# Programmazione 1 - Modulo C

## Metodi ricorsivi

**Marco Beccuti**

*Università degli Studi di Torino*

*Dipartimento di Informatica*

Novembre 2020

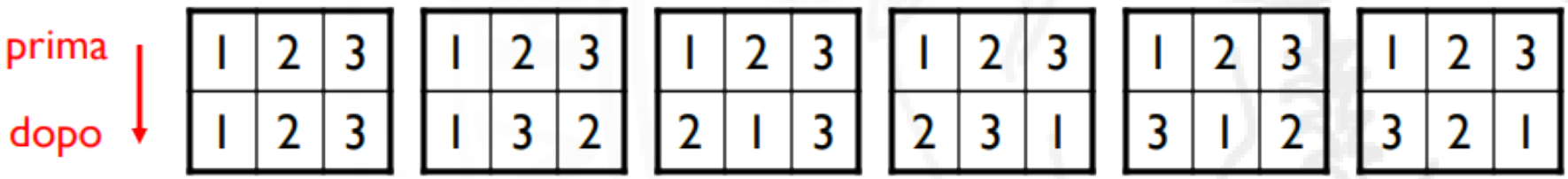
# Fattoriale

Il fattoriale è la funzione definita su N come:

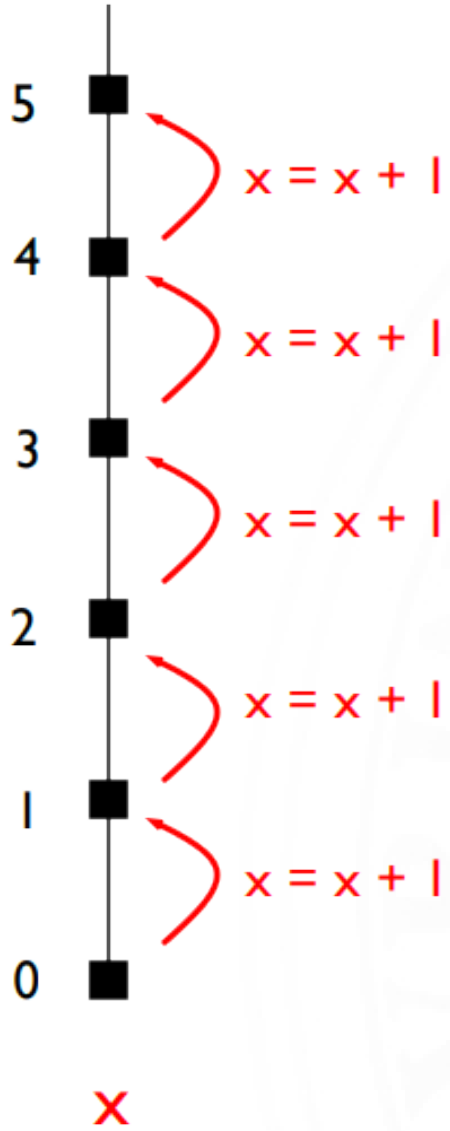
$$n! = 1 * 2 * \ldots * n$$

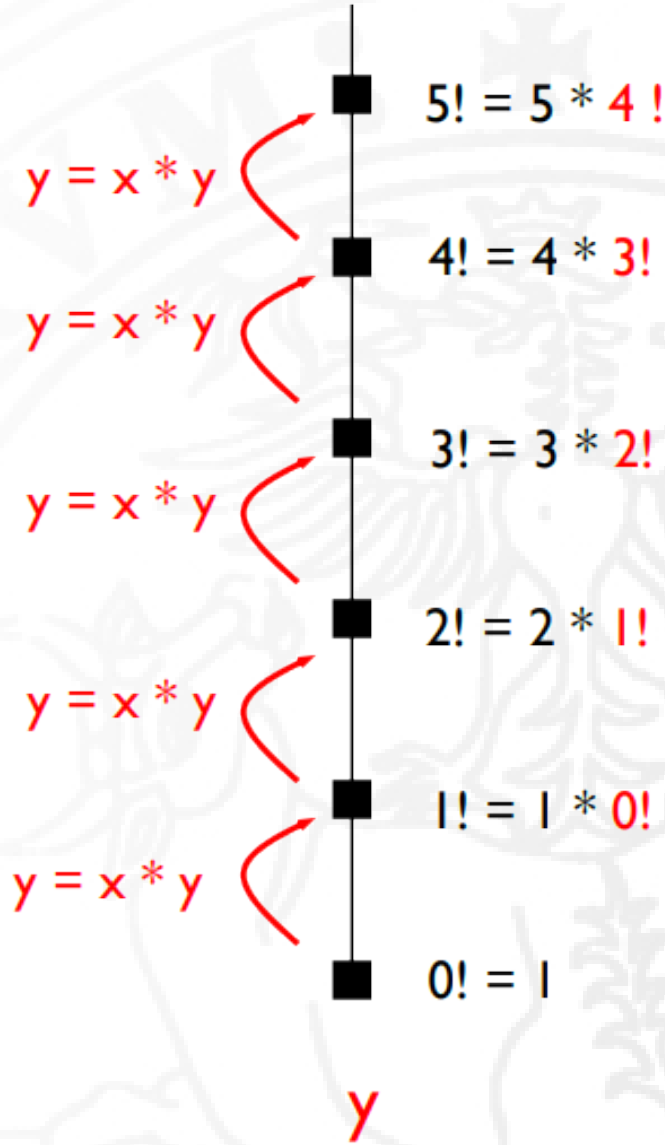
con $0! = 1$

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



# 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:  y = x! **/
13 x = x + 1;
14 y = x * y;
15 }
16 System.out.println("Il fattoriale di " + i + " è: " + y);
17 }
18 }
```

# Formula del fattoriale

La funzione fattoriale può anche essere definita in modo ricorsivo:

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

- É possibile implementare direttamente questa funzione in java?
- É possibile invocare un metodo mentre si esegue lo stesso metodo?

# La ricorsione

- Invocare un metodo mentre si esegue lo stesso metodo é un paradigma di programmazione chiamato **ricorsione**;
- Un metodo che faccia uso della ricorsione si chiama metodo **ricorsivo**;
- La ricorsione é uno strumento **molto potente** per realizzare alcuni algoritmi, ma puó anche essere **fonte di molti errori** di difficile diagnosi.

