

RICORSIONE E ITERAZIONE

Vediamo le principali differenze tra gli approcci iterativi di iterazione e ricorsione.

Prendiamo un esempio il calcolo del fattoriale:

FATT(m)

IF $m=0$ THEN

$r=1$

ELSE

$r=m \cdot \text{FATT}(m-1)$

RETURN r

FATT(m)

$r=1$

FOR $i=1$ TO N DO

$r=i \cdot r$

RETURN

La differenza principale è che nella versione iterativa abbiamo solo un'allocazione di memoria per i ed r mentre per l'algoritmo ricorsivo abbiamo un'allocazione (in spazi di memoria DIVERSI) ad ogni chiamata, il che ha ovviamente delle implicazioni.

OSS

Dato un algoritmo ricorsivo il più efficiente possibile è sempre possibile tradurlo in un algoritmo iterativo che occupi la stessa quantità di memoria (parliamo di efficienza sulla memoria)

STACK DI ATTIVAZIONE

Quando viene effettuata una chiamata a funzione, viene allocata memoria nello **STACK DEI RECORD DI ATTIVAZIONE**

dove viene salvato un record che contiene i valori delle variabili locali. Nel caso di algoritmi ricorsivi, al termine di un' esecuzione locale vengono ripresi dallo stack i valori delle variabili dell'esecuzione sospesa che chiamava la funzione terminata.

DA RICORSIONE A ITERAZIONE

La ricorsione tiene salvati tutti i valori delle variabili locali, cosa che l'iterazione non fa (perché sovrascrive il valore precedente). Se vogliamo evitare tale comportamento dobbiamo

esplicitare la gestione dello stack dei record, per salvare i valori delle "istantanee" di ogni iterazione.

ALGORITMO GENERALE DI CONVERSIONE

Esiste uno schema generale per esplicitare il salvataggio delle variabili nello stack. L'idea generale è:

WHILE (INIZIO || RIPRESA) DO

IF INIZIO THEN

<SIMULA NUOVA CHIAMATA>

ELSE

<RIPRESA CONTESTO DI ESECUZIONE>

[DA DOVE DOBBIAMO RIPRENDERE?]

<RIPRENDI SIMULAZIONE>

[SI DEVE TERMINARE?]

<RIPULIRE STACK>

<ASSEGNARE VALORE DI RITORNO>

<FORZARE INIZIO FALSO>

Se stiamo eseguendo le chiamate

Vogliamo riprendere le chiamate separate

Rivediamo i valori dello stack

Dobbiamo capire dove avevamo interrotto l'esecuzione

} Simulazione RETURN

Dobbiamo proseguire

FATTORIALE ITERATIVO

FATI (m)

CN = m valore corrente di m

CR = 0 // 1

STN = STR = NIL Gli stack per m o 1 sono initializzati vuoti

WHILE (CN >= 0 || STN ≠ NIL) DO

IF (CN >= 0) THEN Prima chiamata (inizio)

CR = 0

IF (CN = 0) Analogico dell' algoritmo ricorsivo

CR = 1

RETURN {
 | RET = CR

Variabile di ritorno

 | CN = -1 Forziamo l'inizio falso

ELSE

 | z = m · FATI (m-1)

 | STN = PUSH (STN, CN)

 | STR = PUSH (STR, CR)

} Sospensione della chiamata corrente

CN = CN - 1

Prepariamo la variabile da passare a FATI (m-1)

ELSE

Risalita

CN = TOP (STN)

CR = TOP (STR)

} Ripresa del contesto

$$CR = CN \cdot RET$$

$$STN = \text{POP}(STN)$$

$$STR = \text{POP}(STR)$$

$$RET = CR$$

$$CN = -1$$

riprendiamo $i = m$. FAUT(m-1)

} Simulazione RETURN

RETURN CR

OTTIMIZZAZIONE DELL' ALGORITMO

E' facile notare che il valore originale di i verrà sempre sovrascritto, quindi lo stack STR è inutile (e di conseguenza i comandi che lo usano).

OSS

Vanno necessariamente solvate le variabili che vengono lette dopo la chiamata ricorsiva (in questo caso m)