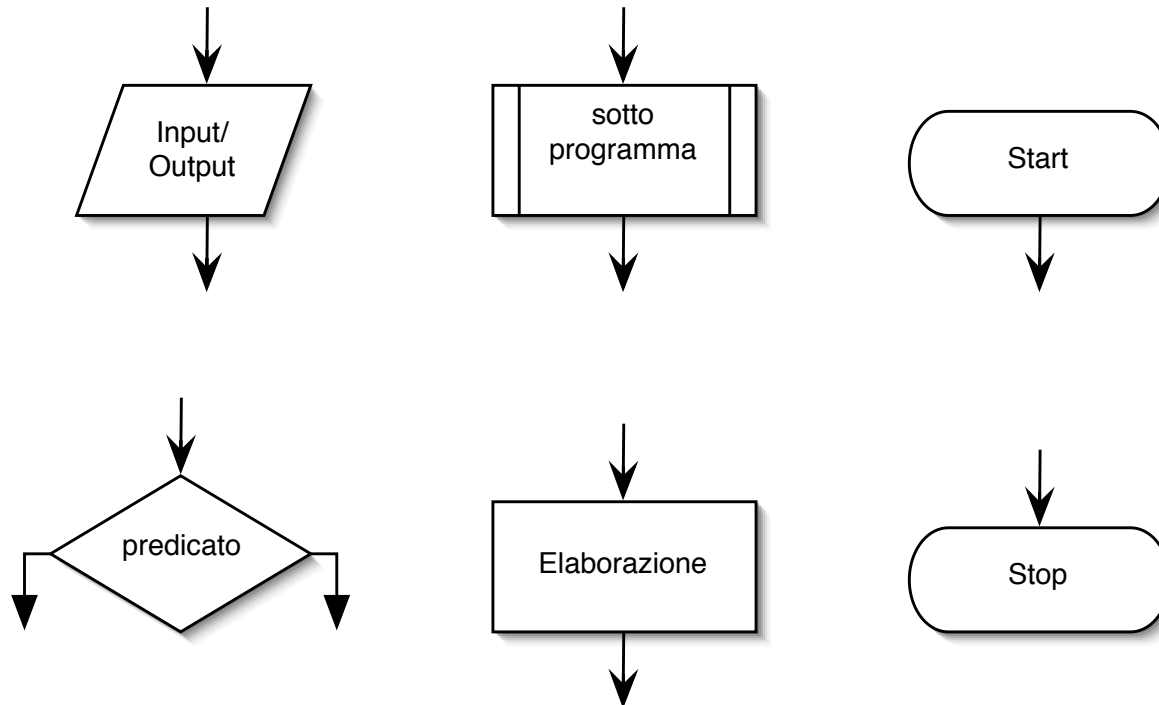


# Diagrammi di flusso

- Un metodo per rappresentare graficamente gli algoritmi.



