Fondamenti di Informatica
Ingegneria Clinica
Lezione 7/12/2009

Raffaele Nicolussi
FUB - Fondazione Ugo Bordoni
Via B. Castiglione 59 - 00142 Roma

/* scambio di due variabili nell'ambiente del chiamante */

```
#include <stdio.h>
void scambia (int *, int *);
int main (void)
{ int i = 3, j = 5;
  scambia (&i, &j);
  printf("%d%d\n", i, j);
                  /*stampa 5 3*/
  return 0;   }
void scambia (int *p, int *q)
{ int temp;
  temp = *p;
  *p = *q;
  *q = temp;  }
```

scambia_ind.c

```
scambia (&i, &j);
void scambia (int *p, int *q)
{  int temp; ……   }
```

- scambia riceve due valori di tipo puntatore a int e non torna nulla; tmp è locale
- in scambia:

```
temp=*p; *p=*q; *q=temp;
```

  - a tmp viene assegnato il valore puntato da p
  - all'oggetto puntato da p viene assegnato il valore puntato da q
    - all'oggetto puntato da q viene assegnato il valore di tmp

- Effetto: scambiare in main i valori cui puntano i e j

```
/* Inizializzazione di un array. I dati sono letti da tastiera
   dalla funzione leggi */
#include <stdio.h>
#define SIZE 10
void leggi (int [], int);
int main (void) {
  int vet[SIZE], i;
  leggi (vet, SIZE);
  for (i=0; i <= (SIZE-1); i++)
     printf("%10d  %10d\n", i,
  vet[i]);
  return 0;     }

void leggi (int vet[], int v){
  int i;
  printf("Introduci i valori
  dell'array%d\n");
     for (i=0; i<= v-1; ++i)
     { printf("valore %d\n", i);
       scanf("%d", &vet[i]);}
  }
```

array3.c

/* Problema: classe di 10 studenti. 5 test con voti da 1 a 10. Si vuole calcolare la frequenza di ciascun voto */

```
void stampa (int [], int);
void leggi (int [], int);
void  calcola_freq (int [], int []);
void  stampa_freq (int []);
```

- Memorizziamo i voti di un test in un vettore di 10 elementi (indice da 0 a 9)
  - voti[10]
  - in voti[i] c'è il voto preso dall'i-esimo studente

voti [0]	voti [1]	voti [2]	voti [3]	voti [4]	voti [5]	voti [6]	voti [7]	voti [8]	voti [9]
4	5	9	10	7	7	7	8	2	10

- Memorizziamo le frequenze dei voti in un vettore di 11 elementi (indice da 0 a 10) per sfruttare la comparazione indice-valore del voto

fre [1]	fre [1]	fre [2]	fre [3]	fre [4]	fre [5]	fre [6]	fre [7]	fre [8]	fre [9]	fre [10]
0	0	0	0	5	3	7	30	5	2	3

```
void stampa (int [], int);
void leggi (int [], int);
void  calcola_freq (int [], int []);
void  stampa_freq (int []);
```

- Non usiamo 5 vettori per memorizzare i voti, ma usiamo sempre lo stesso riscrivendoci sopra

```
for (j=1; j<=TEST; j++)
{
    leggi (voti, j);
    stampa (voti, j);
    calcola_freq (voti, frequenze);
}
```

```c
#include <stdio.h>
#define STUD 10
#define TEST 5
#define VOTI 10

void stampa (int [], int);
void leggi (int [], int);
void calcola_freq (int [], int
  []);
void stampa_freq (int []);
```

frequenza.c

```
int main (void)    {
  int voti[STUD];
  int frequenze[VOTI+1]={0};
  int j;
  for (j=1; j<=TEST; j++)    {
    leggi (voti, j);
          /* i vettori  sono
             passati per
            riferimento*/
    stampa (voti, j);
    calcola_freq (voti,frequenze);
         }
  printf ("\n Le frequenze dei
           voti sono:\n");
  stampa_freq (frequenze);
  return 0;
}
```

```
void stampa (int vet[], int v)  {
  int i;
  printf("Test %d\n", v);
  printf("%10s %10s \n", "Indice",
            "Voto");
  for (i=1; i <= STUD-1; i++)
      printf("%10d  %10d\n", i,
                vet[i]);
                   }

void leggi (int vet[], int v)  {
   int i;
   printf("Introduci i valori del
            test %d\n");
   for (i=0; i<= STUD-1; ++i)
       {
          printf("valore %d\n", i);
          scanf("%d", &vet[i]);
          }
  }
```

```
void calcola_freq (int vet[], int
  freq[])
{
  int j;
  for (j=0; j<= STUD-1; j++)
       ++freq[vet[j]];
}

void stampa_freq (int freq[])
{
  int j;
  printf ("%10s %10s \n", "Voto",
                  "Frequenza");
  for (j=1; j<= VOTI; j++)
      printf ("%10d %10d \n", j,
                  freq[j]);
  }
```

```
/* vettori passati per riferimento, elementi passati per
   valore */
#include <stdio.h>
#define N 10

void  cambia (int[], int, int);
void  cambia2 (int [], int,int);
int main( void ){
  int ar[N] = {0,1,2,3,4,5};
  int i;
  printf ("vettore immesso\n");
  for (i=0; i<=N-1; i++)
      printf ("%4d  ", ar[i]);
  cambia (ar, N, ar[7]);
  printf ("\nvettore dopo
                  cambia\n");
```

