4 GLI ARRAY E LE STRINGHE

4.1 Gli array monodimensionali

Un array è un insieme di variabili dello stesso tipo, cui si fa riferimento mediante uno stesso nome. L'accesso ad un determinato elemento si realizza mediante un indice.
Come ogni variabile, affinché il compilatore possa allocare lo spazio di memoria necessario, l'array deve essere dichiarato. La forma generale di dichiarazione di un array monodimensionale è la seguente:

tipo nome_array [*dimensione*];

Ove *tipo* specifica il tipo di ogni elemento dell'array, mentre *dimensione* definisce il numero di elementi contenuti nell'array.
Nel seguente esempio viene dichiarato un array, con nome Stipendi, di 30 elementi di tipo intero:

```
int Stipendi[30];
```

Per accedere ad un elemento dell'array occorre specificarne l'indice, come nel codice seguente:

```
Stipendi[5]=1250;
```

Si tenga presente che il primo elemento di un array ha indice 0. Pertanto, la dichiarazione dell'array Stipendi implica la creazione di 30 elementi che vanno da `Stipendi[0]` a `Stipendi[29]`.
Il compilatore C non restituisce alcun errore se viene superato il limite di un array; è affidata al programmatore la verifica dell'indice per evitare che questo superi i limiti dimensionali.

La quantità di memoria richiesta da un array monodimensionale è legata alla dimensione ed al tipo:
*byte totali = sizeof(tipo)*dimensione*

4.1.1 Passaggio di un array monodimensionale ad una funzione

Il passaggio di un array ad una funzione si realizza specificando il nome dell'array senza alcun indice, ovvero, specificando un puntatore all'array (i puntatori saranno trattati nel prossimo capitolo).
Il seguente codice passa a Func1() l'array Pippo.

```
int main(void) {
  int Pippo[10] ;
  ...
  Func1(Pippo) ;
  ...
}
```

Il parametro formale della funzione Func1, che deve ospitare l'array, può essere dichiarato in due modi: come array dimensionato o come array non dimensionato. Negli esempi seguenti vengono evidenziate le due modalità:

```
/* array dimensionato */
void Func1(int x[10]) {
```

```
  ...
}


/* array non dimensionato */
void Func1(int x[]) {
  ...
}
```

Nel secondo caso si specifica che la funzione Func1 riceverà un array di tipo int, di lunghezza non dichiarata esplicitamente.
In realtà, esiste un terzo metodo che dichiara il parametro formale come puntatore, ma ciò sarà oggetto del prossimo capitolo.

4.2 Le stringhe

Una stringa consiste in una sequenza di caratteri. A differenza di altri linguaggi di programmazione, che definiscono le stringhe come tipi elementari, in C le stringhe vengono implementate mediante array. In particolare, una stringa in C è costituita da un array di caratteri che termina con il carattere nullo.
Quando si dichiara una stringa occorre specificarne la dimensione, che sarà pari al numero di caratteri che deve contenere aumentato di una unità per il carattere nullo. Ad esempio, la seguente scrittura dichiara una stringa che può contenere 20 caratteri:

```
char strNome[21];
```

Questa istruzione lascia uno spazio per il carattere nullo al termine della stringa.
E' possibile inizializzare una variabile stringa, quotando il testo che deve contenere, mediante il ricorso alla funzione strcpy presente nella libreria string:

```
strcpy(strNome,"Pinco Pallino");
```

Non è necessario aggiungere alla sequenza "*Pinco Pallino*" il carattere nullo per terminare la stringa, questa operazione viene svolta automaticamente dal compilatore C.
Come negli array la scrittura strNome[5] restituirà il sesto carattere contenuto nella stringa strNome (il primo ha indice 0).

Le librerie del C includono una serie di funzioni per la manipolazione delle stringhe. Le più comuni sono:

Nome	**Funzione**
strcpy(s1,s2)	Copia s2 in s1
strcat(s1,s2)	Concatena s2 alla fine di s1
strlen(s1)	Restituisce la lunghezza di s1
strcmp(s1,s2)	Restituisce 0 se s1 e s2 sono uguali; un valore minore di 0 se s1<s2; maggiore di 0 se s1>s2
strchr(s1,ch)	Restituisce un puntatore alla prima occorrenza di ch in s1

strstr(s1,s2)	Restituisce un puntatore alla prima occorrenza di s2 in s1

Queste funzioni possono essere utilizzate includendo nel proprio progetto il file header standard string.h.