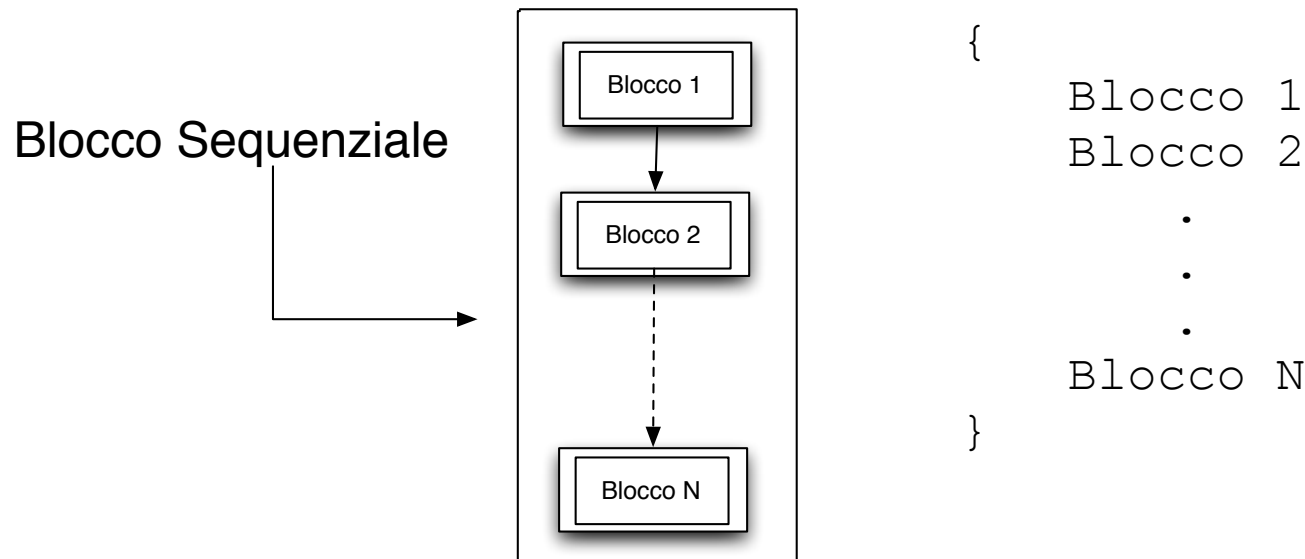
# La programmazione strutturata

Un algoritmo è strutturato in blocchi di istruzioni.

- Vantaggio: maggior leggibilità e pulizia del codice.
- Blocchi: sequenziali, condizionali, iterativi.
- Le istruzioni di input/output e di assegnamento possono essere considerate come dei blocchi sequenziali atomici.

# Blocco sequenziale

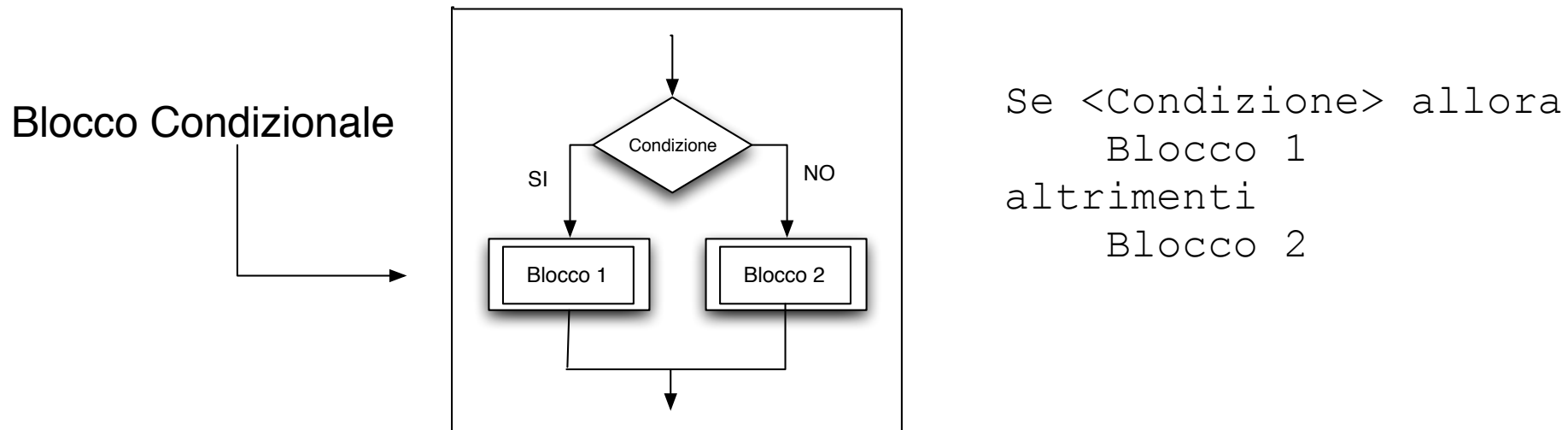
- E' costituito da una sequenza di (sotto-)blocchi che saranno eseguiti sequenzialmente.



