10.1 Iterazione e ricorsione

Quando si vuole ripetere l'esecuzione di un gruppo di istruzioni, un'alternativa alle strutture iterative come `for` o `while` è rappresentata dalle *funzioni ricorsive*. Una funzione si dice ricorsiva se chiama se stessa direttamente o indirettamente.

Per alcune classi di problemi le soluzioni ricorsive sono eleganti, sintetiche e più chiare delle altre. Un esempio di questo fatto si può trovare nel calcolo del fattoriale, esaminato nel Capitolo 3. Ricordiamo che il fattoriale $n!$ del numero n, dove n è un intero maggiore o uguale a 2, è dato da:

$$n! = n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdot \ldots \cdot 2 \cdot 1$$

Inoltre 0! e 1! sono per definizione uguali a 1.

Il fattoriale è ricorsivo per definizione. Può essere espresso come

$$n! = n \cdot (n-1)!$$

ovvero il fattoriale di n è uguale a n moltiplicato per il fattoriale di $n-1$; ricorsivamente, 4! è dunque uguale a 4 moltiplicato per il fattoriale di $(4-1)$, e così via.

Se `fat` è la funzione che calcola il fattoriale, dovrà allora ricevere in ingresso il numero intero su cui operare e dovrà restituirne il fattoriale (Figura 10.1).

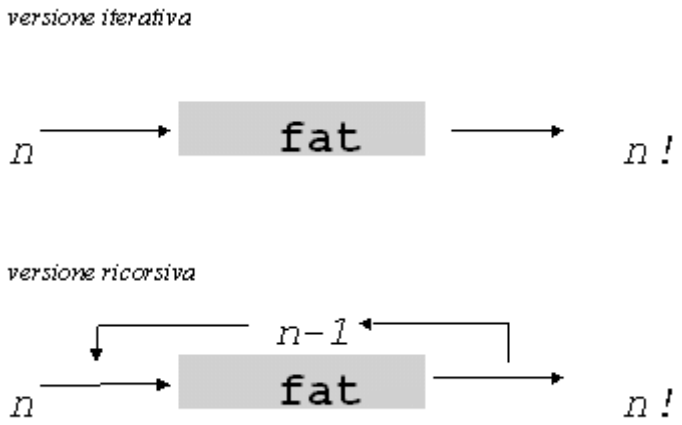

Figura 10.1 La funzione `fat` restituisce il fattoriale di `n`

Per restituire il fattoriale al programma chiamante la funzione utilizzerà l'istruzione `return`:

```
return(n*fat(n-1));
```

che ritorna il valore di `n` moltiplicato per il fattoriale di `n-1`. Il calcolo del fattoriale di `n-1` lo si ottiene invocando ancora una volta la stessa funzione `fat` e passandole come argomento `n-1`; in questo modo si ottiene l'iterazione. Il ciclo, a un certo punto, deve terminare, per cui è necessaria una condizione di fine, che impostiamo così:

```
if(n==0)
       return(1);
     else
       return(n*fat(n-1));
```

Quando il valore passato alla funzione è uguale a 0, non ci sono valori da considerare e `fat` ritorna 1 (0!).

Confrontiamo dunque la funzione ricorsiva `fat` con il programma iterativo del Capitolo 3.

procedura ricorsiva

```
fat(int n)
{
if(n==0)
  return(1);
else
  return(n*fat(n-1));
}
```

procedura iterativa

```
if(n==0)
  fat = 1;
else
  for(fat=n; n>2; n--)
    fat = fat*(n-1);
```

La soluzione ricorsiva corrisponde direttamente alla definizione di fattoriale. Nel Listato 10.1 viene presentato un programma completo per il calcolo del fattoriale.

```
/* Calcolo del fattoriale con una funzione ricorsiva */

#include <stdio.h>

fat(int);

main()
{
int n;

printf("CALCOLO DI n!\n\n");
printf("Inser. n: \t");
```

```
scanf("%d", &n);
printf("Il fattoriale di: %d ha valore: %d\n", n, fat(n));
}

fat(int n)
{
if(n==0)
  return(1);
else
  return(n*fat(n-1));
}
```

Listato 10.1 Calcolo del fattoriale mediante una funzione ricorsiva

Esaminiamo ora più da vicino gli ambienti che la funzione ricorsiva genera a ogni sua chiamata, prendendo l'esempio del calcolo di 4! (Figura 10.2).

