GLI ARRAY

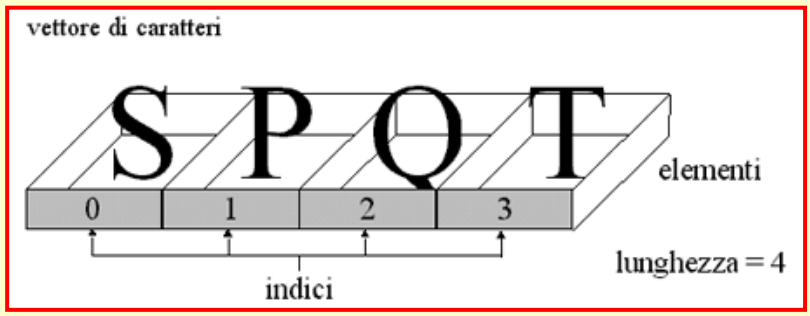
Prof. Orazio Mirabella

Array

- E' una variabile strutturata dove è possibile memorizzare più valori tutti dello stesso tipo.
- un array monodimensionale o *vettore* può essere immaginato come un contenitore suddiviso in tanti scomparti quanti sono i dati che vi si vogliono memorizzare.
- Ognuno di questi scomparti, detti *elementi* del vettore, contiene un unico dato ed è individuato da un numero progressivo, detto *indice*, che specifica la posizione dell'elemento all'interno del vettore stesso.
- L'indice può assumere valori interi da zero al numero totale di elementi meno 1.
- L'indice di base dell'array è sempre zero.
- Il numero complessivo degli elementi del vettore viene detto *lunghezza*.

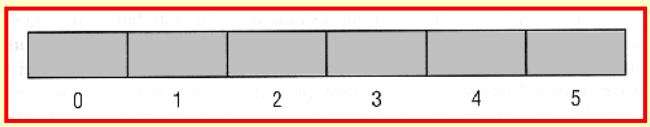
Array

- un vettore è una struttura di dati composta da un numero determinato di elementi tutti dello stesso tipo, ognuno dei quali è individuato da un indice specifico.
- Il tipo dei dati contenuti nel vettore viene detto *tipo del vettore*.



Dichiarazione di un Array

- Occorre innanzitutto definire il tipo, il nome e la lunghezza, cioè il numero di elementi che lo compongono.
- Es: int a[6];
- La lunghezza dev'essere un intero positivo.



- Per accedere a un singolo elemento di un array si deve specificare il nome del vettore seguito dall'indice dell'elemento posto tra parentesi quadre. Es: a[1], oppure a[3] etc.

Lettura e scrittura di un Array

- a[0] = 71; a[1] = 4;
- E' possibile assegnare ad un elemento dell'array il valore di una variabile b,purchè dello stesso tipo: a[3] = b;
- un singolo elemento di un array può essere utilizzato esattamente come una variabile semplice.

  Es: b = b + a[0] * a[5];

- Spesso l'array viene trattato all'interno di iterazioni

```
/* Inizializzazione dell'array */
for(i=0; i<=5; i++) {
printf("Inser. intero: ");
scanf("%d", &a[i]);
}
```

Esempio sugli Array

```
/* Memorizza in un array di interi il
punteggio raggiunto da sei studenti
e ne determina il maggiore, il
minore e la media */

#include <stdio.h>
main()
{
int voti[6];
int i, max, min;
float media;
printf("VOTI STUDENTI\n\n");
/* Immissione voti */
for(i=0; i<=5; i++) {
printf("Voto %dº studente: ", i+1);
scanf("%d", &voti[i]);
}
```

```
/* Ricerca del maggiore */
max = voti[0];
for(i=1; i<=5; i++)
if(voti[i]>max)
max = voti[i];
/* Ricerca del minore */
min = voti[0];
for(i=1; i<=5; i++)
if(voti[i]<min)
min = voti[i];
/* Calcolo della media */
media = voti[0];
for(i=1; i<=5; i++)
media = media + voti[i];
media = media/6;
printf("Maggiore: %d\n", max);
printf("Minore: %d\n", min);
printf("Media: %f\n", media);
}
```

Spazio in memoria degli array

- L'occupazione in memoria di un array ad una dimensione dipende dalle dimensioni e dal tipo, e può essere calcolata nel modo seguente

```
Num.totale byte = sizeof (tipo) x dim. dell'array
```

- Il linguaggio C non esegue una verifica sui limiti degli array, pertanto esiste la possibilità di superarli. Se ciò dovesse accadere si rischierebbe di alterare un'altra variabile oppure verificando dei malfunzionamenti durante l'esecuzione.
- È compito del programmatore evitare il verificarsi di questi casi con gli opportuni controlli.

Caso di segmentation fault

se int ***a*[10]** e' un array, il primo elemento e' ***a[0]*** e l'ultimo e' **a[9]**.

- Il compilatore non puo' rivelare un tale errore: la memoria relativa ad a[10] può essere sempre letta e scritta, ma non fa parte dell'array!!
- In caso di lettura, il valore letto non ha alcun significato logico ai fini del programma, mentre in caso di scrittura si rischia di sovrascrivere qualche altro dato importante che risiede nella locazione di memoria subito dopo le locazioni assegnate all'array.
- Per il compilatore, a[10] è semplicemente la word che si trova a un certo numero di byte dall'indirizzo al quale l'array è memorizzato.

Cosa accade se si tenta di referenziare un elemento che non fa parte dell'array, ad esempio a[10] ?

- Se la locazione di memoria referenziata non appartiene al programma, il programma stesso termina con un errore (segmentation fault)

Inizializzazione di variabili

- L'inizializzazione di una variabile può essere esplicitata direttamente al momento della sua dichiarazione.

