



## Formalismi per la descrizione di algoritmi

---

- Per descrivere in passi di un algoritmo bisogna essere precisi e non ambigui
- Il linguaggio naturale degli esseri umani si presta a interpretazioni non univoche
- Si usano due formalismi:
  - diagrammi di flusso: formalismo grafico
  - pseudo-codice: linguaggio con istruzioni simili a quelle dei linguaggi di programmazione
- si possono usare in alternativa



## Specifica di inizio e di fine

---

- Ogni algoritmo deve avere un inizio ed una fine

start



end

**start**

.....

.....

.....

.....

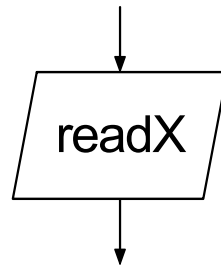
.....

.....

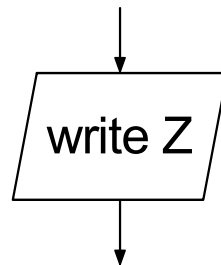
**end**

# Input ed output di dati

- Ogni algoritmo parte da dati in ingresso per produrre dati in uscita (problema computazionale)



.....  
read X  
.....



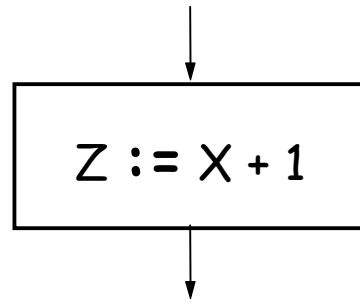
.....  
write Z  
.....



## Specifica delle azioni

---

- Ogni algoritmo specifica azioni che l'esecutore deve compiere del tipo descritto in precedenza



.....  
Z := X + 1  
.....

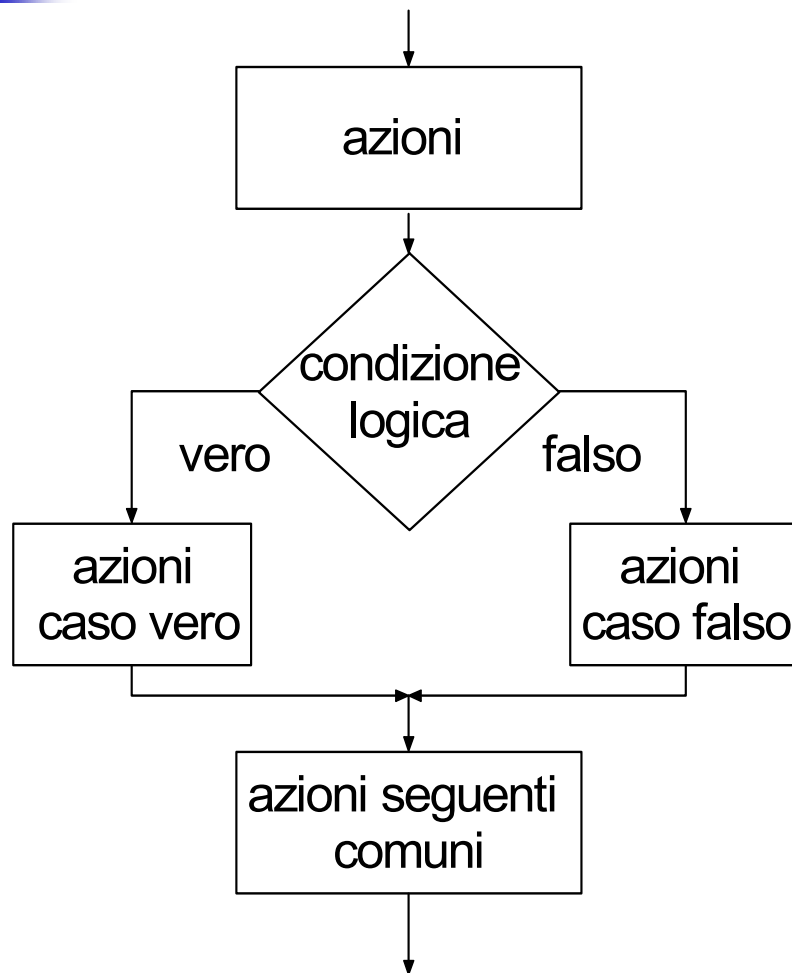


## Specifica delle condizioni logiche

---

- Alcuni passi di un algoritmo si devono eseguire se sono verificate condizioni logiche su valori di variabili
- In genere, questi *salti* dell'algoritmo sono sottoposti ad una *condizione logica* (risposta *vero* o *falso*);
- Si parla di *blocchi decisionali*

