Linguaggio C: Array

Valeria Cardellini

Corso di Calcolatori Elettronici
A.A. 2019/20

Array in C

- Dichiarazione di variabili array
- Inizializzazione e manipolazioni di array monodimensionali
- Inizializzazione e manipolazioni di array multidimensionali

Tipi di dato semplici e strutturati

- Finora tipi di dato semplici:
  - `int, char, float, double`
- Gli array sono un tipo di dato strutturato
- Sono composti da elementi omogenei
  - Tutti gli elementi sono dello stesso tipo
  - Gli elementi sono memorizzati in locazioni di memoria contigue
- Ogni elemento è identificato da un numero d'ordine detto indice dell'elemento
- Il numero di elementi del vettore è detto lunghezza (o dimensione) del vettore

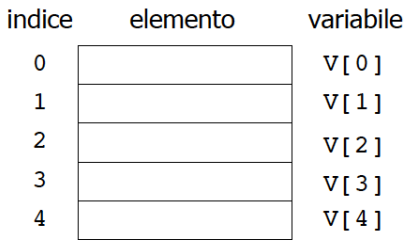
Array monodimensionale (vettore)

- Dichiarazione di variabile di tipo vettore:

  tipoElementi nomeArray[lunghezza];
- Esempio: `int v[5];`
  - Dichiara un vettore di 5 elementi di tipo int
    - ✓ 5 locazioni di memoria contigue, ciascuna contenente un intero
    - ✓ La lunghezza del vettore è 5
- In Ansi C la lunghezza di un vettore è statica (nota a tempo di compilazione)
- In C l'indice degli elementi va da 0 a lunghezza-1
  - Nell'esempio da 0 a 4
  - Se si usa un indice >= lunghezza (es. `v[5]`) warning del compilatore

    warning: array index 5 is past the end of the array

Array monodimensionale (vettore)



- v[i] denota l'elemento del vettore v di indice i
- Ogni elemento del vettore è una variabile

- Esempio

```
int v[10], a;
v[0] = 5;
a = v[0];
v[1] = v[0] + a;
```

- L'indice del vettore deve essere un intero

```
a = 2;
v[a] = 7;
v[a+1] = 13;
```

error: array subscript is not an integer

Manipolazione di vettori

- Avviene solitamente attraverso cicli **for**
  - L'indice del ciclo varia in genere da 0 a lunghezza-1
  - La lunghezza del vettore può essere definita tramite una costante
    - ✓ Es.: `#define LEN 6`

Manipolazione di vettori

- Esempio: lettura e stampa di un vettore

Codice sorgente: vettore.c

```
#include <stdio.h>
#define LEN 5

int main (void) {
   int v[LEN]; /*vettore di LEN elementi, indicizzati da 0 a LEN-1 */
   int i;
   for (i = 0; i < LEN; i++) {
      printf("Inserisci l'elemento di indice %d: ", i);
      scanf("%d", &v[i]);
   }
   printf("Indice\tElemento\n");
   for (i = 0; i < LEN; i++) {
      printf("%d\t%d\n", i, v[i]);
   }
   return 0;
}
```


Inizializzazione di vettori

- Gli elementi del vettore possono essere inizializzati con valori costanti contestualmente alla dichiarazione del vettore
- Esempio: `int vn[4] = {11, 22, 33, 44};`
- Attenzione: l'inizializzazione deve essere contestuale alla dichiarazione

```
int vn[4];
vn = {11, 22, 33, 44}; Errore!
```

- Se vengono inizializzati meno elementi, quelli rimanenti sono posti a 0

```
int vn[10] = {3};
float vf[5] = {1.0, 2.5};
```

Inizializzazione di vettori

- Se ci sono inizializzatori in più, errore di sintassi

```
int vn[2] = {1, 2, 3}; errore!
```

- Se si mette una lista di inizializzatori, si può non specificare la lunghezza (posta pari alla lunghezza della lista)

```
int vn[ ] = {1, 2, 3};      equivale a
int vn[3] = {1, 2, 3};
```

- In C l'unica operazione possibile sugli array è l'accesso agli elementi
- Non si possono effettuare direttamente assegnazioni tra vettori

```
int a[3] = {11, 22, 33};
int b[3];
b = a; errore!
```


Esempi

- Esempio: Calcolare la somma degli elementi di un vettore.

```
int a[10], i, somma = 0;
...
for (i = 0; i < 10; i++)
   somma += a[i];
printf("%d", somma);
```