  Es.: int i = 0;
- L'istruzione dichiara la variabile i di tipo intero e le assegna il valore zero.
- Analogamente si possono assegnare valori agli altri tipi di variabili semplici:

  float x = 567.8927;

  float y = 7e13;

  char risposta = 's';
- Per gli array l'inizializzazione è possibile solamente se sono stati dichiarati come extern o come static;.
- Le variabili extern sono quelle che vengono definite prima di main. L'inizializzazione si ottiene inserendo i valori tra parentesi graffe, separati da una virgola:
- int voti[6] = {11, 18, 7, 15, 21, 9};

Esempio sugli Array(1/3)

- Per determinare la destrezza di *n* concorrenti sono state predisposte due prove, entrambe con una valutazione che varia da 1 a 10; il punteggio totale di ogni concorrente è dato dalla media aritmetica dei risultati delle due prove. Si richiede la visualizzazione di una tabella che contenga su ogni linea i risultati parziali e il punteggio totale di un concorrente.
- /* dichiarazioni*/
- #include <stdio.h>

```
#define MAX_CONC 1000 /* massimo numero di concorrenti */
#define MIN_PUN 1 /* punteggio minimo per ogni prova */
#define MAX_PUN 10 /* punteggio massimo per ogni prova */
main()
{
float prova1[MAX_CONC], prova2[MAX_CONC], totale[MAX_CONC];
int i, n;
    do
{
    printf("\nNumero concorrenti: ");
    scanf("%d", &n);
}
while(n<1 || n>MAX_CONC);
```

Esempio sugli Array(2/3)

- /* Per ogni concorrente, richiesta punteggio nelle due prove */

```
for(i=0; i<n; i++)
{
        printf("\nConcorrente n.%d \n", i+1);
        do {
        printf("Prima prova: ");
        scanf("%f", &prova1[i]);
        }
while(prova1[i]<MIN_PUN || prova1[i]>MAX_PUN);
        do
         {
                printf("Seconda prova: ");
                scanf("%f", &prova2[i]);
        }
while(prova2[i]<MIN_PUN || prova2[i]>MAX_PUN);
}
```

Esempio sugli Array(3/3)

```
/* Calcolo media per concorrente */
for(i=0; i<n; i++)
totale[i] = (prova1[i]+prova2[i])/2;
printf("\n CLASSIFICA\n");
for(i=0; i<n; i++)
printf("%f %f %f \n", prova1[i], prova2[i], totale[i]);
}
```

Esempio di esecuzione

Numero concorrenti: **3**
Concorrente n.1
Prima prova: **8** Seconda prova: **7**
Concorrente n.2
Prima prova: **5** Seconda prova: **9**
Concorrente n.3
Prima prova: **8** Seconda prova: **8**

CLASSIFICA
8.000000 7.000000 7.500000
5.000000 9.000000 7.000000
8.000000 8.000000 8.000000

Array bidimensionali: matrici

- In una matrice bidimensionale i dati sono organizzati per righe e per colonne, come se fossero inseriti in una tabella. Per la memorizzazione si utilizza una variabile di tipo array specificando il numero di componenti per ciascuna delle due dimensioni che la costituiscono:

ES.: int mat[4][3];

- Una matrice è caratterizzata da un numero di righe e di colonne. 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

mat[0][0] mat[0][1] mat[0][2]
mat[1][0] mat[1][1] mat[1][2]
mat[2][0] mat[2][1] mat[2][2]
mat[3][0] mat[3][1] mat[3][2]

Array bidimensionali: matrici

- **Il formato generale della dichiarazione degli array multidimensionali è il seguente:**

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

```
/* Caricamento di una matrice */
#include <stdio.h>
int mat[4][3];
main()
{
int i, j;
printf("\n \n CARICAMENTO DELLA MATRICE \n \n");
for(i=0; i<4; i++)
     for(j=0; j<3; j++)
    {
     printf("Inserisci linea %d colonna %d val: ", i, j);
     scanf("%d", &mat[i][j]);
     };
```

Array bidimensionali: matrici

```
/* Visualizzazione */
for(i=0; i<4; i++) {
printf("\n");
for(j=0; j<3; j++)
printf("%5d", mat[i][j]);
}
}
```

- Si può ottenere il caricamento per colonne invertendo semplicemente i due cicli:
- for(j=0; j<3; j++)
  for(i=0; i<4; i++) {
  printf("Inserisci linea %d colonna %d val:", i, j);
  scanf("%d", &mat[i][j]);
  };
- Il numero totale d'iterazioni è sempre uguale a 12 ottenuta con due cicli for annidati uno nell'altro.

Array bidimensionali: matrici

```
/* Caricamento di una matrice le cui
     dimensioni vengono decise dall'utente */
#include <stdio.h>
#define MAXLINEE 100
#define MAXCOLONNE 100
int mat[MAXLINEE][MAXCOLONNE];
main()
{
int n, m;
int i, j;
/* Richiesta delle dimensioni */
do {
    printf("\nNumero di linee: ");
    scanf("%d", &n);
}
while((n>=MAXLINEE) || (n<1));
do {
    printf("Numero di colonne: ");
    scanf("%d", &m);
}
while((m>=MAXCOLONNE) || (m<1));
```

```
printf("\n \n CARICAMENTO DELLA
     MATRICE \n \n");
for(i=0; i<n; i++)
    for(j=0; j<m; j++) {
    printf("Inserisci linea %d colonna %d val:",
     i, j);
    scanf("%d", &mat[i][j]);
};
/* Visualizzazione */
for(i=0; i<n; i++) {
    printf("\n");
    for(j=0; j<m; j++)
        printf("%5d", mat[i][j]);
}
}
```