# Specifica delle condizioni logiche



.....

azioni

**if** (condizione logica) **then**

    azioni caso vero

**else**

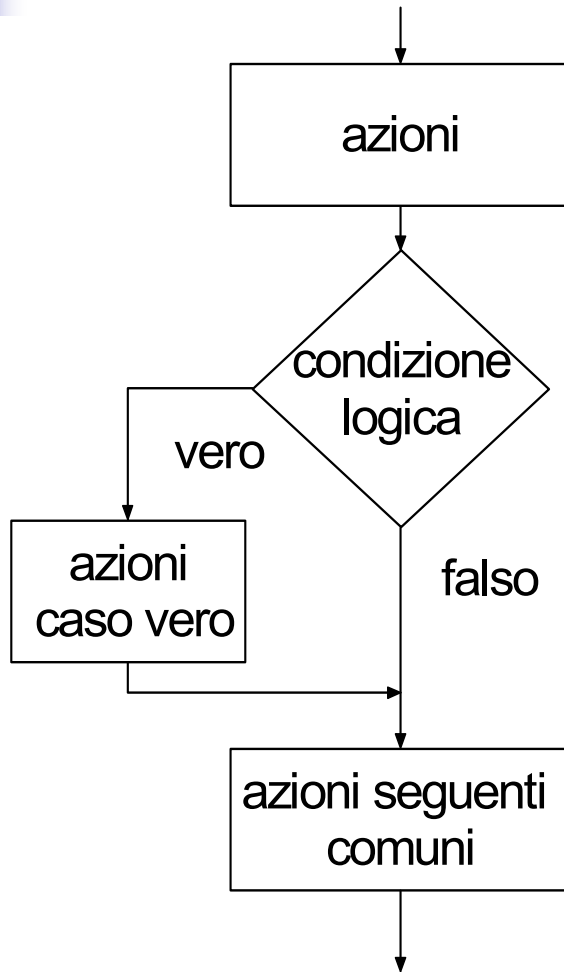
    azioni caso falso

**end if**

azioni seguenti comuni

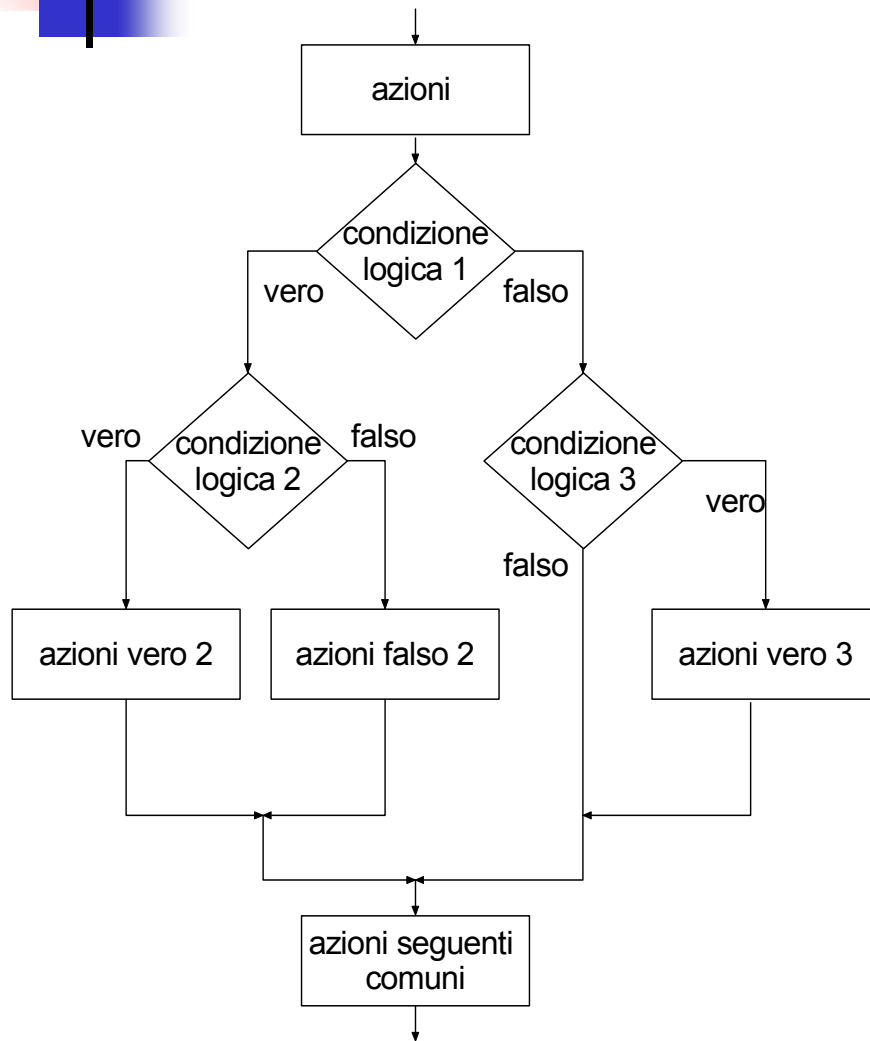
.....

