



UNIVERSITÀ  
degli STUDI  
di CATANIA

# Array in C

Corso di programmazione I (A-E / O-Z) AA 2025/26

Corso di Laurea Triennale in Informatica

---

Fabrizio Messina

[fabrizio.messina@unict.it](mailto:fabrizio.messina@unict.it)

Dipartimento di Matematica e Informatica

# Dichiarazione di Array

Un array è un insieme di locazioni di memoria consecutive dello stesso tipo.

Ogni elemento dell'array è accessibile in lettura/scrittura mediante nome dell'array, parentesi quadre e indice. ES: `V[3]`

Il primo valore per un indice è zero, mentre il valore massimo, per uno array di dimensione DIM è DIM-1.

# Dichiarazione di Array

NB: Dichiarazioni alle linee 7-8 (VLA == Variable Length Array) ammesse a partire da C99. Tuttavia GNU Gcc le ammette come estensione anche per le precedenti versioni del linguaggio..

```
1  #define DIM 10
2  const int dim = 10;
3  short mydim = 10;
4
5  int V1[10]; //10 elementi interi
6  float V2[DIM]; //10 elementi float
7  double V3[dim]; //10 elementi double.
8  long V4[mydim]; //10 elementi long
9
10 V1[7] = 4; V2[0] = 6.7;
11 V2[10] = 0.1234; //NO!
```

# Dichiarazione vs inizializzazione di Array

```
1  #define DIM 10
2
3  int V1[DIM]; //10 elementi interi
4
5  //Quanto vale V1[5] ?? ?
6  print("Elemento di V1 con indice 5: %d", V[5]);
```

**NB: Non si possono fare assunzioni sul valore iniziale delle variabili non inizializzate, array compresi**

# Dichiarazione vs inizializzazione di Array

```
1  //tutti gli elementi inizializzati
2  int V[10] = {1,2,3,4,5,6,7,8,9,10};
3
4  //inizializzazione parziale.
5  int W[10] = {1,2,3,4,5};
6
7  //Dimensione array definita implicitamente!
8  int Z[] = {1,2,3,4,5};
```

Inizializzazione mediante **lista di inizializzatori**.

Alla linea 5 inizializzazione parziale. **Il compilatore inizializza i rimanenti elementi a 0!**

Alla linea 8 **dimensione implicita, fissata mediante inizializzazione.**

# Dichiarazione vs inizializzazione di Array

```
1  int V[1000] = {0}; // tutti a zero!
2  int W[1000] = {}; // tutti a zero!
3  int Z[1000]; // non inizializzati!
4
5  for(int j = 0; j < 1000; j++)
6      Z[j] = j * 2;
```

Inizializzazione parziale può essere sfruttata opportunamente se si vuole che il valore iniziale degli elementi sia semplicemente zero (linee 1-2).

Linee 5-6: Inizializzazione tramite ciclo for.

## Dichiarazione vs inizializzazione di Array

```
1  int n = 100;  
2  int V[n] = {0};  
3  //...  
4  
5  n=250;  
6  //Dimensione di V?? 100!!
```

La dimensione di V viene fissata all'atto della sua dichiarazione. In questo caso 100 elementi.

NB: *Le eventuali successive variazioni del valore conservato nella variabile non hanno alcun effetto sulla dimensione di V!*

# Array multidimensionali

```
1  #define N 3
2  #define M 4
3
4  //matrice dimensioni N x M
5  int V[N][M] = {0}; // tutti a zero
6  float W[N][M] = {}; // tutti a zero
7
8  //inizializzazione delle righe
9  int Z[N][M] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

Per inizializzazione array multidimensionali valgono le stesse regole sintattiche degli array monodimensionali.



# Array multidimensionali

```
1  #define N 3
2  #define M 4
3
4  //Compile-time error!
5  int Z[][] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

Il compilatore sa che si tratta di un array multidimensionale, ma non può determinare la lunghezza della righe. L'inizializzazione da sola non basta!

```
1  #define N 3
2  #define M 4
3
4  //OK!
5  int Z[][M] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

Il compilatore sa che la lunghezza delle righe è M, dunque può determinare lo *shape* della matrice con precisione. Il numero delle righe sarà ricavato dal numero degli inizializzatori.

# Array multidimensionali

```
1  #define N 3
2  #define M 4
3
4  //OK!
5  int Z[][M] = {1,2,3,4,5,6,7,8,9,10}; //quante righe? 3!
```

Il compilatore sa che la lunghezza delle righe è M, dunque anche questa volta può determinare lo *shape* della matrice con precisione.

Il numero minimo di righe tale da far entrare tutti gli inizializzatori.

In generale, per un array multidimensionale:  $\lceil \frac{L}{M} \rceil$ , dove  $L$  è la lunghezza della lista di inizializzazione, ed M il numero di colonne specificato.

## Homework H12.1

Scrivere un programma in C nel quale si chiede all'utente un numero qualunque  $p$  in virgola mobile. Successivamente:

- Se il numero  $p$  è minore di 1, stampare un messaggio di errore.
- Se il numero  $p$  è maggiore o uguale a 1, arrotondare il numero stesso all'intero più vicino, sia  $n$ , e successivamente allocare un array di dimensione  $n$ ;
- Inizializzare l'array con valori a piacere per tre volte mediante un ciclo: i) con il costrutto for, ii) con il costrutto while, iii) con il costrutto do-while.
- Stampare tutti i valori dell'array mediante un ciclo usando un costrutto a scelta.

## Homework H12.2

Scrivere un programma in C nel quale si chiede all'utente di inserire due numeri  $N$  ed  $M$  entrambi maggiori di 1. Successivamente:

- operare i soliti controlli ed eventuali arrotondamenti al fine di ottenere due numeri interi maggiori di zero;
- inizializzare una matrice  $N \times M$  con numeri in virgola mobile scelti a caso;
- calcolare e stampare il prodotto di un numero scelto a caso (es: 3.542) con la matrice;
- definire una ulteriore matrice quadrata  $K \times K$ , in cui  $K = \min(N, M)$ . ES: per una matrice 8x9 le dimensioni della nuova matrice saranno 8x8;

- riempire la nuova matrice con i corrispondenti elementi della prima matrice.

## "Esempi svolti"

12\_01.c

12\_02.c