Tipi di dati derivati: gli Array e i puntatori

Salvatore Filippone
salvatore.filippone@uniroma2.it

Tipi di dati derivati

È possibile specificare dei tipi di dati derivati, ossia diversi dai tipi di dati nativi visti finora. Una prima categoria è quella dei dati *aggregati*: collezioni di oggetti, aventi ciascuno un tipo *base*.

Tipi di dati derivati

È possibile specificare dei tipi di dati derivati, ossia diversi dai tipi di dati nativi visti finora. Una prima categoria è quella dei dati *aggregati*: collezioni di oggetti, aventi ciascuno un tipo *base*.

Array

Un *array* è un tipo di dato aggregato costituito da molteplici dati elementari i quali:

- Hanno tutti lo stesso tipo base;
- Sono identificati da un indice numerico (o da un insieme di indici).

Il numero di indici necessario per identificare un elemento si dice *dimensione*; avremo quindi array monodimensionali, bidimensionali, etc.

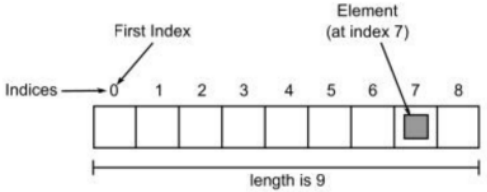
Array

Un esempio di array monodimensionale

```
1  #include <stdio.h>
2  #define N 100
3  int main(int argc, char *argv[])
4  {
5    int a[N],i;
6
7    for (i=0; i<N; i++)
8       a[i]=i;
9
10 }
```

- Il primo elemento corrisponde all'indice 0 (e l'ultimo a $N-1$);
- L'indirizzo in memoria dell'elemento i si trova aggiungendo $i * S$ all'indirizzo base di a, dove S è la dimensione in bytes di ciascun elemento

Un esempio di array monodimensionale



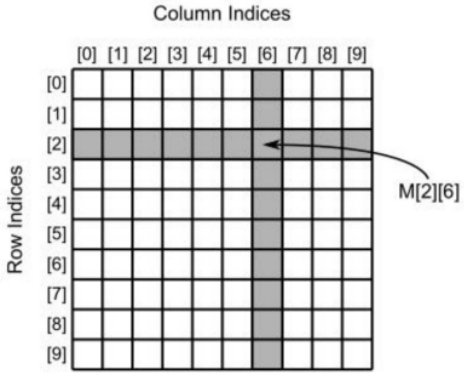
Array

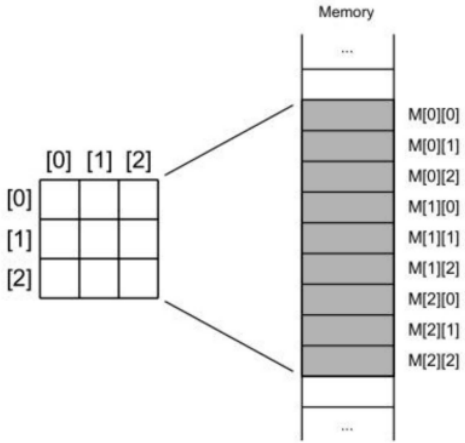
Un esempio di array bidimensionale

```
1  #include <stdio.h>
2  #define M 50
3  #define N 100
4  int main(int argc, char *argv[])
5  {
6    int a[M][N],i;
7
8    for (i=0; i<M; i++)
9      for (j=0; j<N; j++)
10       a[i][j]=M*i+j;
11
12 }
```

- Il primo elemento corrisponde all'indice [0][0] (e l'ultimo a $[M-1][N-1]$);
- La memorizzazione avviene per *righe*;
- L'indirizzo in memoria dell'elemento i, j si trova aggiungendo $((i * N) + j) * S$

Un esempio di array bidimensionale





Array

Come si fa a passare un array ad una funzione? Ci sono due modi:

1. Il modo "tradizionale" usando un puntatore;
2. il modo "moderno" usando lo standard C99 e seguenti

```
1        void fill(int m, int n, int a[m][n])
2        { int i,j
3          for (i=0; i<m; i++)
4          for (j=0; j<n; j++)
5          a[i][j]=m*i+j;
6        }
7        int main(int argc, char *argv[])
8        {
9          int a[M][N],i;
10         fill(M,N,a);
11       }
```

Array e stringhe

Nel linguaggio C una stringa è semplicemente un array di caratteri, terminato con il carattere '\0':

```c
#define M 6
int main(int argc, char *argv[])
{
  char name[M];
  name[0] = 'P';
  name[1] = 'i';
  name[2] = 'p';
  name[3] = 'p';
  name[4] = 'o';
  name[5] = '\0';

  printf("%s\n",name);
}
```

Array

Come si fa a passare un array ad una funzione?
Il modo "tradizionale" usando un puntatore:

```
1  void fill(int m, int *a)
2  { int i,j
3    for (i=0; i<m; i++)
4      a[i]=i;
5  }
6  int main(int argc, char *argv[])
7  {
8    int a[M];
9    fill(M,a);
10 }
```

Ma cosa è un puntatore?