# Specifica delle condizioni logiche



.....  
azioni  
**if** (condizione logica) **then**  
    azioni caso vero  
**end if**  
azioni seguenti comuni  
.....

# Condizioni logiche Annidate: esempio



```
.....  
azioni  
if (condizione logica 1) then  
    if (condizione logica 2) then  
        azioni vero 2  
    else  
        azioni falso 2  
    end if  
else  
    if (condizione logica 3) then  
        azioni vero 3  
    end if  
end if  
azioni seguenti comuni  
.....
```

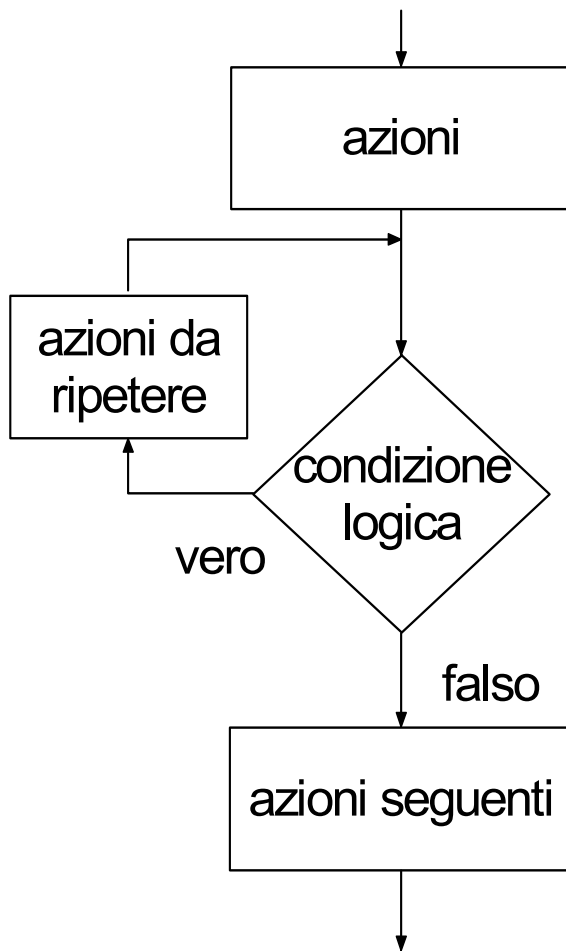


## Flusso di esecuzione

---

- I singoli diagrammi devono essere uniti tramite i *connettori* (linee e frecce in un flow chart);
- L'esecuzione dei passi deve essere fatta *sequenzialmente*, ovvero seguendo i connettori, partendo dall'inizio dell'algoritmo fino a raggiungere la sua fine
- Quando si scrive l'algoritmo bisogna fare molta attenzione alla direzione del flusso di esecuzione

# Strutture di controllo: iterazione



.....