# La ricorsione: caso base

- Il metodo ricorsivo deve fornire la soluzione del problema in almeno un caso particolare, senza ricorrere ad una chiamata ricorsiva;
- Tale caso si chiama **caso base della ricorsione**;
- Nel caso della funzione fattoriale il caso base e':

```
if (n == 0)
    return 1;
```

- Si possono definire piú casi base.

# La ricorsione: passo ricorsivo

- Il metodo ricorsivo deve effettuare la chiamata ricorsiva **dopo aver semplificato il problema**;
- il concetto di "problema piú semplice" varia di volta in volta: in generale, **ad ogni chiamata ricorsiva bisogna avvicinarsi ad un caso base**
- Nel caso della funzione fattoriale il passo ricorsivo é:

```
if (n > 0)
   return n * factorial(n - 1);
```

# Metodo ricorsivo per il fattoriale

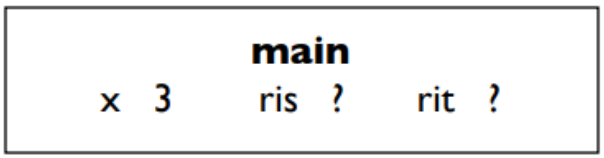
```
1  public class Fattoriale {
2  static int F(int n){
3  /***
4  Due casi per la definizione ricorsiva del fattoriale:
5  0! = 1
6  n! = (n - 1)! * n, se n > 0
7  ***/
8  if (n == 0)
9  return 1;
10 else
11 return n * F(n - 1);
12 }
13
14 public static void main (String[] args){
15 System.out.print("Inserire un numero naturale: ");
16 int x = SavitchIn.readLineInt();
17 System.out.println(F(x));
18 }
19 }
```

# Esecuzione dei metodi ricorsivi

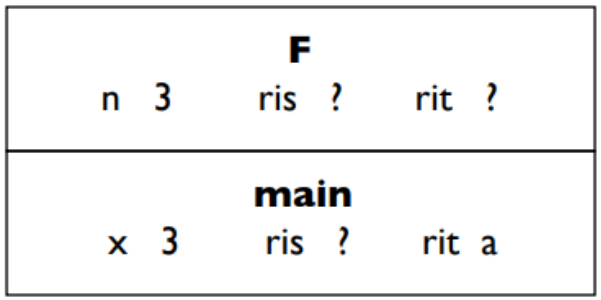
```
static int F(int n){
if (n == 0)
        return 1;
else
        return n * F(n-1);    b
}
```

**Nel main:**

