativi (la lunghezza che ci interessa), oppure del livello di riempimento (ultimo indice significativo).

DNA

- **Algoritmo semplice:**

1. *creo un secondo array di appoggio, B*
2. *scorro il primo e copio nel secondo solo gli elementi che mi interessano*
   - *Attenzione: avrò bisogno di un **indice_array1** e **indice_array2***
     - ***indice_array1 :: scorre A***
     - ***indice_array2 :: scorre B***
3. *Stampo l'array ottenuto*

DNA

- Algoritmo in versione **tosta**:

1. Fino a che non è finito A (ciclo su indice **i**)
2. A[**i**] va eliminato? (if)
   1. **SI**
      a. **sposta** tutto il resto dell'array **A** (da **i+1** fino a **DIM-1**) copiandolo nelle posizioni da **i** a **DIM-2**
      b. **DIM--;**
   2. **NO**
      1. non fare nulla (**if** senza **else**)

3. **sposta** ::
   1. for (k=i; k<DIM-1; k++)
      A[K] = A[K+1]

Ordina

- Scrivere la funzione `ordina` che ricevuti due interi in ingresso, **passati per indirizzo**, assegni alla prima variabile il valore più piccolo e alla seconda il più grande

```c
#include <stdio.h>
void ordina (int *, int *);

int main (void)
{  int a, b;
   printf("Introduci due interi da
              ordinare \n");
   scanf("%d  %d", &a, &b);
   ordina (&a, &b);
   printf("Numeri ordinati %d  %d\n", a, b);
   return 0; }

void ordina (int *p, int *q)
{  int aux;
   if (*p > *q)
       {      aux = *p;
              *p = *q;
              *q = aux;   } }
```


ordina_ind.c

Per casa

- Dato un array A di interi scrivere le funzioni:
  - void carica_array (int ar[], int elem)
    - carica nell'array un numero di interi, chiesti all'utente, pari a elem
  - void stampa_array (int ar[], int elem)
    - Stampa elem elementi dell'array
  - int trova_max (int ar[] , int elem)
    - Trova e restituisce il valore massimo dell'array
  - float calcola_media (int ar[], int elem)
    - Calcola la media dei valori dell'array
  - int stampa_pari (int ar[], int elem)
    - Stampa gli elementi pari e restituisce il numero di elementi pari presenti nell'array

Per casa

- Dato anche un array B di interi di dimensioni uguali ad A scrivere le funzioni:
  - void somma_array (int A[], int B[], int elem)
    - Calcola A = A + B
  - int max_array (int A[], int B[], int elem)
    - Calcola il valore massimo tra i due array usando la funzione int trova_max (int ar[] , int elem)