Programmazione I

A.A. 2002-03

Funzioni

(Lezione XXII)

Ricorsione

Prof. Giovanni Gallo
Dr. Gianluca Cincotti
Dipartimento di Matematica e Informatica
Università di Catania
e-mail : {gallo, cincotti}@dmi.unict.it

Definizione

- La **ricorsione** si basa sulla teoria matematica della induzione.

- Una *funzione* si dice ***ricorsiva*** se è definita in termini di se stessa,
  - cioè se nel corpo della funzione sono presenti chiamate alla funzione stessa.

In matematica …

- Le *relazioni di ricorrenza* sono delle funzioni definite in maniera ricorsiva.
- Esempio
  - Fattoriale di un numero intero positivo n
    - n! = n • (n-1) • (n-2) • … • 3 • 2 • 1
    - Per convenzione 0! = 1
  - Relazione di ricorrenza

$$n! = \begin{cases} 1 & \text{se } n = 0, \\ n \bullet (n-1)! & \text{se } n > 0. \end{cases}$$

In informatica …

```
int fattoriale ( int n )
    {
        if ( n == 0 )    return 1;

        return  n * fattoriale ( n – 1 );
    }
```

La sequenza delle chiamate

fattoriale (4)	
4 * fattoriale (3)	**24**
3 * fattoriale (2)	**6**
2 * fattoriale (1)	**2**
1 * fattoriale (0)	**1**
1	**1**

Es. 01

Caratteristiche della ricorsione

- Ogni definizione *ricorsiva* è caratterizzata dal
  - Caso base (*condizione di terminazione*)
    - Fornisce la condizione per cui la funzione *termina*, cioè smette di richiamare se stessa.
      - Se non fosse definito il caso base la funzione richiamerebbe sempre se stessa e si avrebbe un loop infinito.
  - Passo induttivo (*chiamata ricorsiva*)
    - Viene chiamata la stessa funzione relativamente ad un problema più "*piccolo*"; la soluzione ottenuta viene combinata con altra informazione per produrre la soluzione al problema originale.

Esercizio

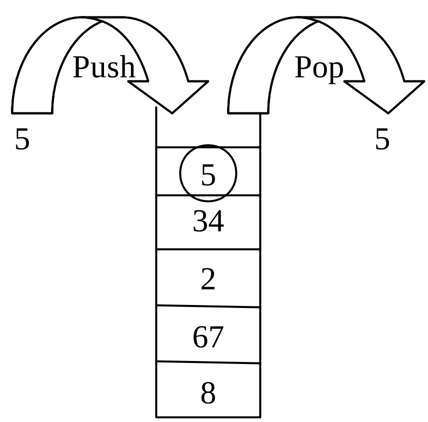
Valutare al calcolatore la seguente relazione di ricorrenza

$$A(n) = \begin{cases} 2\,A(n-1)\,A(n-2) & \textit{se } n \textit{ pari}, \\ 3\,A(n-2) + A(n-1) & \textit{se } n \textit{ dispari}. \end{cases}$$

per ogni $n > 1$, con $A(0) = 1$, $A(1)=3$.

Lo stack (o pila)

- Lo *stack* è una struttura dati (LIFO, Last-In First-Out) in cui le due sole operazioni ammesse per modificare i dati sono:
  - Push (inserimento)
  - Pop (estrazione)



Il meccanismo delle chiamate di funzioni

- Il meccanismo delle chiamate di funzioni fa uso di uno *stack delle chiamate.*
  - Ad ogni chiamata di funzione viene creato un *record di attivazione*, che viene inserito nello stack.
    - In particolare, ogni record di attivazione contiene
      - il *punto di ritorno* alla funzione chiamante,
      - i *parametri formali* della funzione chiamata (in cui sono già stati copiati i parametri attuali),
      - le *variabili locali* alla funzione chiamata,
      - i valori dei *registri* della CPU.

Iterazione vs. Ricorsione

- In generale, le funzioni ricorsive :
  - sono intuitivamente più semplici,
  - sono brevi in termini di linee di codice,
  - possono consumare molta *memoria*,
  - possono richiedere molto *tempo*.

Es. 02 Es. 03

Eliminazione della ricorsione

- **<u>Qualsiasi</u>** funzione *ricorsiva* può essere espressa in forma *non ricorsiva*.
  - Se la ricorsione è di *coda*, cioè la chiamata ricorsiva è l'ultima azione eseguita dalla funzione, allora la ricorsione può essere eliminata con una semplice iterazione.
  - Se la ricorsione è *non di coda*, la ricorsione può essere eliminata solo con l'ausilio di uno stack esterno.
    - In genere, tutte le funzioni con una *doppia chiamata ricorsiva* **<u>non</u>** possono essere riscritte con una banale iterazione.

Eliminazione della ricorsione di "coda"

```
int  fattoriale  ( int  n )
      {
              if  ( n == 0 )     return  1;
              return   n * fattoriale ( n – 1 );
      }
```

```
int  fattoriale  ( int  n )
      {       int   f = 1;
              while (n > 0)     f *= n-- ;
              return   f;
      }
```

- Differenze:
  - Costrutto
  - Terminazione
  - Contatore vs. parametro
  - Loop infinito

Numeri di Fibonacci

➢ I numeri di *Fibonacci* sono definiti dalla seguente relazione di ricorrenza

$$F(n) = F(n\text{-}1) + F(n\text{-}2),$$

per ogni $n>1$, con $F(0) = F(1) = 1$.

- Ecco i primi :

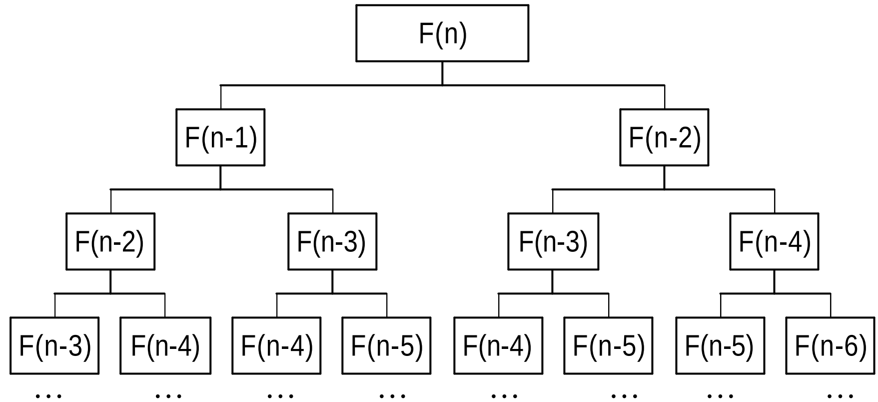
  1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

Una "doppia" ricorsione

```
int  fibonacci  ( int  n )
        {
                if  ( n <= 1 )     return  1;

                return   (  fibonacci ( n – 1 )  +
                               fibonacci ( n – 2 )    );
        }
```

Attenzione alle inefficienze …



La funzione ricorsiva *Fibonacci* provoca un’esplosione combinatorica di chiamate ricorsive “*inutili*”, in quanto molti numeri vengono calcolati più volte.

Esempi

- Segmentazione del righello
- Torre di Hanoi
- Per gli scacchisti
  - Il giro del cavallo
  - Il problema delle otto regine

Es. 04

Es. 05

Fine