Programmazione

Giovanni Da San Martino

Dipartimento of Matematica, Università degli Studi di Padova
giovanni.dasanmartino@unipd.it
A.A. 2021-2022

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Array a Dimensione Fissata

- In C89 la dimensione di un array deve essere una costante intera positiva (no const)
- nota a tempo di compilazione (non una variabile, letta da tastiera o meno)
- Di norma la costante si indica tramite #define
  - #define DIM_ARRAY 3
  - int x[DIM_ARRAY] = {1,2,3} //OK
- Dal C99 è possibile dichiarare array la cui dimensione non è fissata a tempo di compilazione
- Tali array sono implementati internamente in modo diverso dal C. Una delle differenze è che non possono essere inizializzati
  - int n = 3; int x[n] = {1,2,3}; //NO
  - x[0]=1; x[1]=2; x[2]=3; //OK

Previously on Programmazione

Correttezza

- Molti dei problemi su liste che abbiamo risolto con un ciclo fino ad ora presentano la stessa struttura
- c'è una relazione diretta tra l'invariante e la post:
- all' iterazione i-esima del ciclo si è risolto il problema fino all'i-esimo elemento della lista (invariante)
- la post si osserva applicando l'invariante a tutta la lista

Ricorsione

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Funzioni Ricorsive

- Una funzione è detta ricorsiva se chiama se stessa
- Se due funzioni si chiamano l'un l'altra, sono dette mutuamente ricorsive
- La funzione ricorsiva sa risolvere direttamente solo casi particolari di un problema detti casi di base:
- se viene invocata passandole dei dati che costituiscono uno dei casi di base, allora restituisce un risultato
- Se invece viene chiamata passandole dei dati che NON costituiscono uno dei casi di base, allora chiama se stessa (passo ricorsivo) passando dei DATI semplificati/ridotti

Funzioni Ricorsive

- Fatt(0) = 1 (caso base)
- Fatt(n) = n* Fatt(n-1) (caso ricorsivo)

- Fatt(n-1) = (n-1)* Fatt(n-2) quindi
- Fatt(n) = n*(n-1)*...*1

Funzioni Ricorsive

- Fatt(3) = 3*Fatt(2)
- Fatt(2) = 2*Fatt(1)
- Fatt(1) = 1*Fatt(0)
- Fatt(0) = 1

quindi

- Fatt(1) = 1*1
- Fatt(2) = 2*(1*1)
- Fatt(3) = 3*(2*1)

Ricorsione: Quando Utilizzarla

- Spesso la ricorsione permette di risolvere un problema anche molto complesso con poche linee di codice
- La ricorsione è poco efficiente perché richiama molte volte una funzione e questo:
- richiede tempo per la gestione dello stack
- consuma molta memoria (alloca un nuovo stack frame a ogni chiamata, definendo una nuova ulteriore istanza delle variabili locali e dei parametri ogni volta)

Ricorsione: Quando Utilizzarla

- Qualsiasi problema ricorsivo può essere risolto in modo non ricorsivo (ossia iterativo),
- ma la soluzione iterativa potrebbe non essere facile da individuare oppure essere (molto) più complessa
- L'approccio ricorsivo è in genere da preferire se:
- non ci sono particolari problemi di efficienza e/o di memoria
- è più intuitivo di quello iterativo
- la soluzione iterativa non è evidente o agevole

Ricorsione: Esempio


```
int fibo(int n) {
    if (n<=2)
    return 1;
    else
    return fibo(n-1) + fibo(n-2);
}
```

Ricorsione: Quando Non Utilizzarla

- La nostra funzione fibo ripete gli stessi calcoli più volte -> inefficiente