azioni

**while** (condizione logica)

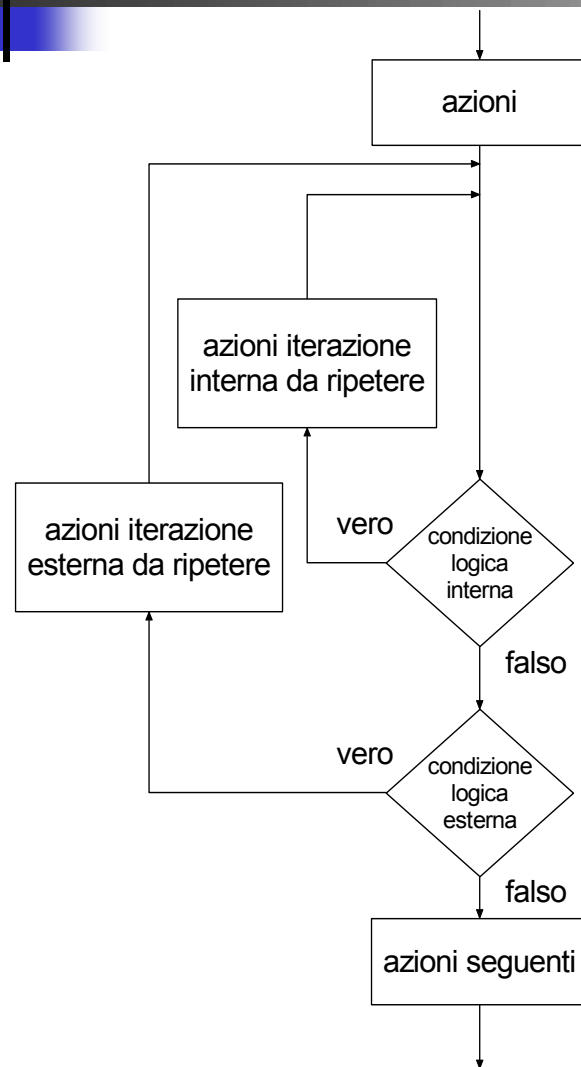
azioni da ripetere

**end while**

azioni seguenti

.....

# Strutture di controllo: iterazioni annidate



.....

azioni

**while** (condizione logica esterna)

**while** (condizione logica interna)

azioni iterazione interna da ripetere

**end while**

azioni iterazione esterna da ripetere

**end while**

azioni seguenti

.....

