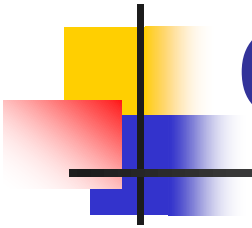
# Gli Array o Vettori

**Concetti chiave**

- Array
- Elementi dell'array
- Indici e lunghezza dell'array
- Inizializzazione di vettori
- Array multidimensionali



# Gli Array o Vettori

- Fino ad ora abbiamo utilizzato ***variabili scalari***, ossia riferite ai tipi scalari: char, int, float, double, etc.
- Esistono in linguaggio C anche le ***variabili aggregate***, riferite ai tipi aggregati
- L'array o vettore è una variabile aggregata
- Si dice anche che l'array o vettore è una variabile di **tipo derivato**, perché costruito a partire da altri tipi

# Gli Array o Vettori

- Il vettore o *array* è costituito da ***elementi*** in cui è possibile memorizzare valori di tipo omogeneo.
- Ogni elemento è individuato da un numero progressivo, detto ***indice***.
- L'indice può assumere valori interi da zero al numero totale di elementi meno 1.
- Il numero complessivo degli elementi del vettore viene detto ***lunghezza***.

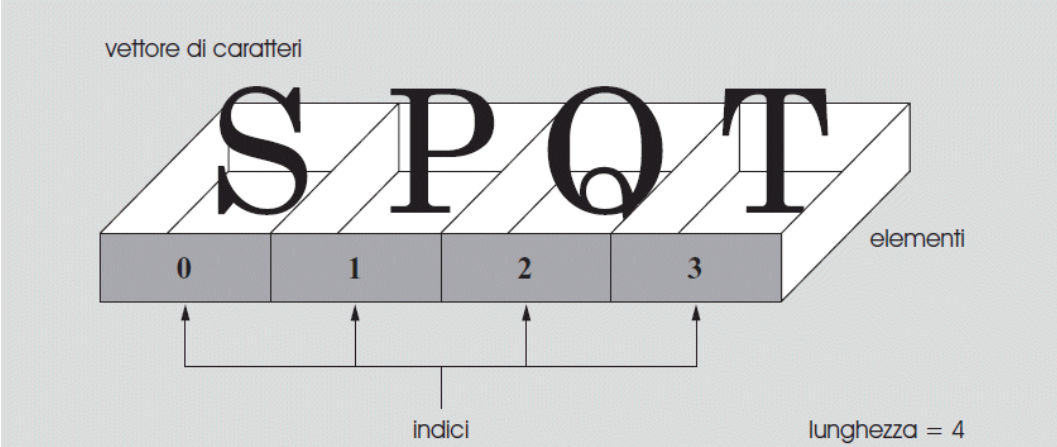



***Figura 10.1*** *Rappresentazione intuitiva di un array.*

# Gli Array o Vettori

- I vettori sono *variabili strutturate*
- Il tipo dei dati contenuti nel vettore viene detto ***tipo del vettore***.
- Sintassi: *tipo nome [dimensione] = {elenco valori tra ,};*
- Esempio: int a[6];

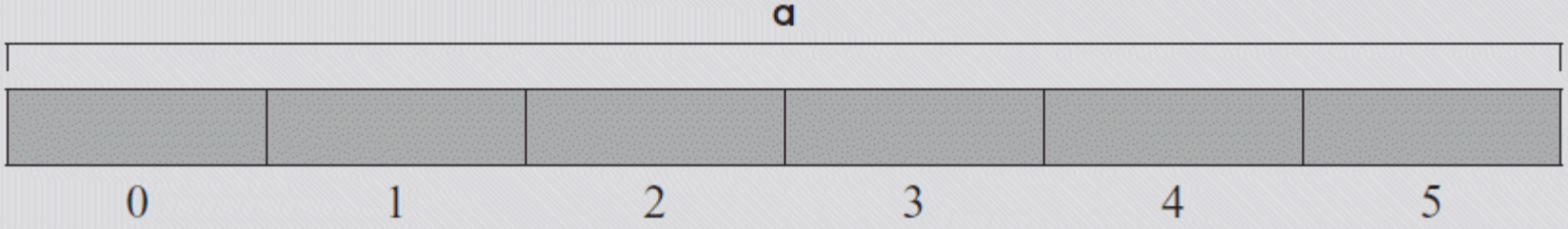



***Figura 10.2*** *Struttura dell'array* a [6].

# Gli Array o Vettori

**Dichiarazione e inizializzazione**

- L'inizializzazione è opzionale e se presente, il numero di elementi usati per l'inizializzazione deve essere pari (al massimo) alla dimensione del vettore:

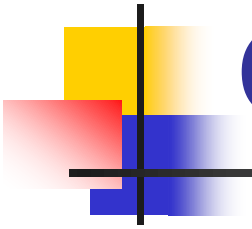
```
int voti[6] = {18, 30, 27, 25, 21, 19};
```

- Se è presenta l'inizializzazione, allora la dimensione può essere omessa (viene ricavata dal numero di elementi usati per l'inizializzazione):

```
int voti[] = {18, 30, 27, 25, 21, 19};
```

- È possibile inizializzare solo una parte (i primi x elementi):

```
int voti[6] = {18, 30};
```



# Gli Array o Vettori

**Dichiarazione e inizializzazione**

- A partire dal C99, è possibile indicare espressamente gli elementi da inizializzare tramite l'uso dei cosiddetti **inizializzatori designati**
- **Esempio:**

```
int voti[6] = {[3] = 18, [5] = 30};
```

# Gli Array o Vettori

**Dichiarazione e inizializzazione**

- Tutte le versioni di C: La dimensione del vettore, se presente, è in genere una costante (anche **definita con #define**)

```
int v[10]          oppure          #define dim 10
                                   int v[dim];
```

- C99: La dimensione può essere una variabile che va inizializzata a run time prima della dichiarazione (VLA=Variable-Length Array):

```
int dim;
scanf("%d", &dim);
int v[dim];
```

- C11: VLA è opzionale, dunque meglio non usare

# Gli Array o Vettori

- Istruzione di Assegnazione applicata ai vettori:
  - ✓ a[0] = 71; /*assegna al primo elemento del vettore a il valore 71 */
  - ✓ a[1] = 4; /*assegna al secondo elemento del vettore a il valore 4.*/
  - ✓ b += a[0] * a[5]; /* il valore di b è sommato al prodotto tra il primo e il sesto elemento di a e il risultato è assegnato a b.*/
- **Assegnazioni Proibite !:**
  - ✓ a=71; /*scopriremo che a è l'indirizzo dell'inizio del vettore*/
  - ✓ a[]=4;

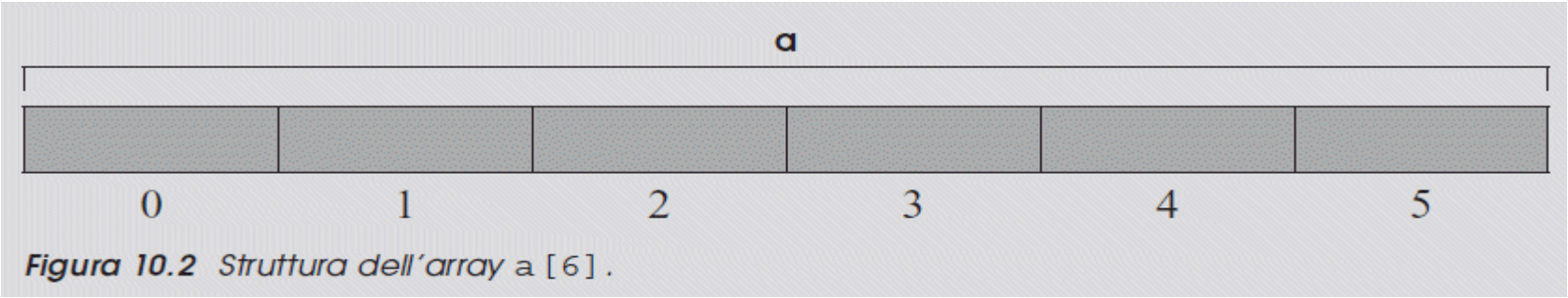
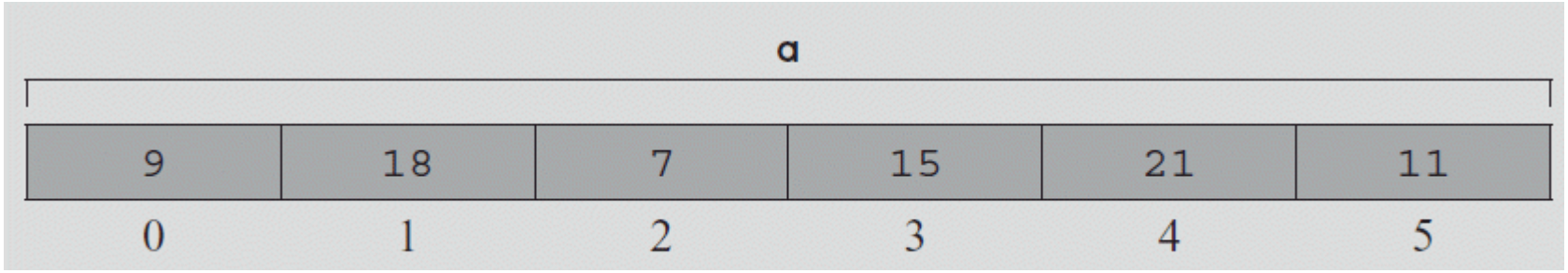