4.3 Gli array bidimensionali

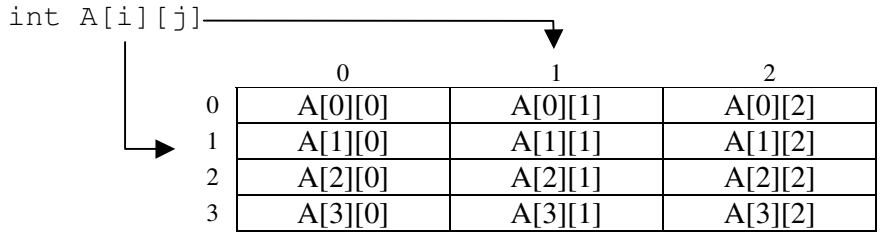
Un array bidimensionale viene dichiarato nel seguente modo:

tipo Nome[*dimensione1*][*dimensione2*]

Esso viene memorizzato in una matrice di righe e colonne: *dimensione1* specifica il numero delle righe, mentre *dimensione2* il numero delle colonne.
Nell'esempio seguente viene dichiarato un array, di tipo intero, costituito da 4 righe e 3 colonne.

```
int A[4][3]
```

In figura sono evidenziati i singoli elementi dell'array:



Lo spazio di memoria richiesto da un array bidimensionale è espresso dalla seguente formula:
*byte totali = sizeof(tipo) * dimensione1 * dimensione2*

In una funzione il parametro formale, che riceve un array bidimensionale, deve definire almeno le dimensioni che si trovano più a destra, in quanto il compilatore C deve conoscere la lunghezza di ogni riga per accedere correttamente ai vari elementi dell'array. Affinché una funzione possa ricevere l'array `A` dell'esempio precedente, dev e essere dichiarata nel seguente modo:

```
void Func1(int x[][3]) {
  ...
}
```

4.3.1 Gli array di stringhe

Per creare un array di stringhe, si deve fare ricorso ad un array bidimensionale di caratteri. La dimensione dell'indice di sinistra determina il numero delle stringhe, quella dell'indice di destra specifica il numero massimo di caratteri di ciascuna stringa.
La seguente dichiarazione crea un array di 20 stringhe lunghe massimo 29 caratteri:

```
char Elenco[20][30];
```

La scrittura:

```
strcpy(Elenco[5],"Pinco Pallino");
```

assegna alla sesta stringa dell'array Elenco la sequenza di caratteri indicata.

4.4 L'inizializzazione degli array

Come ogni variabile, è possibile inizializzare un array al momento della dichiarazione. La forma generale dell'inizializzazione di un array è la seguente:

tipo nome_array [*dim1*] [*dim2*]… [*dimN*]={*elenco_valori*};

ove *elenco_valori* è un elenco di valori separati da virgole, in cui il primo valore sarà posizionato nella primo elemento dell'array, il secondo nel secondo elemento e così via.
Nel seguente esempio viene dichiarato ed inizializzato un array di 10 elementi di tipo intero:

```
int A[10]={10,20,30,40,50,60,70,80,90,100};
```

Dopo la dichiarazione `A[0]` conterrà il valore 10, `A[1]` il valore 20 e così via tutti gli altri.

È possibile inizializzare gli array di caratteri (stringhe) in una maniera più immediata:

char *nome_array* [*dim*] = "*stringa*";

Nell'esempio seguente viene dichiarata una stringa di 20 caratteri ed inizializzata con la frase "Buona giornata":

```
char Prova[20]="Buona giornata";
```

Ciò equivale a scrivere:

```
char Prova[20]={'B', 'u', 'o', 'n', 'a', ' ', 'g', 'i', 'o', 'r', 'n', 'a', 't', 'a', '\0'};
```

Poiché in C tutte le stringhe terminano con il carattere nullo, bisogna assicurarsi che l'array dichiarato sia lungo abbastanza per contenere tale carattere.

È possibile inizializzare gli array bidimensionali in maniera analoga agli array monodimensionali; nell'esempio seguente si inizializza `A` con i numeri da 1 a 10 ed i rispettivi quadrati:

```
int Sqr[10][2]={1,1,2,4,3,9,4,16,5,25,6,36,7,49,8,64,9,81,10,100};
```

In generale, è preferibile raggruppare tutti gli elementi che si riferiscono ad una dimensione tra parentesi quadre. La precedente dichiarazione può, quindi, essere espressa nel seguente modo:

```
int Sqr[10][2]={
  {1,1},
  {2,4},
  {3,9},
  {4,16},
```

```
  {5,25},
  {6,36},
  {7,49},
  {8,64},
  {9,81},
  {10,100}
};
```

4.4.1 Gli array non dimensionati

In C è possibile creare array non dimensionati unicamente se associati ad una inizializzazione. Se nell'istruzione di inizializzazione di un array non si specificano le dimensioni, il compilatore C creerà automaticamente un array sufficientemente grande per contenere tutti gli inizializzatori presenti. La seguente istruzione genera un array (stringa) non dimensionato, grande a sufficienza per ospitare la sequenza di caratteri indicata a destra dell'uguale:

```
char messaggio[]="Ciao Ciao";
```