Programmazione 1 - Modulo C

Invarianti di Ciclo

Marco Beccuti

Università degli Studi di Torino

Dipartimento di Informatica

Ottobre 2020

Divisione intera

- Esempio di codice Java per il calcolo iterativo della divisione intera

```
1  class Divisione{
2  /**
3  Dati in ingresso: interi X,D
4  Dati in uscita: interi q,r
5  Condizione di ingresso: X >= 0, D > 0
6  Condizione di uscita: X = q * D + r && r < D
7  */
8  public static void main (String [] args){
9  int X = 14;
10 int D = 3;
11 int q = 0;
12 int r = X;
13 while (r >= D) { /** invariante: ?????? **/
14 q = q + 1;
15 r = r - D;
16 }
17 System.out.println(q + " " + r);
18 }
19 }
```

Invarianti

Dato un ciclo generico, per esempio della forma

```
while (C)
      S
```

dove C è una espressione booleana (la condizione del ciclo) e S una istruzione (il corpo del ciclo), un

invariante del ciclo

è una relazione R tra alcune delle variabili utilizzate nel ciclo (ed eventualmente altri valori) che è vera dopo un numero arbitrario $n \geq 0$ di iterazioni del corpo S del ciclo.

Invarianti

- Fra tutti gli invarianti noi siamo interessati ad uno che sia significativo per la correttezza del programma.

```
1  class Divisione{
2  /**
3  Dati in ingresso: interi X,D
4  Dati in uscita: interi q,r
5  Condizione di ingresso: X >= 0, D > 0
6  Condizione di uscita: X = q * D + r && r < D
7  */
8  public static void main (String [] args){
9  int X = 14;
10 int D = 3;
11 int q = 0;
12 int r = X;
13 while (r >= D) { /** invariante: r>=D  non e' significativo**/
14 q = q + 1;
15 r = r - D;
16 }
17 System.out.println(q + " " + r);
18 }
19 }
```

Divisione intera

- Esempio di codice Java per il calcolo iterativo della divisione intera.

```
1  class Divisione{
2  /**
3  Dati in ingresso: interi X,D
4  Dati in uscita: interi q,r
5  Condizione di ingresso: X >= 0, D > 0
6  Condizione di uscita: X = q * D + r && r < D
7  */
8  public static void main (String [] args){
9  int X = 14;
10 int D = 3;
11 int q = 0;
12 int r = X;
13 while (r >= D) { /** invariante:  X = q * D + r **/
14 q = q + 1;
15 r = r - D;
16 }
17 System.out.println(q + " " + r);
18 }
19 }
```

Divisione intera

```
1  class Divisione{
2  /**
3   Dati in ingresso: interi X,D
4   Dati in uscita: interi q,r
5   Condizione di ingresso: X >= 0, D > 0
6   Condizione di uscita: X = q * D + r && r < D
7   */
8  public static void main (String [] args){
9   int X = 14;
10  int D = 3;
11  int q = 0;
12  int r = X;
13 while (r >= D) { /** invariante:  X = q * D + r **/
14  q = q + 1;
15  r = r - D;
16  }
17  System.out.println(q + " " + r);
18  }
19  }
```

- In generale l'invariante dovrebbe essere un'espressione che congiunta con la negazione della condizione del ciclo ci fornisce la condizione di uscita.

Divisione intera

- Dimostriamo che l'espressione trovata sia sempre vera in ogni iterazione

Supponiamo che **prima** di una iterazione generica la relazione $X = q * D + r$ sia vera, e dimostriamo che è vera anche con i valori presi da q e r **dopo** questa iterazione.

Dopo l'iterazione:

$$q' = q + 1$$
$$r' = r - D$$

Allora:

$$\begin{aligned} q' * D + r' &= (q + 1) * D + (r - D) \\ &= q * D + D + r - D \\ &= q * D + r \\ &= X \end{aligned}$$

Inoltre, poiché $D > 0$ per la condizione di ingresso, ad ogni iterazione $r' < r$, quindi il ciclo termina.

Esercizi sugli invarianti

- Somma $\Rightarrow$ Somma.java
- Sottrazione$\Rightarrow$ Sottrazione.java

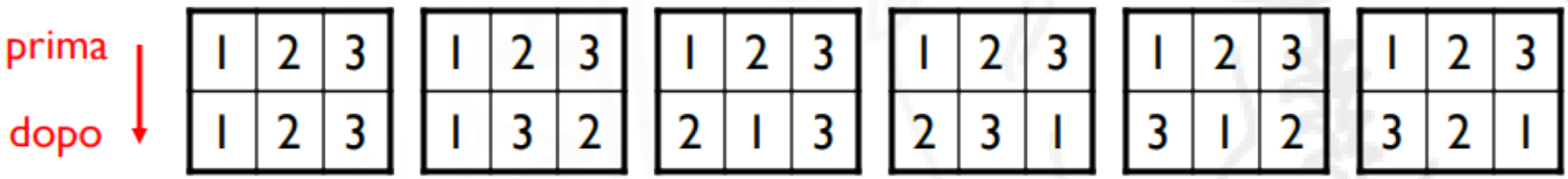
Fattoriale

Il fattoriale è la funzione definita su N come:

$$n! = 1 * 2 * \ldots * n$$

con $0! = 1$

Conta il numero delle permutazioni di un insieme di n oggetti; per esempio, per n = 3:



