Impiego di metodi e loro invocazione

Eugenio G. Omodeo

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Trieste, 06/10/2014

Il progr. che stiamo per vedere, 'in azione'

Il modo *piú diretto* di lanciare un processo è di chiederne l'avvio all'*interprete di comandi* del sistema operativo.

Ad es., con Mac OS X:

Il progr. che stiamo per vedere, 'in azione'

Il modo *piú diretto* di lanciare un processo è di chiederne l'avvio all'*interprete di comandi* del sistema operativo.

Ad es., con Mac OS X:

```
Last login: Sun Oct 13 10:19:04 on console
Eugenios-MacBook-Air:~ eomodeo$ cd Desktop/Eugenio/JAVA/131015
Eugenios-MacBook-Air:131015 eomodeo$ javac ConvertiNumeroInNumerale.java
Eugenios-MacBook-Air:131015 eomodeo$ java ConvertiNumeroInNumerale 2 10 32 371
1010
100000
101110011
Eugenios-MacBook-Air:131015 eomodeo$
```

Il progr. che stiamo per vedere, 'in azione'

Il modo *piú diretto* di lanciare un processo è di chiederne l'avvio all'*interprete di comandi* del sistema operativo.

Ad es., con Mac OS X:

```
Last login: Sun Oct 13 10:19:04 on console
Eugenios-MacBook-Air:~ eomodeo$ cd Desktop/Eugenio/JAVA/131015
Eugenios-MacBook-Air:131015 eomodeo$ javac ConvertiNumeroInNumerale.java
Eugenios-MacBook-Air:131015 eomodeo$ java ConvertiNumeroInNumerale 2 10 32 371
1010
100000
101110011
Eugenios-MacBook-Air:131015 eomodeo$
```

(Ma vi sono interazioni fra programma e utente ben piú pratiche !)

Programma 'monolitico' (Manca l'essenziale!)

```
class ConvertiNumeroInNumerale {

    public static void main( String[] aaa ) {

    int numero; // verra` trasformato

    int base;   // presiedera` alla trasformazione

    String numerale; // accumuleremo qui il risultato

    base = Integer.parseInt( aaa[0] );

    String cifre = "0123456789abcdefghijklmnopqrstuvwxyz"; // tabella di conversione

    assert base > 1 && base <= cifre.length();

    for (int i = 1; i < aaa.length; i = i + 1) {

        numero = Integer.parseInt( aaa[i] ); // qui ha inizio la conversione

                :       :       :       :       :       :       :

        System.out.println( numerale );

        }
    }
}
```

Per completezza: Il nucleo del convertitore

```
numerale = "";

do {
     numerale = cifre.charAt( numero % base ) + numerale;

     numero = numero / base;
   }

while ( numero != 0);
```

(Da sostituire ai puntini rossi del lucido precedente)

Riorganizzare il programma esposto nei due lucidi precedenti come un

main che invoca un metodo.

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

1. (a+b)
2. ()
3. (a+b)*(3-7)
4. (a+(bc)())
5. (a+(bc)()
6. (a+(bc)())(d
7. a+(bc)()
8.)(a+b)(

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

1. (a+b) — Sí
2. ()
3. (a+b)*(3-7)
4. (a+(bc)())
5. (a+(bc)()
6. (a+(bc)())(d
7. a+(bc)()
8.)(a+b)(

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

1. (a+b)
2. ()
3. (a+b)*(3-7)
4. (a+(bc)())
5. (a+(bc)()
6. (a+(bc)())(d
7. a+(bc)()
8.)(a+b)(

Sí

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

1. (a+b)
2. ()
3. (a+b)*(3-7) — Sí
4. (a+(bc)())
5. (a+(bc)()
6. (a+(bc)())(d
7. a+(bc)()
8.)(a+b)(

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

1. (a+b)
2. ()
3. (a+b)*(3-7)
4. (a+(bc)()) — Sí
5. (a+(bc)()
6. (a+(bc)())(d
7. a+(bc)()
8.)(a+b)(

