Titolo unità didattica: Algoritmi ricorsivi [14]

Titolo modulo : tecniche di programmazione ricorsiva [01-T]

Iterazione vs. ricorsione

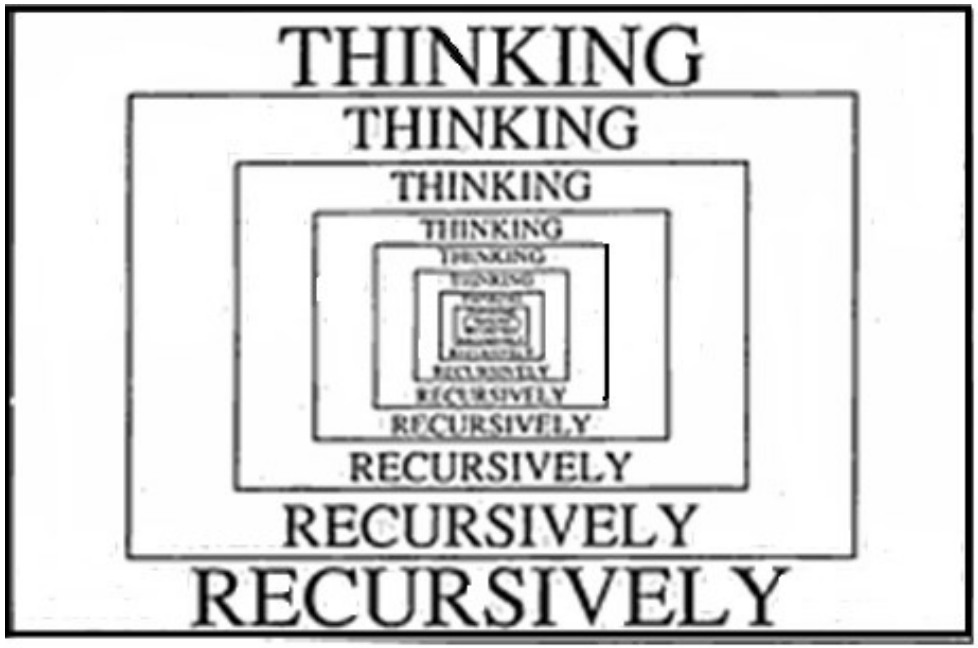
Argomenti trattati:

- ✓ l'idea di ricorsività
- ✓ autoattivazione di function
- ✓ formule ricorrenti e algoritmi ricorsivi
- ✓ struttura di un algoritmo ricorsivo
- ✓ algoritmo ricorsivo per la somma dei primi n numeri naturali
- ✓ algoritmo ricorsivo per il fattoriale

Prerequisiti richiesti: P1-05-01-T, P1-06-02-T, P1-12-01-T

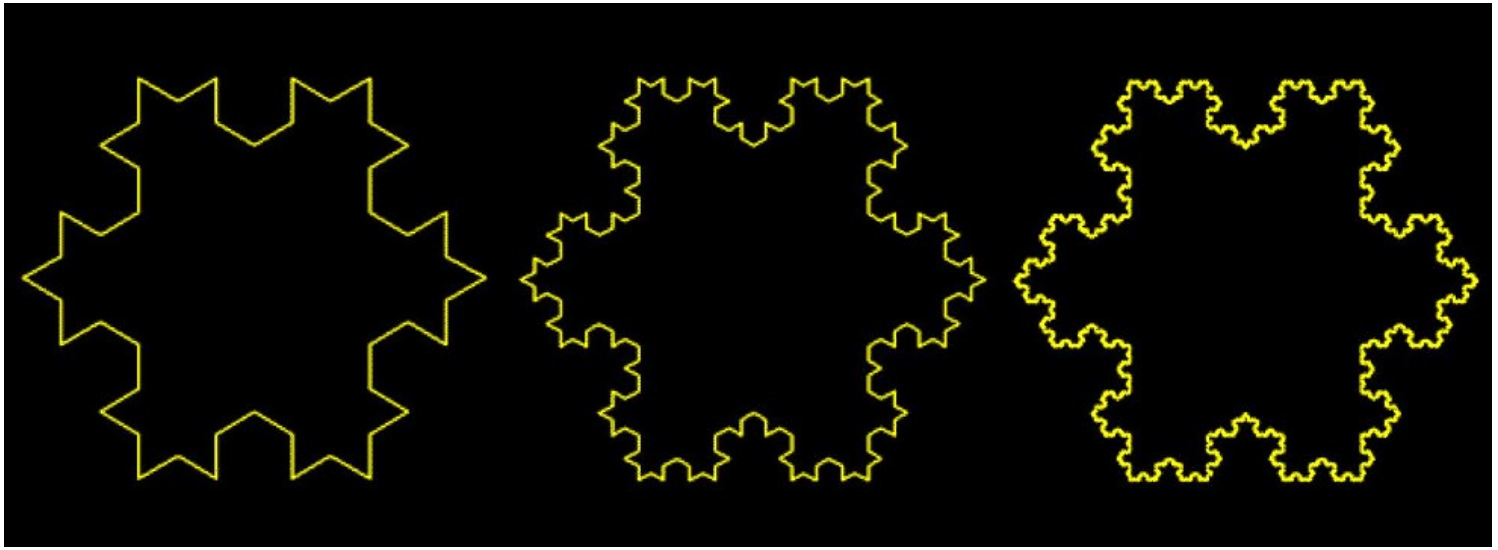
tecnica di **programmazione ricorsiva** (**algoritmi ricorsivi**)

un modo per denotare
la **ripetizione** **di una azione**
(**alternativo** alla descrizione **iterativa**)

