



Corso di Programmazione a oggetti

Array multidimensionali

a.a. 2014/2015

Francesco Fontanella

Array Bidimensionali

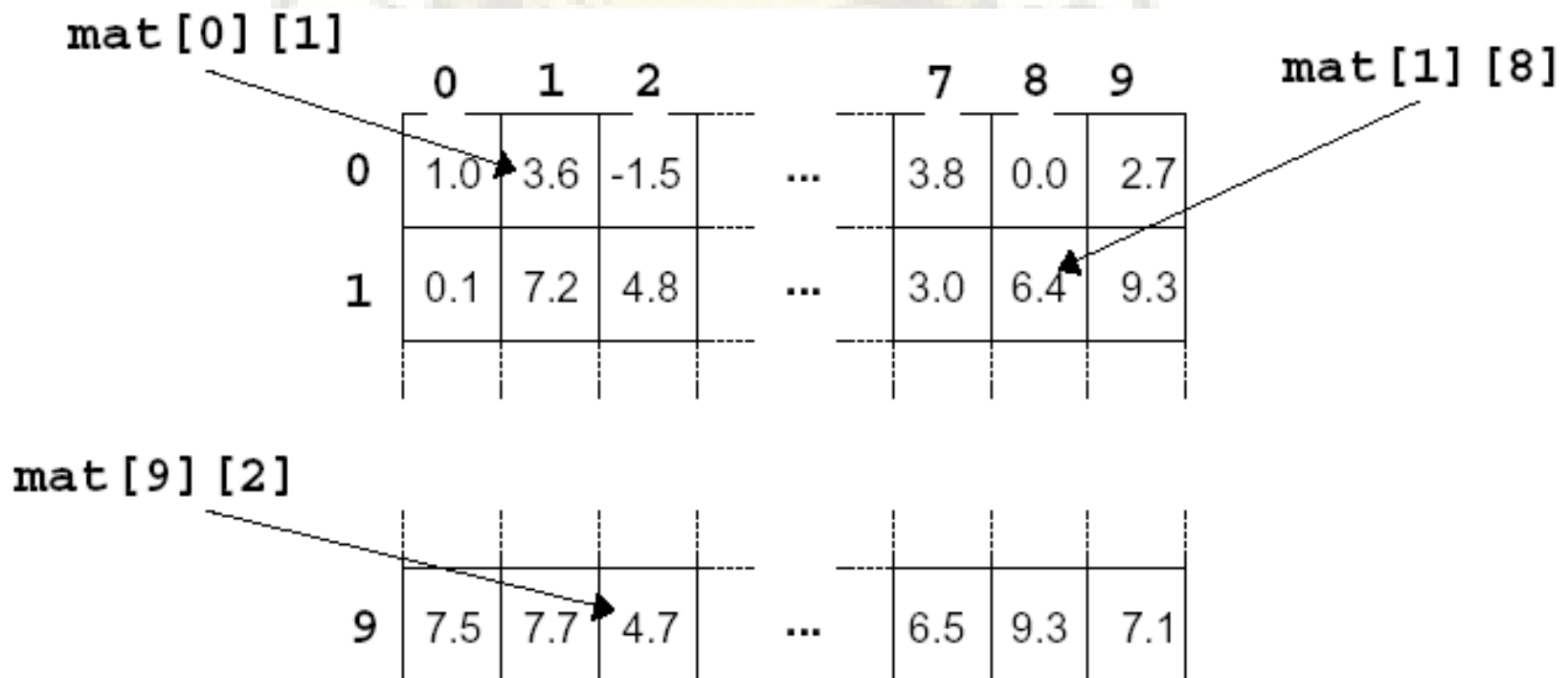
- Il C++ permette di definire anche array a due (e più) dimensioni;
- In tal caso, sono necessari due indici per identificare un singolo elemento dell'array.
- Per definire un array bidimensionale, è necessario specificare:
 - ♦ il nome della variabile array;
 - ♦ il tipo degli elementi;
 - ♦ il numero degli elementi presenti per ogni dimensione;

■ Esempio

- ◆ Definizione di una variabile array `mat` contenente 10x10 elementi double:
`float mat[10][10];`
- ◆ Per accedere ad un singolo elemento:
`mat[2][3] = 4.5;`

Array bidimensionali: organizzazione

- Per accedere ad un array bidimensionale, il primo indice determina la riga, il secondo indica la colonna:



Array bidimensionali: allocazione in memoria

- Anche gli array bidimensionali vengono allocati in memoria in posizione contigue.
- L'allocazione avviene per righe.