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

1. (a+b)
2. ()
3. (a+b)*(3-7)
4. (a+(bc)())
5. (a+(bc)() — No
6. (a+(bc)())(d
7. a+(bc)()
8.)(a+b)(

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

1. (a+b)
2. ()
3. (a+b)*(3-7)
4. (a+(bc)())
5. (a+(bc)()
6. (a+(bc)())(d
7. a+(bc)()
8.)(a+b)(

No

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

1. (a+b)
2. ()
3. (a+b)*(3-7)
4. (a+(bc)())
5. (a+(bc)()
6. (a+(bc)())(d
7. a+(bc)()
8.)(a+b)(

Sí

Problema: Riconoscere il corretto *bilanciamento* di parentesi.

Semplificazione: Solo parentesi tonde.

Per fissare le idee: Quali delle seguenti sono ben bilanciate?

1. (a+b)
2. ()
3. (a+b)*(3-7)
4. (a+(bc)())
5. (a+(bc)()
6. (a+(bc)())(d
7. a+(bc)()
8.)(a+b)(

No

Modo concreto di stabilire il bilanciamento...

```
class Bilan0{ // parentesi tonde bilanciate

    public static void main( String[] aaa ){

        String daTestare = aaa[ 0 ]; // seq. di caratteri da esaminare

        int contAperte = 0; // divario fra quante parentesi sono state
                            // aperte e quante ne sono state chiuse:
                            // in caso di corretto bilanciamento
                            // non diverra` mai negativo

        for( int i = 0; contAperte >= 0 && i < daTestare.length( ); i++ )

            if ( daTestare.charAt( i ) == '(' ) contAperte += 1; else

            if ( daTestare.charAt( i ) == ')' ) contAperte -= 1;

        System.out.print( ( contAperte == 0 ) ? "" : 'S'  );

        System.out.println("bilanciata");
    }
}
```

Implem. modulare del test di bilanciamento

```
class Bilan1{ // verifica sul corretto bilanciamento di parentesi

    public static void main( String[] aaa ){

        System.out.print( ( bilanciata( aaa[ 0 ] ) ) ? "" : 'S'  );

        System.out.println("bilanciata");
    }

    public static boolean bilanciata( String daTestare ){

        int contAperte = 0; // divario fra parentesi aperte e chiuse:
                            // in caso di corretto bilanciamento
                            // non diverra` mai negativo

        for( int i = 0; contAperte >= 0 && i < daTestare.length( ); i++ )

            if ( daTestare.charAt( i ) == '(' ) contAperte += 1; else

            if ( daTestare.charAt( i ) == ')' ) contAperte -= 1;

        return contAperte == 0;
    }
}
```

SALTARE I PROSSIMI 4 LUCIDI[1]

(Solo momentaneamente, in questa fase del corso;
ma piú avanti tornateci su. . .)

[1] 'SLIDE'

Come spiegare a parole il bilanciamento ?

Specifica del problema

Diremo che una sequenza D di caratteri è **bilanciata**
(rispetto alle parentesi tonde) se:

1. in D non figurano parentesi tonde (né aperte né chiuse);
   oppure
2. la prima parentesi tonda che vi figura e' una '(',
   1. tale '(' è seguita una seq **bilanciata**, a sua volta seguita da ')',
   2. tale ')' è a sua volta seguita da una seq. **bilanciata**
      che si protrae sino alla fine della D.

Osservazione: tanto 2.1 che 2.2 trattano di seq. bilanciate. . .

. . . *quanto piú lunghe possibile.*

È una specifica o un circolo vizioso ?

Implem. ricorsiva della specifica vista ora

```
// Il metodo che segue determina la sotto-sequenza della stringa D sotto esame:
//    ** che inizia dalla posizione indicata,
//    ** che e` bilanciata,
//    ** che si protrae (per posiz. consecutive) quanto piu` possibile.
// Ci da` la posizione che viene subito dopo tale sotto-sequenza.
public static int sbilanciam( String sottoEsame, int posiz ){

    assert sottoEsame.length() >= posiz; // se non e` cosi`, crash!

    if ( posiz == sottoEsame.length() ||

         sottoEsame.charAt( posiz ) == ')' ) return posiz; // caso base

    int riemers = sbilanciam( sottoEsame, posiz + 1 );
 //  System.out.println("posiz = "+ posiz + "; riemers = "+riemers);

    if ( sottoEsame.charAt( posiz ) == '(' )

        return ( riemers < sottoEsame.length() &&

                    sottoEsame.charAt( riemers ) == ')' ) ?

                    sbilanciam( sottoEsame, riemers + 1 ) : posiz ;
    return riemers;
 }
```

