

PREVIOUSLY, on Tecniche della Programmazione

Problemi e loro “progettazione di soluzione” mediante algoritmi.

Programma elementare ... con DICHIARAZIONE,
ASSEGNAZIONE, INPUT e OUTPUT (I/O)

Qui vediamo

- Algoritmi
- Definizione / rappresentazione di algoritmi
- Programmazione Strutturata



Algoritmo (again ...)

Abbiamo detto che e' una sequenza di PASSI, operazioni, istruzioni ...
per risolvere un problema per il quale abbiamo una formalizzazione matematica

Più schematicamente ci sono delle componenti da considerare

- **INFORMAZIONI** relative al problema: rappresentate come **DATI**
 (nel calcolatore, nell'algoritmo, nel programma)
- **INPUT** ...
- **Procedura Computazionale** passi di calcolo ...
- **OUTPUT** ...

Progettazione Algoritmo: attività` creativa ...
Esecuzione Algoritmo: attività` meccanica



Algoritmo (again ...)

Abbiamo detto che e' una sequenza di PASSI, operazioni, istruzioni ...
per risolvere un problema per il quale abbiamo una formalizzazione matematica

Più schematicamente ci sono delle componenti da considerare

- **INFORMAZIONI** relative al problema: rappresentate come **DATI**
 (nel calcolatore, nell'algoritmo, nel programma)
- **INPUT** i dati, che distinguono un'**istanza** da un'altra del problema,
 e che sono da usare nel calcolare la soluzione
- **Procedura Computazionale** passi di calcolo ...
- **OUTPUT** ...

Progettazione Algoritmo: attività` creativa ...
Esecuzione Algoritmo: attività` meccanica



Algoritmo (again ...)

Abbiamo detto che e' una sequenza di PASSI, operazioni, istruzioni ... per risolvere un problema per il quale abbiamo una formalizzazione matematica

Più schematicamente ci sono delle componenti da considerare

- **INFORMAZIONI** relative al problema : rappresentate come **DATI** (nel calcolatore, nell'algoritmo, nel programma)
- **INPUT** i dati, che distinguono un'istanza da un'altra del problema, e che sono da usare nel calcolare la soluzione
- **Procedura Computazionale** questa e' la sequenza di passi ...
 - 1) operazioni su dati di input
 - 2) operazioni su dati intermedi (ottenuti con calcoli su dati di input e altri dati intermedi)
 - 3) Produzione di dati di OUTPUT
- **OUTPUT**



Algoritmo (again ...)

Abbiamo detto che e' una sequenza di PASSI, operazioni, istruzioni ...
per risolvere un problema per il quale abbiamo una formalizzazione matematica
Più schematicamente ci sono delle componenti da considerare

- **INFORMAZIONI** relative al problema : rappresentate come **DATI**
(nel calcolatore, nell'algoritmo, nel programma)
- **INPUT** i dati, che distinguono un'istanza da un'altra del problema,
e che sono da usare nel calcolare la soluzione
- **Procedura Computazionale** questa e' la sequenza di passi ...
 - 1) operazioni su dati di input
 - 2) operazioni su dati intermedi (ottenuti con calcoli su dati di input e altri dati intermedi)
 - 3) Produzione di dati di **OUTPUT**
- **OUTPUT** i dati emessi a valle della procedura computazionale,
interpretabili dall'utente come informazioni sulla soluzione
dell'istanza del problema

Torniamo ad ourProgram.c

- 0) Servono primoNumero e secondo e ris (struttura dati ...)
- 1) Assegna primoNumero con 68
- 2) Assegna 64 a secondo
- 3) Assegna a ris il risultato di primoNumero+secondo
- 4) Stampa ris
- 5) Fine

Versione con INPUT

- 0) (primoNum, secondoNum, ris ... i dati che usiamo)
- 1) **INPUT** primoNum, secondoNum
- 2) $\text{ris} = \text{primoNum} + \text{secondoNum}$
- 3) **OUTPUT** ris
- 4) Fine

NB quando l'algoritmo viene "eseguito" la sequenza di passi che vengono eseguiti denota quell che si chiama "Flusso di Esecuzione"

"Esecuzione dell'algoritmo" (*Flusso di esecuzione*)

- 0) primoNumero, secondo, ris ...
- 1) **Assegna primoNumero con 68**
- 2) **Assegna 64 a secondo**
- 3) **Assegna a ris il risultato di primoNumero+secondo**
- 4) **Stampa ris**
- 5) **Fine**

tutte le esecuzioni sono uguali ... i risultati non cambiano da istanza ad istanza ... e nemmeno il flusso

Flusso di esecuzione: 1) 2) 3) 4) 5)

Versione con INPUT

- 0) (primoNum, secondoNum, ris ... i dati che usiamo)
- 1) **INPUT primoNum, secondoNum**
- 2) **ris = primoNum + secondoNum**
- 3) **OUTPUT ris**
- 4) **Fine**

esecuzioni diverse, a seconda dell'INPUT, ma comunque la sequenza di passi e` sempre la medesima ...