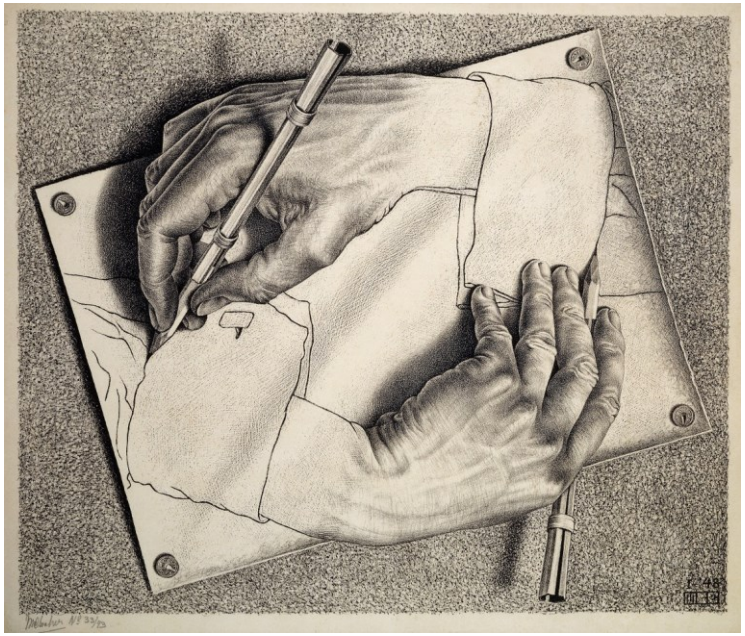


immagini ricorsive

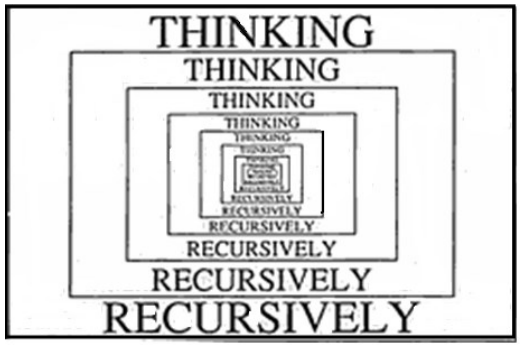
immagini ricorsive

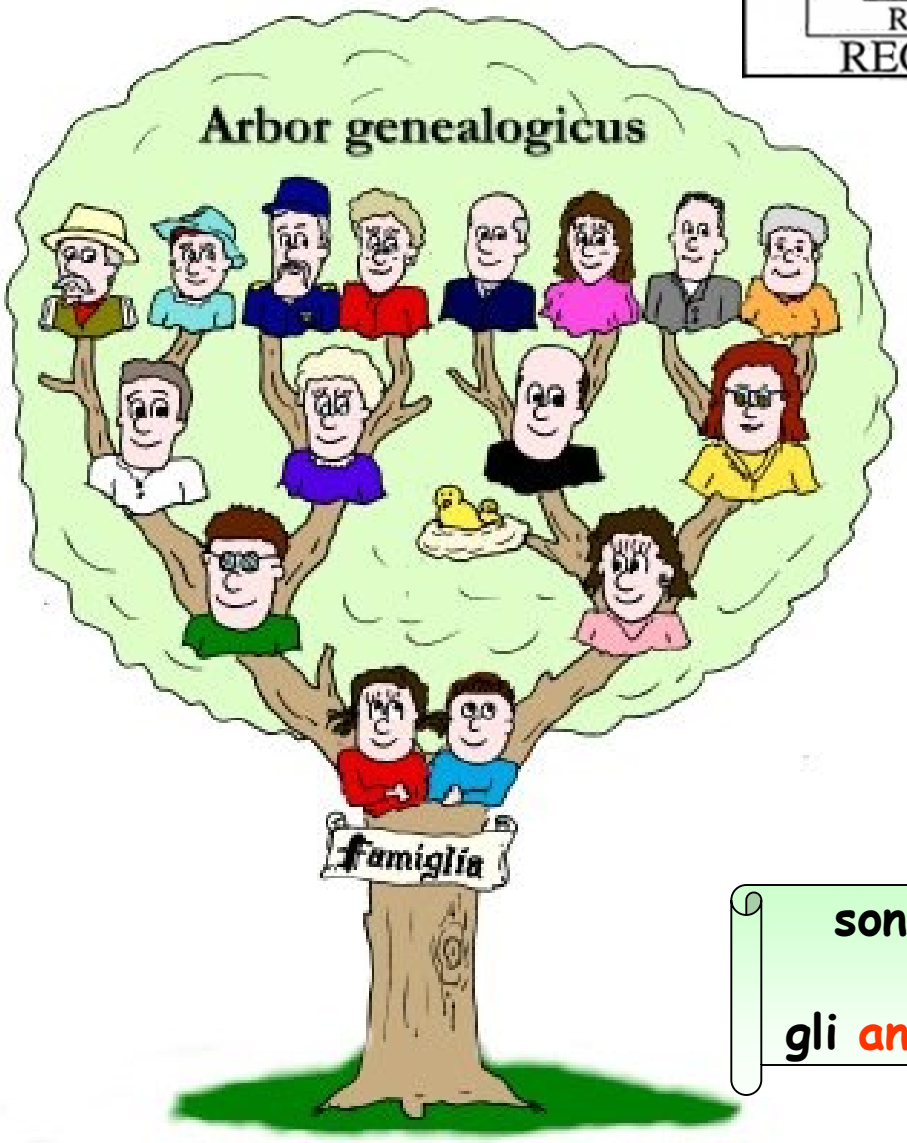


immagini ricorsive

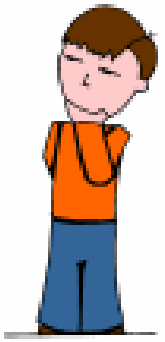


definizioni ricorsive



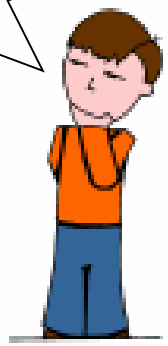


chi sono gli antenati di Isotta e Nico?



sono i loro genitori
+
gli antenati dei genitori

chi sono gli antenati dei loro genitori?



sono i loro genitori
+
gli antenati dei genitori

definizioni ricorsive

una persona è di famiglia reale se è un re
oppure se è figlio di una persona di famiglia reale

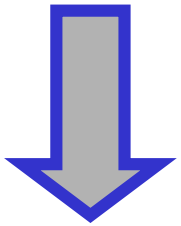
una persona è cittadino italiano se è nata in Italia
oppure se ha acquisito la cittadinanza italiana
oppure se è figlio di una persona che è cittadino italiano

caso base (caso banale)

definizioni ricorsive

ogni definizione fa riferimento a se stessa

si possono ripetere all'infinito ?



sequenza finita di autoriferimenti

caso base (caso banale)

definizioni ricorsive

una espressione aritmetica è:

1. un numero, oppure
2. una variabile, oppure
3. una espressione aritmetica racchiusa tra parentesi tonde, oppure
4. due espressioni aritmetiche separate da un operatore aritmetico (+,-,*,/)

caso base

`7*(6+2)`

è una espressione aritmetica?

definizioni ricorsive

una espressione aritmetica è:

1. un numero, oppure
2. una variabile, oppure
3. una espressione aritmetica racchiusa tra parentesi tonde, oppure
4. due espressioni aritmetiche separate da un operatore aritmetico (+,-,*,/)

caso base

7*(6+2)

è una espressione aritmetica?

4

7 — espressione aritmetica (caso base 1)

***** — operatore aritmetico

(6+2) — espressione aritmetica ?

definizioni ricorsive

una espressione aritmetica è:

1. un numero, oppure
2. una variabile, oppure
3. una espressione aritmetica racchiusa tra parentesi tonde, oppure
4. due espressioni aritmetiche separate da un operatore aritmetico (+,-,*,/)

caso base

`7*(6+2)`

è una espressione aritmetica?