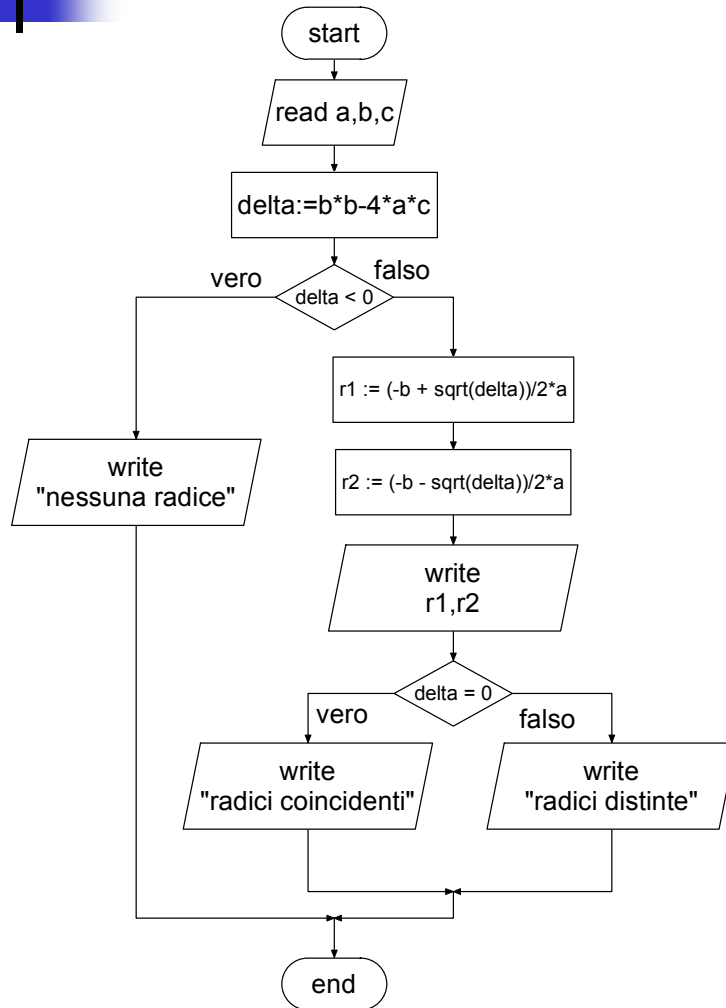


# Sviluppo di programmi

---

- FASE 1: Dare un nome al problema partendo dall'analisi del problema
- FASE 2: Scrivere la specifica funzionale
- **FASE 3: Scrittura dell'algoritmo**
  - FASE 3.1: Introduzione delle variabili (contenitori di dati) necessarie e delle relative operazioni elementari
  - FASE 3.2: Specifica di un diagramma di flusso ( o di uno pseudo codice) che descrive in modo preciso e non ambiguo la sequenza di operazioni da eseguire
- FASE 4: Traduzione del diagramma di flusso ( o dello pseudo codice) in un programma in un linguaggio di programmazione

# Esempio di algoritmo: calcolo radici



**start**

read a,b,c

delta := b\*b-4\*a\*c

**if** (delta < 0) **then**

write "nessuna radice"

**else**

r1 := (-b+sqrt(delta))/2\*a

r2 := (-b-sqrt(delta))/2\*a

write r1,r2

**if** (delta = 0) **then**

write "radici coincidenti"

**else**

write "radici distinte"

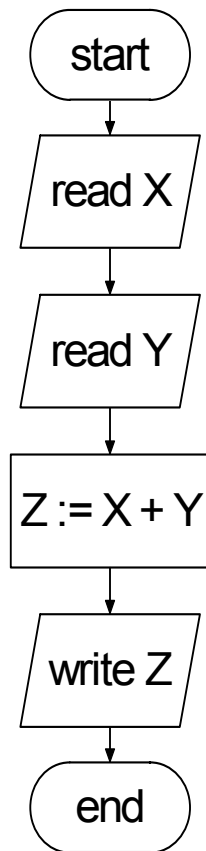
**end if**

**end if**

**end**

## Esempio di algoritmo

- Scrivere l'algoritmo che esegue la somma di due numeri



**start**

read X

read Y

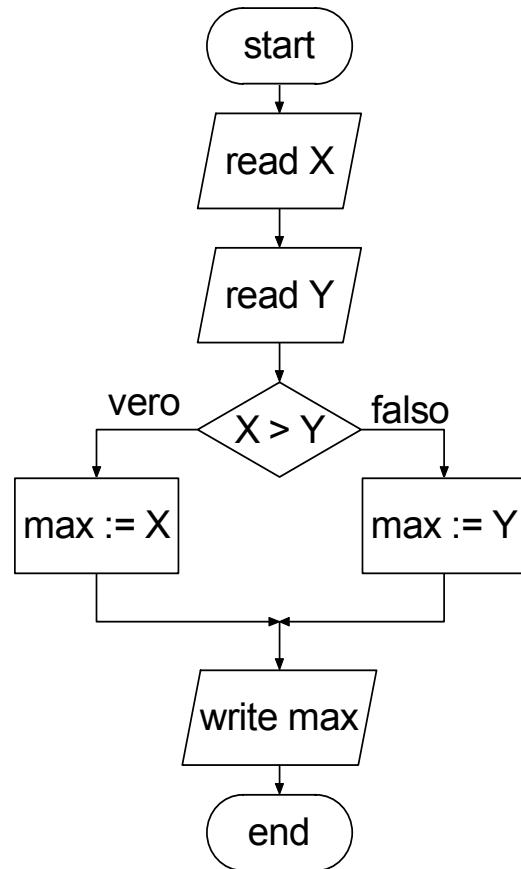
Z := X + Y

write Z

**end**

## Esempio: massimo tra due numeri

Dati due numeri, dire qual è il massimo tra i due.