**Blocco i**,  $i=1,..N$ , può essere un blocco sequenziale, condizionale, iterativo

# Blocco condizionale

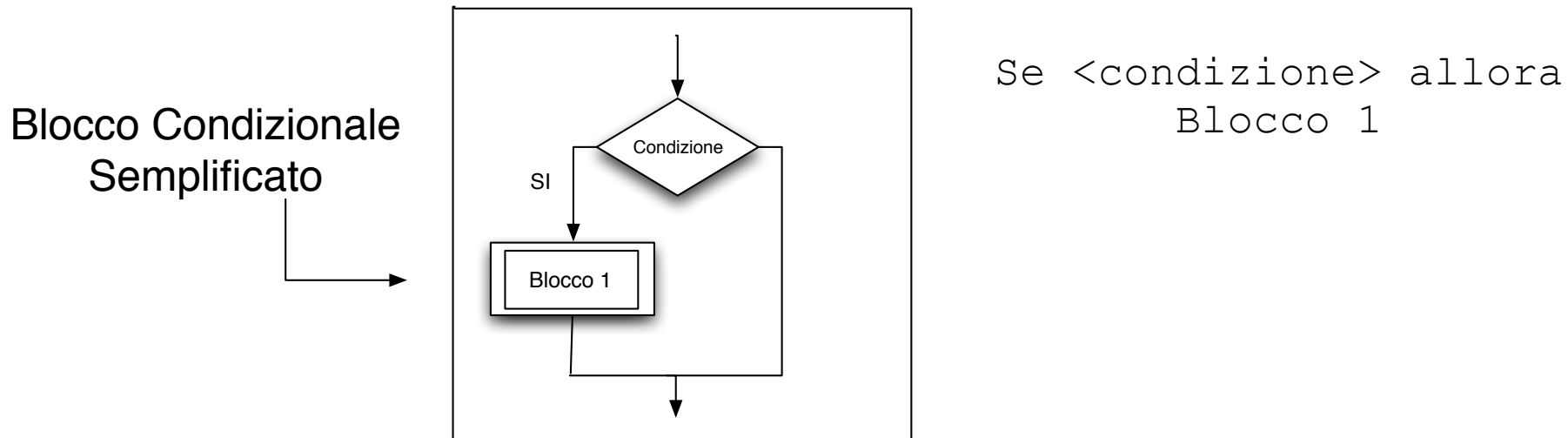
- Il blocco condizionale permette di scegliere tra due azioni in base a una condizione logica.



**Blocco i**,  $i=1,..2$ , può essere un blocco sequenziale, condizionale, iterativo

# Blocco condizionale (semplificato)

- Tale blocco permette di decidere di eseguire un'azione in base a una condizione logica.

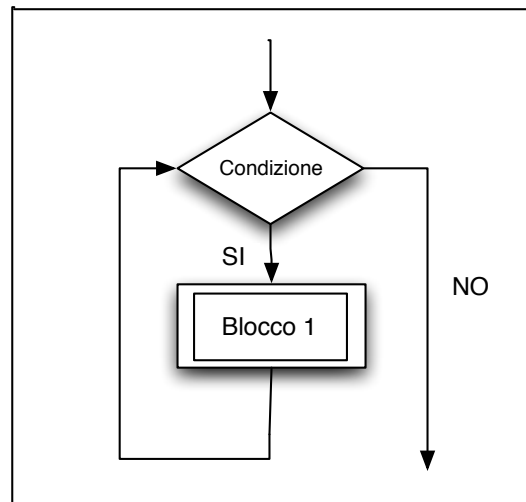


**Blocco 1** può essere un blocco sequenziale, condizionale, iterativo

# Blocco iterativo (controllo in testa)

- Il blocco iterativo con controllo in testa permette di ripetere un blocco di istruzioni finché la condizione logica rimane vera

Blocco Iterativo  
(controllo in testa)



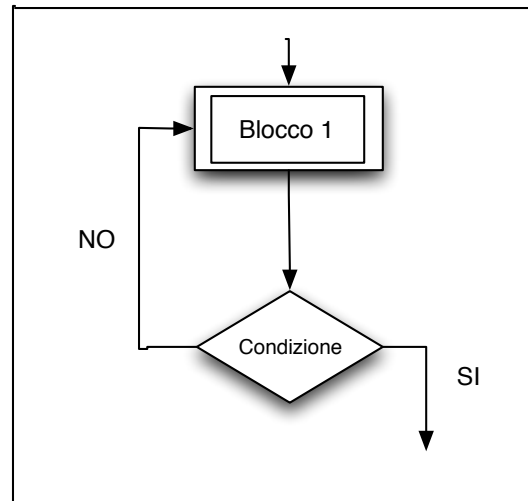
Mentre <condizione> fai  
Blocco 1

**Blocco 1** può essere un blocco sequenziale, condizionale, iterativo

# Blocco iterativo (controllo in coda)

- Permette di eseguire almeno una volta il blocco contenuto nel ciclo. Si ripete il blocco finché la condizione logica viene verificata.