3 `(6+2)`

`6+2`

espressione aritmetica ?

definizioni ricorsive

una espressione aritmetica è:

1. un numero, oppure
2. una variabile, oppure

← **caso base**

3. una espressione aritmetica racchiusa tra parentesi tonde, oppure
4. due espressioni aritmetiche separate da un operatore aritmetico (+,-,*,/)

7*(6+2)

è una espressione aritmetica?

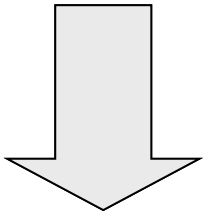
si

4

- **6** — espressione aritmetica (caso base 1)
- **+** — operatore aritmetico
- **2** — espressione aritmetica (caso base 1)

tecnica di **programmazione ricorsiva**
(**algoritmi ricorsivi**)

si realizza mediante una **function**
(o una procedura) che nel
proprio corpo contiene
una **chiamata** alla **function stessa**
(o alla procedura stessa)



autoattivazione di una function

tecnica di **programmazione ricorsiva** (**algoritmi ricorsivi**)

può essere utilizzata in alternativa alla **programmazione iterativa** (**algoritmi iterativi**) che descrive la ripetizione attraverso i **costrutti di ripetizione** (**for**, **while**, **do-while**)

può essere utilizzata per descrivere

- ✓ sia **algoritmi incrementali**
- ✓ sia **algoritmi divide et impera**

tecnica di **programmazione ricorsiva** (**algoritmi ricorsivi**)

struttura generale di un algoritmo ricorsivo elementare

```
if  ( caso base)
     {return soluzione del problema banale }
   else
       {
       autoattivazione / autoattivazioni
       return combinazione del risultato/i
                    dell'autoattivazione/i
       }
```

problema della **somma**
dei primi n numeri naturali

$$s_{nat} = \sum_{i=1}^{n} i$$

formula ricorrente

$$y_k = y_{k-1} + k \qquad y_0 = 0$$

la soluzione è $s_{nat} \equiv y_n$

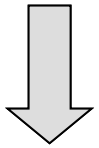
approccio **incrementale**

problema della **somma**
dei primi n numeri naturali

$$s_{nat} = \sum_{i=1}^{n} i$$

formulazione equivalente

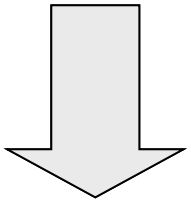
$$s_{nat} = n + \sum_{i=1}^{n-1} i$$



n + soluzione del problema della somma
dei primi $(n\text{-}1)$ numeri naturali

$$S_n = n + (n-1) + (n-2) + \ldots + 3 + 2 + 1$$

$$28 = 7 + 6 + 5 + 4 + 3 + 2 + 1$$



$$S_n = n + S_{n-1}$$

$$28 = 7 + (6 + 5 + 4 + 3 + 2 + 1)$$

"somma dei primi $(n\text{-}1)$ numeri naturali"
è una **istanza più semplice** del problema
"somma dei primi n numeri naturali"

osservazione fondamentale

la risoluzione di tale istanza può essere rimandata, a sua volta, alla risoluzione dell'istanza **ancora più semplice**

$$28 = 7 + \left(6 + \left(5 + 4 + 3 + 2 + 1\right)\right)$$

$$\sum_{i=1}^{n-1} i = \left(n-1\right) + \sum_{i=1}^{n-2} i$$

$$S_n = n + \left(n-1\right) + S_{n-2}$$

rimandare sempre alla risoluzione dell'istanza **ancora più semplice**

fino all'**istanza banale** del problema

$$\sum_{i=1}^{1} i = 1$$

tecnica di programmazione ricorsiva per l'algoritmo incrementale di **somma** dei primi n numeri naturali

```
int somma_ric(int n) {
if (n == 1)
 /* soluzione del problema banale */
       { return 1; }
    else
       /* autoattivazione    */
       { return n + somma_ric(n-1); }
}
```