**start**

read X

read Y

**if** (X > Y) **then**

max := X

**else**

max := Y

**end if**

write max

**end**

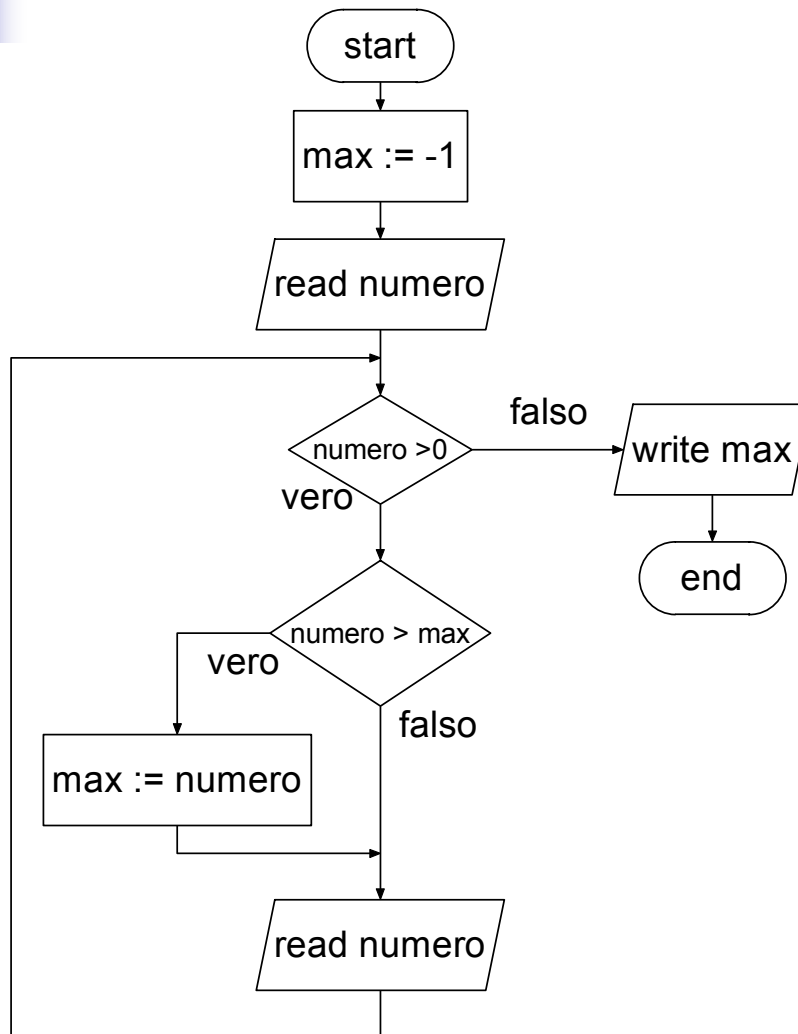


## Esercizio: massimo di una sequenza

---

- Si supponga di fornire in input ad un programma un numero indefinito di interi positivi. L'inserimento verrà terminato dall'utente quando questi inserirà uno zero (0). Il programma deve restituire il valore massimo tra quelli introdotti. Disegnare il diagramma di flusso di tale programma.