Fattoriale

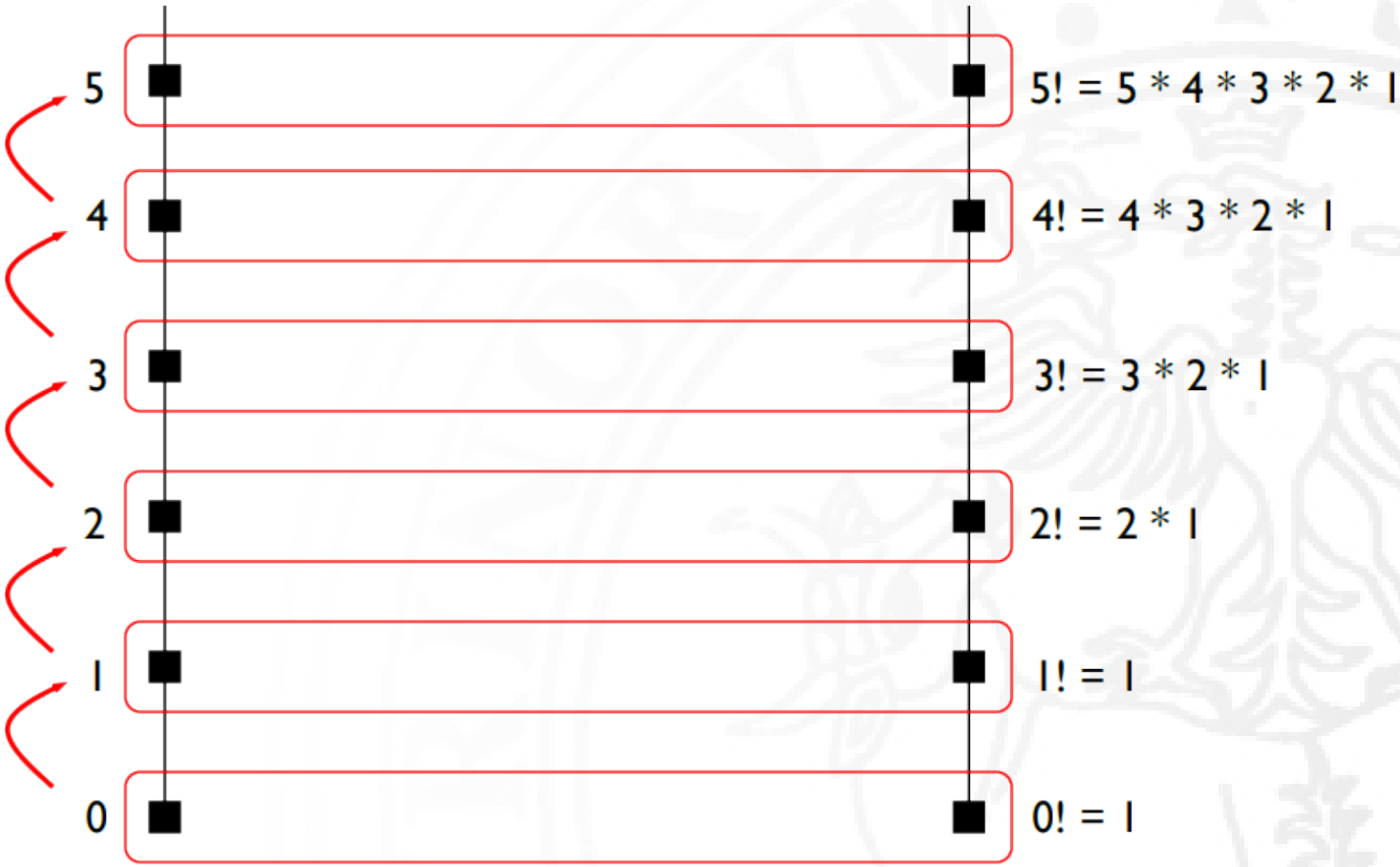
Una funzione $f : \mathbf{N} \rightarrow \mathbf{N}$ è descritta da un insieme di coppie ordinate della forma (x,y), dove y = f(x).

Questo insieme si chiama il **grafo** di f.

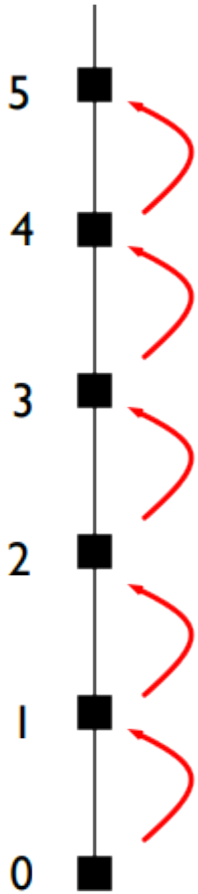
Per implementare il fattoriale, si descrive mediante un ciclo while l'esplorazione del suo grafo attraverso transizioni della forma

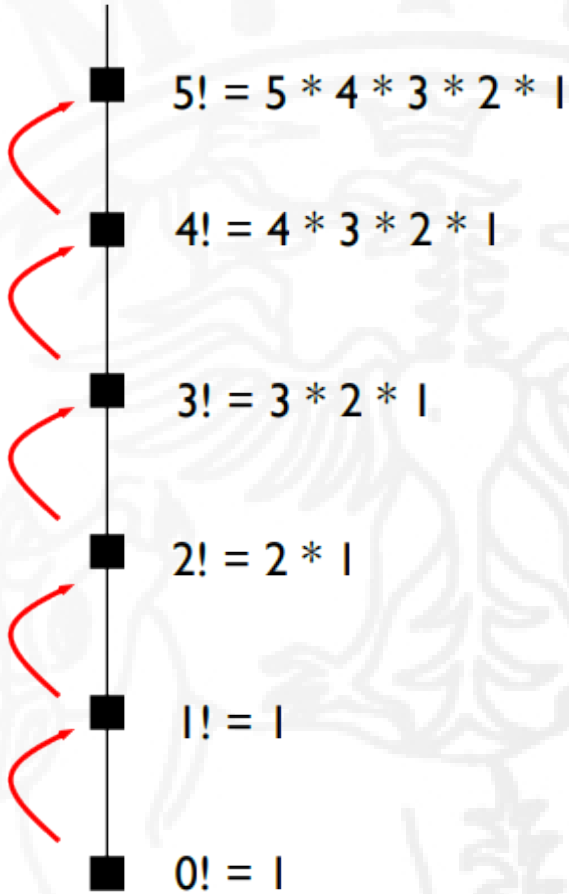
$$(x,y) \rightarrow (x + 1 , (x + 1) * y)$$