vettore immesso

0 1 2 3 4 5 0 0 0 0

cambiaarr.c

```
/* vettori passati per riferimento, elementi
   passati per valore */

   for (i=0; i<=N-1; i++)
       printf ("%4d  ", ar[i]);
   cambia2(ar, N, ar[7]);
    printf("\nvettore dopo
   cambia2\n");
   for (i=0; i<=N-1; i++)
       printf("%4d  ", ar[i]);
      return 0 ;
}
```

cambiaarr.c

```
void  cambia (int vet[ ] , int
  size, int a)
{ int j;
  for (j=0;j<=5; j++)
      vet[j] =vet[j]-2;
   a= 26; }

void  cambia2 (int vet[ ] , int
  size, int a)
{
  vet[7]= 26;
}
```

vettore immesso

0 1 2 3 4 5 0 0 0 0

vettore dopo cambia

-2 -1 0 1 2 3 0 0 0 0

vettore dopo cambia2

-2 -1 0 1 2 3 0 26 0 0

Qualificatore const

- derivato da C++
- indica che una variabile non può essere modificata dopo la sua inizializzazione
  - const int i =100;
  - i non potrà essere alterata da nessuna istruzione
- può essere usato con i puntatori (e i vettori) soprattutto in riferimento alle chiamate di funzione per riferimento per evitare alterazione dei valori

- Va specificato nell'intestazione della funzione, prima del tipo della variabile che si vuole rendere immodificabile

```
#include <stdio.h>
void  provaAcambiare (const int
   []);
int main (void )
{ int ar[] = {10, 20, 30};
   provaAcambiare(ar);
   printf("%d %d %d \n ", ar[0],
   ar[1],ar[2]);
       return 0 ;}
void  provaAcambiare (const int
   b[])
{ b [0] = 1;
   b [1] = 2;
   b [2] = 3;}
```

prova_cambiare.c

Error : illegal assignment to constant
dei6.14.c line 23 b [0] = 1;
Error : illegal assignment to constant
dei6.14.c line 24 b [1] = 2;
Error : illegal assignment to constant
dei6.14.c line 25 b [2] = 3;

Array multidimensionali

- Gli array possono essere array di array
  - Ogni elemento dell'array è a sua volta un array
  - Possono avere qualsiasi dimensione: array multidimensionali
- Caso tipico: array bidimensionale o matrice (2 indici)

4	2
2	1
4	5

array 3x2

3 righe

2 colonne

- Sintassi

```
<tipo> <nome> [ <#_righe>][ <#_colonne>];
```

```
int ar [ 3 ] [ 2];        3 righe x 2 colonne
float mat [ 5] [ 5];       5 righe x 5 colonne
```

Array multidimensionali

int ar [3] [2];

- **Gli indici partono da 0**
- **Quindi l'ultimo indice di ogni dimensione è pari a quella dimensione meno 1**

a[0][0]	a[0][1]
a[1][0]	a[1][1]
a[2][0]	a[2][1]

Sono memorizzati

in celle contigue

per righe

- **a[0][0], a[0][1], a[1][0], a[1][1], a[2][0], a[2][1]**

- **Se si aggiungono parentesi quadre, si aggiunge una dimensione all'array**

  int ar [N][M][R]

- **Il numero di elementi di un array è pari al prodotto delle sue dimensioni**

Inizializzazione

- Può essere inizializzato insieme alla dichiarazione
- E' memorizzato per righe
  - All'interno delle righe è memorizzato per colonne

```
int mat[3][2] = {{4, 2}, {2, 1}, {4, 5}}
```


[0][0] [0][1] [1][0] [1][1] [2][0] [2][1]