*Figura 10.2* *Struttura dell'array* a[6].

# Gli Array o Vettori

- Riempimento di un Vettore

```
for(i=0; i<6; i++) {
    printf("\nInserisci un numero intero: ");
    scanf("%d", &a[i]);
}
```




- Stampa di un Vettore

```
for(i=0; i<6; i++)
    printf("\nElemento di Indice %u = %d ", i, a[i]);
```

# Gli Array o Vettori

Esercizio: riempimento e stampa di un vettore (VLA- SOLO NETBEANS)

```
#include <stdio.h>


int main(void) {

    unsigned long i,dim;
    printf("\nInserisci la dimensione del vettore ");
    scanf("%u", &dim);
    int v[dim];

    for (i=0;i<dim;i++) {
       printf("\nInserisci l'elemento di indice %u ",i);
       scanf("%d", &v[i]);
    }

    for (i=0;i<dim;i++)
       printf("\nElemento di indice %u = %d ",i,v[i]);

}
```

# Gli Array o Vettori

Esercizio: riempimento e stampa di un vettore (NO VLA-tutti i compilatori)

```
#include <stdio.h>

#define dim 5

int main(void) {

    unsigned long i;
    int v[dim];

    for (i=0;i<dim;i++) {
       printf("\nInserisci l'elemento di indice %u ",i);
       scanf("%d", &v[i]);
    }

    for (i=0;i<dim;i++)
       printf("\nElemento di indice %u = %d ",i,v[i]);

}
```

# Gli Array o Vettori

Esercizio: ricercare il valore massimo in un array:

```
#define N 6
unsigned int i;
int a[N]={1,5,9,4,3,7},max;

int main(void)
{
  max = a[0];
  for(i=1; i<6; i++)
        if(a[i]>max) max = a[i];
  printf("\nIl valore massimo e' %d ",max);
}
```

# Gli Array o Vettori

Esercizio: calcolare la media in un array

```
#define N 6
unsigned int i;
int a[N]={1,2,3,4,5,6};
float media;

int main(void)
{
  for(i=0; i<N; i++)
        media+=a[i];
  printf("\nIl valore medio e' %f ",media/N);
}
```

# Gli Array o Vettori

Esercizio:

- Riempire in modo casuale un vettore
- Uso di rand()
- Libreria stdlib.h
- RAND_MAX= 32767

```
#include<stdio.h>
#include<stdlib.h>
#define N 30

unsigned int i, a[N];

int main(void)
{
        printf("\nRiempio il vettore con numeri casuali maggiori o uguali a zero");
        printf("\nIl massimo numero che posso generare e' = %u ", RAND_MAX);
        for (i = 0; i<N; i++) {
                        a[i] = rand();
                        printf("\nL'elemento di indice %u e' %u ", i, a[i]);
       }
}
```

# Gli Array o Vettori

Esercizio:

- Riempire in modo casuale un vettore con valori compresi tra 0 e MAX-1
- MAX < RAND_MAX
- Uso di srand()