■ Esempio

- ◆ Poniamo di avere definito una matrice di interi 3X3:

```
int M[3][3];
```

RAM		
1000	M[0][0]	Riga 1
1004	M[0][1]	
1008	M[0][2]	
1012	M[1][0]	Riga 2
1016	M[1][1]	
1020	M[1][2]	
1024	M[2][0]	Riga 3
1028	M[2][1]	
1032	M[2][2]	

accesso agli elementi dell'array

- Dato un array bidimensionale M, di r righe e c colonne, di tipo TipoValue, il compilatore calcola l'indirizzo dell'elemento M[i][j] applicando la formula:

$$\&M[i][j] = M + i * c * \text{sizeof}(\text{TipoValue}) + j$$

- **Esempio (vedi slide 6)**

```
int mat[3][3];
```

se il compilatore alloca la matrice a partire dall'indirizzo 1000 (valore assegnato alla variabile M), l'indirizzo dell'elemento M[1][2] sarà 1020

Inizializzazione

- Un array bidimensionale può essere inizializzato in fase di definizione:

```
int mat[2][3] = {{10,20,30},  
                {40,50,60}};
```

- Altre inizializzazioni equivalenti:

```
int mat[2][3] = {10,20,30,40,50,60};
```

```
int mat[][3] = {{10,20,30},  
                {40,50,60}};
```

Input/output: esempio

```
#include <iostream>
```

```
const int MAX = 100;
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int i,j,r,c;
```

```
    double a[MAX][MAX], x[MAX], b[MAX];
```

```
    cout<<"inserire il numero di righe"<<endl;
```

```
    cin >> r;
```

```
    cout << "inserire il numero di colonne \n";
```

```
    cin >> c;
```

```
cout << "inserire la matrice (per righe) \n";
for(i = 0; i < r; i++)
    for(j = 0; j < c; j++)
        cin >> a[i][j];

cout << "La matrice inserita e':" << endl;
for(i = 0; i < r; i++) {
    cout << "riga " << i << ": ";
    for(j = 0; j < c; j++)
        cout << a[i][j] << " ";
    cout << endl;
}

return 0;
}
```

Array bidimensionali come parametri

- Quando si passa un array bidimensionale come argomento di una funzione, così come per i vettori si passa l'indirizzo del primo elemento.
- È necessario specificare il numero di colonne dell'array (il numero di righe può essere non specificato), per calcolare l'indirizzo degli elementi della matrice (vedi slide 8)

■ ESEMPIO

```
void func(int M[][10])  
{  
    .  
    .  
    .  
    M[2][1] = 0;  
}
```

Per fare riferimento alla seconda riga è necessario conoscere la lunghezza delle colonne

Ricerca del Massimo in una Matrice

```
const int MAX = 100;
```

```
int ric_max(int mat[][MAX], int r, int c)
```

```
{
```

```
    int i,j, m;
```

```
    m = mat[0][0];
```

```
    for(i = 0; i < r; i++)
```

```
        for(j = 0; j < c; j++)
```

```
            if (mat[i][j] > m)
```

```
                m = mat[i][j]
```

```
    return m;
```

