Introduzione al C

Corso di Fondamenti di Informatica

Ingegneria Clinica

Esercitazione 9

DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Domenico Daniele Bloisi
Raffaele Nicolussi

Menù di oggi

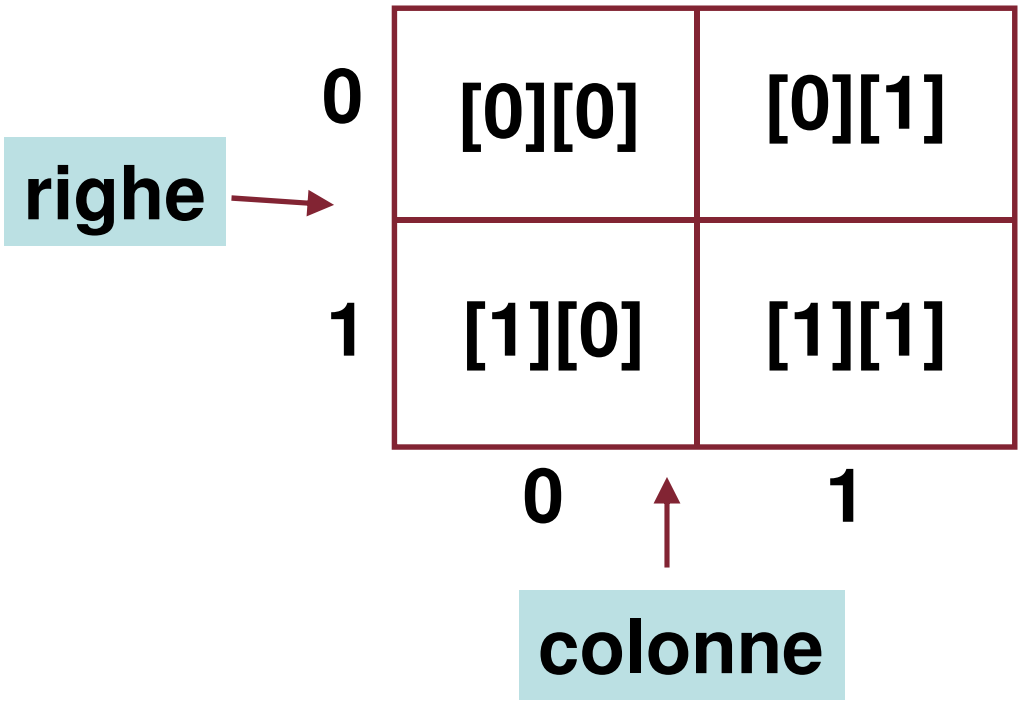
Matrici

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

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

Esercizio calc array (1/2)

Progettare una funzione che, ricevuti:

1. un carattere ch
2. un array A di reali
3. un array B di reali
4. un array C di reali
5. un intero d indicante la dimensione di A,B,C,

interpreti ch come l'operazione da eseguire tra A[k] e B[k] (k=0,1,..,d-1) per produrre il valore da assegnare a C[k]. Le operazioni valide sono: somma ('+'), differenza ('-'), prodotto ('*'), e divisione ('/'). In caso di operatore valido, la funzione restituirà un carattere indicante l'operazione effettuata. In caso di operatore non valido, la funzione restituirà il carattere '?' e non effettuerà alcuna operazione tra gli elementi di A e B

Esercizio calc array(2/2)

ESEMPI

Se ch='+', d=4, A = [0.0, 1.0, 0.1, 1.1],
B = [0.0, 1.0, 0.2, -1.7], verrà restituito
'+' e sarà C = [0.0, 2.0, 0.3, -0.6]

Se ch='%', d=4, A = [0.0, 1.0, 0.1, 1.1],
B = [0.0, 1.0, 0.2, -1.7], verrà restituito
'?' e sarà C = [-, -, -, -]

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, B e C ed il contenuto degli array A e B.
- Dopo l'acquisizione del carattere che rappresenta l'operazione da effettuare sugli elementi degli array, si acquisisce la dimensione degli array e i valori degli elementi degli array A e B (funzione "arrayscan").
- Alla fine viene visualizzato un messaggio che segnala se il carattere inserito rappresenta un operatore valido e, in caso affermativo, viene stampato il contenuto dell'array C (funzione "arrayprint")

Soluzione

```
/* definizione funzione arrayscan */
void arrayscan( float 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( "%f" , &v[ i ] ) ;
    }
}
```