**Cast Operator**

(tipo) variabile/espressione

Esempio:

```
int x=3,y=5;
float z;

int main(void) {

    z=x/ (float) y;

}
```

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define N 1000
#define MAX 10

unsigned int i, a[N];

int main(void)
{
     // inizializzazione del generatore pseudo-casuale dei numeri
     srand((unsigned int) time(NULL));

     printf("\nAdesso riempio il vettore con numeri tra 0 e %u", MAX - 1);
     for (i = 0; i<N; i++) {
          a[i] = rand() % MAX;
          printf("\nL'elemento di indice %u e' %u ", i, a[i]);
     }
}
```

# Gli Array o Vettori

**Esercizio:**

**Prodotto di due vettori**

**(1xN) * (Nx1):**

```
#include<stdio.h>
#define N 6

unsigned long i;
int v1[N], v2[N], prodotto;

int main(void){
   printf("\nRiempimento dei Vettori ");
   for (i=0;i<N;i++) {
      printf("\nInserisci l'elemento del primo vettore di indice %u ",i);
      scanf("%d",&v1[i]);
      printf("\nInserisci l'elemento del secondo vettore di indice %u ",i);
      scanf("%d",&v2[i]);
   }
   prodotto=0;
   for (i=0; i<N; i++)
      prodotto+=v1[i]*v2[i];
   printf("Il prodotto dei due vettori e' %d ",prodotto);
}
```

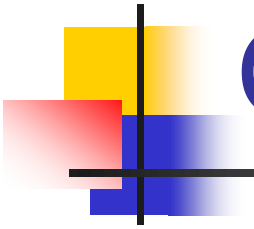
# Gli Array o Vettori

**Esercizio:**

**Fusione di 2 Vettori Ordinati**

```
#include <stdio.h>
#define DIM1 5
#define DIM2 7
float vettore1[DIM1], vettore2[DIM2], vettore3[DIM1+DIM2];
unsigned int i, j, k, FineVettore;

int main(void)
{
       for (i=0;i<DIM1;i++)  do {
              printf("\n Inserisci l'elemento di indice %u del primo vettore ",i);
              scanf("%f",&vettore1[i]);
           } while (i>0 && vettore1[i]<=vettore1[i-1]);
       for (j=0;j<DIM2;j++)  do {
              printf("\n Inserisci l'elemento di indice %u del secondo vettore ",j);
              scanf("%f",&vettore2[j]);
           } while (j>0 && vettore2[j]<=vettore2[j-1]);
       i=j=k=0;
       while (i<DIM1 && j<DIM2)
             if (vettore1[i]<vettore2[j]) vettore3[k++]=vettore1[i++];
             else if (vettore1[i]>vettore2[j]) vettore3[k++]=vettore2[j++];
                    else {
                       vettore3[k++]=vettore1[i++];
                       j++;
                    }
       while (i<DIM1) vettore3[k++]=vettore1[i++];
       while (j<DIM2) vettore3[k++]=vettore2[j++];
       FineVettore=k;
       for (i=0;i<FineVettore; i++)
              printf("\n Elemento di indice %u del terzo vettore = %f",i,vettore3[i]);
}
```



# Gli Array Multidimensionali

Il formato della dichiarazione degli **array *multidimensionali*** è:

***tipo nome*[*dimensione1*][*dimensione2*]...[*dimensioneN*];**

In una *matrice* o array *bidimensionale* i dati sono organizzati per righe e per colonne.

int mat[4][3];

- mat contiene 4 righe e 3 colonne per un totale di dodici elementi;
- per accedere a ciascuno di essi si utilizzano due indici:
  il primo specifica la riga, il secondo la colonna.
- Gli indici variano rispettivamente tra 0 e $r - 1$ e tra 0 e $c - 1$, dove $r$ e $c$ sono il numero di righe e il numero di colonne.
- Anche per le matrici, le dimensioni possono essere definite con il **#define**

# Gli Array Multidimensionali o Matrici

- E' possibile utilizzare le parentesi {} per l'inizializzazione:

***int m[3][2]={{1,2},{3,4},{5,6}};***

prima riga ⟶ ***1 2***

***3 4***

***5 6***

- Se la lista è incompleta, allora i rimanenti elementi sono messi a 0:

***int m[3][2]={{1},{3,4}};***

prima riga ⟶ ***1 0***

***3 4***

***0 0***

# Gli Array Multidimensionali o Matrici

- E' possibile omettere SOLO la prima dimensione

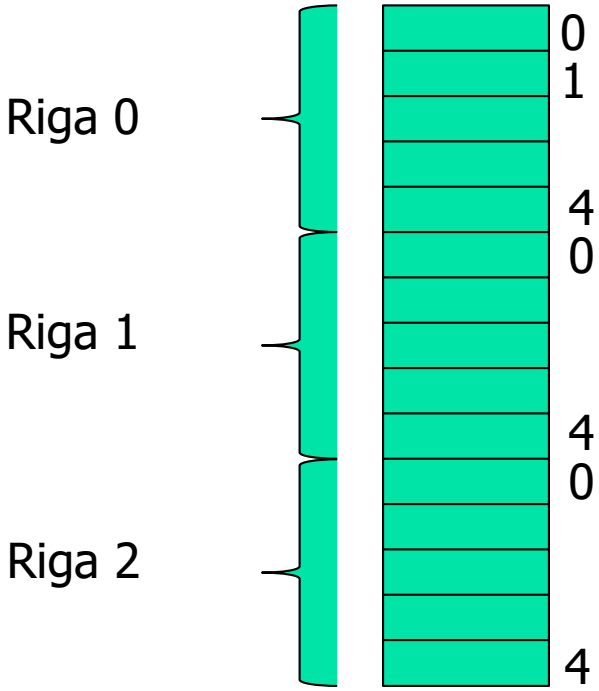
**int m[][2]={{1,2},{3,4},{5,6}};**

- E' possibile assegnare i singoli elementi (tutti gli altri sono zero):

**int m[3][2] = { [1][1] = 1, [2][1] =2 };**

# Gli Array Multidimensionali o Matrici

- Nella pratica gli elementi di una matrice sono memorizzati in modo sequenziale, riga per riga (*row-major order*), ovvero, nell'ordine: prima tutti gli elementi della riga 0, poi tutti gli elementi della riga 1 e così via per gli elementi delle rimanenti righe




***int m[3][5];***

# Gli Array Multidimensionali o Matrici

**Assegnazione**