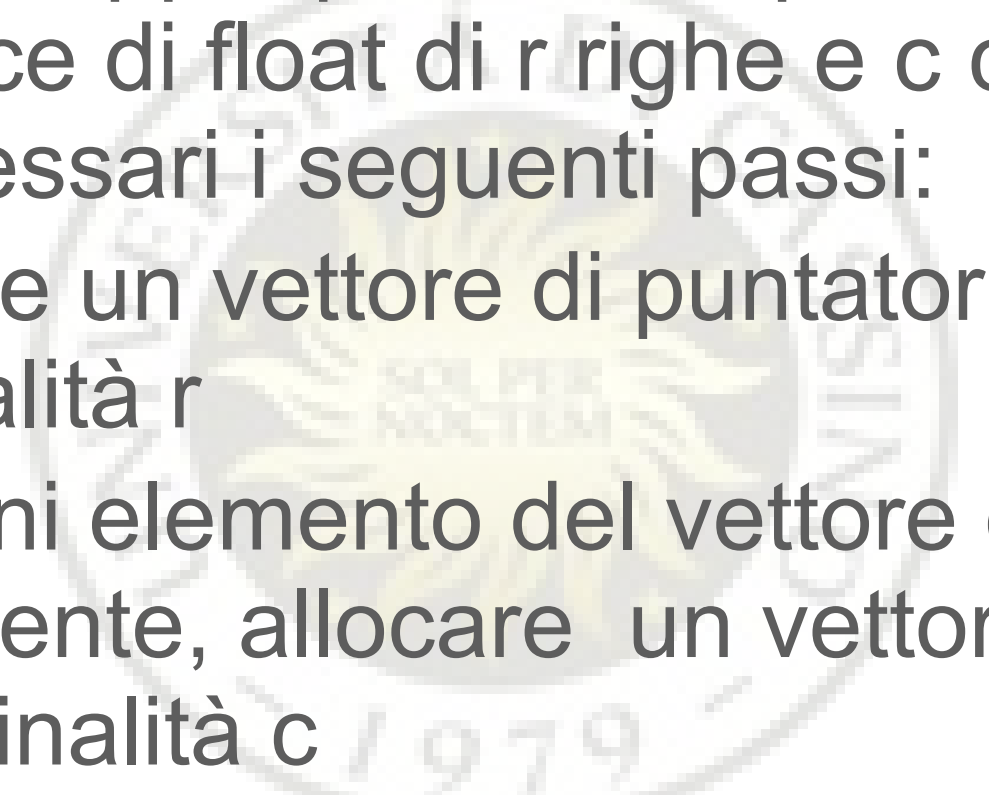
```
}
```

allocazione dinamica

- Gli array bidimensionali possono essere allocati anche in maniera dinamica
- A tale scopo è necessario definire un doppio puntatore:

```
int    **mat_i;
```

