Esempio: ricorsione

Esercizio: Calcolo della somma dei primi N numeri pari

- Scrivere l'algoritmo e il codice C di una funzione che, dato un numero *N*, calcola la somma dei primi *N* numeri pari in maniera ricorsiva.

- La funzione scritta è tail ricorsiva? Perche'? Se non lo è, è possibile riscriverla in maniera tail-ricorsiva? Se sì, si riscriva la funzione in versione tail-ricorsiva.

Soluzione ricorsiva

Calcolo della somma dei primi N numeri pari

```
int somma_pari(int N)
{
    if (N==0)
        {return 0; }
    else
        return 2*(N-1)+somma_pari(N-1);
}
```

La funzione scritta non è tail ricorsiva, infatti dopo la chiamata ricorsiva deve ancora essere effettuata la somma. Il compilatore non potrebbe quindi effettuare l'ottimizzazione tail, cioè riutilizzare il record di attivazione di una chiamata per eseguirne un'altra.

Soluzione Tail ricorsiva

Per avere una versione tail ricorsiva è necessario inserire un nuovo parametro che rappresenta la somma calcolata fino a quel momento.

```
int somma_pari_tail(int N, S)
{
    if (N==0)
        {return S; }
    else
        return somma_pari_tail(N-1,S+2*(N-1));

}
```

Esempio: potenza di un numero

Esercizio: Potenza di un numero

Scrivere una funzione C che calcola, dati due numeri interi *M* ed *N*, la potenza M^N. Si cerchi, in particolare, di minimizzare il numero dei prodotti che vengono effettuati, sfruttando opportunamente la seguente osservazione:

- Se *N* è dispari, allora $M^N=(M^{N-1})*M$
- Se *N* è pari, allora $M^N=(M^{N/2})^2$

- La funzione scritta è ricorsiva? È tail-ricorsiva? È possibile scriverla in maniera tail-ricorsiva? È possibile scriverla in maniera non ricorsiva?

Soluzione ricorsiva

Potenza di un numero

```
float potenza(float M, int N)
{   float P;
    if (N==0)
        { return 1; }
    else
        if (N%2 == 0) /* Se N è pari */
            { P = potenza(M,N/2);
              return P*P; }
        else
            { return potenza(M,N-1)*M; }
}
```

La funzione è ricorsiva, ma non tail ricorsiva, infatti dopo la chiamata ricorsiva devono ancora essere effettuati dei prodotti (sia nel caso di *N* pari sia in quello di *N* dispari).

Soluzione Tail ricorsiva

Potenza di un numero

È possibile scriverla in maniera tail ricorsiva, osservando che, nella funzione precedente, vengono calcolate tutte le potenze del tipo *M*^(2*i*), per tutti gli *i* tali che $2i<N$ ovvero *M*1, *M*2, *M*4, *M*8, ... Aggiungeremo quindi due parametri. P rappresenta la potenza calcolata fino a questo momento; P2 rappresenta la potenza *M*^(2*i*) accumulata fino a questo momento. Se *N* è pari, allora dovremo moltiplicare P per P2.

```
float potenza_tail(float M, int N, float P, float P2)
{
    if (N==0)
        { return P; }
    else
        if (N%2 == 0)
            { P2=P2*P2;
              return potenza_tail(M,N/2,P,P2);
            }
        else
            { P = P*P2;
              return potenza_tail(M,N-1,P,P2);
            }
}
```

```
float potenza_2(float M, int N)
{
    return potenza_tail(M,N,1,M);
}
```

Soluzione Iterativa

Potenza di un numero

Visto che è stato possibile scriverla in maniera tail ricorsiva, è evidentemente possibile scriverla anche in maniera iterativa. Infatti un compilatore che esegue l'ottimizzazione tail effettuerà il calcolo in maniera iterativa. In questo caso, dovremo essere noi a scrivere il ciclo corrispondente. Prendiamo una variabile *i* che ci servirà come indice e che farà le veci del parametro *N* nella versione tail ricorsiva.

```
float potenza_non_ric(float M, int N)
{ float P, P2;
  int i;
  i = N;
  while (i>0)
   { if (i%2 == 0)
          { P2=P2*P2;
            i = i/2;
          }
      else
          { P = P*P2;
            i = i-1;
          }
   }
}
```