Calcolo del fattoriale

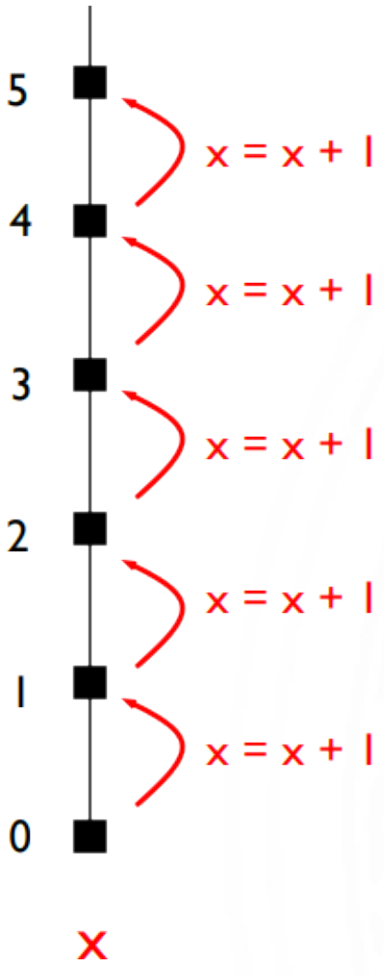


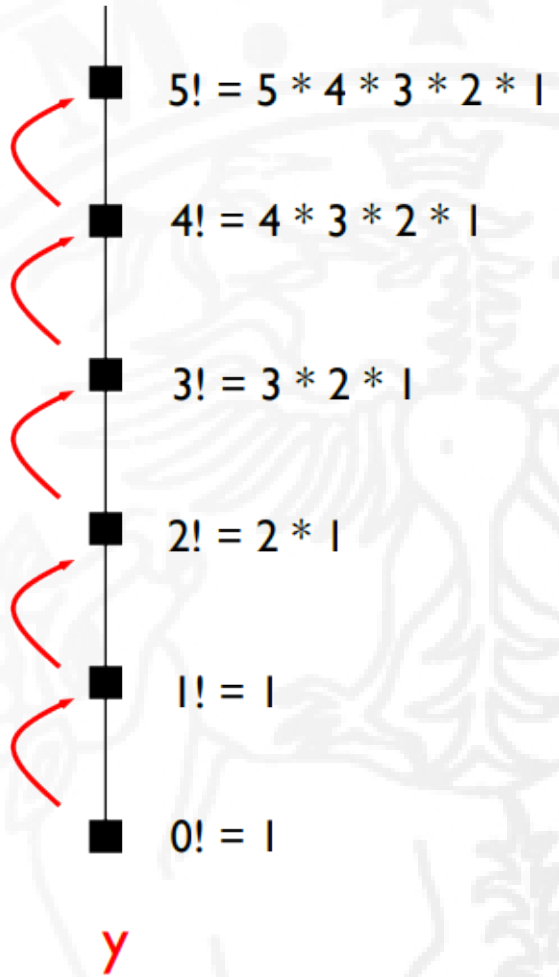
Calcolo del fattoriale



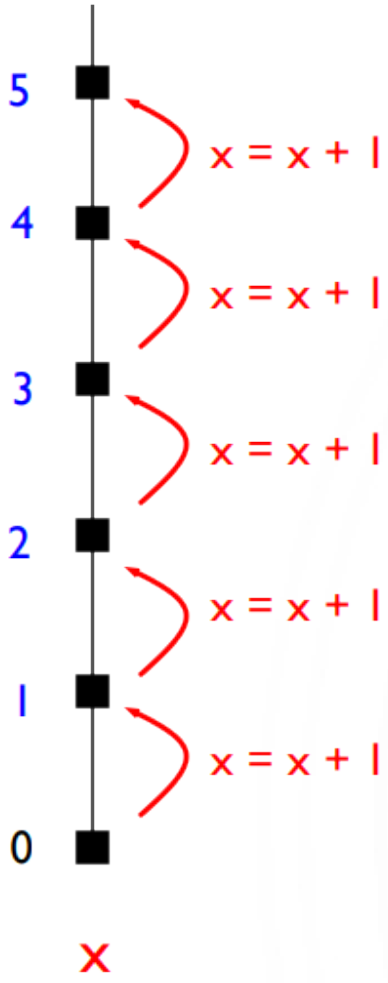


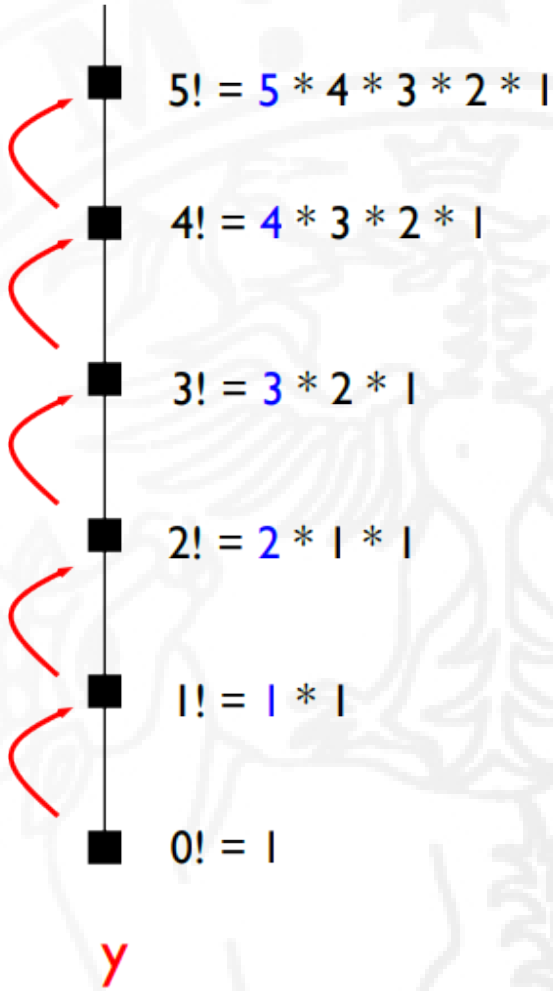
Calcolo del fattoriale



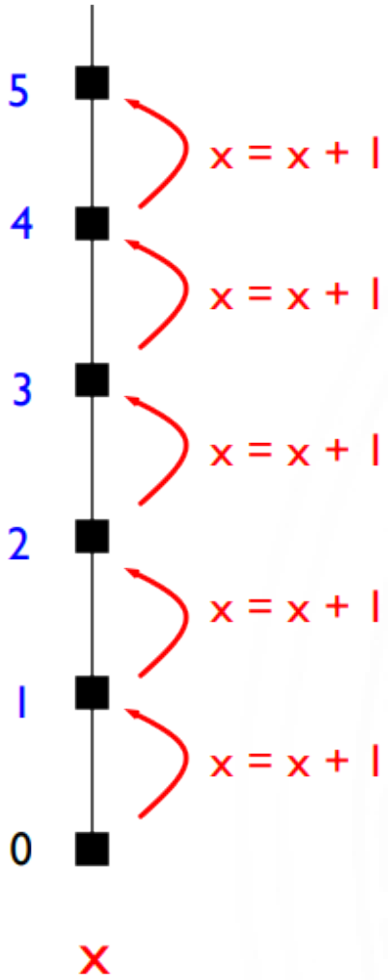


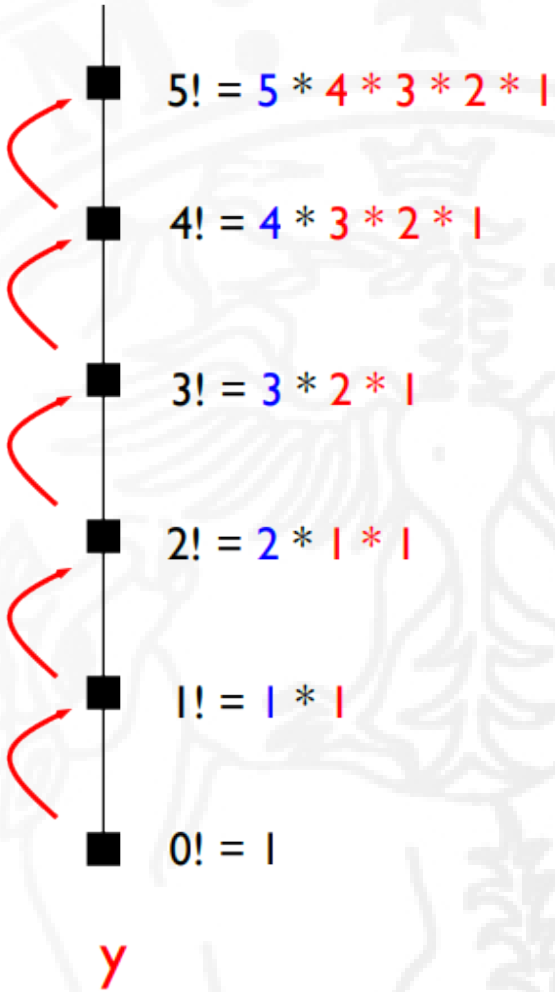
Calcolo del fattoriale



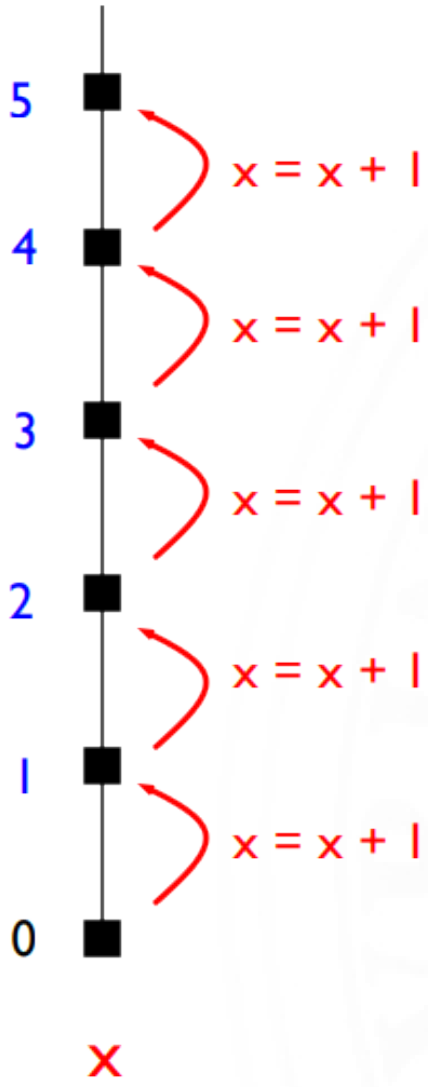


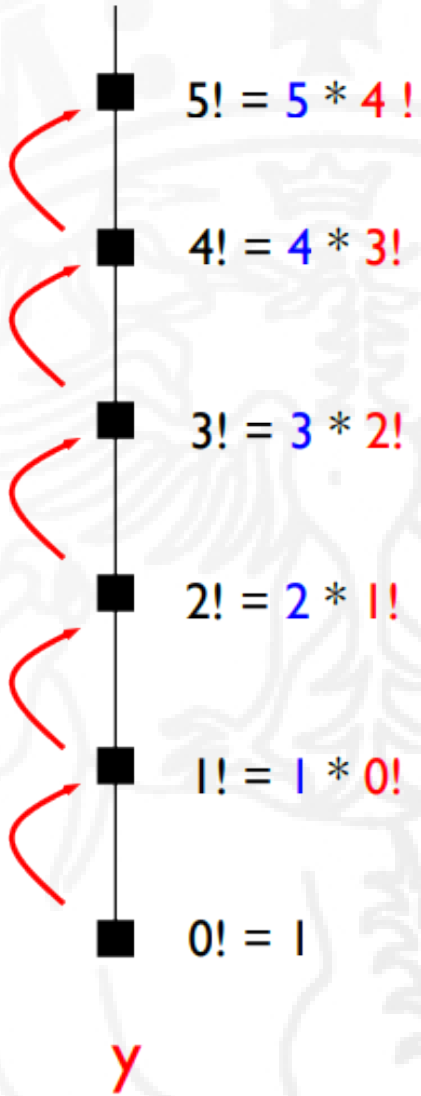
Calcolo del fattoriale



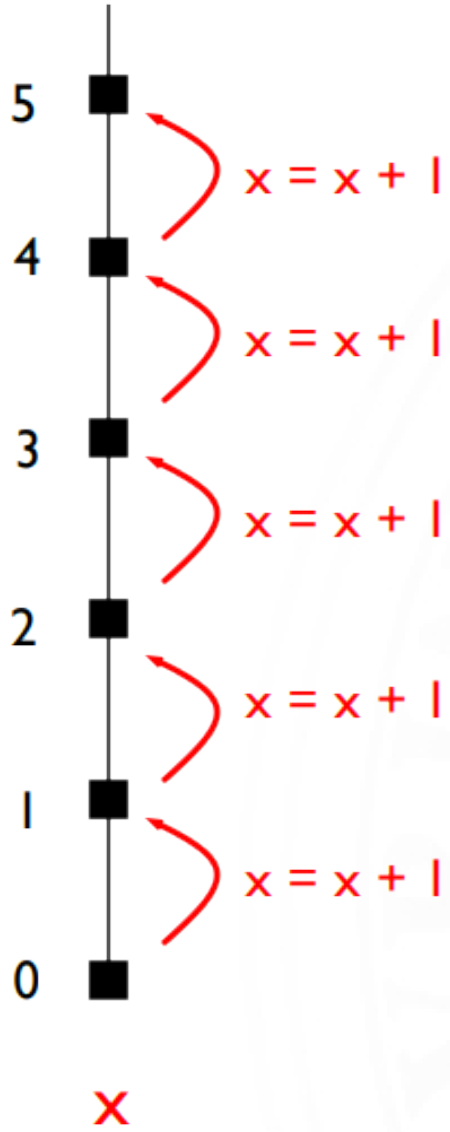


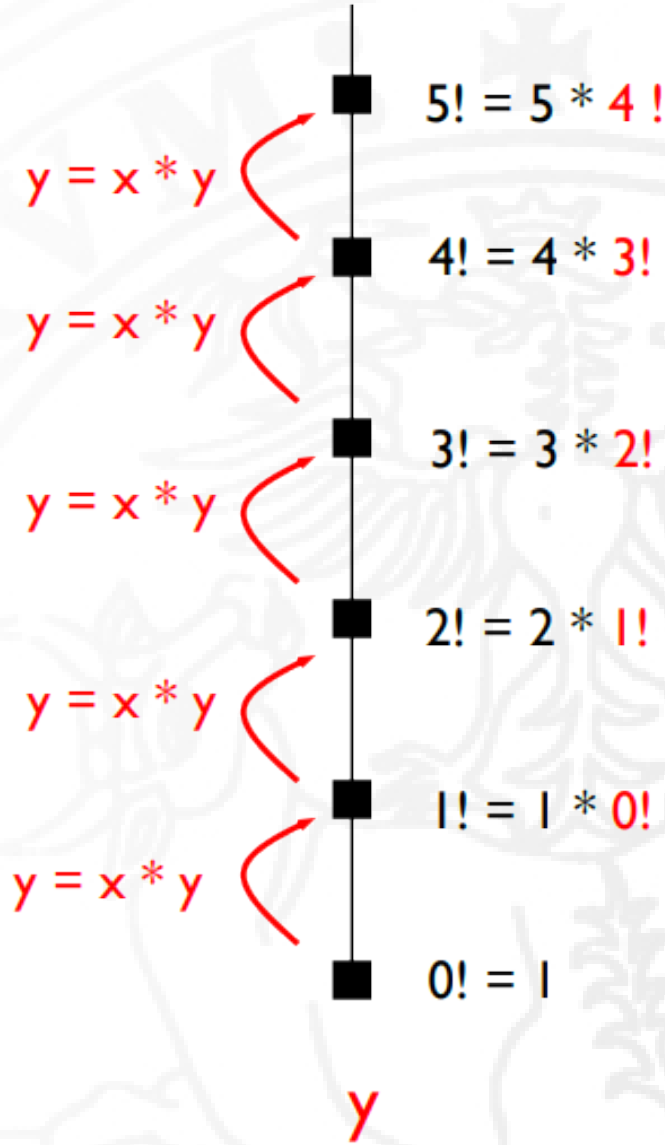
Calcolo del fattoriale





Calcolo del fattoriale





Il codice per il calcolo iterativo del fattoriale

