

# Programmazione (imperativa)

Corso di Laurea in Informatica

Roberto Cordone

DI - Università degli Studi di Milano



Lezioni: Lunedì 12.00 - 13.00 e 14.00 - 16.00 Mercoledì 14.00 - 17.00  
Laboratorio: Giovedì 12.00 - 13.00 e 14.00 - 17.00  
Ricevimento: su appuntamento  
Tel.: 02 503 16235  
E-mail: roberto.cordone@unimi.it  
Web page: <http://homes.di.unimi.it/~cordone/courses/2015-prog/2015-prog.html>

# Tipi strutturati

Rappresentano informazioni composte dall'**insieme di più valori concettualmente legati**

- **vettori** mono o pluridimensionali: se sono **valori omogenei associati a oggetti diversi** identificati da uno o più indici numerici (i risultati di un esame, un listino prezzi)
- **strutture**: se sono **valori eterogenei associati a uno stesso oggetto** (le misure di una stazione meteorologica, giorno mese e anno di una data) identificati da nomi simbolici

Applicando `sizeof` a una variabile di tipo vettore o struttura si ottiene la memoria occupata dall'intero vettore o struttura

**Vettore** o **array** è una **struttura** contenente un dato numero di oggetti dello stesso **tipo** detti **elementi**

Si dichiara specificando

*tipo variabile*[numero];

- il **tipo** degli elementi: predefinito o definito da utente, semplice o strutturato
- il **numero** degli elementi: **un'espressione costante positiva**

Esempio:

```
#define N 10  
int V[N], A[100];  
int B[10*N+4];
```

# Indicizzazione (1)

Per accedere a un elemento di un vettore, si specifica il vettore e l'**indice** dell'elemento, cioè **la sua posizione nel vettore**

```
Esempio: #define N 10
          int V[N];
          i = V[4];
```

L'**indice** di un elemento **può essere qualsiasi espressione intera, anche variabile** (ad es.,  $V[N-j]$ )

L'indice può anche essere un'espressione con effetti collaterali, ma è meglio evitarlo (è pericoloso combinare due effetti collaterali)

```
    i = 1; somma = 0;
    while (i <= N-1)
        somma += V[i++];
```

**Le parentesi quadre sono un operatore ad altissima priorità**  
(come ++ e --)

## Indicizzazione (2)

L'indice di un vettore di  $N$  elementi va da 0 a  $N-1$

Per definire vettori con estremi diversi (positivi), si allarga il vettore:  
un vettore con estremi  $S$  e  $D$  tali che  $0 \leq S \leq D$  si dichiara con

```
int V[D+1];
```

cioè lasciando i primi  $S$  elementi inutilizzati

Ogni elemento di un vettore è una variabile, e quindi può comparire a sinistra di un assegnamento (ad es.,  $V[3] = 7;$ )

In C non c'è controllo che l'indice cada entro l'intervallo dichiarato  
Si può scrivere in aree di memoria incontrollate (tipica causa di errori)

# Vettori multidimensionali (1)

Un vettore può avere qualsiasi numero di dimensioni, cioè i suoi elementi possono essere identificati da qualsiasi numero di indici

Matrice è un vettore a due o più dimensioni

La dichiarazione specifica il numero di valori per ciascun indice

*tipo variabile [numero1] [numero2] [numero3];*

Per una matrice di M righe, da 0 a M-1, e N colonne, da 0 a N-1:

```
#define M 5
#define N 10
int V[M][N];
```

Nota:

$V[i][j]$  è l'elemento nella riga  $i$  e colonna  $j$

$V[i, j]$  equivale a  $V[j]$  (vedi operatore virgola) ed è la riga  $j$

# Vettori multidimensionali (2)

I vettori multidimensionali sono memorizzati ordinati per righe

Data la dichiarazione

```
#define M 5  
#define N 10  
int V[M][N];
```

V[0][0]	...	V[0][9]	V[1][0]	...	V[1][9]	...	V[4][0]	...	V[4][9]
---------	-----	---------	---------	-----	---------	-----	---------	-----	---------

Se l'indice di colonna eccede N, si accede alla riga seguente

I vettori non si possono copiare con il semplice assegnamento:  
vanno copiati elemento per elemento

Sbagliato

```
#define M 5
#define N 10
int A[M+1][N+1];
int B[M+1][N+1];
```

```
B = A;
```

Corretto

```
#define M 5
#define N 10
int A[M+1][N+1];
int B[M+1][N+1];
int i, j;

for (i = 0; i <= M; i++)
    for (j = 0; j <= N; j++)
        B[i][j] = A[i][j];
```