$$y_k = y_{k-1} + k$$

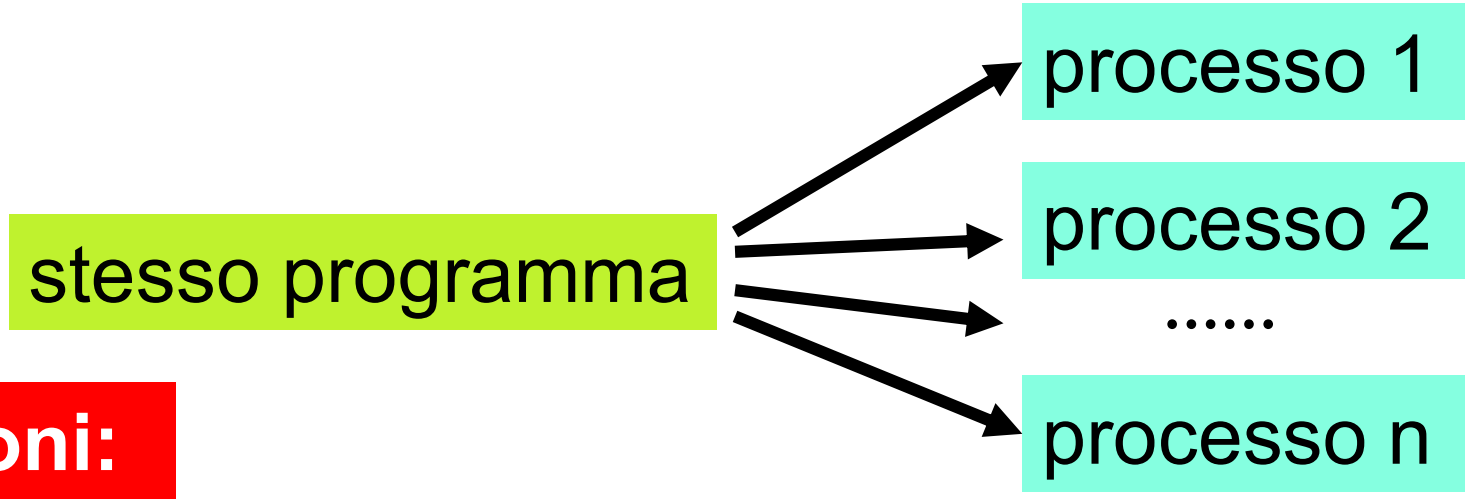
$$y_0 = 0$$

$$y_k = y_{k-1} + k$$

$$y_1 = 1$$

concetti importanti per la programmazione ricorsiva

PROCESSO è un programma in esecuzione

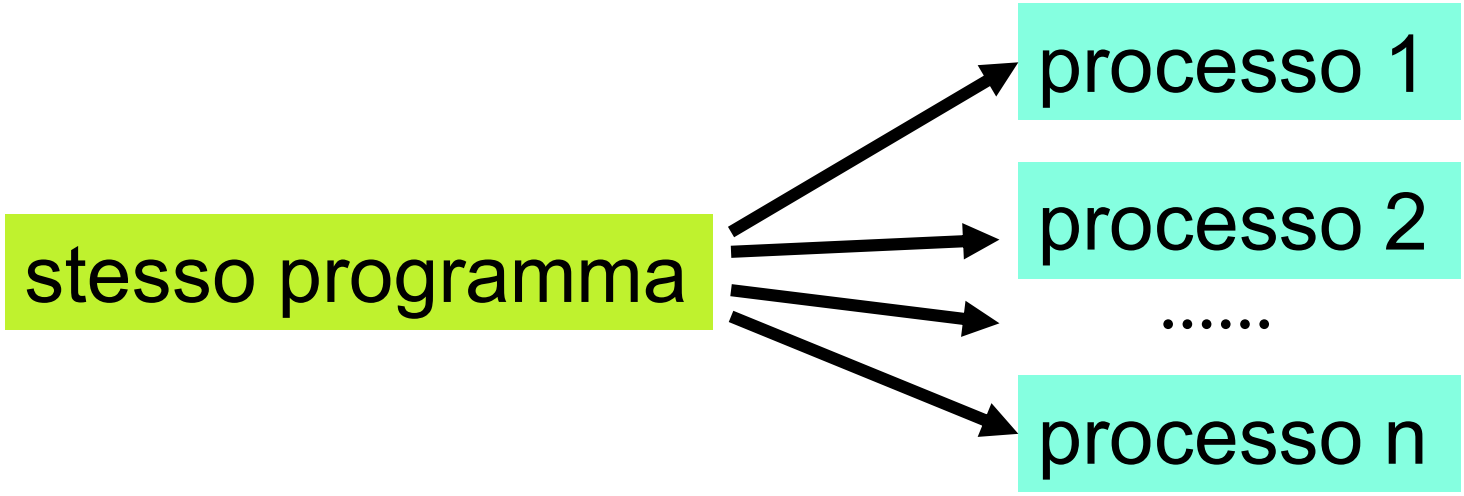


azioni:

- un processo è **creato**
- un processo è **attivo**
- un processo è **sospeso**
- un processo è **concluso**

concetti importanti per la programmazione ricorsiva

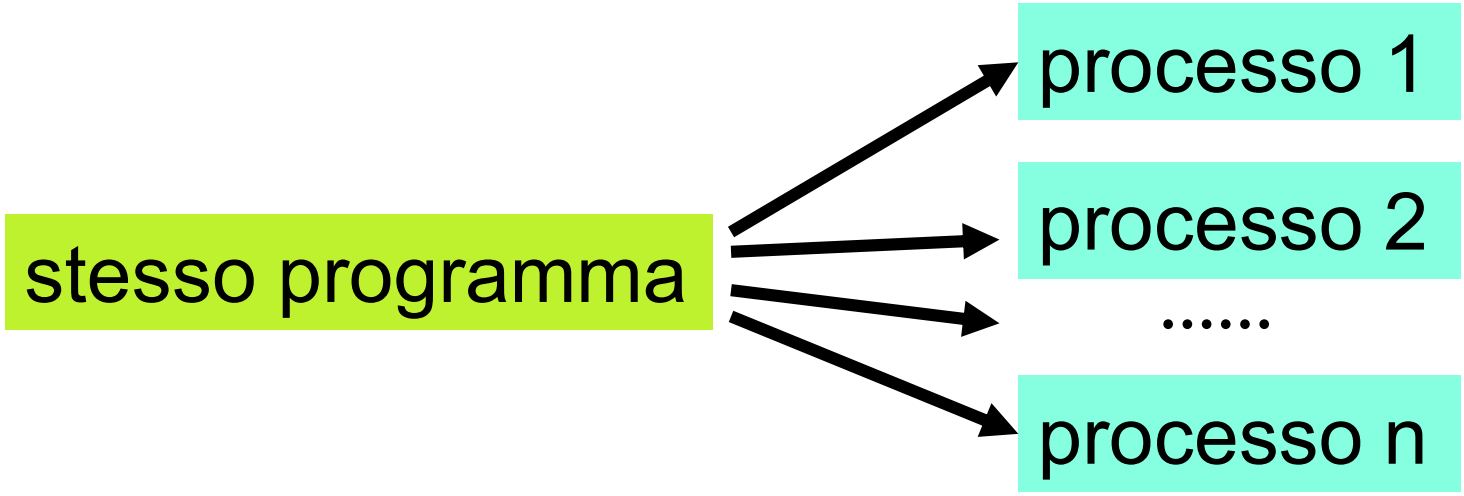
PROCESSO è un programma in esecuzione



una **autoattivazione** di una function **crea** un nuovo processo e lo rende **attivo**

concetti importanti per la programmazione ricorsiva

PROCESSO è un programma in esecuzione



il processo che **autoattiva** la function è **sospeso** e **crea** un nuovo processo **attivo**

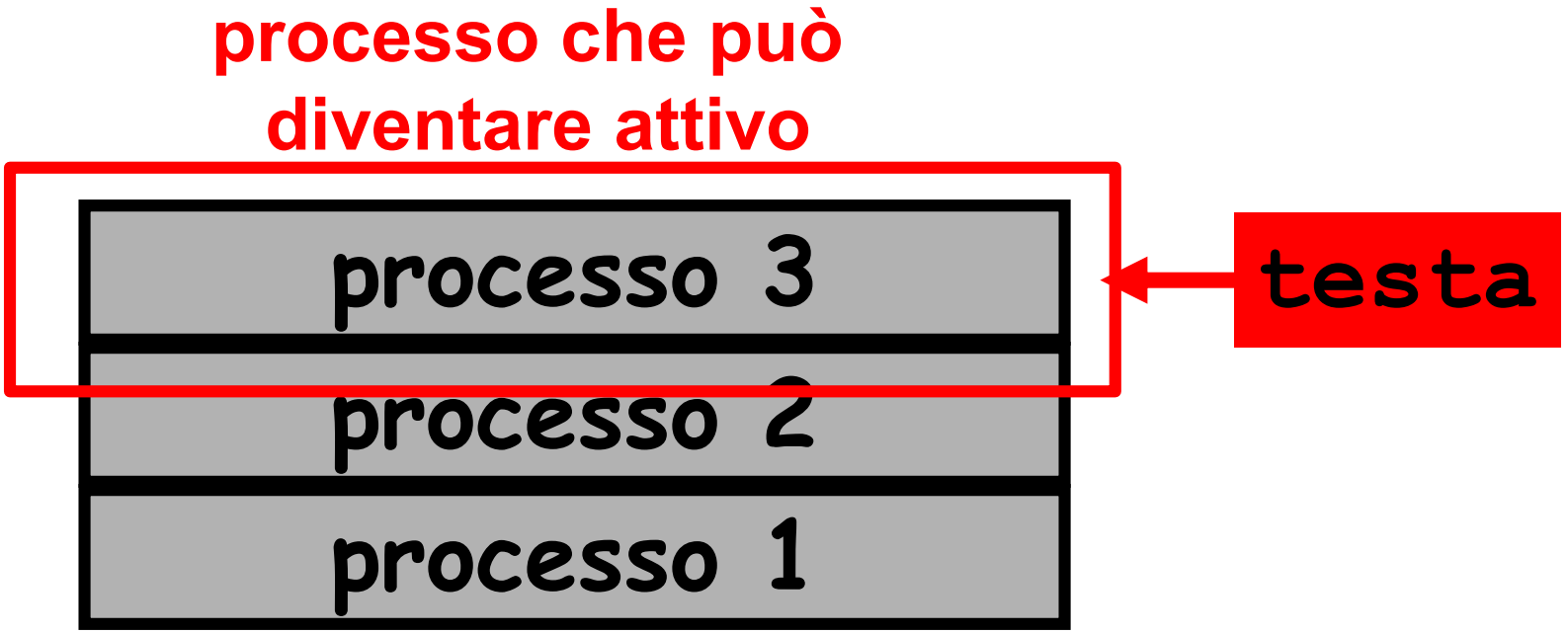
concetti importanti per la programmazione ricorsiva

