14.2 Liste lineari

Una lista lineare è una successione di elementi omogenei che occupano in memoria una posizione qualsiasi. Ciascun elemento contiene un'informazione e un puntatore per mezzo del quale è legato al successivo. L'accesso alla lista avviene con il puntatore al primo elemento. Si ha dunque:

```
Elemento = informazione + puntatore
```

Il puntatore è il riferimento a un elemento, il suo valore è l'indirizzo dell'elemento nella memoria del sistema. Indichiamo con `inf` la parte informazione di ogni elemento, con `pun` la parte puntatore e con `punt_lista` il puntatore al primo elemento della lista, come in Figura 14.2.

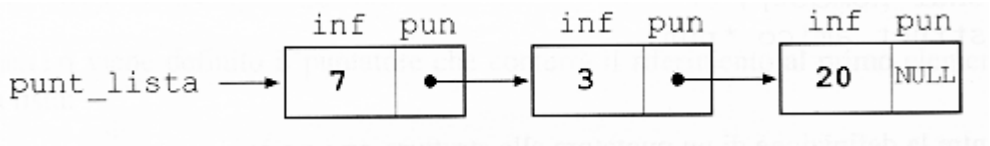
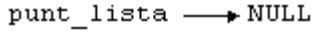


Figura 14.2 Esempio di lista lineare

Il campo puntatore dell'ultimo elemento della lista non fa riferimento a nessun altro elemento; il suo contenuto corrisponde a un segnale di fine lista che in C è il valore `NULL`. Una *lista vuota* non ha elementi ed è rappresentata da `punt_lista` che punta a `NULL`.



La parte informazione dell'elemento dipende dal tipo di dati che stiamo trattando. In C può essere costituita da uno qualsiasi dei tipi semplici che conosciamo: `int`, `float` ecc. Nel caso della lista lineare di Figura 14.2 il cui campo informazione è di tipo intero, la dichiarazione della struttura di ogni elemento può essere la seguente:

```
struct elemento {
       int inf;
       struct elemento *pun;
     };
```

dove `inf` è di tipo `int` e `pun` è un puntatore a una struttura di tipo elemento. La scelta del nome della struttura e dei nomi dei campi, in questo caso elemento, `inf` e `pun`, è libera, nell'ambito degli identificatori ammessi dal linguaggio. La precedente dichiarazione descrive la struttura di elemento ma non alloca spazio in memoria. La definizione:

```
struct elemento *punt_lista;
```

stabilisce che `punt_lista` è un puntatore che può riferirsi a variabili `elemento`. Non esistono variabili puntatore in generale, ma variabili puntatore che fanno riferimento a oggetti di un determinato tipo.

La parte informazione può essere anche composta da più campi, ognuno dei quali di un certo tipo. Per esempio, se si desiderano memorizzare nella lista nomi ed età degli amici la parte informazione diventa:

```
informazione = nome + anni
```

con nome di tipo array di `char` e `anni` di tipo `int`. La dichiarazione di un elemento diventa allora:

```
struct amico {
       int anno;
       char nome[30];
       struct amico *pun;
     };
```

mentre la definizione di un puntatore alla struttura amico è:

```
struct amico *p_amico;
```

È da sottolineare la posizione in memoria non sequenziale: quando si aggiunge un ulteriore elemento alla lista si deve allocare uno spazio di memoria, connetterlo all'ultimo elemento della lista e inserirvi l'informazione relativa.
Nella struttura lista lineare, così come definita, non esiste alcun modo di risalire da un elemento al suo antecedente. La lista si può *scandire* solo in ordine, da un elemento al successivo, per mezzo dei puntatori. *Scandire* una lista significa esaminare uno per uno i suoi elementi, dove per esaminare si può intendere: leggere, stampare ecc. Il segnale di fine lista `NULL` è importante perché permette di verificare durante la scansione se la lista è terminata.

I problemi che vengono presentati e risolti in questo capitolo permettono di familiarizzare con le liste lineari.