Spiegazione del metodo RICORSIVO visto ora

Della stringa `sottoEsame` il metodo esamina il segmento che inizia in posizione `posiz`.

Deve individuare in quale posizione $\geqslant$ `posiz` il segmento smette di essere bilanciato. Alla peggio (se il segmento vuoto è *il solo* bilanciato), restituirà `posiz` stessa come risposta. Ciò accade nelle due situazioni rilevabili all'inizio, ma potrà anche essere causato dal fatto che `sottoEsame.charAt(posiz)` è una '(' scompagnata.

I casi favorevoli sono due:

1. `sottoEsame.charAt(posiz)` *non è* una parentesi:
   basta procedere ricorsivamente dalla posizione `posiz` $+ 1$;
2. `sottoEsame.charAt(posiz)` *è* una '(', seguita da una porzione bilanciata, seguita da una ')':
   si procederà ricorsivamente dalla posizione a valle di tale ')'

Come sfruttare il metodo RICORSIVO visto ora

Direttamente da un main:

```
class Bilan2{ // parentesi tonde bilanciate

    public static void main( String[] aaa ){

        int sbil =  sbilanciam( aaa[ 0 ], 0 );

        System.out.print("La posizione di sbilanciamento e` la ");

        System.out.println( sbil );

        System.out.print( ( sbil == aaa[ 0 ].length( ) ) ? "" : 'S'  );

        System.out.println("bilanciata");
    }
```

'Mascherandolo' (ma allora perché non renderlo privato ?):

```
    public static boolean bilanciata( String daTestare ){

        return sbilanciam( daTestare, 0 ) == daTestare.length( );
    }

    private static int sbilanciam( String sottoEsame, int posiz ){
```

Esercizio: Isolare un metodo lancia-moneta

```
class GettaMoneta{ // Esercizio. Di che tipo e`:
                   // *** il risultato di System.out.println ?
                   // *** il risultato di Math.random ?
                   //
                   // Cosa verra` stampato ?

   public static void main( String[] aaa ){

      for( int i = 0; i < 10; i++)

         System.out.println( i + ".  " +

                             ((Math.random() >= 0.5) ? "Testa" : "Croce")

         );
   }
}
```

Esercizio: Si implementi un contatore ad `lu` bit

Ad es., per $\texttt{lu} = 3$ (operando da sn a dx) per semplicità, il contatore deve generare la successione:

0. 000
1. 100
2. 010
3. 110
4. 001
5. 101
6. 011
7. 111

Cosa accade di preciso quando, durante l'esecuzione di un programma Java, un metodo *A* ne invoca un altro, *B* ?

* * * * * *

Cosa accade di preciso quando, durante l'esecuzione di un programma Java, un metodo A ne invoca un altro, B ?

* * * * * *

Prima ancora: *Dove figura* l'invocazione ?

R.: Dove figura un'invocazione di metodo –I

Forma dell'invocazione:

$B(\ \text{espr}_1\ ,\ldots,\ \text{espr}_n\ )$ con $n \geqslant 0$ (Parentesi cmq obbligatorie)

Le espr_i sono chiamate parametri *attuali*

R.: Dove figura un'invocazione di metodo –I

Forma dell'invocazione:

$B(\ \text{espr}_1\ ,\ldots,\ \text{espr}_n\ )$ con $n \geqslant 0$ (Parentesi cmq obbligatorie)

Le espr_i sono chiamate parametri *attuali*

R.: Dove figura un'invocazione di metodo –I

Forma dell'invocazione:

$B(\ \text{espr}_1\ ,\ldots,\ \text{espr}_n\ )$ con $n \geqslant 0$ (Parentesi cmq obbligatorie)

