Ricorsione

L’uso della ricorsione nella programmazione

Ricorsione

Una funzione è definita in modo ricorsivo se è definita in termini di se stessa.

Nella definizione ricorsiva di una funzione è possibile identificare uno o più casi base e uno o più casi ricorsivi:

- i casi base permettono di calcolare direttamente il valore della funzione, anche se solo nei casi più semplici
- i casi ricorsivi permettono di calcolare la funzione mediante altre valutazioni della funzione

Definizione ricorsiva del fattoriale

La funzione fattoriale `n!`, definita per `n` naturale, è definita in modo "iterativo" come segue:

$$n! = 1 \cdot 2 \cdot \ldots \cdot (n-2) \cdot (n-1) \cdot n$$

La funzione fattoriale può essere definita in modo "ricorsivo" come segue

$$n! = \begin{cases} 1 & \text{se } n==0 \text{ — } \textit{caso base} \\ n \cdot (n-1)! & \text{se } n>0 \text{ — } \textit{caso ricorsivo} \end{cases}$$

La definizione ricorsiva e quella iterativa della funzione fattoriale sono equivalenti.

Ricorsione

- La ricorsione consente di progettare metodi mediante la decomposizione di un problema in problemi dello stesso tipo, ma da risolvere utilizzando dati per cui la soluzione del problema è "più semplice".
  - Ad esempio, il problema del calcolo di 3! viene ricondotto a quello del calcolo di 2!
- La ricorsione puà essere diretta o indiretta

```
void metodoA(…) {
    …
    metodoA(…);
    …
}
```

```
void metodoB(…) {
    … ; metodoC(…); …
}
void metodoC(…) {
    … ; metodoB(…); …
}
```

Il fattoriale

forma iterativa

```
int fattoriale(int n) {
    int nfatt = 1, i; // il fattoriale di n
    for (i=1; i<=n; i++)
        nfatt = nfatt * i;
    return nfatt;
}
```

forma ricorsiva

```
1. // Calcola il fattoriale del numero naturale n

2.  int fattoriale(int n)
3.  {
4.      int nfatt; // il fattoriale di n
5.          if (n==0) // caso base
6.              nfatt = 1;
7.          else // caso ricorsivo
8.              nfatt = n * fattoriale(n-1);
9.      return nfatt;
10. }
```

Esecuzione di funzioni ricorsive: fattoriale(2)

Prima attivazione — `fattoriale(2)`

- **si controlla se n==0**
  - in questo caso n vale 2, e deve essere eseguita la parte else dell'istruzione condizionale
- **per poter assegnare un valore a nfatt, bisogna valutare l'espressione**
  `n*fattoriale(n-1)`
  - n vale 2 — bisogna valutare l'espressione `fattoriale(1)`
- **questa attivazione della funzione viene temporaneamente sospesa e viene invocato nuovamente la funzione `fattoriale(1)`.**

Esecuzione funzioni ricorsive: fattoriale(2)

Seconda attivazione — `fattoriale(1)`

- **si controlla se n==0**
  - in questo caso n vale 1, e deve essere eseguita la parte else dell'istruzione condizionale
- **per poter assegnare un valore a nfatt, bisogna valutare l'espressione `n*fattoriale(n-1)`**
  - n vale 1 — bisogna valutare l'espressione `fattoriale(0)`
- **questa attivazione viene temporaneamente sospesa e viene invocato nuovamente `fattoriale(0)`.**

Esecuzione funzioni ricorsive: fattoriale(2)

Terza attivazione — `fattoriale(0)`

- **si controlla se n==0**
  - in questo caso n vale 0 e deve essere eseguita la parte if dell'istruzione condizionale
- **viene assegnato a nfatt il valore 1**
- **questa attivazione termina restituendo il valore 1**
  - l'esecuzione riprenderà dalla seconda attivazione di `fattoriale`, dal punto in cui era stata sospesa

Esecuzione funzioni ricorsive: fattoriale(2)

Seconda attivazione — `fattoriale(1)`

- **riprende l'esecuzione della seconda attivazione di `fattoriale`, dal punto in cui era stata sospesa**
- **l'espressione `fattoriale(0)` è stata valutata 1, ed è ora possibile eseguire la moltiplicazione con n (che in questa attivazione vale 1) e assegnare il risultato a `nfatt` — che ora vale 1**
- **questa attivazione termina restituendo il valore 1**
  - l'esecuzione riprenderà dalla prima attivazione di `fattoriale`, dal punto in cui era stata sospesa

Esecuzione funzioni ricorsive: fattoriale(2)

Prima attivazione — `fattoriale(2)`

- riprende l'esecuzione della prima attivazione di `fattoriale`, dal punto in cui era stata sospesa
- l'espressione `fattoriale(1)` è stata valutata `1`, ed è ora possibile eseguire la moltiplicazione con `n` (che in questa attivazione vale `2`) e assegnare il risultato a `nfatt` — che ora vale `2`
- questa attivazione termina restituendo il valore `2`

– **Si arriva quindi alla conclusione che `fattoriale(2)` vale `2`**

Definizione ricorsiva della funzione `cifre`

Si consideri la funzione `cifre(n)`, definita per `n` naturale, con il seguente significato:

`cifre(n)` è il numero di cifre decimali nella rappresentazione decimale del numero naturale `n`.