- Se in una riga non ci sono valori sufficienti, i rimanenti vengono inizializzati a 0.

```
#include <stdio.h>
int main (void)
{
   int mat [2] [2] = {{1,2}, {3}};
   int ar [2] = {2, 5};
   int i, j;

   for (i=0; i<= 1; i++)
      for (j=0; j<= 1; j++)
      printf("Indici:%2d%2d Valore:
            %4d\n", i, j,
            mat[i][j]*ar[i]);

    return 0 ;
}
```

matrice.c

1	2
3	0

2
5

mat[0][0]*a[0], mat[0][1]*a[0],
mat[10]*a[1], mat[1][1]*a[1],

Array bidimensionali e puntatori

- **Date le relazioni tra array e puntatori, ci sono diversi modi per accedere ad un elemento di un array.**
- **Il nome dell'array è il puntatore al primo elemento**

  &a[0][0] o &a[0]
- **In generale**

  a[i][j] equivale a *(a[i] +j)

  a[1][2] → *(a[1]+2)

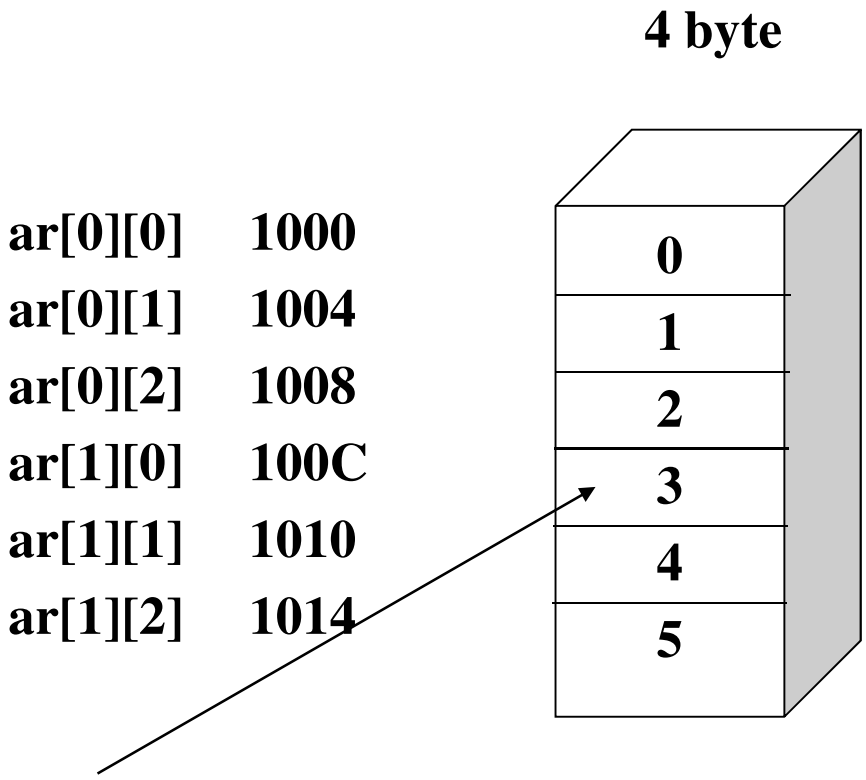
- **Espressioni equivalenti ad a[i][j]**

  ***(a[i]+j)**

  (*(a+i))[j]

Array bidimensionali e puntatori

- **int ar[2][3] = {{0, 1, 2}, {3, 4, 5}}**



ar[1]=&ar[1][0] (puntatore alla seconda riga)

Inizializzazione array a 3 dimensioni

```
#include <stdio.h>
int main (void)  {
  int mat [3][5][6];
  int i, j, k;

  for (i=0; i<= 2; i++)
    for (j=0; j<= 4; j++)
          for (k=0; k<= 5; k++)
                mat[i][j][k]= 2*k +i +j;

  for (i=0; i<= 2; i++)
    for (j=0; j<= 4; j++)
          for (k=0; k<= 5; k++)
          printf ("Indici: %2d %2d %2d Valore:
                %4d\n", i, j, k,  mat[i][j][k]);
 return 0 ;
}
```

tri_array.c

Inizializzazione array a 3 dimensioni

Indici: 0 0 0 Valore: 0

……..

……..

Indici: 0 1 1 Valore: 3

Indici: 0 1 3 Valore: 7

Indici: 0 0 1 Valore: 2

Indici: 0 2 0 Valore: 2

Indici: 0 1 0 Valore: 1

Indici: 0 3 2 Valore: 7...

……..