```
int x = SavitchIn.readLineInt();
System.out.println(F(x));       a
```



# Esecuzione dei metodi ricorsivi

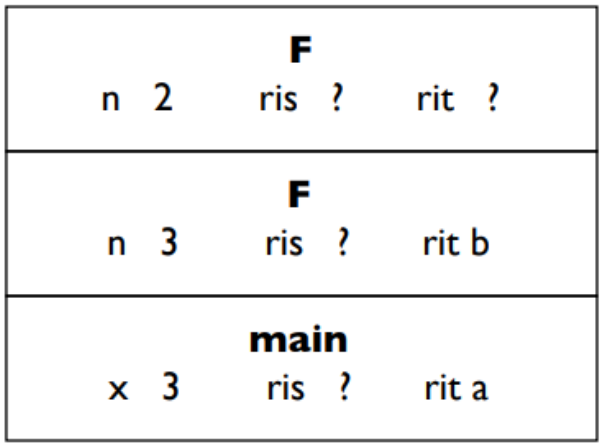



```
static int F(int n){
if (n == 0)
        return 1;
else
        return n * F(n-1);    b
}
```

**Nel main:**

```
int x = SavitchIn.readLineInt();
System.out.println(F(x));        a
```

# Esecuzione dei metodi ricorsivi




```
static int F(int n){
if (n == 0)
        return 1;
else
        return n * F(n-1);    b
}
```

**Nel main:**

```
int x = SavitchIn.readLineInt();
System.out.println(F(x));        a
```

# Esecuzione dei metodi ricorsivi

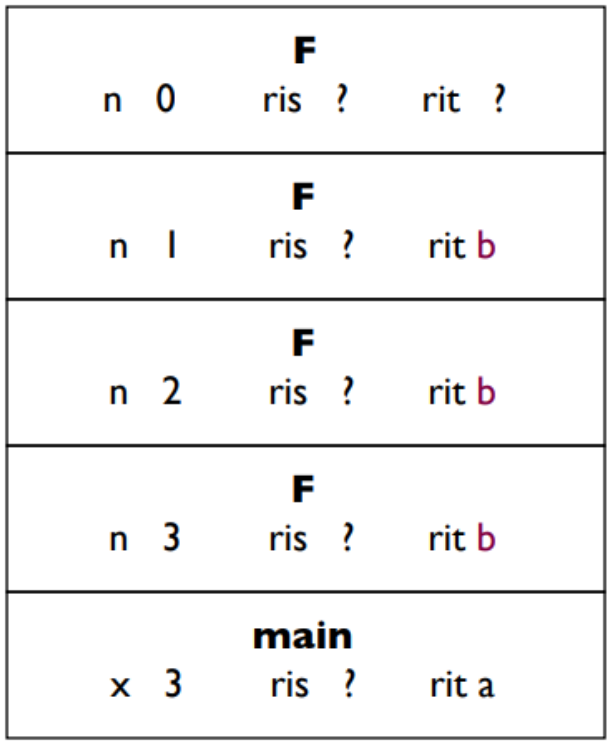
| | | |
|---|---|---|
| **F** | | |
| n 1 | ris ? | rit ? |
| **F** | | |
| n 2 | ris ? | rit b |
| **F** | | |
| n 3 | ris ? | rit b |
| **main** | | |
| x 3 | ris ? | rit a |

```
static int F(int n){
if (n == 0)
        return 1;
else
        return n * F(n-1);    b
}
```

**Nel main:**

```
int x = SavitchIn.readLineInt();
System.out.println(F(x));       a
```

# Esecuzione dei metodi ricorsivi




