

I VETTORI MULTIDIMENSIONALI

Il linguaggio C

1



DEFINIZIONE DI VETTORE

- Un vettore può essere definito come una “collezione organizzata di oggetti”:
 - **“collezione”**: implica che gli oggetti contenuti nel vettore siano dello stesso tipo;
 - **“organizzata”**: implica che sia possibile identificare univocamente tutti gli oggetti del vettore.



I VETTORI MULTIDIMENSIONALI

- I **vettori multidimensionali** (o matrici) sono vettori i cui elementi sono a loro volta dei vettori.
- Il numero delle dimensioni non è limitato a priori, anche se normalmente non si superano le tre dimensioni.
- Ogni elemento di un vettore multidimensionale è identificato da **tanti indici quante sono le dimensioni del vettore**:
 - vettore monodimensionale: 1 indice;
 - vettore bidimensionale: 2 indici;
 - vettore tridimensionale: 3 indici;
 - ecc.



I VETTORI MULTIDIMENSIONALI

<u>Indici</u> →	0	1	2	3	4	5	6	7
	3	5	-2	7	6	4	25	-3

	<i>Indici</i> →	0	1	2	3
↓	0	3	5	-2	7
→	1	6	4	25	-3
	2	16	-5	5	-23



COME SI DICHIARA UN VETTORE MULTIDIMENSIONALE?

- La dichiarazione di una vettore multidimensionale è simile a quella di un vettore e si compone di 3 elementi:
 1. il tipo di dati;
 2. il nome del vettore multidimensionale;
 3. il numero di elementi, per ogni dimensione del vettore del vettore, scritto tra parentesi quadre.

```
float vett_bidim[25][10];
```

→ Crea un vettore bidimensionale con 25 x 10 elementi

```
float vett_tridim[5][10][7];
```

Crea un vettore tridimensionale con 5 x 10 x 7 elementi



INIZIALIZZAZIONE DI UN VETTORE MULTIDIMENSIONALE

- Come per tutti gli altri tipi di vettori, anche i vettori multidimensionali, all'atto della dichiarazione, **non vengono inizializzati**, a meno che non lo si faccia in modo esplicito.
- L'inizializzazione di un vettore multidimensionale può avvenire in molti modi, i due principali sono:
 - **contemporaneamente alla dichiarazione dello stesso;**
 - **in un secondo momento, assegnando esplicitamente ad ogni elemento del vettore il valore voluto.**



INIZIALIZZAZIONE DI UN VETTORE MULTIDIMENSIONALE

- Per **dichiarare e contemporaneamente inizializzare** un vettore multidimensionale si deve procedere come nell'esempio seguente:

```
int vet[3][4] = {{-1, 4, 27, -5}, {10, -4, 2, -15}, {-21, 14, 7, 12}}
```

- Per inizializzare un vettore multidimensionale assegnando esplicitamente ad ogni elemento il valore voluto:

```
int vet[10][5], i, j;  
for(i = 0; i < 10; i++)  
{  
    for(j = 0; j < 5; j++)  
    {  
        vet[i][j] = 0;  
    }  
}
```



VETTORI DI STRINGHE

- Un vettore di stringhe non è altro che un vettore bidimensionale di tipo «char» in cui ogni stringa viene memorizzata in una diversa «riga» del vettore.

```
char vett_str[3][8];
```

Indici →	0	1	2	3	4	5	6	7
0	C	i	a	o	\0			
1	P	r	o	v	a	\0		
2	V	e	t	t	o	r	e	\0



ESEMPIO - 1

- Crea e visualizza una tavola Pitagorica 10 x 10.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int tavolaPitagorica[10][10], i, j;

    /* Creo la tavola Pitagorica */

    for(i = 0; i < 10; i++)
    {
        for(j = 0; j < 10; j++)
        {
            tavolaPitagorica[i][j] = (i + 1) * (j + 1);
        }
    }
}
```



ESEMPIO - 1

```
/* Visualizzo la tavola Pitagorica */
```

```
printf("\nTAVOLA PITAGORICA 10X10\n\n");
```

```
→ for(i = 0; i < 10; i++)
```

```
{
```

```
→ for(j = 0; j < 10; j++)
```

```
{
```

```
printf("%3d", tavolaPitagorica[i][j]);
```

```
}
```

```
printf("\n");
```

```
}
```

```
return 0;
```

```
}
```



ESEMPIO - 2

- Dato un «dizionario» di 20 parole (max 30 caratteri) inserite dall'utente, calcolare la lunghezza media delle parole inserite.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char dizionario[20][31], i;
    float media = 0;

    /* Creo il dizionario */

    printf("\nInserisci 20 parole:\n");
    for(i = 0; i < 20; i++)
    {
        scanf("%s", dizionario[i]);
    }
}
```



ESEMPIO - 2

```
/* Calcolo la lunghezza media delle parole */
```

```
for(i = 0; i < 20; i++)
```

```
{
```

```
    media += strlen(dizionario[i]);
```

```
}
```



```
printf printf("\nLa lunghezza media delle parole e': %f", media / 20);
```

```
    return 0;
```

```
}
```



MATRICI QUADRATE

- Una matrice quadrata è un vettore bidimensionale che ha un ugual numero di righe e di colonne.

3-1
↓

Indici → 0 1 2 3

↓

0	3	5	-2	7
1	6	4	25	-3
2	16	-5	5	-23
3	1	45	-5	-23

i →

Diagonale secondaria
indici di riga e di colonna «opposti»
 $matr[i][i_max - i]$

Diagonale principale
stesso indice di riga e di colonna
 $matr[i][i]$



GENERAZIONE DI NUMERI (PSEUDO)CASUALI

- ***int rand(void)***: ritorna un numero intero (pseudo)casuale nell'intervallo $[0, \text{RAND_MAX}]$ con $\text{RAND_MAX} \geq 32767$.
- ***void srand(unsigned int seed)***: usa *seed* come seme per generare una nuova sequenza di numeri (pseudo)casuali, numeri che verranno poi ritornati dalla funzione *rand*.
- Le funzioni *rand* e *srand* sono definite nella libreria «***stdlib.h***» che deve essere inclusa.
- ***time_t time(time_t *pt)***: ritorna il numero di secondi trascorsi a partire dal 01/01/1970.
- La funzione *time* è definita nella libreria «***time.h***» che deve essere inclusa.



GENERAZIONE DI NUMERI (PSEUDO)CASUALI

- Per generare un numero pseudo casuale, normalmente, si eseguono i seguenti passaggi:
 1. si richiama, una sola volta prima della prima chiamata a *rand*, la funzione *srand* passandole come parametro il valore ritornato dalla funzione *time*.

```
srand(time(NULL));
```

2. si richiama, quante volte è necessario, la funzione *rand*. Ogni chiamata ritornerà un nuovo numero (pseudo)casuale.

```
num = rand();
```



ESERCIZI

- **Esercizio 1:** scrivere un programma che, data una matrice (vettore bidimensionale) 25×25 , assegni ad ogni elemento della matrice il valore 1 se la somma degli indici è pari, 0 altrimenti.
- **Esercizio 2:** scrivere un programma che data una matrice di numeri interi 10×10 generata casualmente, calcoli e visualizzi:
 - La media di ogni riga.
 - La media di ogni colonna.
 - La media dei valori della diagonale principale e di quella secondaria.
- **Esercizio 3:** scrivere un programma che chieda all'utente di inserire 20 parole (max 30 caratteri) e verifichi se ci siano o meno parole duplicate tra le parole inserite. Le parole devono essere memorizzate in un vettore di stringhe di dimensioni opportune.