Le espr_i sono chiamate parametri *attuali*

Una tale invocazione può:

1. costituire essa stessa, da sola, un'istruzione;

R.: Dove figura un'invocazione di metodo –I

Forma dell'invocazione:

$B(\ \text{espr}_1\ ,\ldots,\ \text{espr}_n\ )$ con $n \geqslant 0$ (Parentesi cmq obbligatorie)

Le espr_i sono chiamate parametri *attuali*

Una tale invocazione può:

1. costituire essa stessa, da sola, un'istruzione;
2. figurare come espressione, ad es.:

R.: Dove figura un'invocazione di metodo –I

Forma dell'invocazione:

$B(\ \text{espr}_1\ ,\ldots,\ \text{espr}_n\ )$ con $n \geqslant 0$ (Parentesi cmq obbligatorie)

Le espr_i sono chiamate parametri *attuali*

Una tale invocazione può:

1. costituire essa stessa, da sola, un'istruzione;
2. figurare come espressione, ad es.:
   - comparire a membro destro di un'assegnazione ,
   - entrare come sotto-espressione in un'espressione piú grande ,
   - essere parametro attuale in un'altra invocazione .

R.: Dove figura un'invocazione di metodo –II

1. La prima circostanza ha luogo quando, nella firma del metodo B, il risultato è dichiarato **void**.

R.: Dove figura un'invocazione di metodo –II

1. La prima circostanza ha luogo quando, nella firma del metodo B, il risultato è dichiarato **void**.
2. Nella seconda circostanza, B dovrà "restituire" al metodo invocante un risultato del tipo atteso: potrà trattarsi di un tipo primitivo, o anche del tipo di una classe di oggetti.

R.: Dove figura un'invocazione di metodo –II

1. La prima circostanza ha luogo quando, nella firma del metodo B, il risultato è dichiarato **void**.
2. Nella seconda circostanza, B dovrà "restituire" al metodo invocante un risultato del tipo atteso: potrà trattarsi di un tipo primitivo, o anche del tipo di una classe di oggetti.

Esempi di tipi rappresentati tramite classi:

- `String`
- `int[]`
- `String[]`

Da cosa viene erogato, un metodo ?

Da cosa viene erogato, un metodo ?

R: Dalla classe cui il metodo B appartiene, se B è *statico*; altrimenti, da un oggetto della classe di quel metodo — tale oggetto viene detto *parametro implicito* dell'invocazione.

Da cosa viene erogato, un metodo ?

R: Dalla classe cui il metodo B appartiene, se B è *statico*; altrimenti, da un oggetto della classe di quel metodo — tale oggetto viene detto *parametro implicito* dell'invocazione.

Il compilatore stesso avrà controllato che B sia 'visibile' ad A

Da cosa viene erogato, un metodo ?

R: Dalla classe cui il metodo B appartiene, se B è *statico*; altrimenti, da un oggetto della classe di quel metodo — tale oggetto viene detto *parametro implicito* dell'invocazione.

Il compilatore stesso avrà controllato che B sia 'visibile' ad A

Esempi

1. `Math.random()` viene erogato dalla classe `Math`

Da cosa viene erogato, un metodo ?

R: Dalla classe cui il metodo B appartiene, se B è *statico*; altrimenti, da un oggetto della classe di quel metodo — tale oggetto viene detto *parametro implicito* dell'invocazione.

Il compilatore stesso avrà controllato che B sia 'visibile' ad A

Esempi

1. `Math.random()` viene erogato dalla **classe** `Math`
2. `System.out.println( n )` viene erogato da `System.out`, che è un **oggetto** di classe `PrintStream`

Da cosa viene erogato, un metodo ?

R: Dalla classe cui il metodo B appartiene, se B è *statico*; altrimenti, da un oggetto della classe di quel metodo — tale oggetto viene detto *parametro implicito* dell'invocazione.