Blocco Iterativo  
(controllo in coda)



```
Ripeti  
    Blocco 1  
finché <condizione>
```

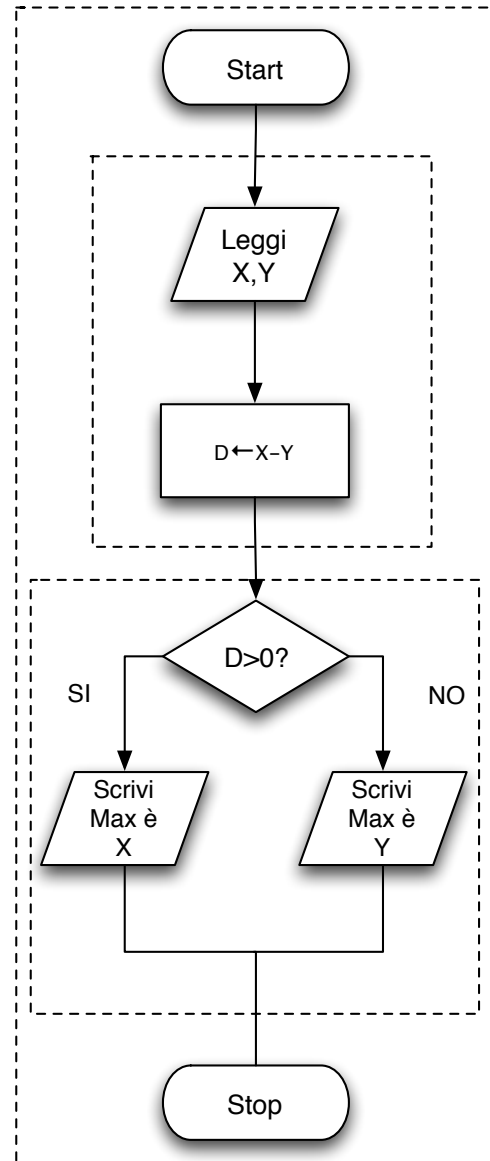
**Blocco 1** può essere un blocco sequenziale, condizionale, iterativo

# Esempio di algoritmo

## Calcolo del massimo fra 2 numeri

1. Leggi un numero dall'esterno e mettilo nella variabile X.
2. Leggi un numero dall'esterno e mettilo nella variabile Y.
3. Calcola la differenza d fra x e y ( $d \leftarrow X - Y$ )
  1. Se ( $d > 0$ ) allora stampa "il massimo è X", altrimenti stampa "il massimo è Y".