STACK DEI PROCESSI

è la **pila** (stack) che contiene i processi sospesi, rispettando l'ordine temporale di creazione

concetti importanti per la programmazione ricorsiva

concetti importanti per la programmazione ricorsiva



sequenza delle attivazioni (processi) della function somma_ric

caso n=5

```
somma_ric(5) := 5+somma_ric(4)
                  ||
              4+somma_ric(3)
                  ||
               3+somma_ric(2)
                  ||
                2+somma_ric(1)
```

somma_ric(5) :=

5 + somma_ric(4)

5 + 4 + somma_ric(3)

5 + 4 + 3 + somma_ric(2)

5 + 4 + 3 + 2 + somma_ric(1)

5 + 4 + 3 + 2 + 1

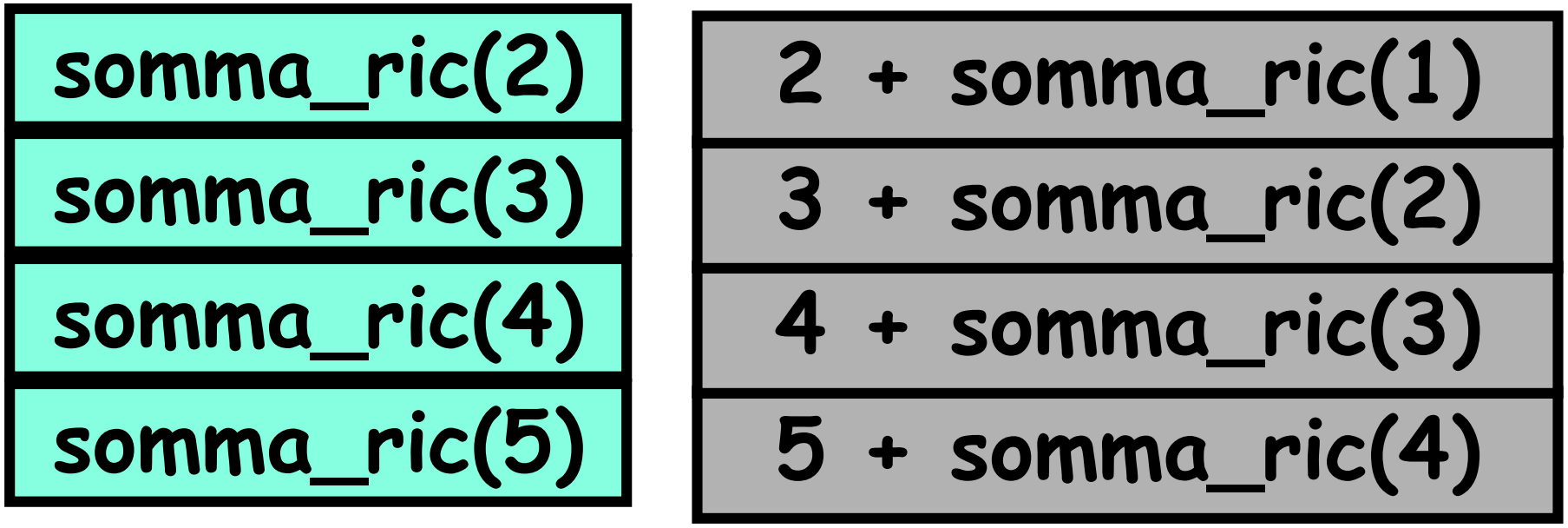
5 + 4 + 3 + 3

5 + 4 + 6

5 + 10

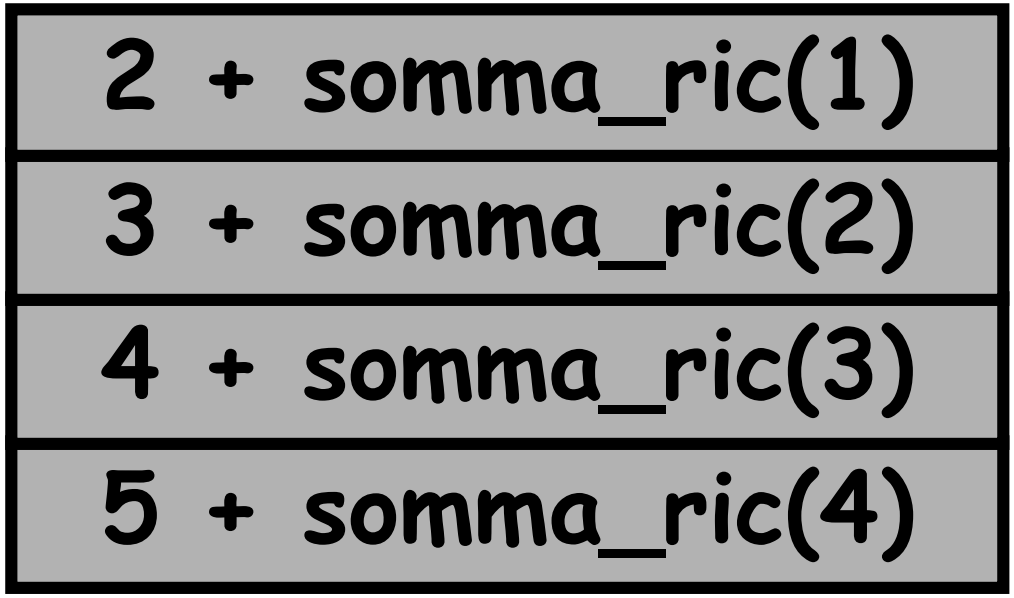
15

stack (pila) dei processi sospesi e delle attivazioni (processi) della function somma_ric