Il compilatore stesso avrà controllato che B sia 'visibile' ad A

Esempi

1. `Math.random()` viene erogato dalla **classe** `Math`
2. `System.out.println( n )` viene erogato da `System.out`, che è un **oggetto** di classe `PrintStream`
3. `tastiera.nextInt( n1 )` viene erogato da `tastiera`, che è un **oggetto** di classe `Scanner`

Oggetti e classi

- Gli oggetti sono entità di un programma che si possono manipolare invocando metodi.
- Tali oggetti appartengono a diverse classi. Per esempio l'oggetto *System.out* appartiene alla classe *PrintStream*.

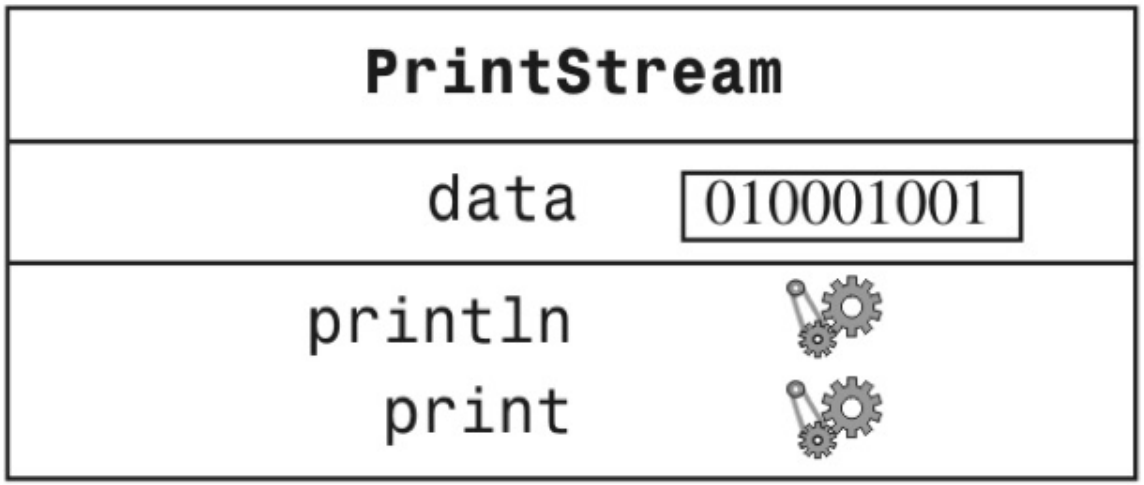

Figura 3: Rappresentazione dell'oggetto `System_out`

Cay S. Horstmann Concetti di informatica e fondamenti di Java, 4 ed. Apogeo ©2007

Rappresentazione di due oggetti di tipo String

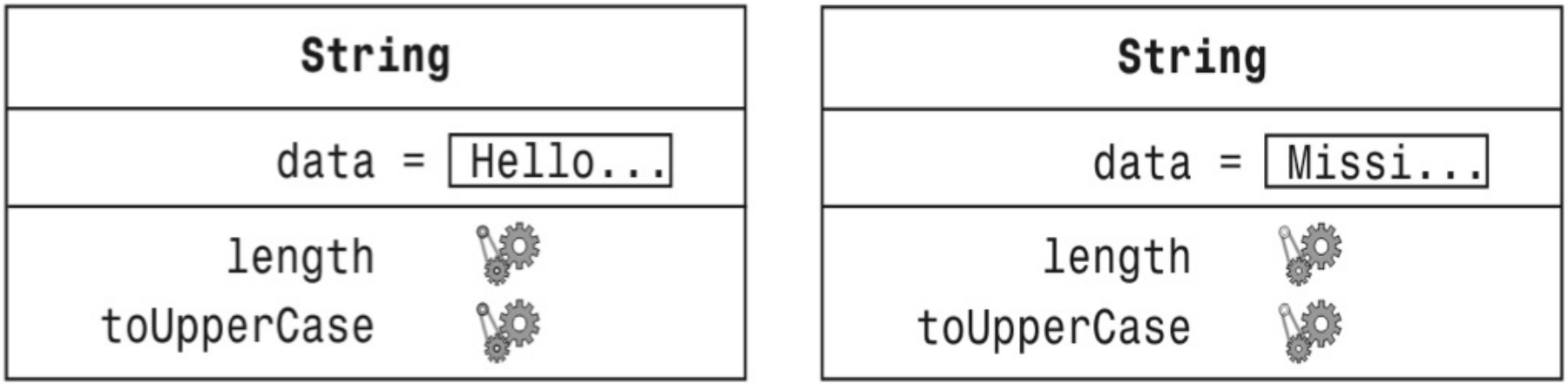

Figura 4
Rappresentazione di due oggetti di tipo `String`

Cay S. Horstmann Concetti di informatica e fondamenti di Java, 4 ed. Apogeo ©2007

