12.6 Tipi derivati composti

Uno dei punti di forza del C è la capacità di combinare insieme tipi derivati e fondamentali per ottenere strutture dati complesse a piacere, in grado di modellare entità del mondo reale. Per esempio

```
Auto salone[100];
```

definisce un array di 100 variabili di tipo `Auto`, che potrebbe essere usato per mantenere memoria di tutti i tipi di auto e del relativo venduto, ponendo un limite massimo di 100 autovetture disponibili.
Il C è dunque flessibile e permette la costruzione di tipi derivati complessi. Questa caratteristica, comoda nella fase di creazione di un programma, può tuttavia essere fonte di problemi soprattutto quando si rilegge un programma scritto da altri. Per esempio, una dichiarazione del tipo

```
char *(*(*var)())[10];
```

potrebbe risultare non immediatamente comprensibile. Esistono per fortuna delle semplici regole che consentono di interpretate una volta per tutte qualsiasi tipo di dichiarazione. Infatti una dichiarazione di una variabile va sempre letta dall'interno verso l'esterno secondo i seguenti passi:

1. 1. si individua all'interno della dichiarazione l'identificatore della variabile e si controlla se alla sua destra ci sono parentesi aperte, tonde o quadrate;
2. 2. si interpretano le eventuali parentesi tonde o quadrate, e poi si guarda alla sinistra per vedere se c'è un asterisco;
3. 3. se si incontra durante un qualsiasi passo una parentesi tonda chiusa, si torna indietro e si riapplicano le regole 1 e 2 a tutto ciò che si trova all'interno delle parentesi;
4. 4. infine si applica lo specificatore di tipo.

Proviamo subito ad applicare queste regole al precedente esempio:

```
char *(*(*frog)()) [10];
      7   6 4 2 1   3     5
```

I vari passi sono etichettati da 1 a 7 e sono interpretati nel modo seguente:

1 l'identificatore di variabile `frog` è un
2 puntatore a
3 una funzione che ritorna un
4 puntatore a
5 un array di 10 elementi, che sono
6 puntatori a
7 valori di tipo `char`

È lasciato al lettore il compito di decifrare le seguenti dichiarazioni:

```
int *frog[5];
     int (*frog)[5];
```

Di tutte le possibili combinazioni di tipi derivati ne esistono alcune più frequentemente usate:

1. 1. tipi derivati composti tramite struttura;
2. 2. tipi derivati composti tramite funzione;
3. 3. tipi derivati composti tramite puntatore.

Per ogni classe di tipo composto studieremo alcune possibili composizioni, facendo particolare riferimento a quelle in cui intervengono puntatori e array. Molte di queste composizioni sono già state trattate nei capitoli e paragrafi precedenti. Altre, come per esempio i puntatori a funzione, sono invece introdotte per la prima volta