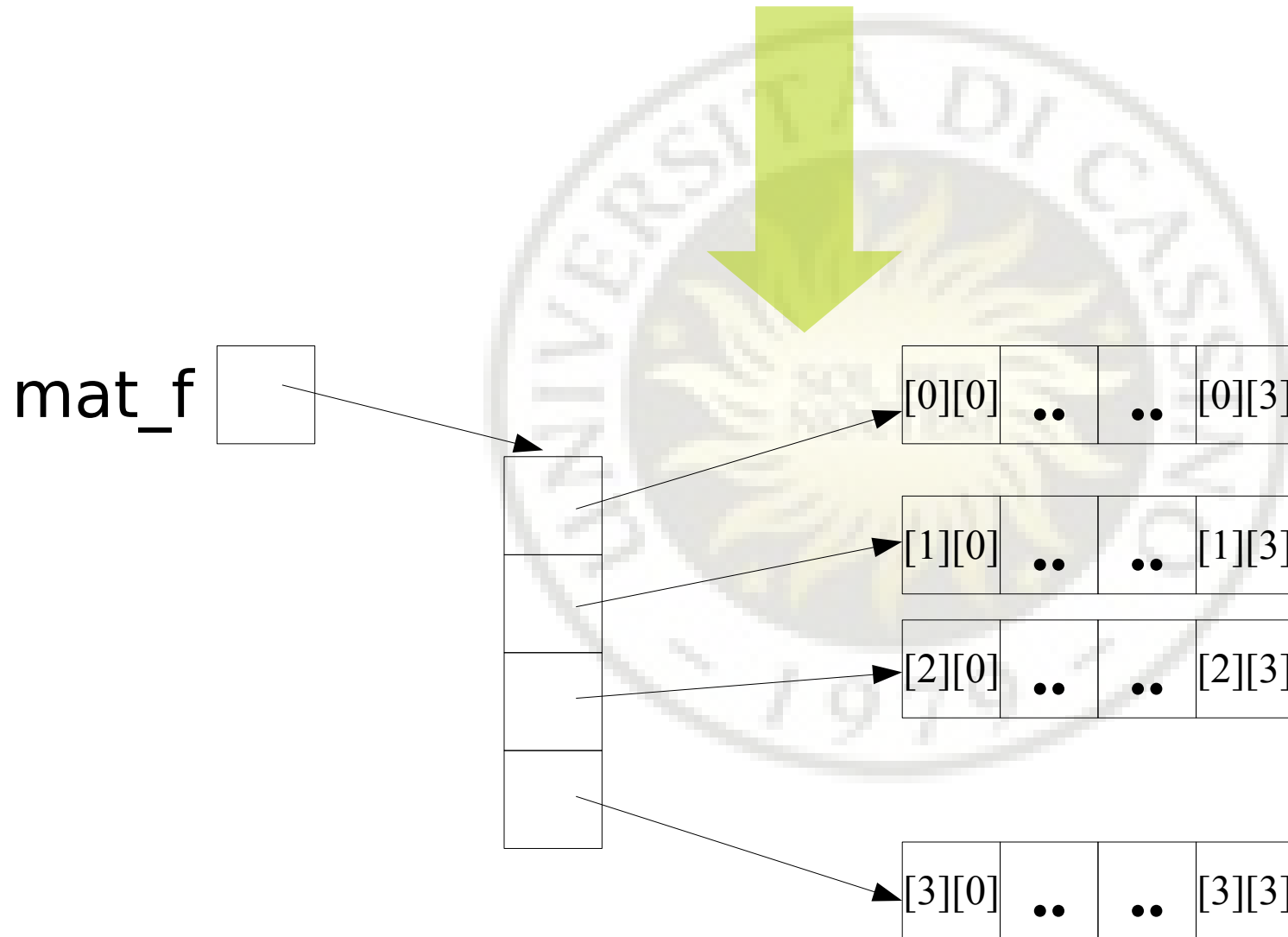
```
float **mat_f;
```

- 
- Definito il doppio puntatore, per allocare una matrice di float di r righe e c colonne sono necessari i seguenti passi:
 - ♦ Allocare un vettore di puntatori a float di cardinalità r
 - ♦ Per ogni elemento del vettore del punto precedente, allocare un vettore di float di cardinalità c

- La seguente funzione restituisce un puntatore a una matrice di float di r righe e c colonne

```
float** alloc_mat(int r, int c) {  
    float **m;  
    int i;  
  
    m = new float*[r];  
  
    for (i=0; i < c; ++i)  
        m[i] = new float[c];  
  
    return m;  
}
```

```
float **mat_f;  
mat_f = alloc_mat(4,4);
```



Esempio 2

```
const int MAX = 100;
```

```
void print_col(int mat[][MAX], int r, int c)
```

```
{
```

```
    int i, col;
```

```
    cout << endl << "Quale colonna vuoi stampare? ";
```

```
    cin >> col;
```

```
    cout << endl;
```

```
    for(i = 0; i < r; i++)
```

```
        cout << mat[i][col] << endl;
```

```
    cout << endl;
```

```
    return ;
```

```
}
```

Esercizio

- Scrivere la funzione:

bool is_simmetric(**int** mat[][MAX], **int** r, **int** c)

che restituisce **true** se la matrice è
simmetrica, **false** altrimenti

Array Multidimensionali

- Il C++ permette la definizione di array multidimensionali con più di due indici:
`int mat3[5][10][5];`

- Con le dovute modifiche valgono le considerazioni sulla definizione, inizializzazione, assegnazione, accesso fatte per gli array monodimensionali e bidimensionali.