PROGRAMMAZIONE

AA 2006-07

LA RICORSIONE

LA RICORSIONE

- Il concetto di ricorsione nasce dalla matematica
- Una funzione matematica è definita ricorsivamente quando nella sua definizione compare un riferimento a se stessa

Esempio: $n!$ Il fattoriale di n

– Definizione iterativa $n! = n*(n-1)*(n-2)*...*2*1$

– Definizione ricorsiva:

$$f(n) = n * f(n-1) \text{ se } n > 0 \quad n! = n*(n-1)!$$

$$f(n) = 1 \quad \text{se } n=0 \quad 0! = 1 \quad (\text{assioma})$$

LA RICORSIONE

- Una definizione ricorsiva è specificata tramite
 - Clausola ricorsiva (relazione di ricorrenza)
 - Assioma
- Principio di induzione
 - Se un predicato P sull'insieme N gode delle seguenti proprietà:
 - $P(0)$ è vero
 - da $P(k)$ vero discende la verità di $P(k+1)$allora P è vero per qualsiasi n appartenente ad N
- Essenziale disporre dell'assioma o passo base
- Alcuni problemi sono intrinsecamente ricorsivi

PROBLEMI RICORSIVI

Esempi

- Somma dei primi numeri naturali
 - $\text{somma}(n) = n + \text{somma}(n-1)$
 - $\text{somma}(n) = 0$ se $n=0$
- Generare l'n-esimo termine della serie di Fibonacci
 - $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$
 - $\text{fib}(n) = 1$ se $n=1$
 - $\text{fib}(n) = 0$ se $n=0$

VARIE PROSPETTIVE

- Definizione ricorsiva
 - Un oggetto è definito in termini di se stesso
 - Esempio: la definizione di grafo di flusso strutturato
 - Esempio: la definizione di identificatore
- Scomposizione ricorsiva di problemi
 - Il problema di partenza è scomposto in sottoproblemi formalmente identici al problema di partenza, ma di ordine man mano inferiore sino a giungere ad un sottoproblema banale di cui si conosce la soluzione
- Algoritmo ricorsivo
 - Un compito è risolto applicando lo stesso algoritmo ad un dominio man mano ridotto rispetto a quello originale
- Sottoprogrammi ricorsivi
 - Unità di programma nel cui corpo invocano se stessi (Ricorsione lineare o non lineare)
 - Ricorsione mutua (A chiama B che, a sua volta, chiama A)

PROGRAMMAZIONE RICORSIVA

- Molti linguaggi (tra cui il C) offrono la possibilità di definire sottoprogrammi ricorsivi.
- Dipende dalla modalità di gestione delle attivazioni
- In C, le chiamate ricorsive vengono gestite come chiamate “normali”

FATTORIALE DI N

```
int fattoriale(unsigned int n)  
{  
int result;  
if (n == 0) result = 1;  
else result = n * fattoriale (n-1);  
/* ricorsione */  
return result;  
}
```

```
int iter_fatt (unsigned int n)  
{  
int i, result=1;  
for (i=2, i<=n; i++)  
    result = result * i;  
return result;  
}
```

SOTTOPROGRAMMI ATTIVATI RICORSIVAMENTE

- Per ogni chiamata della stessa unità viene caricato nello Stack un diverso r.a.
- Prima che sia terminata l'attivazione precedente avviene una nuova attivazione della stessa unità di programma

Esempio

fattoriale (3) innesca una catena di invocazioni sino a fattoriale (0). Nello Stack ci saranno 4 r.a.

fattoriale (0)

fattoriale (1)

fattoriale (2)

fattoriale (3)

SOTTOPROGRAMMI ATTIVATI RICORSIVAMENTE

fattoriale(3)

fattoriale (2)

fattoriale (1)

fattoriale (0) ULTIMA CHIAMATA produce 1

return 1*1

return 2*1

return 3*2 viene passato 6

- L'ultima chiamata corrisponde al **passo base** (condizione di terminazione)
- Il valore di ritorno viene costruito “a ritroso”: man mano che vengono deallocati i r.a. sino all'innesco delle chiamate ricorsive

DISCUSSIONE

- Algoritmi eleganti, sinteticità e semplici
 - Soluzioni ricorsive sono “vicine” alla definizione matematica di molte funzioni
- La implementazione è “pesante”
 - Le versioni iterative sono generalmente più efficienti delle ricorsive
- **Qualunque algoritmo ricorsivo è risolubile anche iterativamente**

RICORSIONE LINEARE

Somma dei primi n numeri naturali

```
int sum (unsigned int n)  
{  
int risult;  
if ( $n == 0$ ) risult = 0;  
    else risult = n + sum (n-1); /* ricorsione */  
return risult;  
}
```

RICORSIONE NON LINEARE

N-esimo numero di Fibonacci

```
int fib (unsigned int n)  
{  
int risult;  
if (n == 0) risult = 0;  
    else if (n == 1) risult = 1;  
else risult = fib (n-1) + fib (n-  
    2);  
return risult;  
}
```

```
int fib (unsigned int n)  
{  
    switch(n) {  
        case 0: return 0; break;  
        case 1: return 1; break;  
        default: return (fib (n-1) + fib (n-2) );  
    }  
}
```

ATT.NE alla STRUTTURAZIONE
(più di un *return*)

Esercizio 1

```
int funz (unsigned int n);
main () {
int x, y;
scanf ("%d", &x);
y = funz (x);
printf (y);
}
int funz (unsigned int n)
{
int result;
if (n == 0) result = 1;
    else result = n * (funz (n-1) + 1);
return result;
}
```

Verificare che l'output prodotto dal programma in corrispondenza di input 4 sia 88

Esercizio 2

Scrivere una funzione C che riceva due parametri interi x ed y e restituisca il valore $f(x, y)$ dove la funzione f è definita come segue:

$$f(x, y) = \begin{cases} 1 & \text{se } y = 0 \\ 3 + f(x, y-1) & \text{se } y > 0 \end{cases}$$

Risolvere l'esercizio proposto

ESERCIZI sulla RICORSIONE

- Stampare al contrario una sequenza di caratteri terminata da “.” letta in ingresso (senza usare il tipo array e stringa) Es. ROMA. AMOR
- Reformuliamo il problema: “Invertire una sequenza di caratteri”
 - E’ una procedura (non una funzione)
 - Una soluzione iterativa ha bisogno di un array di caratteri
 - Vedere Algoritmo iterativo per invertire un array

ESERCIZI sulla RICORSIONE

Se la sequenza contiene una sola lettera **allora**

- Scrivila (il problema è risolto)

altrimenti:

- Rimuovi la prima lettera dalla sequenza
- *Inverti* la sequenza rimanente
- Appendi in coda la lettera rimossa

ESERCIZI sulla RICORSIONE

- Invertire “ROMA”
 - “ROMA” (4 lettere): rimuovi “R”, inverti “OMA”
 - “OMA” (3 lettere): rimuovi “O”, inverti “MA”
 - “MA” (2 lettere): rimuovi “M”, inverti “A”
 - » “A” (1 lettera): è già invertita
 - Appendi “M”: “AM”
 - Appendi “O”: “AMO”
 - Appendi “R”: “AMOR”

ESERCIZI sulla RICORSIONE

- Determinare del massimo/minimo in un array
- Calcolo della lunghezza di una sequenza di elementi
- Generare le permutazioni degli elementi di un insieme
- Ricerca sequenziale e binaria
- Algoritmi di ordinamento (per selezione, per partizionamento, per fusione,...)