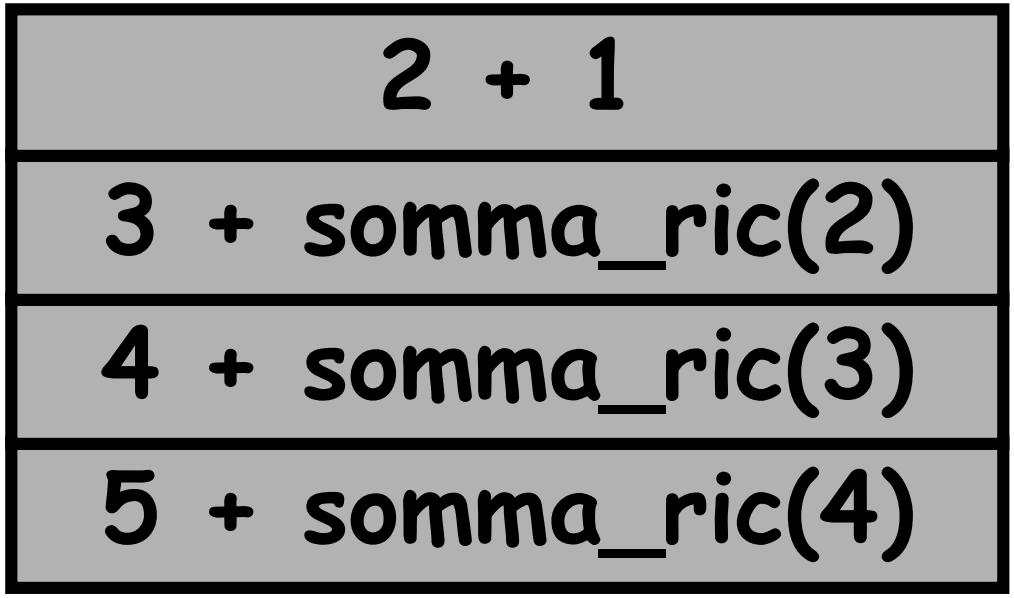


stack dei processi sospesi

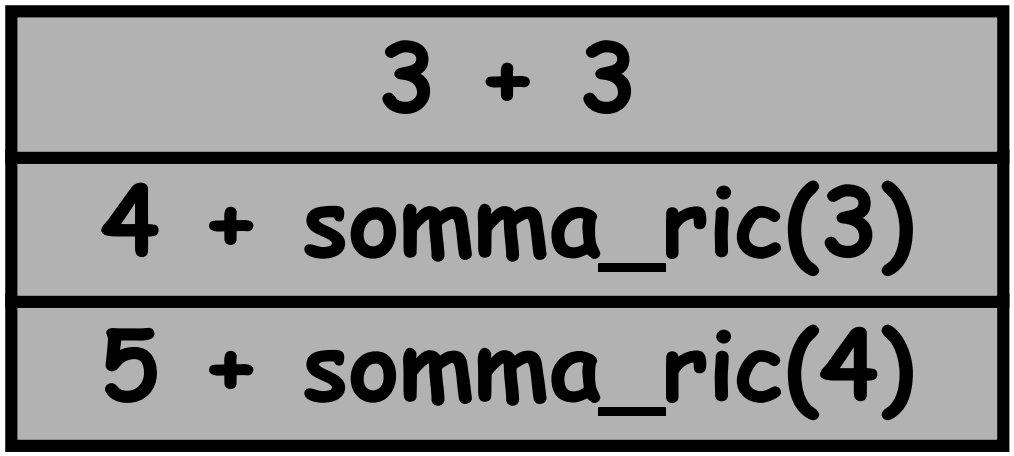
stack (pila) delle attivazioni (processi) della function `somma_ric`



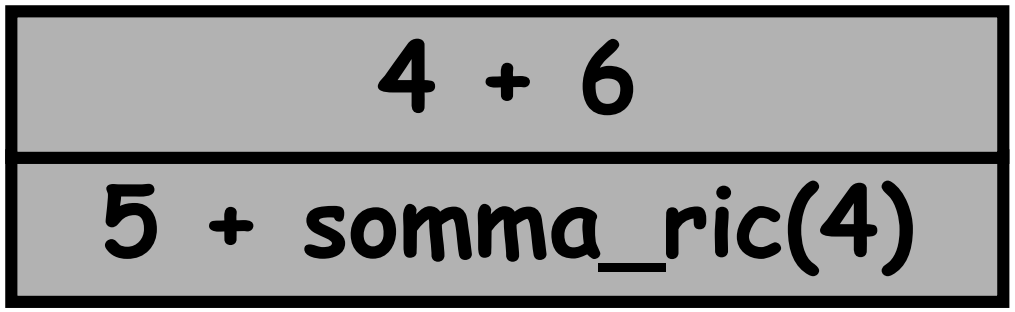
stack (pila) delle attivazioni (processi) della function somma_ric



stack (pila) delle attivazioni (processi) della function somma_ric



stack (pila) delle attivazioni (processi) della function `somma_ric`



stack (pila) delle attivazioni (processi) della function somma_ric

5 + 10

T(n) = n-1
somme

problema del **fattoriale** di n

$$f_n = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$$

formula ricorrente

$$y_k = k y_{k-1} \qquad y_0 = 1$$

la soluzione è $f_n \equiv y_n$

approccio **incrementale**

tecnica di programmazione ricorsiva per l'algoritmo incrementale per il **fattoriale** di n

```
int fattoriale_ric (int n) {
if (n <= 1)
/* soluzione del problema banale */
        { return 1; }
    else
        /* autoattivazione  */
        { return n*fattoriale_ric(n-1) ; }
}
```