```
m[0][1] = 71;
m[1][0] = 4;
```

**scanf**

```
scanf("%d",&m[0][1]);
scanf("%d",&m[1][0]);
scanf("%d",&m[i][j]);
```

**print**

```
printf("%d", m[0][1]);
```

# Gli Array Multidimensionali o Matrici

**Riempimento di una Matrice Bidimensionale**

```
for (i=0; i<3; i++)
  for (j=0; j<2; j++) {
     printf("\nElemento di indice %u,%u ", i,j);
     scanf("%d", &m[i][j]);
  }
```

**Stampa di una Matrice Bidimensionale**

```
printf("\nLa Matrice e' ");
for (i=0;i<N;i++){
   printf("\n");
   for (j=0;j<M;j++)
      printf("M[%u][%u] = %d   ",i,j,m[i][j]);
}
```

# Prodotto di due Matrici

```
PRIMA MATRICE
     1      0      0
    22     -6      3
     5      2      0
    11      4      7

SECONDA MATRICE
     2      0      4      0      3
     0      1      5      1      4
    21      1      2      2      5

MATRICE PRODOTTO
     2      0      4      0      3
   107     -3     64      0     57
    10      2     30      2     23
   169     11     78     18     84
```

Date m1[N][T] e m2[T][M], la *matrice prodotto* m3 è dunque costituita da *N* righe e *M* colonne, ossia m3[N][M].

Il Generico elemento m3[i][j] è dato da:

$$m3[i][j] = \sum_{K=0}^{T-1} m1[i][k]*m2[k][j]$$

per i=0….N-1, j=0….M-1

**m3[2][4] =**

**5*3 + 2*4 + 0*5 = 23**

# Prodotto di due Matrici

Codice in C che realizza il prodotto:

```
for (i=0; i<N; i++)
    for (j=0; j<M; j++) {
        m3[i][j]=0;
        for (k=0;k<T; k++)
            m3[i][j]+=m1[i][k]*m2[k][j];
    }
```