Programmazione ricorsiva

In quasi tutti i linguaggi di programmazione moderni è ammessa la possibilità di definire funzioni/procedure **ricorsive**: durante l'esecuzione di una funzione **F** è possibile chiamare la funzione **F** stessa.

Ciò può avvenire:

- ⇨ **direttamente**: il corpo di **F** contiene una chiamata a **F** stessa.
- ⇨ **indirettamente**: **F** contiene una chiamata a **G** che a sua volta contiene una chiamata a **F**.

Questo può sembrare strano: se pensiamo che una funzione è destinata a risolvere un sottoproblema **P**, una definizione ricorsiva sembra indicare che per risolvere **P** dobbiamo . . . saper risolvere **P**.

In realtà, la programmazione ricorsiva si basa sull'osservazione che per molti problemi **la soluzione per un caso generico può essere ricavata sulla base della soluzione di un altro caso, generalmente più semplice, dello stesso problema.**

Un esempio classico di algoritmo ricorsivo

Individuare, in un gruppo di palline, l'unica pallina di peso maggiore delle altre facendo uso di una bilancia "a basculla" (Per semplicità: il numero di palline sia una potenza di 3)

Algoritmo Pesate:

Se il gruppo di palline consiste in una sola pallina, allora essa è banalmente la pallina cercata, altrimenti procedi come segue:

- Dividi il gruppo di palline in tre e confronta due dei tre sottogruppi.
- Se i due gruppi risultano di peso uguale scarta entrambi, altrimenti scarta il gruppo non pesato e quello risultato di peso minore.
- Applica l'algoritmo Pesate al gruppo rimanente.

Principio di induzione ben fondata

La programmazione di algoritmi di natura ricorsiva trova radici teoriche nel **principio di induzione ben fondata**, che può essere visto come una generalizzazione del principio di induzione sui naturali.

Sia A(n) una proprietà vera sull'insieme N dei numeri naturali e supponiamo che:

1. A(0) è vera
2. Per ogni k ∋ N, se è vera A(k) allora è vera A(k+1)

Allora A(n) è vera per ogni n ∋ N.

La soluzione di un problema viene individuata supponendo di saperlo risolvere su casi più semplici.
Bisogna poi essere in grado di risolvere direttamente il problema sui casi più semplici di qualunque altro.

Funzioni ricorsive sono convenienti per implementare funzioni matematiche definite in **modo induttivo**.

Esempio:

Funzione fattoriale.

definizione iterativa:

```
fatt(n) = n • (n - 1) • (n - 2) • ….. • 2 • 1
```

➔ `fatt(n) = 1; se n = 0 (caso base)`

definizione induttiva:

➔ `fatt(n) = n • fatt(n-1); se n > 0 (caso induttivo)`

È essenziale il fatto che, applicando ripetutamente il caso induttivo, ci riconduciamo prima o poi al caso base.

```
fatt(3)= 3•fatt(2) = 3•(2•fatt(1)) = 3•(2•(1•fatt(0))) = 3•(2•(1••1)) = 6
```

Esempio: Funzione fattoriale.

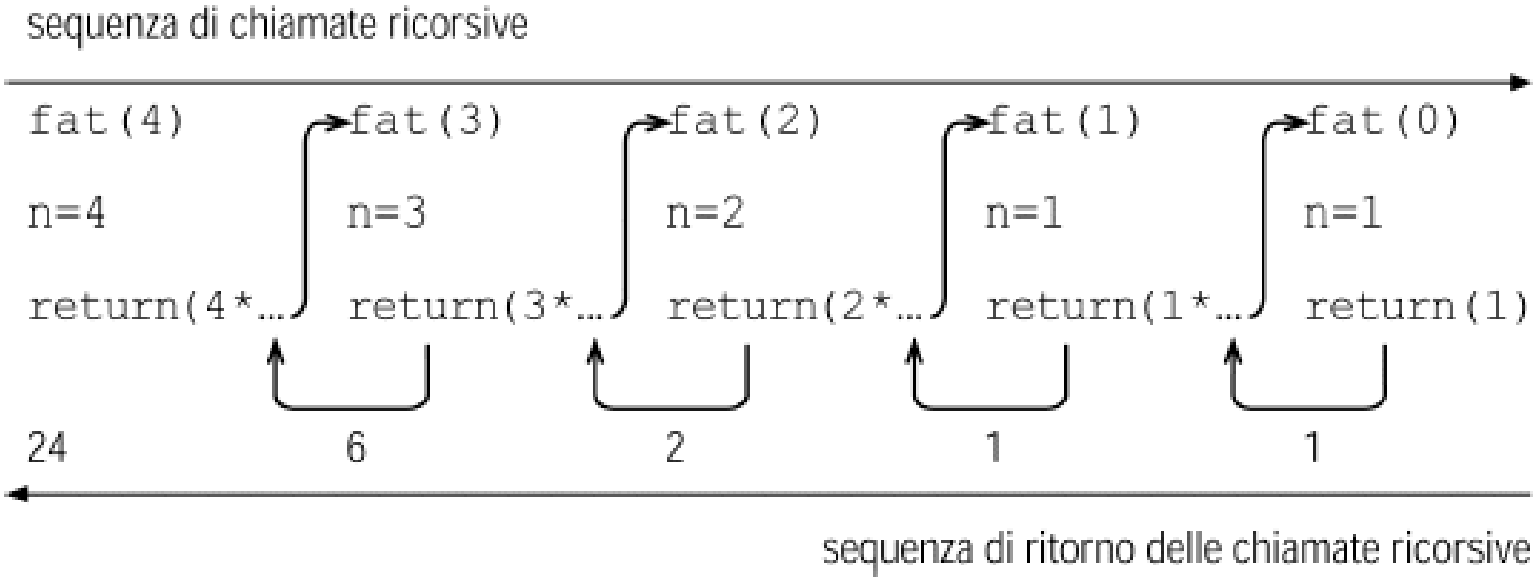
Il codice delle due diverse versioni

definizione iterativa:

```
int fatt(int n)
    {
        int i,ris;
        ris=1;
        for (i=1;i<=n;i++)
        ris=ris*i;
        return ris;
    }
```

definizione ricorsiva:

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



Programmazione ricorsiva

Evoluzione della pila (supponendo x=3).