```
1  class Fattoriale {
2  /**
3  Dati in ingresso: interi i >= 0
4  Dati in uscita: interi y >= 0
5  Condizione di ingresso: true
6  Condizione di uscita: y = i!
7  */
8  public static void main (String [] args){
9  int i = 10;
10 int x = 0;
11 int y = 1;
12 while (x < i) { /** invariante: ??? **/
13 x = x + 1;
14 y = x * y;
15 }
16 System.out.println("Il fattoriale di " + i + " è: " + y);
17 }
18 }
```

Il codice per il calcolo iterativo del fattoriale

```
1  class Fattoriale {
2  /**
3  Dati in ingresso: interi i >= 0
4  Dati in uscita: interi y >= 0
5  Condizione di ingresso: true
6  Condizione di uscita: y = i!
7  */
8  public static void main (String [] args){
9  int i = 10;
10 int x = 0;
11 int y = 1;
12 while (x < i) { /** invariante:  y = x! **/
13 x = x + 1;
14 y = x * y;
15 }
16 System.out.println("Il fattoriale di " + i + " è: " + y);
17 }
18 }
```

Fattoriale

- Dimostriamo che l'espressione trovata sia sempre vera in ogni iterazione

Supponiamo che **prima** di una iterazione generica la relazione y = x! sia vera, e dimostriamo che è vera anche con i valori presi da x e y **dopo** questa iterazione.

Dopo l'iterazione:

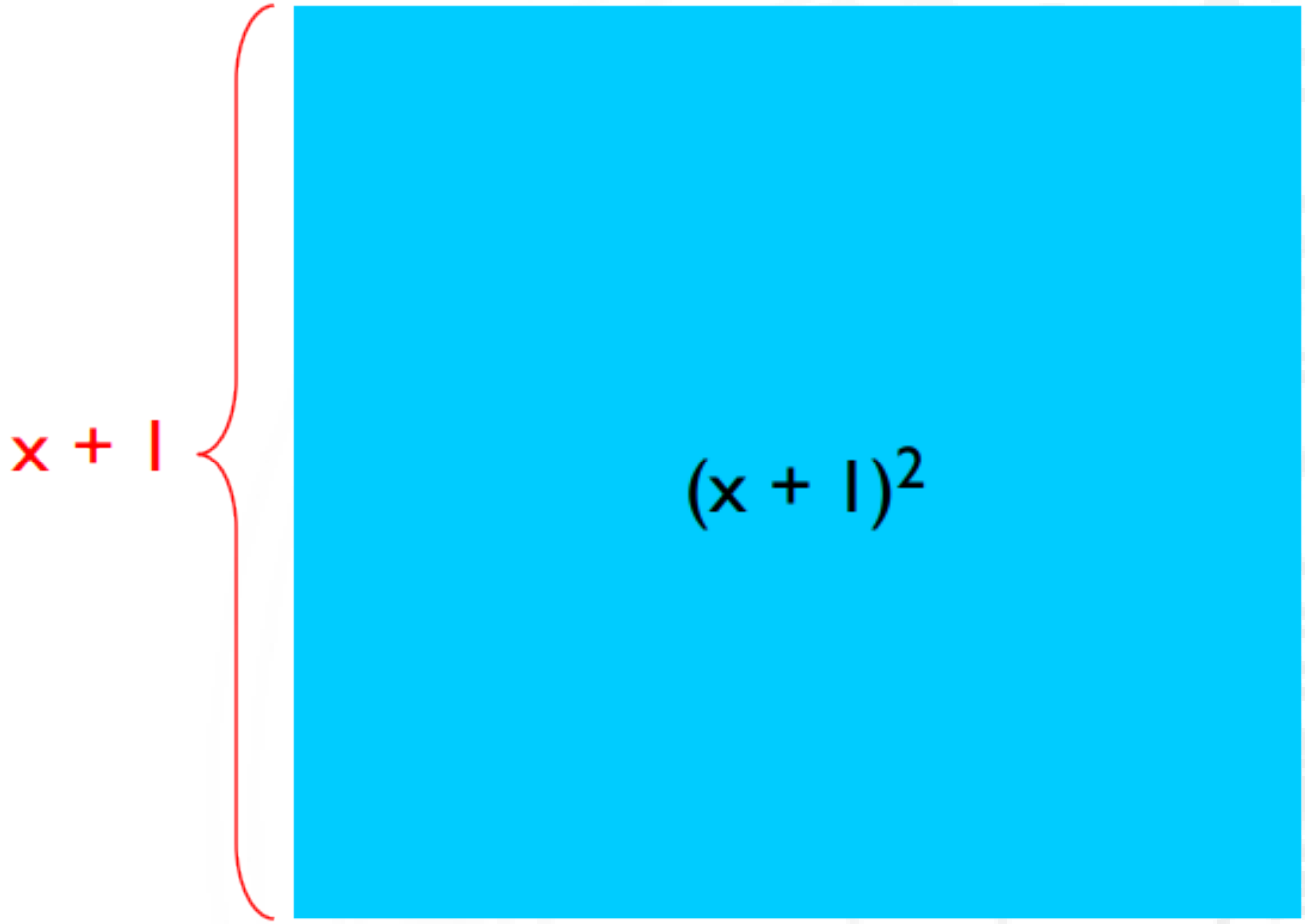
$$x' = x + 1$$
$$y' = x' * y$$

Allora:

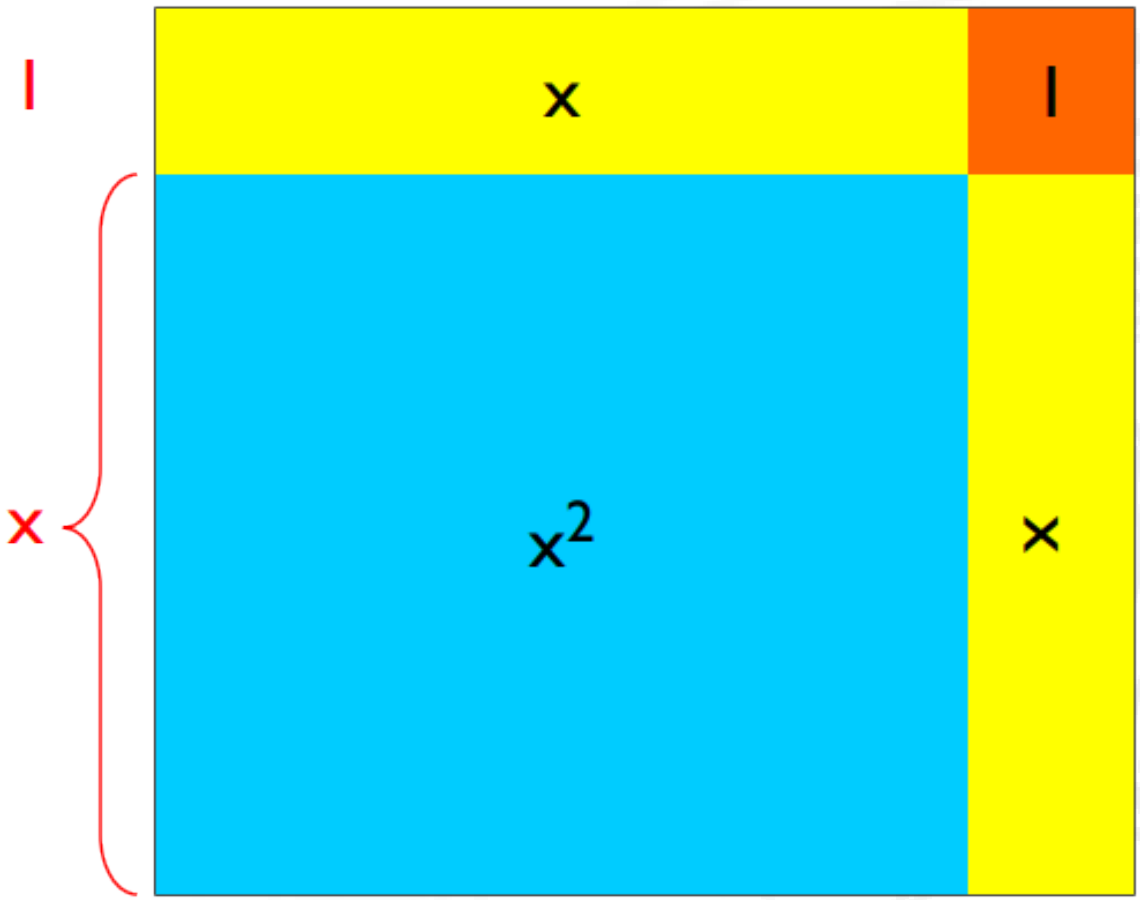
$$\begin{aligned} y' &= x' * y \\ &= (x + 1) * y \\ &= (x + 1) * x! \\ &= (x + 1)! \end{aligned}$$

All'uscita dal ciclo, x = i, quindi y = x! = i!
Il ciclo termina perché la quantità i - x decresce strettamente ad ogni iterazione, perciò il programma è totalmente corretto.

Quadrato



Quadrato



$$(x + 1)^2 = x^2 + 2x + 1$$

Usiamo la formula del quadrato di un binomio

Il codice per il calcolo iterativo del quadrato

```
1  class Quadrato{
2  /**
3  Dati in ingresso: interi x >= 0
4  Dati in uscita: interi q >= 0
5  Condizione di ingresso: true
6  Condizione di uscita: q = x*x
7  */
8  public static void main (String[] args){
9  int x = 7;
10 int n = 0;
11 int q = 0;
12 while (n < x){ /** invariante: ? **/
13 q = q + 2 * n + 1;
14 n = n + 1;
15 }
16 System.out.println("Il quadrato di " + x + " è: " + q);
17 }
18 }
```

Il codice per il calcolo iterativo del quadrato

```
1  class Quadrato{
2  /**
3  Dati in ingresso: interi x >= 0
4  Dati in uscita: interi q >= 0
5  Condizione di ingresso: true
6  Condizione di uscita: q = x*x
7  */
8  public static void main (String[] args){
9  int x = 7;
10 int n = 0;
11 int q = 0;
12 while (n < x){ /** invariante:  q = n*n **/
13 q = q + 2 * n + 1;
14 n = n + 1;
15 }
16 System.out.println("Il quadrato di " + x + " è: " + q);
17 }
18 }
```

Quadrato

- Dimostriamo che l'espressione trovata sia sempre vera in ogni iterazione

Assumiamo che q = n*n prima della n-esima iterazione, dimostriamo che l'equazione
è vera dopo la n-esima iterazione. Durante la n-esima iterazione:

$$n' = n + 1$$
$$q' = q + 2n + 1$$

quindi alla fine della n-esima iterazione abbiamo

$$\begin{aligned} q' &= q + 2n + 1 \\ &= (n+1)(n+1) \text{ (per il disegno di prima)} \\ &= n' * n' \end{aligned}$$

Quando si esce dal ciclo si ha n = x, che stabilisce la condizione di uscita.

Terminazione: la quantità x - n decresce strettamente ad ogni iterazione.

Il Principio di Induzione

Principio di induzione - Proposizioni

Quando abbiamo una proposizione P (una frase affermativa con un valore di verità definito), possiamo creare un predicato P(x) rimpiazzando un nome (di individuo) che compare nella proposizione con una variabile x. Per esempio:

P = Cesare ha varcato il Rubicone $\rightsquigarrow$ P(x) = x ha varcato il Rubicone
Q = $3^2 > 2^2$ $\rightsquigarrow$ Q(x) = $x^2 > 2^2$

In entrambi questi esempi P(x) è una proposizione vera o falsa secondo i valori della variabile x:

P(Cesare) è una proposizione vera, in realtà coincide con la proposizione P, mentre P(Trump) è falsa. Analogamente Q(3) è vera mentre Q(2) è falsa.

Quando un predicato viene derivato in questo modo da una proposizione, la variabile che prende il posto del nome varia su un insieme fissato di possibili valori: nome di persone nel primo caso, nomi di numeri nel secondo.