```
static int F(int n){
if (n == 0)
        return 1;
else
        return n * F(n-1);    b
}
```

**Nel main:**

```
int x = SavitchIn.readLineInt();
System.out.println(F(x));        a
```

# Esecuzione dei metodi ricorsivi

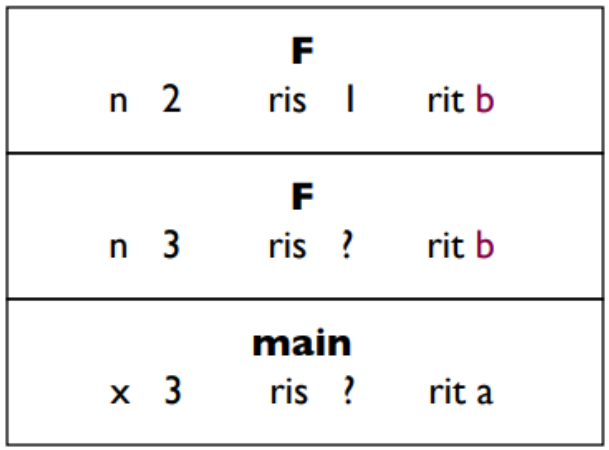
| | | |
|---|---|---|
| **F** | | |
| n 1 | ris 1 | rit b |
| **F** | | |
| n 2 | ris ? | rit b |
| **F** | | |
| n 3 | ris ? | rit b |
| **main** | | |
| x 3 | ris ? | rit a |

```
static int F(int n){
if (n == 0)
        return 1;
else
        return n * F(n-1);    b
}
```

**Nel main:**

```
int x = SavitchIn.readLineInt();
System.out.println(F(x));        a
```

# Esecuzione dei metodi ricorsivi

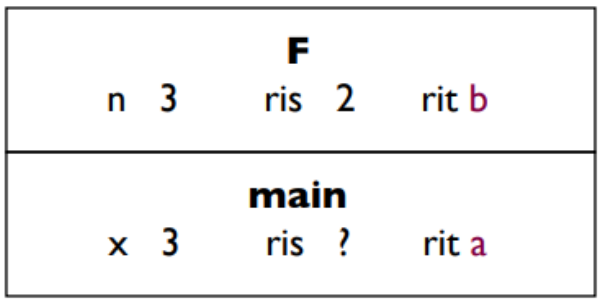



```
static int F(int n){
if (n == 0)
        return 1;
else
        return n * F(n-1);    b
}
```

**Nel main:**

```
int x = SavitchIn.readLineInt();
System.out.println(F(x));        a
```

# Esecuzione dei metodi ricorsivi




```
static int F(int n){
if (n == 0)
        return 1;
else
        return n * F(n-1);    b
}
```

**Nel main:**

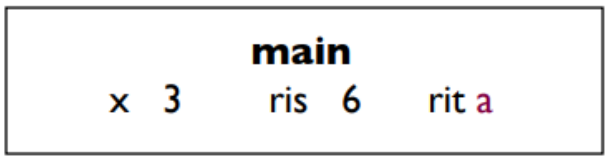
```
int x = SavitchIn.readLineInt();
System.out.println(F(x));        a
```

# Esecuzione dei metodi ricorsivi

```
static int F(int n){
if (n == 0)
        return 1;
else
        return n * F(n-1);    b
}
```

**Nel main:**