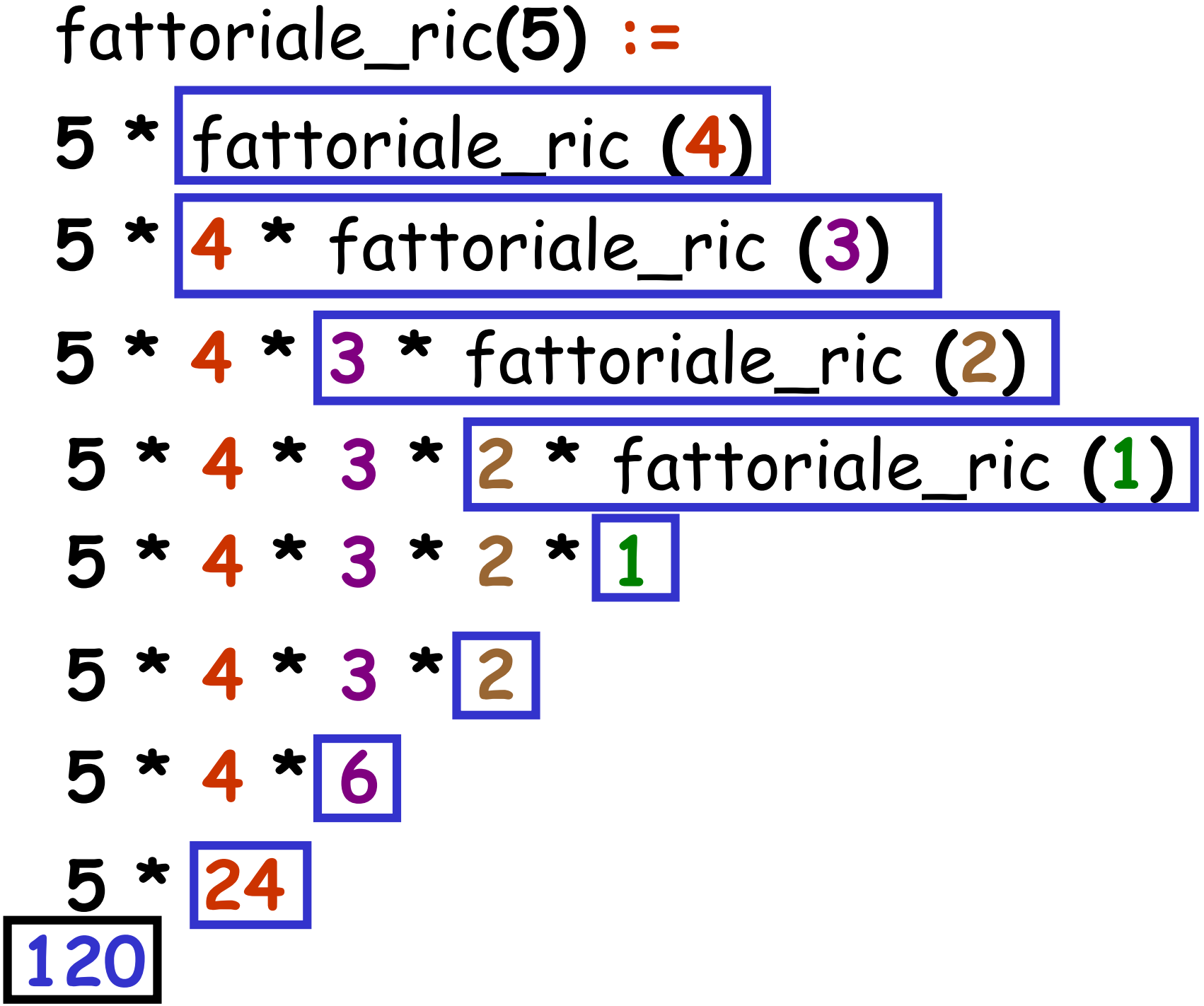
$$y_k = k y_{k-1}$$

$$y_0 = 1$$

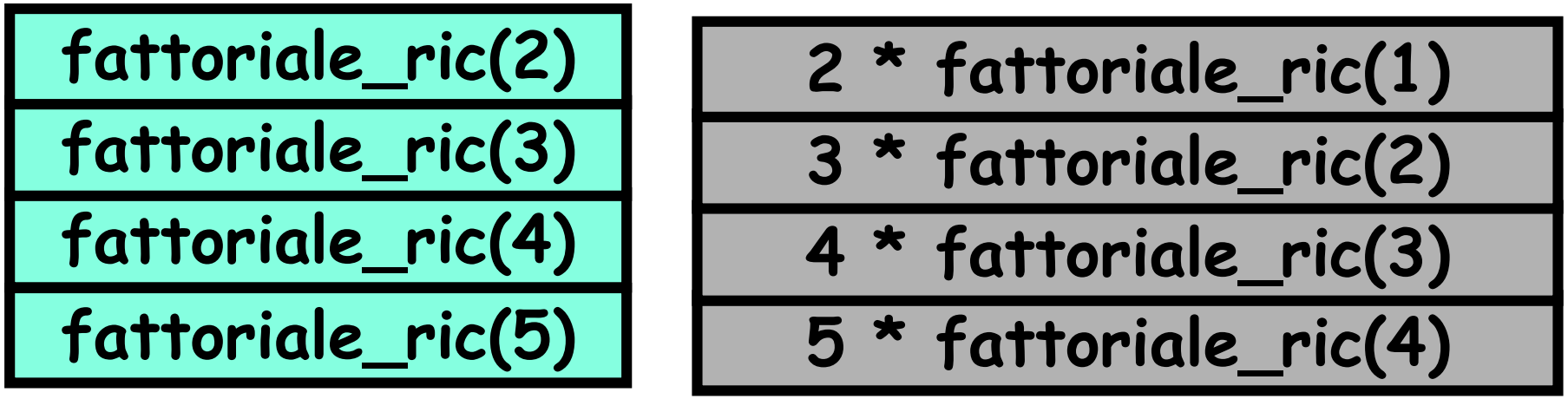
sequenza delle attivazioni (processi) della function fattoriale_ric

caso n=5

```
fattoriale_ric(5) := 5*fattoriale_ric(4)
                      ||
                     4*fattoriale_ric(3)
                      ||
                     3*fattoriale_ric(2)
                      ||
                     2*fattoriale_ric(1)
```

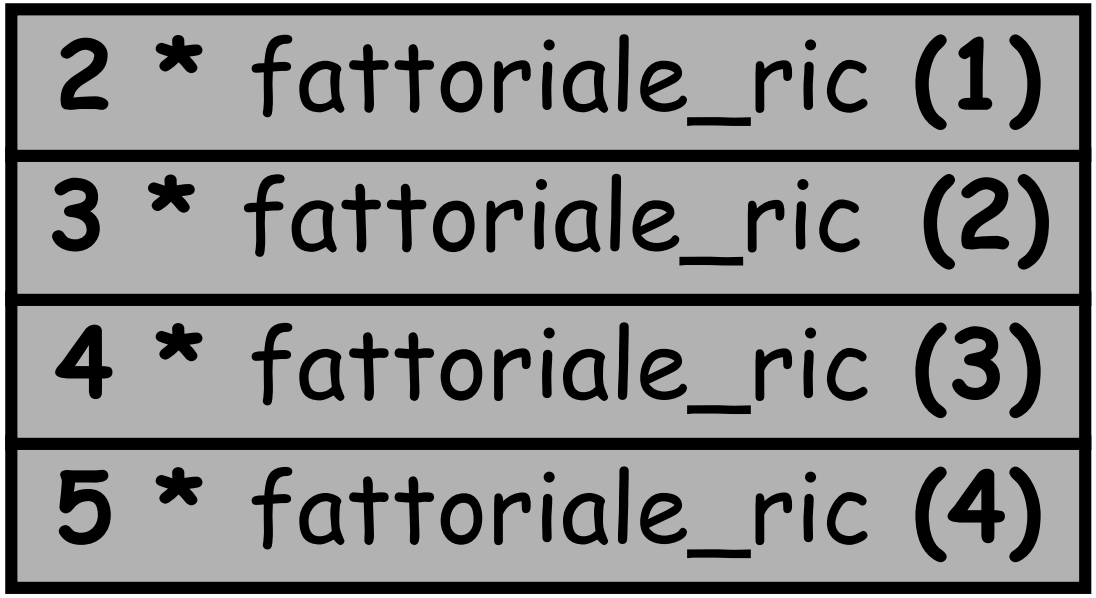

fattoriale_ric(5) :=
5 * fattoriale_ric (4)
5 * 4 * fattoriale_ric (3)
5 * 4 * 3 * fattoriale_ric (2)
5 * 4 * 3 * 2 * fattoriale_ric (1)
5 * 4 * 3 * 2 * 1
5 * 4 * 3 * 2
5 * 4 * 6
5 * 24
120

stack (pila) dei processi sospesi e delle attivazioni (processi) della function fattoriale_ric

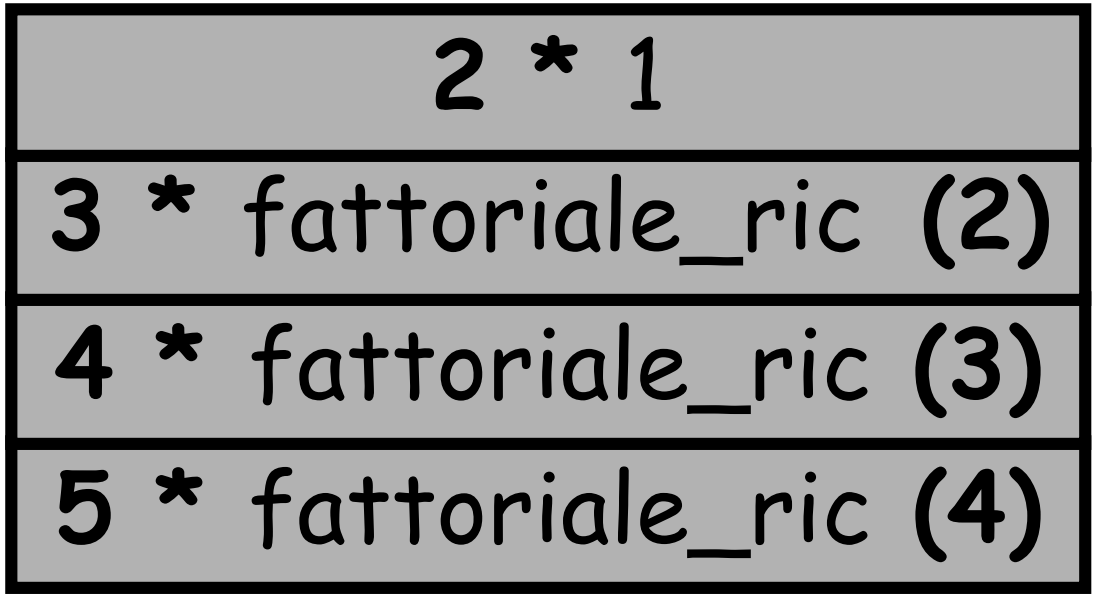


stack dei processi sospesi

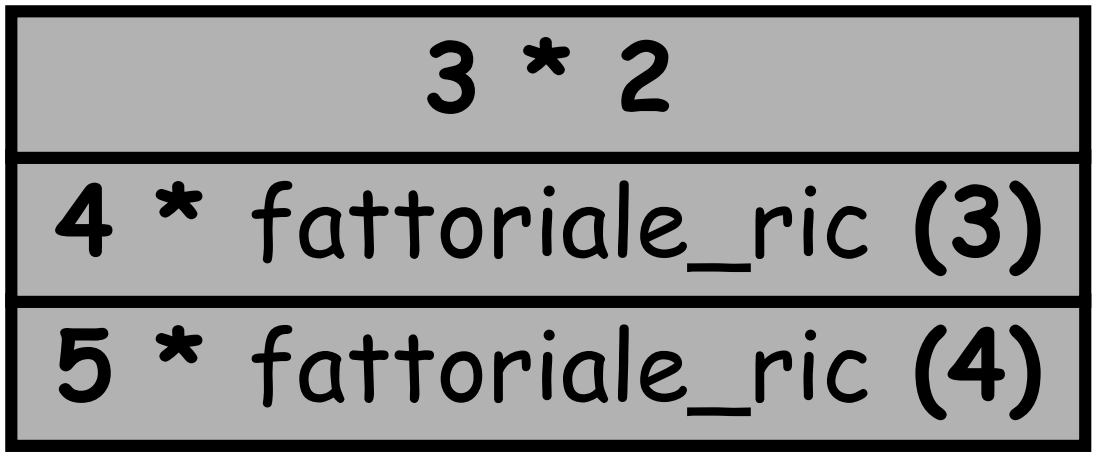
stack (pila) delle attivazioni (processi) della function fattoriale_ric



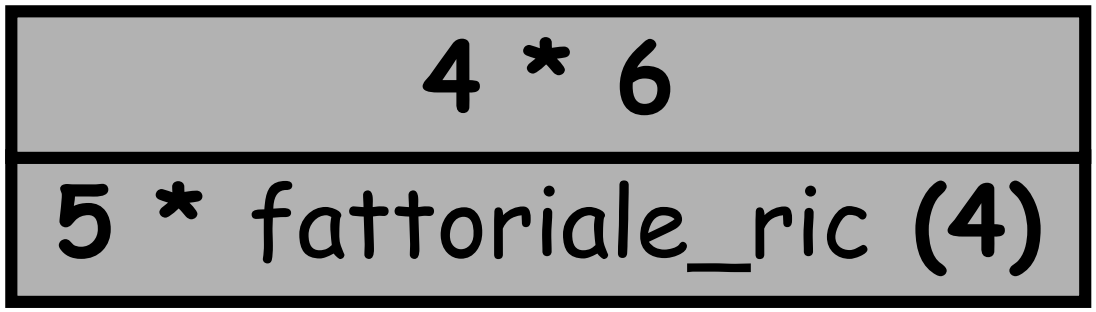
stack (pila) delle attivazioni (processi) della function fattoriale_ric



stack (pila) delle attivazioni (processi) della function fattoriale_ric



stack (pila) delle attivazioni (processi) della function fattoriale_ric



stack (pila) delle attivazioni (processi) della function fattoriale_ric

5 * 24

T(n) = n-1
prodotti