Esempio: `cifre(765) = 3, cifre(18) = 2, cifre(0) = 1`

La funzione `cifre(n)`, per `n` naturale, può essere definita in modo ricorsivo come segue:

$$\texttt{cifre(n)} = \begin{cases} 1 & \text{se } n<10 \\ 1 + \texttt{cifre(n/10)} & \text{se } n>=10 \end{cases}$$

Ad esempio:

```
cifre(765) = 1+cifre(76) = 1+(1+cifre(7)) = 1+(1+(1)) = 3
```

Il metodo `cifre`

```
1.  /* Calcola il numero di cifre nella rappresentazione
2.   * decimale di n.
3.  */
4.  int cifre(int n) {
5.     if (n<10) // caso base
6.        return 1;
7.     else // caso ricorsivo
8.        return 1 + cifre(n/10);
9.  }

10. // provate a generalizzare ad una base qualunque
```

Si osservi come, anche in questo caso, i casi ricorsivi invocano la funzione usando come argomento un valore "più piccolo" di quello con cui è stato invocato. I dati su cui **cifre** viene invocato diventano sempre più piccoli finche si riducono ad un valore minore di 10. A questo punto la ricorsione ha termine.

Definizione ricorsiva della funzione `mcd`

Il massimo comun divisore di due numeri interi positivi può essere calcolato sulla base della seguente definizione ricorsiva (nota anche come algoritmo di Euclide):

- **la funzione massimo comun divisore mcd(n,m) — con n e m numeri interi positivi — può essere definita come segue:**

$$\text{mcd}(n, m) = \begin{cases} n \text{ (oppure } m) & \text{se } n==m \text{ — } \textit{caso base} \\ \textbf{mcd(n–m, m)} & \textbf{se n>m — } \textbf{\textit{caso ricorsivo}} \\ \text{mdc}(n, m–n) & \text{se } n<m \text{ — } \textit{caso ricorsivo} \end{cases}$$

ad esempio:

- **mcd(16, 12) = mcd(4,12) = mcd(4,8) = mcd(4,4) = 4**

notare la presenza di un caso base e di due casi ricorsivi

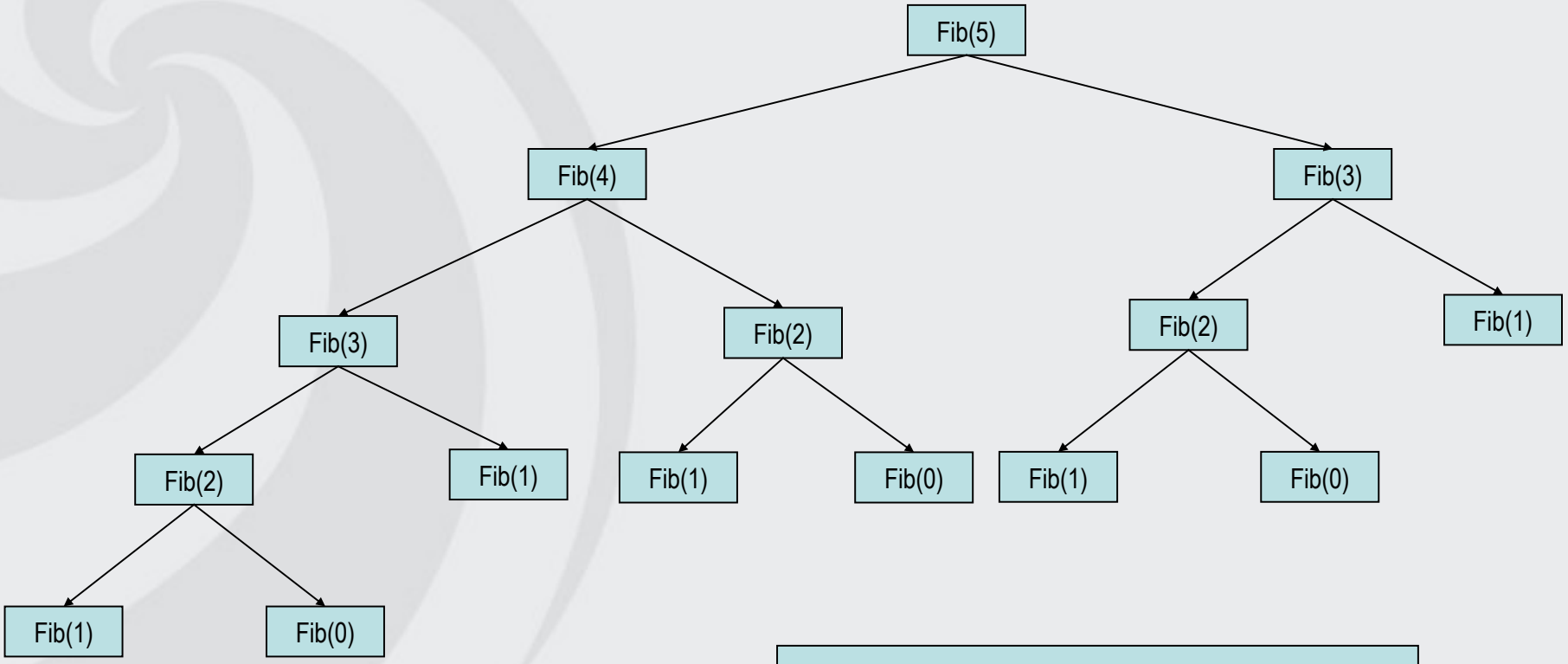
Il metodo mcd

```
1.  /* Calcola il massimo comun divisore di n e m usando
    l'algoritmo di Euclide */

2.  int mcd(int n, int m) {
3.     // pre-condizione: n>0 && m>0
4.     if (n==m) {
5.        // caso base
6.        return n;
7.     } else if (n>m) {
8.        // caso ricorsivo
9.        return mcd(n-m, m);
10.    } else {
11.       // altro caso ricorsivo, n<m
12.       return mcd(n, m-n);
13.    }
14. }
```

Il metodo mcd

```
1.  /* Variante dell'algoritmo di Euclide */

2.  int mcd(int n, int m) {
3.     // pre-condizione: n>0 && m>0
4.     if (n==m) {
5.        // caso base
6.        return n;
7.     } else if (n>m) {
8.        // caso ricorsivo
9.        return mcd(n % m, m);
10.    } else {
11.       // altro caso ricorsivo, n<m
12.       return mcd(n, m % n);
13.    }
14. }
```

Fibonacci ricorsivo



Fib(0)=0, Fib(1)=1
N>1 Fib(N)=Fib(N-1)+Fib(N-2)

Errori comuni

Nella scrittura di funzioni ricorsive è possibile commettere alcuni errori

errori nella gestione dei casi base e ricorsivi

- sbagliare la struttura dell'istruzione condizionale che controlla la ricorsione

errori nella definizione del caso base

- calcolare valori non corretti nel caso base

errori nella definizione del caso ricorsivo

- non procedere verso un caso base (il metodo non termina correttamente, a causa di una ricorsione infinita)

Modello di esecuzione dei metodi

Il modello di esecuzione delle funzioni è basata sul meccanismo della pila (o stack – memoria gestita secondo la logica LIFO) di attivazione

ogni volta che viene invocato una funzione, viene creato un nuovo record di attivazione, associato all'attivazione della funzione, e inserito nella pila di attivazione

il record di attivazione per una funzione contiene tutte le informazioni necessarie all'esecuzione di una particolare attivazione:

- **informazioni relative all'attivazione della funzione utilizzate per:**
  - effettuare correttamente il rientro dalle funzioni richiamate
- **dati della funzione:**
  - i parametri della funzione
  - le variabili locali

Modello di esecuzione di metodi ricorsivi

Il modello di esecuzione basato sulla pila di attivazione permette l'esecuzione di metodi ricorsivi

- questo modello permette infatti l'esecuzione di programmi in cui sono necessarie molteplici *attivazioni contemporanee* di uno stesso metodo
- una invocazione ricorsiva di un metodo viene vista semplicemente come ogni altra invocazione di metodo

```
1.  Public class Fattoriale {
2.     static public void main(String[] args) {
3.        int x=2;
4.        int nfat = fattoriale(x); //1
5.        System.out.println(nfat);
6.     }
7.     public static int fattoriale(int n){
8.        if(n>1)
9.           return n * fattoriale(n-1); //2
10.       else
11.          return 1;
12.    }
13. }
```

```
        main
        args
   x            2
   nfat         ?
   risultato    ?
   ritorno     //1

     fattoriale
n            2
risultato    ?
ritorno     //2

     fattoriale
n            1
risultato    ?
ritorno      ?
```