# Diagramma di Flusso



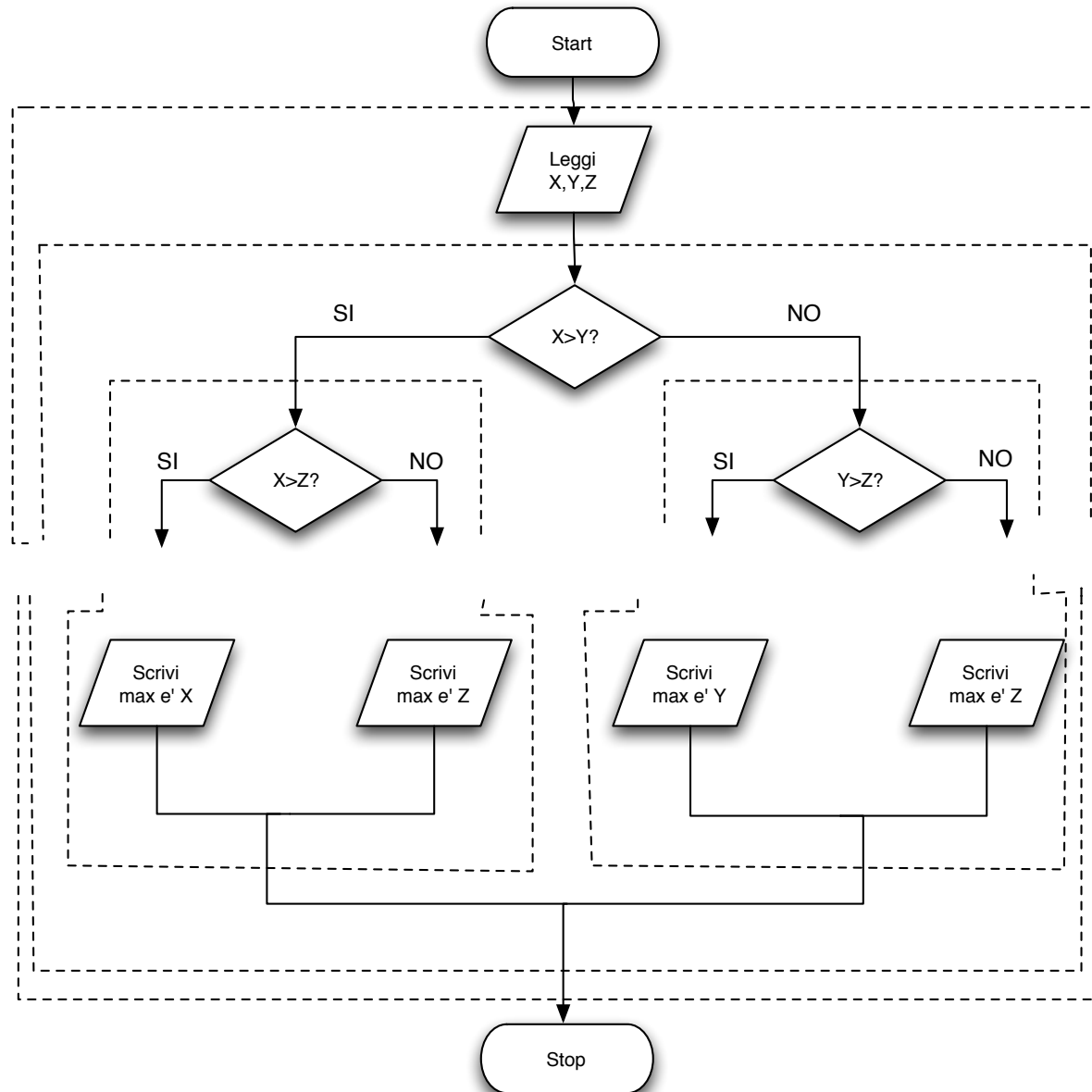
# Max fra 3 numeri

Input: tre numeri distinti

Output: il massimo fra i tre numeri

```
1. Leggi X,Y,Z
2. Se (X>Y) allora
    2.1. Se (X>Z) allora
        2.1.1. Stampa "il max e' X"
        altrimenti
            2.1.2. Stampa "il max e' Z"
    altrimenti
        2.2. Se (Y>Z) allora
            2.2.1. Stampa "il max e' Y"
            altrimenti
                2.2.2. Stampa "il max e' Z"
```

# Diagramma di Flusso



# Somma di n numeri

Input: un naturale n e n naturali

Output: la somma degli n naturali

```
1. leggi n
2. sum ← 0
3. mentre (n>0) fai {
    3.1 leggi x
    3.2 sum ← sum + x
    3.3 n ← n-1
}
4. stampa sum
```

# Esercizio

Calcolare la media aritmetica di n naturali

$$\text{Media} = \frac{(X_1 + X_2 + X_3 + \dots + X_n)}{n}$$

# Sottoprogrammi

- Per calcolare la media aritmetica fra  $n$  naturali sarebbe comodo riutilizzare il codice per la somma di  $n$  naturali.
- I sottoprogrammi permettono di scrivere codice
  - modulare
  - leggibile
  - riutilizzabile

