

# Array multidimensionali e stringhe

Andrea Marin

Università Ca' Foscari Venezia  
Laurea in Informatica  
Corso di Programmazione part-time

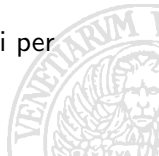
a.a. 2011/2012

## Array uni-dimensionali (richiami)

- ▶ Dichiarazione:

```
int vet[100]; float x[50];
```

- ▶ Gli elementi sono indicizzati da 0 a  $N - 1$  dove  $N$  è la dimensione dell'array ( $N = 100$  nei precedenti casi)
- ▶ `vett[i]` denota l'elemento in posizione  $i$  del vettore
  - ▶  $i$  deve essere un numero intero compreso tra 0 e  $N - 1$
  - ▶ `x[i]` è una variabile di tipo `float`
  - ▶ `vet[i]` è unidimensionale una variabile di tipo `int`
- ▶ Negli array multi-dimensionali possiamo usare più indici per indicare un elemento



## Esempio

- ▶ Si vogliono memorizzare i livelli di inquinamento in una città mediante delle sonde
- ▶ Nella città sono presente 100 sonde rilevatrici
- ▶ Ciascuna sonda rileva il livello di inquinamento ogni 15 minuti
- ▶ Come memorizzare le rilevazioni di una giornata?
  - ▶ Una giornata ha 24 ore, quindi genera  $24 \times 4 = 96$  rilevazioni
  - ▶ Possiamo usare una tabella di 100 righe (una per ogni sonda) e 96 colonne (un per ogni rilevazione)
  - ▶ Ci sono in totale  $96 \times 100 = 9600$  dati misurati
  - ▶ Per accedere ad una particolare rilevazione bisogna specificare
    - ▶ L'indice  $i$  della sonda  $0 \leq i \leq 99$
    - ▶ L'indice  $j$  della rilevazione desiderata  $0 \leq j \leq 95$



## Dichiarazione in C

```
#define NSONDE 100
#define NRILIEVI 96
```

```
float livello_inquinamento [NSONDE][NRILIEVI];
```

- ▶ La **direttiva** `#define` può essere usata per dichiarare costanti
- ▶ L'uso delle quadre può essere esteso ad oltre 2 dimensioni. Ad esempio:

```
double molte_dimensioni [100][20][23][50];
```

- ▶ Nel caso di due dimensioni possiamo parlare di **matrice**
  - ▶ Nell'esempio `NSONDE` è convenzionalmente considerato il numero di righe e `NRILIEVI` il numero di colonne

## Scorrere una matrice: esempio di stampa

```
#define RIGHE 50
#define COLONNE 60

float matrice[RIGHE][COLONNE];

int main(){
    int r,c;
    . . .
    for (r=0; r<RIGHE; r=r+1) {
        for (c=0; c<COLONNE; c=c+1) {
            printf("%f ", matrice[r][c]);
        }
        printf("/n");
    }
    return 0;
}
```



## Passaggio di un array multi-dimensionale ad una funzione

- ▶ Quando si dichiara una funzione che prende come parametro un array multi-dimensionale, solo la prima dimensione può essere lasciata non specificata
- ▶ Esempio:

```
void stampa_matrice(int mat[][COLONNE]);
```

- ▶ La dimensione specificata nel parametro formale deve coincidere con quella specificata nella dichiarazione del parametro attuale
- ▶ **Approfondimenti...**: come rappresentare un array bi-dimensionale con un array uni-dimensionale?

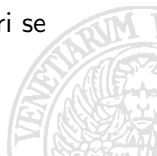


## Esercizi

1. Scrivere una funzione che data una matrice di interi e un vettore di interi metta nella cella  $i$  del vettore la somma degli elementi pari della riga  $i$  della matrice:

```
void somma_righe(int matrice[][COLONNE], int dim,  
                int vett[]);
```

2. Scrivere una funzione che scambi le righe e le colonne di una matrice
3. Scrivere una funzione che decida se due matrici di interi se sono uguali



# Le stringhe in C

## Definizione (Stringa in C)

*In C una stringa è una sequenza contigua di caratteri che termina con il carattere `'\0'`.*

- ▶ Le stringhe sono implementate come array di caratteri `char`
- ▶ Come per gli array la dimensione è decisa in fase di compilazione
- ▶ Il carattere di terminazione denota la porzione (iniziale) dell'array che contiene i caratteri significativi della stringa





## Dichiarazione di una stringa

- ▶ Stringa con al massimo 99 caratteri (posizioni 0 - 98):  
`char stringa [100];`
- ▶ Inizializzazione di una stringa  
`char mio_saluto [100] = "Hello world!";`
- ▶ Attenzione: l'operatore = può essere usato in fase di inizializzazione ma **non di assegnamento**



## Passaggio di una stringa ad una funzione

- ▶ Il passaggio di una stringa ad una funzione segue le regole dell'array
- ▶ Attenzione: non è necessario passare la dimensione perchè c'è il carattere di fine stringa
- ▶ Esempio:

```
void stampa_stringa(char str[]);
```



## Esempio: Stampa di una stringa carattere per carattere

```
void stampa_stringa(char s[]) {  
    int i=0;  
    while (s[i]!='\0') {  
        printf("%c", s[i]);  
        i = i+1;  
    }  
}
```

- ▶ Nota: normalmente si usa `printf("%s", stringa);`



## Calcolare la lunghezza di una stringa

```
int strlen(char s[]) {  
    int len=0;  
    while (s[len] != '\0') {  
        len = len + 1;  
    }  
    return len;  
}
```



## Alcune note

- ▶ L'operatore `==` non può essere usato per stabilire se due stringhe contengono gli stessi caratteri
- ▶ L'operatore `=` non può essere usato per copiare il contenuto di una stringa in un'altra
- ▶ Per facilitare queste operazioni si usa la libreria standard `string.h`



## Esercizi

1. Scrivere una funzione che decida se due stringhe sono identiche
2. Scrivere una funzione che decida se una stringa contiene almeno una vocale minuscola
3. Scrivere una funzione che decida se una stringa ha più di tre consonanti consecutive