# Esercizio: massimo di una sequenza



**start**

max := -1

read numero

**while** (numero > 0)

**if** (numero > max) **then**

    max := numero

**end if**

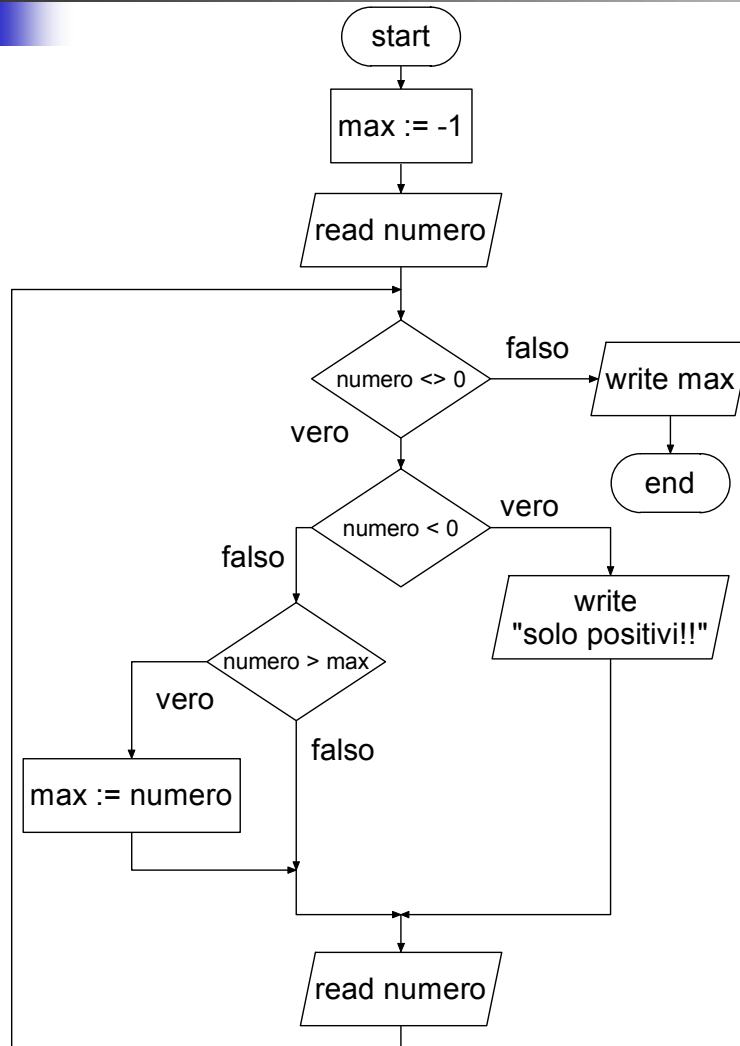
  read numero

**end while**

write max

**end**

# Esercizio: massimo di una sequenza



**start**

max := -1

read numero

**while** (numero <> 0)

**if** (numero < 0) **then**

write "solo positivi!!"

**else**

**if** (numero > max) **then**

max := numero

**end if**

**end if**

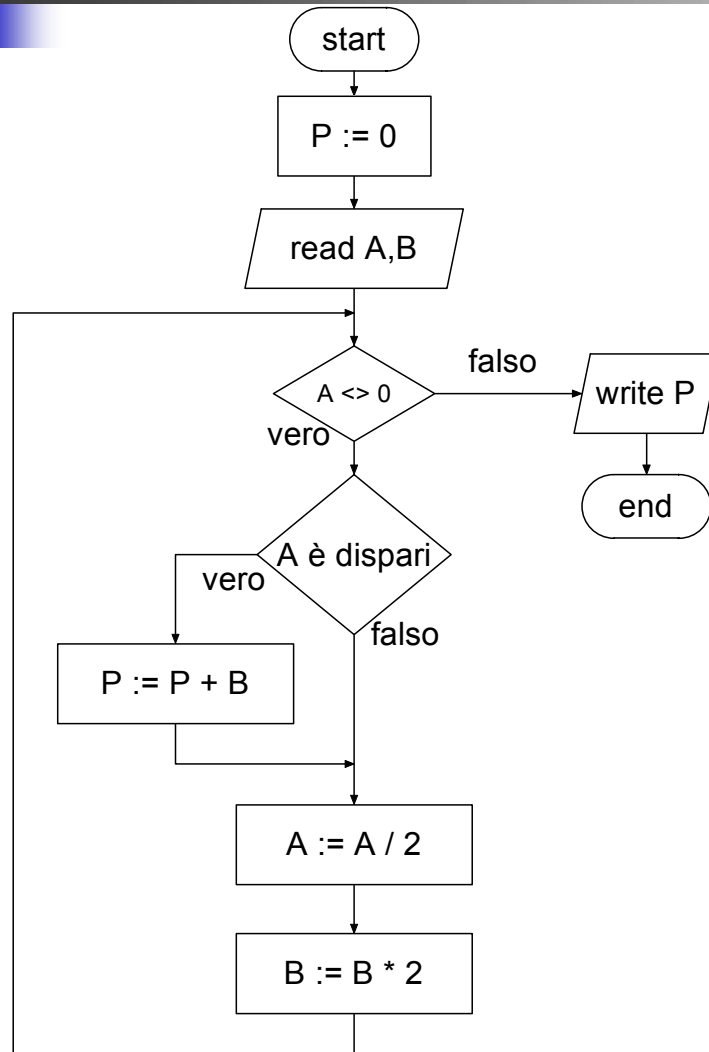
read numero

**end while**

write max

**end**

# Esercizio: cosa fa questo algoritmo?



**start**

$P := 0$

read A,B

**while** ( $A \neq 0$ )

**if** (A è dispari) **then**

$P := P + B$

**end if**

$A := A / 2$

$B := B * 2$

**end while**

write P

**end**