# Sottoprogrammi (definizione)

- In un sottoprogramma è necessario definire quali sono i parametri di input e i parametri di output (definizione dell'interfaccia)
- Tali parametri sono chiamati parametri formali.

```
Sottoprogramma <nome sottoprg> (in: x1, x2, ... xn;  
                                out: y1, y2, ... ym)
```

Blocco

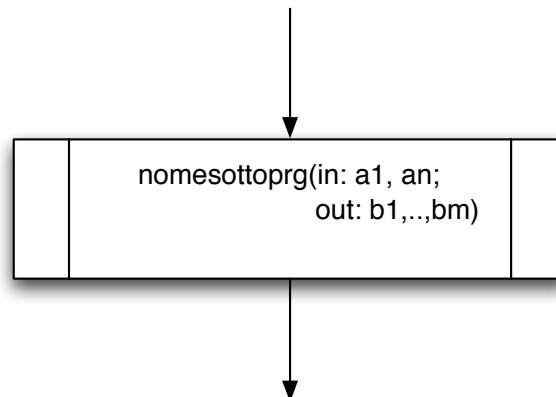
# Esempio

```
sottoprogramma somma (in:n; out: sum)
{
1. sum ← 0
2. mentre (n>0) fai {
    2.1 leggi x
    2.2 sum ← sum + x
    2.3 n ← n-1
    }
}
```

# Sottoprogrammi (chiamata)

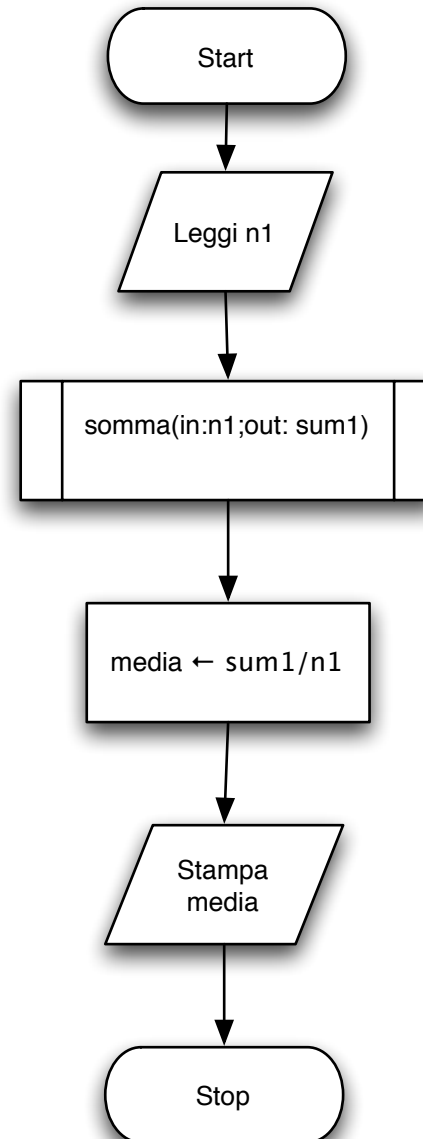
- Un sottoprogramma per essere eseguito deve essere “chiamato” da un’altro programma che lo utilizza esplicitando la lista di parametri di input e output (parametri attuali).
- i nomi dei parametri formali e attuali possono essere distinti. Al momento della chiamata vengono messi in corrispondenza in base all’ordine in cui sono stati dichiarati.