```
int x = SavitchIn.readLineInt();
System.out.println(F(x));        a
```



# Altri esempi

- Quale funzione sui numeri naturali é definita dalle seguenti clausole ricorsive?

$$
\begin{aligned}
f(0) &= 0 \\
f(n+1) &= (n+1) + f(n)
\end{aligned}
$$

- Implementare la funzione con un metodo ricorsivo: codice Java in Ex1.java

# Ricorsione VS Iterazione

- É sempre possibile trovare un corrispondente iterativo di un programma ricorsivo;
- Un algoritmo iterativo sará piú veloce di uno ricorsivo, a causa delle sovrastrutture come le chiamate alle funzioni e la ripetuta registrazione delle stack;
- Gli algoritmi ricorsivi sono prevalentemente usati per risolvere problemi complicati quando la loro implementazione é più naturale e facile.

# Ricorsione VS Iterazione

Diversa visione dello stesso problema:

Una giovane donna o una vecchia?

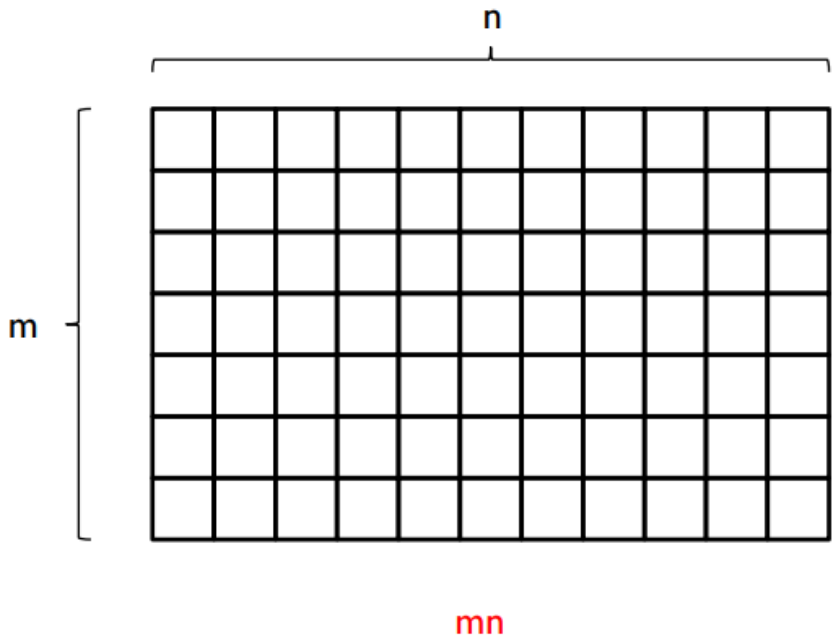
# Ricorsione VS Iterazione

Diversa visione dello stesso problema:

Un indiano o un eschimese?

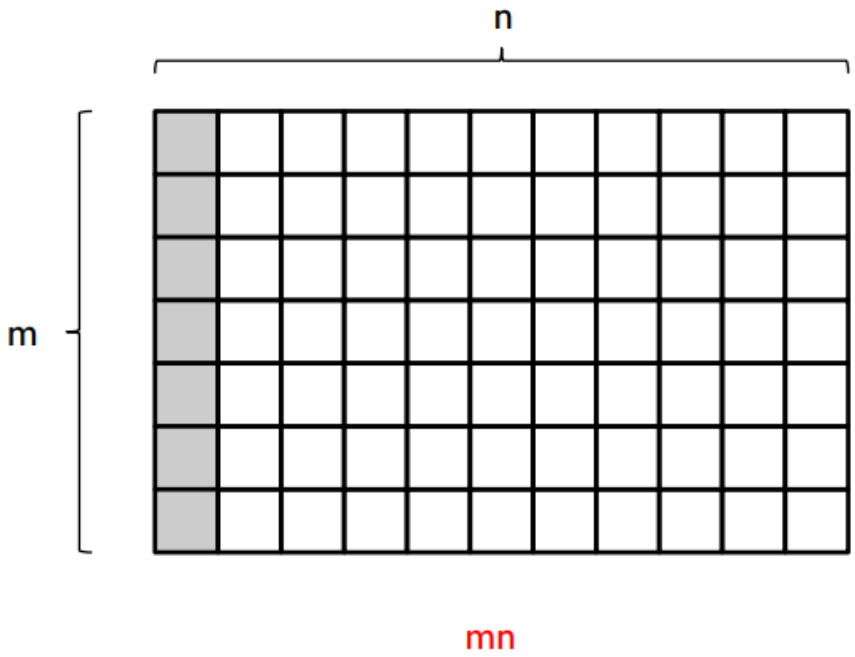
# Ricorsione VS Iterazione

Moltiplicazione con la ricorsione:



# Ricorsione VS Iterazione

Moltiplicazione con la ricorsione:



# Ricorsione VS Iterazione

Moltiplicazione con la ricorsione: