

# Array Multidimensionali

Antonella Santone

1

## Tipi di array

Monodimensionali  
(vettore)

Multidimensionali



2

## Array multidimensionale

Un array multidimensionale è un array, i cui elementi sono a loro volta degli array, i cui elementi ecc.. fino ad esaurire le dimensioni volute

3

## Matrice: array bidimensionale

Facciamo riferimento per semplicità al caso degli array bidimensionali: **le matrici**

I dati sono organizzati per righe e per colonne, come se fossero inseriti in una tabella

4

## Dichiarazione di una matrice

```
tipo NomeVariabile[dim_costante1][dim_costante2];
```

**tipo**  
è il tipo degli elementi della matrice

**dim\_costante1, dim\_costante2**  
costanti che indicano il numero di componenti per ciascuna delle due dimensioni

5

## Esempio

```
int mat[3][4]
```

La variabile **mat** contiene 3 righe e 4 colonne, per un totale di 12 elementi

	Colonna 0	Colonna 1	Colonna 2	Colonna 3
Riga 0	mat[0][0]	mat[0][1]	mat[0][2]	mat[0][3]
Riga 1	mat[1][0]	mat[1][1]	mat[1][2]	mat[1][3]
Riga 2	mat[2][0]	mat[2][1]	mat[2][2]	mat[2][3]

6

## Rappresentazione interna

Le matrici in C sono locazioni di memoria contigue che contengono i dati memorizzati per righe

La matrice, anche se viene pensata come un oggetto a due dimensioni del tipo ad es. 3 righe per 4 colonne, cioè come un vettore di righe, ciascuna delle quali è un vettore,

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3

è in realtà un oggetto disposto linearmente in memoria, riga dopo riga, come un vettore

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

7

## Rappresentazione interna (cont.)

0,0	0,1	0,2	0,3	1,0	1,1	1,2	1,3	2,0	$r,c$	2,2	2,3
									2,1		

$$2 * 4 + 1 = 9 \quad \text{NR}=3$$

$$\text{NC}=4$$

Indicizzando righe e colonne a partire da 0, se **NR** è il numero di righe, ed **NC** è il numero di colonne, allora l'elemento in posizione  $(r,c)$  della matrice sta fisicamente nella posizione  $r * \text{NC} + c$  del vettore

8

## Inizializzazione di un array

All'atto della dichiarazione è possibile inizializzare le matrici come di seguito indicato:

```
int matrice [2][3] =
{
    { 1,2,3 } ,
    { 4,5,6 }
};
```

L'inizializzazione viene fatta scrivendo tra parentesi graffe il valore di ciascun elemento, separati da virgole. Come si vede, essendo la matrice un vettore di righe, ciascuna riga viene inizializzata in modo analogo, scrivendo tra parentesi graffe i valori di ciascun elemento separati da virgole

9

## Accedere ad un elemento di una matrice

La funzione di accesso ad un elemento è del tipo

```
nomearray[espressione1][espressione2]
```

L'accesso al dato in posizione  $r,c$  della matrice viene effettuato mediante l'operatore `[]` già usato per i vettori

**Esempi**

```
matrice[1][3]=7;
```

scrive il valore 7 nell'elemento che sta nella seconda riga in quarta colonna

```
int i=2;
```

```
matrice[1][i+3]=7;
```

scrive il valore 7 nell'elemento che sta nella seconda riga in sesta colonna

10

## Accedere ad un elemento (cont.)

```
int mat[2][2];
```

```
mat[0][0]= 1;
```

```
mat[0][1]= 2;
```

```
mat[1][0]= 3;
```

```
mat[1][1]= 4;
```

1	2
3	4

11

## Errore tipico

Far riferimento all'elemento `mat[i][j]` di una matrice usando la forma `mat[i,j]`

12

## Esempi di uso di

matrici

13

## Riempire un matrice dall'esterno

```
#include<stdio.h>
#define NR 4
#define NC 3
main()
{
    int mat[NR][NC], i, j;
    for (i=0; i<NR; i++)
        for (j=0; j<NC; j++)
        {
            printf("dammi el. riga %d col. %d\n", i,j);
            scanf("%d", &mat[i][j]);
        }
}
```

14

## Visualizzazione

```
for (i=0; i<NR; i++)
{
    printf("\n");
    for (j=0; j<NC; j++)
        printf("%d \t ", mat[i][j]);
}
```

15

## Esercizi

... sulle matrici

link nella home page

[esercizi sulle matrici](#)

16

## Esercizio

Inserire in una tabella la tavola pitagorica e poi stamparla

17

## Esercizio

```
#include <stdio.h>
main() {
    int tabellina[10][10];
    int i, j;
    for (i = 0 ; i < 10 ; i++)
        for (j = 0 ; j < 10 ; j++)
            tabellina[i][j] = (i + 1) * (j + 1);

    // stampa
    for (i=0; i<10; i++)
    {
        printf("\n");
        for (j=0; j<10; j++)
            printf("%4d", tabellina[i][j]);
    }
}
```

Dimensione di campo

18

## Visualizzazione

```
antonella@SANTONE ~
$ ./a.exe
 1  2  3  4  5  6  7  8  9 10
 2  4  6  8 10 12 14 16 18 20
 3  6  9 12 15 18 21 24 27 30
 4  8 12 16 20 24 28 32 36 40
 5 10 15 20 25 30 35 40 45 50
 6 12 18 24 30 36 42 48 54 60
 7 14 21 28 35 42 49 56 63 70
 8 16 24 32 40 48 56 64 72 80
 9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
antonella@SANTONE ~
$
```

19

## Senza printf("%4d", ...

```
#include <stdio.h>
main() {

    int tabellina[10][10];
    int i, j;
    for (i = 0 ; i < 10 ; i++)
        for (j = 0 ; j < 10 ; j++)
            tabellina[i][j] = (i + 1) * (j + 1);

    // stampa
    for (i=0; i<10; i++)
    {
        printf("\n");
        for (j=0; j<10; j++)
            printf("%d", tabellina[i][j]);
    }
}
```

20

## Visualizzazione

```
antonella@SANTONE ~
$ ./a.exe
12345678910
2468101214161820
36912151821242730
481216202428323640
5101520253035404550
6121824303642485460
7142128354249566370
8162432404856647280
9182736455463728190
102030405060708090100
antonella@SANTONE ~
$
```

21

## Esercizio

Un quadrato magico è una matrice quadrata  $n \times n$  tale che la somma degli elementi su ogni riga, ogni colonna ed ognuna delle diagonali principali sia la stessa.

8 3 4	3 8 4	1 5 9
1 5 9	9 5 1	8 3 4
6 7 2	6 7 2	6 7 2
I	II	III

I è un quadrato magico.

II non è: la somma degli elementi sulla prima colonna (18) è diversa dalla somma degli elementi su ognuna delle righe (15).

III non è: la somma degli elementi su una delle diagonali principali (6) è diversa dalla somma degli elementi su ognuna delle righe e colonne (15).

Scrivere un programma che, data una matrice quadrata  $n \times n$  di interi positivi stampi "magico" se questa rappresenta un quadrato magico, "no magico" altrimenti

22

## Soluzione

```
#include <stdio.h>
#define N 3

main() {
    int mat[N][N];
    int i, j, somma, s, magico;

    printf("Caricamento matrice\n");
    for (i=0; i<N; i++){
        for (j=0; j<N; j++){
            printf("dammi el. riga %d col. %d\n", i, j);
            scanf("%d", &mat[i][j]);
        }
    }
}
```

23

## Soluzione (cont.)

/\* Verifica se la matrice è un quadrato magico, appena trova una riga o una colonna o una diagonale con una somma diversa esce \*/