`nomesottoprg(in:a1,...,an; out:b1,...,bm)`



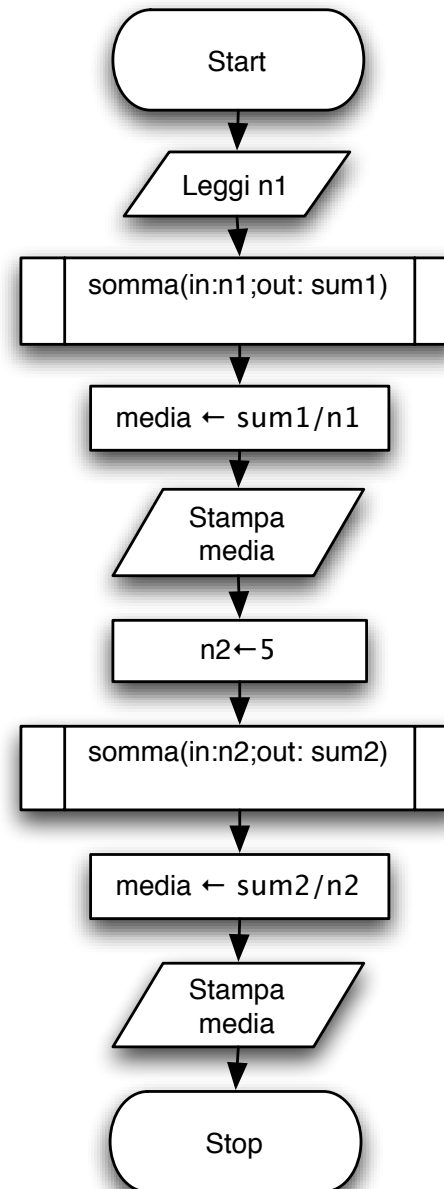
# Media aritmetica

```
1. leggi n1  
2. somma(in:n1,out: sum1)  
3. media ← sum1/n1  
4. stampa media
```

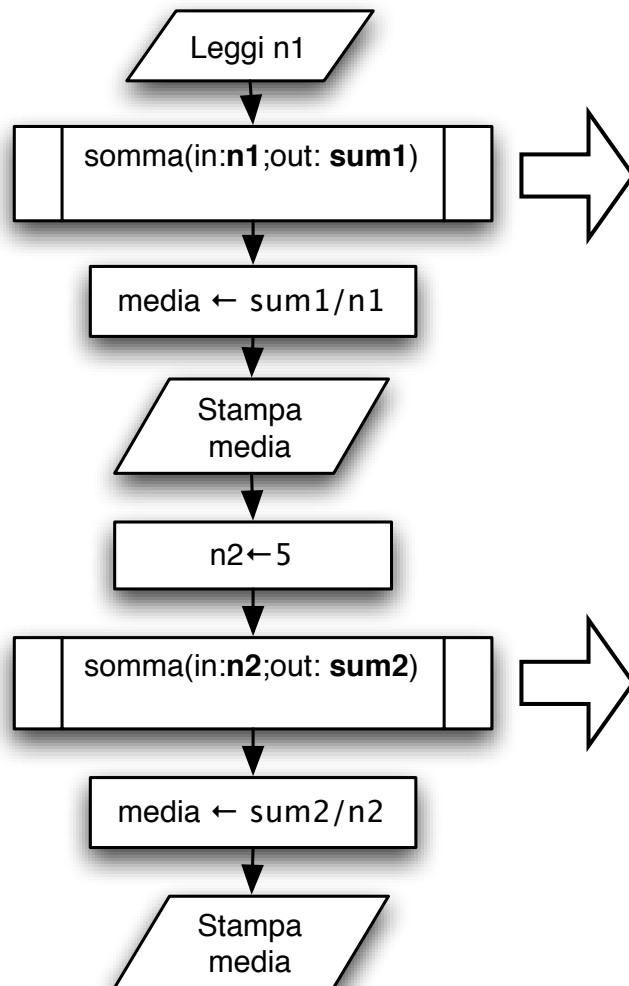


# Media aritmetica

```
1. leggi n1
2. somma(in:n1;out: sum1)
3. media ← sum1/n1
4. stampa media
5. n2 ← 5
6. somma(in:n2;out: sum2)
7. media ← sum2/n2
8. stampa media
```



# Media aritmetica



```
sottoprogramma somma (in:n; out: sum)  
{  
  1. sum ← 0  
  2. mentre (n>0) fai {  
    2.1 leggi x  
    2.2 sum ← sum + x  
    2.3 n ← n-1  
  }  
}
```

```
sottoprogramma somma (in:n; out: sum)  
{  
  1. sum ← 0  
  2. mentre (n>0) fai {  
    2.1 leggi x  
    2.2 sum ← sum + x  
    2.3 n ← n-1  
  }  
}
```