Indici: 2 4 5 Valore: 16

Somma degli elementi di una matrice

```
#include <stdio.h>
int main (void )
{
  int a [3][2]= {{2,1}, {0, 2}, {5, 7}};
  int i, j;
  int sum = 0;

  for (i=0; i<= 2; i++)
    for (j=0; j<= 1; j++)
          sum+= a [i][j];

  printf ("Somma elementi: %4d\n", sum);
  return 0 ;
}

/* sum = a[0][0]+a[0][1]
    +a[1][0]+a[1][1]+a[2][0]+a[2][1] */
```

somma_mat.c

Array multidimensionale come argomento di funzione

- Per passare un array multidimensionale come argomento ad una funzione è sufficiente specificarne il nome
- Nella dichiarazione della funzione si devono specificare tutti gli indici a parte il primo (posso ometterlo, se inserito viene ignorato come l'indice nel caso del vettore)

```
void f1 (void)  {
    int ar [5][6][7];
    ……….
    f2 (ar);  ……….  }
void f2 (int mat [][6][7])
{    ……….  }
```

- Posso omettere la prima dimensione, ma devo specificare la mappa dell'array tramite le altre dimensioni
- Es. int mat [ ] [3]
- In ogni riga ci sono tre elementi a partire da &mat[0][0]

Problema: **In una matrice sono memorizzati i voti di un gruppo di studenti. I voti sono letti da un file. Si deve calcolare il voto minimo, massimo oltre che media dei voti riportati da ciascuno studente.**

```
#include <stdio.h>
#define NSTUD 10
#define ESAMI 12
```

- Supponiamo che il gruppo sia composto da 10 studenti e ogni studente abbia effettuato 12 esami
- In una matrice riportiamo in ogni riga i voti di uno studente
  - Quindi 10 righe (10 studenti) e 12 colonne (12 esami per ogni studente)
- Definiamo le seguenti funzioni:
  - Scrittura array
  - Stampa array
  - Calcolo del voto minimo
  - Calcolo del voto massimo
  - Calcolo della media

studenti_f.c

```
#include <stdio.h>
#define NSTUD 10
#define ESAMI 12

int minimo (int [][ESAMI]);
int massimo (int [][ESAMI]);
float media (int [ ]);
void stampa_array (int [][ESAMI]);
void leggi_array (int [] [ESAMI]);
```

studenti_f.c

```
int main (void ) {
    int  voti_stud[NSTUD][ESAMI];

    leggi_array(voti_stud);
    printf ("L'array ha i seguenti valori:\n");
    stampa_array (voti_stud,);
    printf ("\n\nVoto minimo: %d\n
          Voto massimo: %d\n",
          minimo(voti_stud),
          massimo(voti_stud));
   for (stud=0; stud<= NSTUD-1; stud++)
          printf ("Voto medio studente
                %5d: %.4f\n", stud, media
                (voti_stud[stud]));
    return 0;

}
```

```
/* Memorizza l'array leggendo i dati dal file */

void leggi_array (int voti[][ESAMI])
{
    int i, j;
    FILE *exm;

    exm=  fopen("dat.dat", "r");

    if (!(NULL==exm)) {
          for (i=0; i<= NSTUD-1; i++)
            for (j=0; j<= ESAMI-1; j++)
                fscanf(exm, "%d", &voti[i][j]);
          fclose (exm);
    else
          printf ("\nErrore di apertura!");

}
```

```
/* Trova il minimo */
int minimo (int voti[][ESAMI])   {
    int i, j, low=100;
    for (i=0; i<= NSTUD-1; i++)
        for (j=0; j<= ESAMI-1; j++)
            if (voti[i][j] < low)
                low = voti[i][j];
        return low;   }

/* Trova il massimo */
int massimo (int voti[][ESAMI])  {
    int i, j, max=0;
    for (i=0; i<= NSTUD-1; i++)
        for (j=0; j<= ESAMI-1; j++)
            if (voti[i][j] > max)
                max = voti[i][j];
        return max;
}
```

```
/* Calcola il voto medio per uno studente */
float media (int voti_stud[ ])
{
    int j, totale=0;
    for (j=0; j<= ESAMI-1; j++)
                totale += voti_stud[j];
    return (float) totale/ESAMI;  }

/* Stampa la matrice */
void stampa_array (int voti[][ESAMI])
{
    int i, j;
    for (i=0; i<= NSTUD-1; i++) {
          printf("\nVoti_studente[%d]", i+1);
    for (j=0; j<= ESAMI-1; j++)
          printf("%7d", voti[i][j]);
          }
 }
```

L'array ha i seguenti valori:
Voti_studente[1]
18 24 30 25 27 27 27 30 29 29 29 30
Voti_studente[2]
18 18 21 22 23 23 23 18 19 23 23 22
……..
Voti_studente[10]
30 30 29 29 28 30 23 30 30 29 29 28

Voto minimo: 18
Voto massimo: 30
Voto medio studente 1: 27.0833
Voto medio studente 2: 21.0833
Voto medio studente 3: 26.0833
Voto medio studente 4: 24.8333
Voto medio studente 5: 29.5000
Voto medio studente 6: 30.0000
Voto medio studente 7: 27.7500
Voto medio studente 8: 29.2500
Voto medio studente 9: 26.5000
Voto medio studente 10:28.7500