// Verifica Orizzontale: somma Prima Riga

```
i=0;
magico=1;
somma = 0;
for (j = 0; j < N; j++)
    somma += mat[i][j];
```

// Controllo altre righe

```
i=1;
while (i<N && magico){
    s = 0;
    for (j = 0; j < N; j++) s += mat[i][j];
    if (s != somma) magico=0;
    i++;
}
```



0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

24

## Soluzione (cont.)

// Verifica Verticale

```

j=0;
while (j<N && magico){
    s = 0;
    for (i = 0; i < N; i++) s += mat[i][j];
    if (s != somma) magico=0;
    j++;
}

```

25

## Soluzione (cont.)

// Verifica I Diagonale

```

s = 0;
i = 0;
while (i<N && magico){
    s += mat[i][i];
    i++;
}
if (s != somma) magico=0;

```

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

26

## Soluzione (cont.)

// Verifica II Diagonale

```

s = 0;
i = 0;
while (i<N && magico){
    s += mat[i][N-i-1];
    i++;
}
if (s != somma) magico=0;

```

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

N = 5

// Stampa del risultato

```

if (magico) printf("Magico");
else printf("Non magico");
} // chiude il main

```

27

## Esercizio

Date due matrici  $mat1 [N] [P]$  e  $mat2 [P] [M]$ , calcolare la matrice prodotto in cui ogni elemento è dato da:

$$pmat[i][j] = \sum_{k=1}^P mat1[i][k] * mat2[k][j]$$

Per  $i = 1..N$ ,  $j = 1..M$

$pmat$  è una matrice costituita da  $N$  righe ed  $M$  colonne

28

## Esempio

N == 2          P == 3          M == 3

1	3	2
1	0	2

mat1: NxP (2x3)

$$1*1 + 3*0 + 2*5 = 11$$

$$1*0 + 3*2 + 2*2 = 10$$

ecc.

1	0	1
0	2	3
5	2	0

mat2: PxM (3x3)

11	10	10
11	4	1

pmat1: NxM (2x3)

29

## Soluzione

```

#include <stdio.h>
#define N 2
#define P 3
#define M 3

```

```

main() {
    int mat1 [N][P];
    int mat2 [P][M];
    int pmat [N][M];
    int i, j, k;
    printf("Caricamento I matrice\n");
    for (i=0; i<N; i++)
        for (j=0; j<P; j++){
            printf("dammi el. riga %d col. %d\n", i,j);
            scanf("%d", &mat1[i][j]);
        }
}

```

30

## Soluzione (cont.)

```
printf("Caricamento II matrice\n");
for (i=0; i<P; i++)
  for (j=0; j<M; j++){
    printf("dammi el. riga %d col. %d\n", i,j);
    scanf("%d", &mat2[i][j]);
  }

/* Calcolo del prodotto */

for (i=0; i<N; i++)
  for (j=0; j<M; j++){
    pmat[i][j]=0;
    for(k=0; k<P; k++)
      pmat[i][j] = pmat[i][j]+mat1[i][k]*mat2[k][j];
  }
```

31

## Soluzione (cont.)

```
printf("Matrice prodotto\n");
for (i=0; i<N; i++) {
  printf("\n");
  for (j=0; j<M; j++)
    printf("%5d", pmat[i][j]);
}

/* chiude il main */
```

32

```
for (i=0; i<N; i++)
  for (j=0; j<M; j++){
    pmat[i][j]=0;
    for(k=0; k<P; k++)
      pmat[i][j] =
pmat[i][j]+mat1[i][k]*mat2[k][j];
  }
```

N == 2  
P == 3  
M == 3

1	1	2
1	0	2

mat1: NxP (2x3)

1	3	1
0	2	3
5	2	0

mat2: PxM (3x3)

11	10	10
11	4	1

i=0

j=1

k=2

<del>1</del>	<del>1</del>	11	<del>1</del>	<del>1</del>	10	

pmat1: NxM (2x3)

33