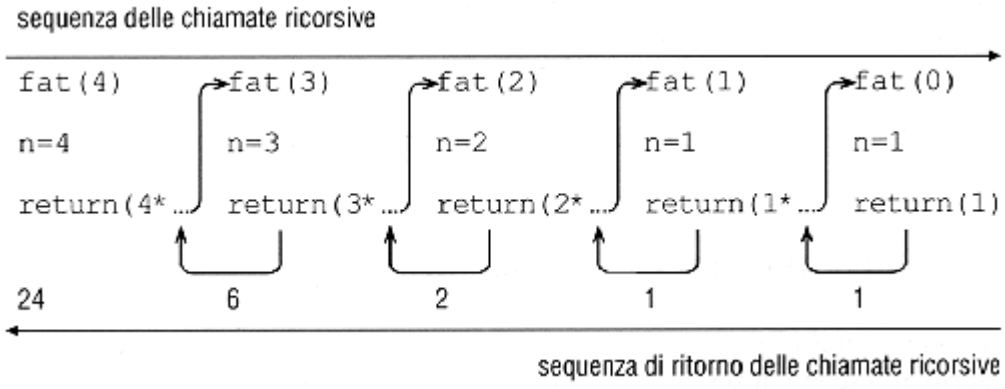

Figura 10.2 Ambienti creati dalla funzione `fat` con ingresso `n`=4

Osservando la figura possiamo vedere che a ogni chiamata di **`fat`** viene creata una nuova variabile `n` locale a quell'ambiente. Quando termina il ciclo delle chiamate, ogni ambiente aperto si chiude e passa all'ambiente precedente il valore calcolato.

> ✓ **NOTA**
> *La zona di memoria riservata alle chiamate viene gestita con la logica di una pila, concetto che tratteremo in modo specifico più avanti, quando parleremo di strutture dati. A ogni invocazione di* `fat`*, il sistema alloca uno spazio di memoria libero in testa alla pila riservato al suo parametro formale* `n`*. In Figura 10.2 si osserva come la sequenza di chiamate e il ritorno delle stesse vengano gestiti come una pila, in cui l'ultimo elemento creato è il primo a essere eliminato. Lo spazio di memoria allocato fa parte dell'ambiente locale a ogni chiamata di* `fat`*.*

Avendo dichiarato `n` e `fat` di tipo `int` (si ricordi che quando nella dichiarazione di funzione non ne viene specificato il tipo, viene implicitamente assunto `int`) si ottengono risultati significativi con valori di `n` piuttosto bassi; per aumentare questo limite si può utilizzare il tipo `long int`, che ha dimensione maggiore o uguale a un `int`. Dobbiamo specificarlo in fase dichiarativa, all'inizio del file:

```
long int fat(long int);
```

nella definizione della funzione:

```
long int fat(long int n) {...};
```

e dobbiamo modificare la `printf` nel programma principale, in modo da indicare il formato in cui si desidera la visualizzazione (`%ld`) di `fat`:

```
printf("Il fattoriale di: %d ha valore: %ld\n", n, fat(n));
```

Se poi si desidera calcolare fattoriali ancora più alti si deve usare una funzione di tipo `float` o `double`. Dato che il risultato è sempre un intero, è meglio specificare nella `printf` di non visualizzare cifre dopo la virgola:

```
printf("Il fattoriale di: %d ha valore: %.0f\n", n, fat(n));
```