Soluzione

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

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

Soluzione

```
char arrayoper (char op, float v1[], float v2[], float v3[], int d)
{
   int i;

   switch (op) {
      case '+': for (i = 0; i < d; i++)  v3[i] = v1[i] + v2[i]; break;
      case '-': for (i = 0; i < d; i++)  v3[i] = v1[i] - v2[i]; break;
      case '*': for (i = 0; i < d; i++)  v3[i] = v1[i] * v2[i]; break;
      case '/': for (i = 0; i < d; i++)  v3[i] = v1[i] / v2[i]; break;
      default: return '?';
   }

   return op;
}
```

Soluzione

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

#define DIM 25

/* prototipo funzione arrayoper */
char  arrayoper(char op, float v1[], float v2[], float v3[], int d);

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

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

Soluzione

```
int main()
{
   float vA[DIM], vB[DIM], vC[DIM];
   unsigned int dim;
   char oper;


   printf("Inserire operatore (carattere): ");
   scanf("%c", &oper);

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

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

   printf( "\nAcquisizione elementi array B:\n" ) ;
   arrayscan( vB , dim ) ;
```

Soluzione

```
  if ( arrayoper(oper, vA, vB, vC, dim) != '?') {
      printf("Operatore  valido\n");
      printf("Stampa elementi array C:\n");
      arrayprint(vC, dim);
    }
    else
      printf ("Operatore non valido\n");

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

Esercizio Matrix (1/2)

Progettare una funzione che, ricevuti:

1. un array bidimensionale *A* di mxn interi (m righe, n colonne)
2. un array B di n interi
3. un array *C* di m interi
4. i due interi m e n,

assegni, per k = 0, 1, …, m-1, all'elemento C[k] il valore $\sum_i$ A[k][i] * B[i], i = 0, .., n-1

Dichiarare e definire la funzione specificando il massimo valore di m ed n usando la costante MAX_D

Esercizio Matrix (2/2)

ESEMPIO:

Se m=2, n=3,
A = [[1, 2, 3], [4, 5, 6]]
B = [2, 1, 0],
allora *C* = [4, 13]

Inserire la funzione all'interno di un programma contenente quanto necessario per verificare se il comportamento della funzione è corretto
Definire, tramite direttiva al compilatore, il valore 25 per la costante MAX_D

Soluzione

- La soluzione proposta definisce 25 come massima dimensione della matrice e degli array.
- E' possibile configurare le dimensioni della matrice A e di conseguenza degli array V1 e V2.
- Dopo l'acquisizione dei valori contenuti nella matrice (funzione "matrixscan") e dei valori contenuti nell'array (funzione "arrayscan") viene visualizzato il contenuto del vettore V2 dopo l'attivazione della funzione "matrixpervect“.

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 matrixscan */
void matrixscan(int m[][DIM_MAX], unsigned int r, unsigned
     int c)
{
   unsigned int i, j;

   for (i = 0; i < r; i++)
      for (j = 0; j < c; j++)
      {
         printf( "Inserire elemento int di indice %u,%u\n" , i, j ) ;
         scanf("%d", &m[i][j]);

      }
   return;
}
```

Soluzione

```
/* definizione funzione matrixpervect */
void matrixpervect (int m[][DIM_MAX], int v1[], int v2[],
     unsigned int r, unsigned int c)
{
   unsigned int i, j;

   /* inizializzazione del vettore v2 */
   for (i = 0; i < r; i++)
      v2[i] = 0;

   for (i = 0; i < r; i++)
       for (j = 0; j < c; j++)
          v2[i] += m[i][j] * v1[j];

   return;
}
```

Soluzione

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

#define DIM_MAX 25

/* prototipo funzione matrixpervect */
void matrixpervect (int m[][DIM_MAX], int v1[], int v2[], unsigned int r,
     unsigned int c);

/* prototipo funzione matrixscan */
void matrixscan(int m[][DIM_MAX], unsigned int r, unsigned int c);

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

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

Soluzione

```
int main()
{
   int mat[DIM_MAX][DIM_MAX], vett1[DIM_MAX],
     vett2[DIM_MAX];
   unsigned int righe, colonne;

   printf( "\nInserire numero righe della matrice (max %u):\n" ,
     DIM_MAX ) ;
   scanf( "%u" , &righe ) ;

   printf( "\nInserire numero colonne della matrice (max %u):\n" ,
     DIM_MAX ) ;
   scanf( "%u" , &colonne ) ;

   printf( "\nAcquisizione elementi matrice :\n" ) ;
   matrixscan( mat , righe, colonne ) ;
```

Soluzione

```
  printf( "\nAcquisizione elementi vettore :\n" ) ;
   arrayscan( vett1 , colonne ) ;

   matrixpervect(mat, vett1, vett2, righe, colonne);

   printf( "\nStampa elementi array v2:\n" ) ;
   arrayprint( vett2 , righe ) ;

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