Diciamo che un predicato P(x) è relativo ai numeri naturali se forma proposizioni quando la variabile x viene rimpiazzata da espressioni numeriche che hanno come valori numeri naturali. Per esempio, Q(x) è un predicato relativo ai naturali se ci restringiamo ai valori naturali di x: non è vero di ogni numero naturale, ma ogni proposizione Q(n) è vera o falsa secondo il valore n.

Principio di induzione - idea

Per dimostrare che una proprietà è vera di ogni numero naturale, sono disponibili alcune tecniche.

Per esempio, se P(x) = il quadrato del numero 2x è divisibile per 4, possiamo dimostrare che $\forall x$ P(x) semplicemente osservando che , per ogni numero naturale x, $(2x)^2 = 4x^2$, che è chiaramente divisibile per 4.

In altri casi, occorre invece riuscire a trovare una descrizione finita di un numero infinito di inferenze della forma

$$\dfrac{\dfrac{\dfrac{P(0) \qquad P(0) \longrightarrow P(1)}{P(1)} \qquad P(1) \longrightarrow P(2)}{P(2)} \qquad P(2) \longrightarrow P(3)}{P(3)} \qquad \vdots$$

Il Principio di Induzione è un modo di riassumere in un solo schema un numero infinito di inferenze come questo.

Principio di induzione

Sia P(n) un predicato relativo ai numero naturali, $n \geq 0$. Supponiamo che:

- (**Base dell'induzione**) P(0) sia vera, e che
- (**Passo induttivo**) assumendo che P(k) sia vera, si riesca a dimostrare che P(k+1) è vera.

Il **principio di induzione** afferma che da queste premesse segue che

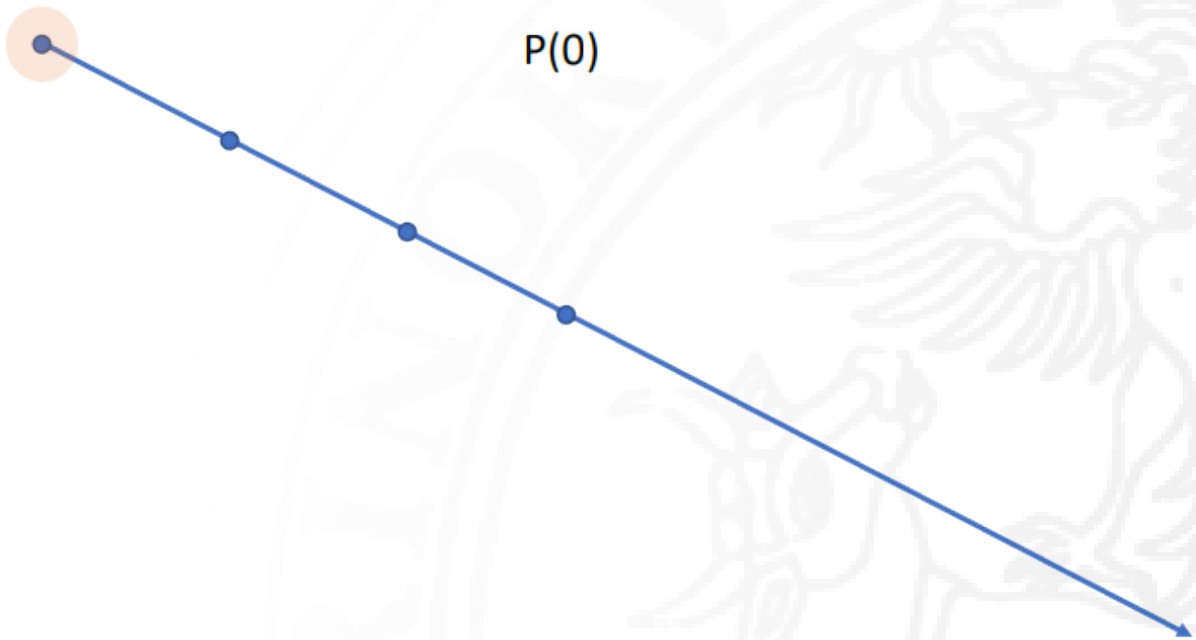
- P(n) è vera per tutti i valori naturali di n.

Una giustificazione (intuitiva) del principio di induzione si può avere dall'osservazione che per ogni numero naturale n:

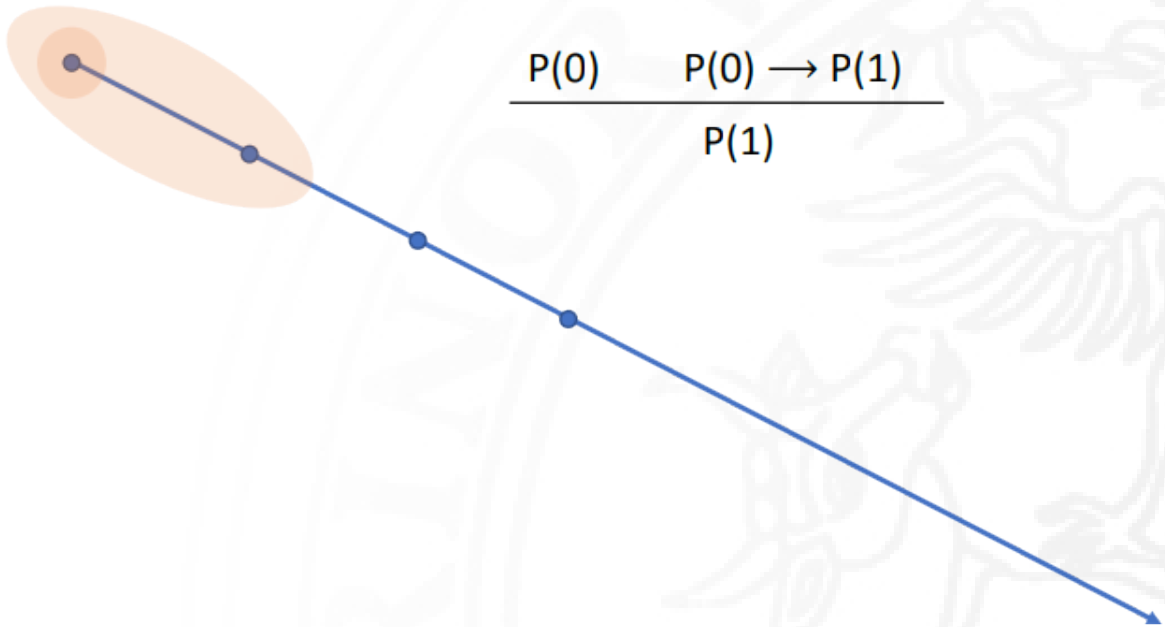
- n = 0, oppure
- n > 0, quindi n = k+1 per un unico k.

La base dell'induzione ed il passo induttivo permettono allora di generare la serie infinita di inferenze che consentono di dimostrare P(k) per ogni specifico valore naturale di k.