- Esempio: Calcolare il massimo di un vettore di 10 elementi interi.

```
int a[10], i, max;
...
max = a[0];
   for (i = 1; i < 10; i++)
      if (a[i] > max) max = a[i];
printf("%d", max);
```

Esempi

- Esempio: Leggere N valori reali e stampare i valori inferiori al 50% della media
  - Esercizio: calcolare la media in una funzione

Codice sorgente: vettore_media.c

- Esempio: Leggere una sequenza di caratteri terminata da `'\n'` e stampare le frequenze delle cifre da 0 a 9

Codice sorgente: vettore_char.c

Array multidimensionali (matrici)

- Dichiarazione di array multidimensionali

  tipo-elementi nome-array [lung-1][lung-2]…[lung-n]
- Esempio: `int A[3][4];`
  - array bidimensionale di 3 righe per 4 colonne (ovvero matrice 3x4)
- Per ogni dimensione *k*, l'indice va da 0 a $lung_k$-1
- Esempio: `int A[3][4];`

	0	1	2	3
0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
2	A[2][0]	A[2][1]	A[2][2]	A[2][3]

Accesso agli elementi di una matrice

- Esempio:

```
int i, A[3][4];   /* matrice A di dimensioni 3x4 */
…
i = A[0][0]; /* elem. di riga 0 e colonna 0 (primo elem.)*/
A[2][3] = 28; /* elem. di riga 2 e colonna 3 (ultimo elem.)*/
A[2][1] = A[0][0] * A[1][3];
```

- Come per i vettori, l'unica operazione possibile sulle matrici è l'accesso agli elementi tramite l'operatore []
  - Esempio: somma di due matrici A e B di dimensione MxN

```
for (i = 0; i < M; i++)    /* ciclo su righe */
   for (j = 0; j < N; j++) /* ciclo su colonne */
      C[i][j] = A[i][j] + B[i][j];
```


Esempio: matrice

- Esempio: lettura e stampa di una matrice

```
#include <stdio.h>
#define RIG 2
#define COL 3
int main() {
 int mat[RIG][COL];
 int i, j;
 printf("Lettura matrice %d x %d;\n", RIG, COL); /* lettura matrice */
 for (i = 0; i < RIG; i++)
  for (j = 0; j < COL; j++) scanf("%d", &mat[i][j]);
 printf("La matrice e':\n"); /* stampa matrice */
 for (i = 0; i < RIG; i++) {
  for (j = 0; j < COL; j++)
   printf("%6d ", mat[i][j]);
  printf("\n");
 }
 return 0;
}
```

Codice sorgente: matrice_read.c

Rappresentazione in memoria di matrici

- Le matrici possono essere considerate come array i cui elementi sono a loro volta array
- In C le matrici sono memorizzate **per righe**

	0	1	2	3
0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
2	A[2][0]	A[2][1]	A[2][2]	A[2][3]

A[0][0]
A[0][1]
A[0][2]
A[0][3]
A[1][0]
A[1][1]
A[1][2]
A[1][3]
A[2][0]
A[2][1]
A[2][2]
A[2][3]

Inizializzazione di matrici

- Esempio:

```
int A[2][3] = {{1,2,3}, {4,5,6}};
int A[2][3] = {1,2,3,4,5,6};
```

1	2	3
4	5	6

- Esempio:

```
int A[2][3] = {{1,2,3}};
int A[2][3] = {1,2,3};
```

1	2	3
0	0	0

- Esempio:

```
int A[2][3] = {{1}, {2,3}};
```

1	0	0
2	3	0

Esempio: prodotto di matrici

- Programma che legge una matrice A (m x p) ed una matrice B (p x n), calcola e stampa la matrice C risultante dal prodotto delle due matrici
- La matrice C è di dimensione m x n
- Il generico elemento c_{ij} di C è dato da:

$$c_{ij} = \sum_{k=0}^{p-1} a_{ik} b_{kj}$$

Codice sorgente: matrici_prod.c

Esempio: prodotto di matrici

- Notare nel codice il passaggio di parametri della matrice (allocata staticamente nella funzione chiamante)
  - Esamineremo altre soluzioni possibili

```
#define MDIM 3
#define PDIM 4
#define NDIM 2

void prod_matr(double A[][PDIM], double B[][NDIM], double C[][NDIM]);

int main(void) {
  double matA[MDIM][PDIM], matB[PDIM][NDIM], matC[MDIM][NDIM];
  …
  prod_matr(matA, matB, matC);
  …
}
```

Array e puntatori

- Analizzeremo nella lezione successiva:
  - la relazione tra array e puntatori
  - il passaggio di parametri di tipo array