Cosa avviene all'attivazione di B ?

A grandi linee:

Salvataggio della situazione corrente:

Ambientazione:

Passaggio di parametri:

Avvio di B

Cosa avviene all'attivazione di B ?

A grandi linee:

Salvataggio della situazione corrente: perché, al termine dell'invocazione, l'esecuzione di A possa riprendere là dove ora viene sospesa

Ambientazione:

Passaggio di parametri:

Avvio di B

Cosa avviene all'attivazione di B ?

A grandi linee:

Salvataggio della situazione corrente:

Ambientazione: sta iniziando il *ciclo di vita* delle variabili di B; dunque occorre spazio in memoria

Passaggio di parametri:

Avvio di B

— spazio piú o meno grande, per ciascuna variabile, a seconda del suo tipo

Cosa avviene all'attivazione di B ?

A grandi linee:

Salvataggio della situazione corrente:

Ambientazione: sta iniziando il *ciclo di vita* delle variabili di B; dunque occorre spazio in memoria

Passaggio di parametri:

Avvio di B

Se B verrà invocata piú volte, ogni volta alle sue variabili (inclusi i *parametri formali*) verrà riservato spazio

E nel caso ricorsivo, anche piú spazi omologhi in contemporanea...

Cosa avviene all'attivazione di B ?

A grandi linee:

Salvataggio della situazione corrente:

Ambientazione:

Passaggio di parametri: i parametri attuali vanno copiati in quelli formali

Avvio di B

I parametri attuali espr_i che compaiono nell'invocazione (scritta sopra), piú un'eventuale altro espr_0 (nel caso di parametro implicito), devono corrispondere per numero ai parametri formali presenti nella firma (detta anche "segnatura") di B. In base alla loro posizione, vengono *copiati* nei corrispondenti parametri formali $\nu_1, \ldots, \nu_n$, quasi stessero avvenendo le assegnazioni $\nu_i = \text{espr}_i$ (una per ogni valore dell'indice i, dal piú piccolo fino ad n).

Cosa avviene all'attivazione di B ?

A grandi linee:

Salvataggio della situazione corrente:

Ambientazione:

Passaggio di parametri: i parametri attuali vanno copiati in quelli formali

Avvio di B

Il compilatore stesso avrà controllato la rispondenza di tipo fra ciascun parametro attuale e il corrispondente param. formale e predisposto le "forzature di tipo" eventualmente necessarie.

Cosa avviene all'attivazione di B ?

A grandi linee:

Salvataggio della situazione corrente:

Ambientazione:

Passaggio di parametri:

Avvio di B (dalla sua prima istruzione)

Cosa avviene al termine di B (purché termini) ?

Deallocazione:

Recupero:

Restituzione:

Riavvio di A

Cosa avviene al termine di B (purché termini) ?

Deallocazione: dello spazio di memoria che era stato assegnato come suo ambiente a B.

Recupero:

Restituzione:

Riavvio di A

Cosa avviene al termine di B (purché termini) ?

Deallocazione:

Recupero: (e rimozione dalla pila) di quelle informazioni che ora vengono sfruttate per ripristinare l'esecuzione di A.

Restituzione:

Riavvio di A

Cosa avviene al termine di B (purché termini) ?

Deallocazione:

Recupero:

Restituzione: del risultato di B ad A.

Riavvio di A

Cosa avviene al termine di B (purché termini) ?

Deallocazione:

Recupero:

Restituzione:

Riavvio di A (dal punto dov'era stato sospeso)