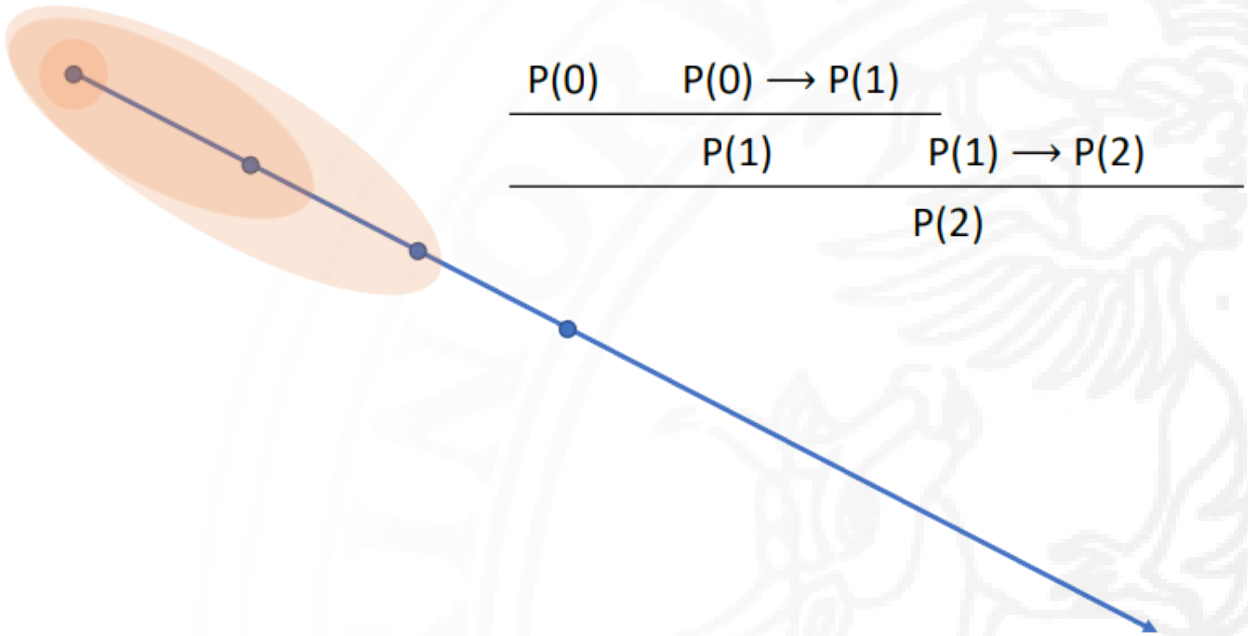
Principio di induzione: graficamente



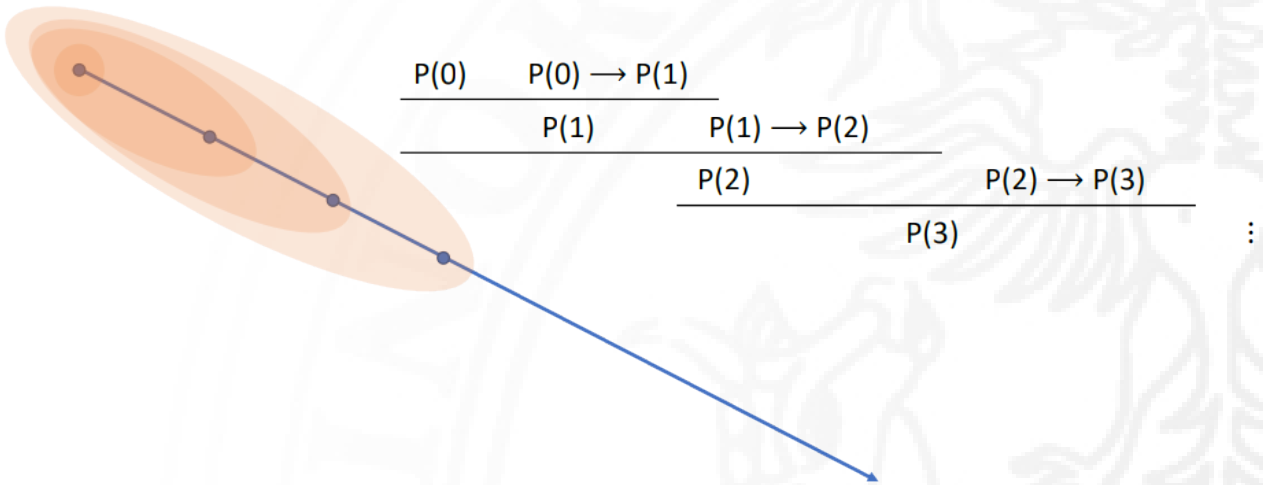
Principio di induzione: graficamente



Principio di induzione: graficamente



Principio di induzione: graficamente



Principio di induzione

Usare il principio di induzione per dimostrare la correttezza di programmi tramite proprietà invarianti

Vediamo come il principio di induzione ci guida nella dimostrazione di correttezza del codice per il calcolo del fattoriale:

```
int x = 0;
int y = 1;
while (x < n) {   //invariante: y = x!
   x = x + 1;
   y = x * y;
}
```

Per dimostrare l'invarianza del predicato , occorre dimostrare che

$$\forall k \geq 0 \; P(k)$$

dove

P(k) = dopo la k-esima iterazione, `y = x!`

Base: P(0), cioè dopo la 0-esima iterazione, `y = x!`, cioè prima dell'esecuzione del ciclo `y = x!`.
Infatti y = 1 = 0! = `x!`

Principio di induzione

Passo induttivo:

```
while (x < n) {
    x = x + 1;
    y = x * y;
}
```

Prendiamo un generico k ≥ 0. **Assumiamo** che:

P(k), cioè
dopo la k-esima iterazione, `y = x!`

allora:

dopo l'iterazione successiva (la k+1-esima) abbiamo

```
x' = x + 1
y' = x' * y = (x + 1) * y
           = (x + 1) * x!
           = (x + 1)!
```

per ipotesi induttiva

quindi: P(k+1)

Principio di induzione

Passo induttivo:

```
while (x < n) {
    x = x + 1;
    y = x * y;
}
```

Prendiamo un generico k ≥ 0. **Assumiamo** che:

P(k), cioè
dopo la k-esima iterazione, `y = x!`

allora:

dopo l'iterazione successiva (la k+1-esima) abbiamo

$$x' = x + 1$$
$$y' = x' * y = (x + 1) * y$$
$$= (x + 1) * x!$$
$$= (x + 1)!$$

per ipotesi induttiva

quindi: P(k+1)

Quindi, per il Principio di Induzione: ∀k ≥ 0 P(k)
Questo significa che la proprietà `y = x!` è **invariante**.
La **correttezza** del programma segue dal fatto che, all'uscita dal ciclo, `x = n`, quindi per l'invariante `y = x! = n!`