Flusso di esecuzione: 1) 2) 3) 4)

Diagrammi di Flusso

Anche *Diagrammi a Blocchi*.

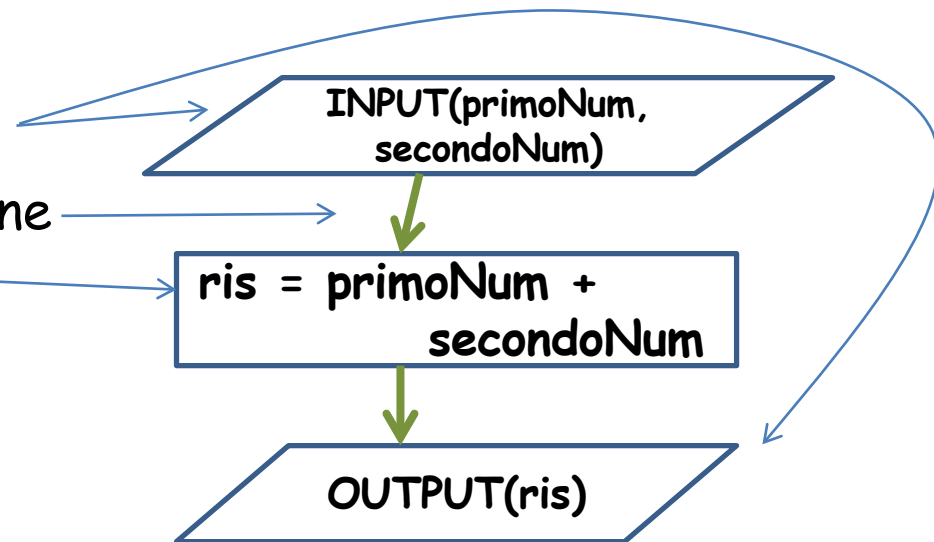
Un formalismo grafico per rappresentare algoritmi
(e seguire il flusso di esecuzione)

Istruzioni di INPUT/ OUTPUT

Direzione del Flusso di esecuzione

Istruzione generica

- 1) INPUT primoNum, secondoNum
- 2) $ris = primoNum + secondoNum$
- 3) OUTPUT ris
- 4) Fine



Flusso di esecuzione: 1) → 2) → 3) → 4)

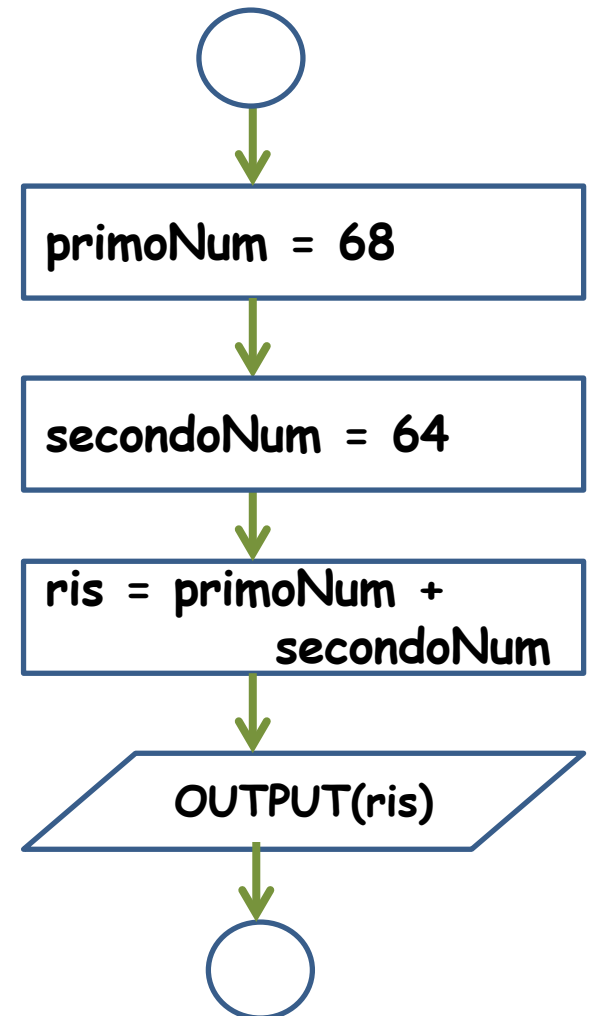
4) ? → STOP?

Diagrammi di Flusso

Anche *Diagrammi a Blocchi*. Un formalismo grafico per rappresentare algoritmi (e seguire il flusso di esecuzione)

- 1) Assegna primoNum con 68
- 2) Assegna 64 a secondoNum
- 3) Assegna a ris il risultato di primoNum+secondoNum
- 4) Stampa ris
- 5) Fine

Flusso di esecuzione: 1) → 2) → 3) → 4) → 5)



Diagrammi di Flusso

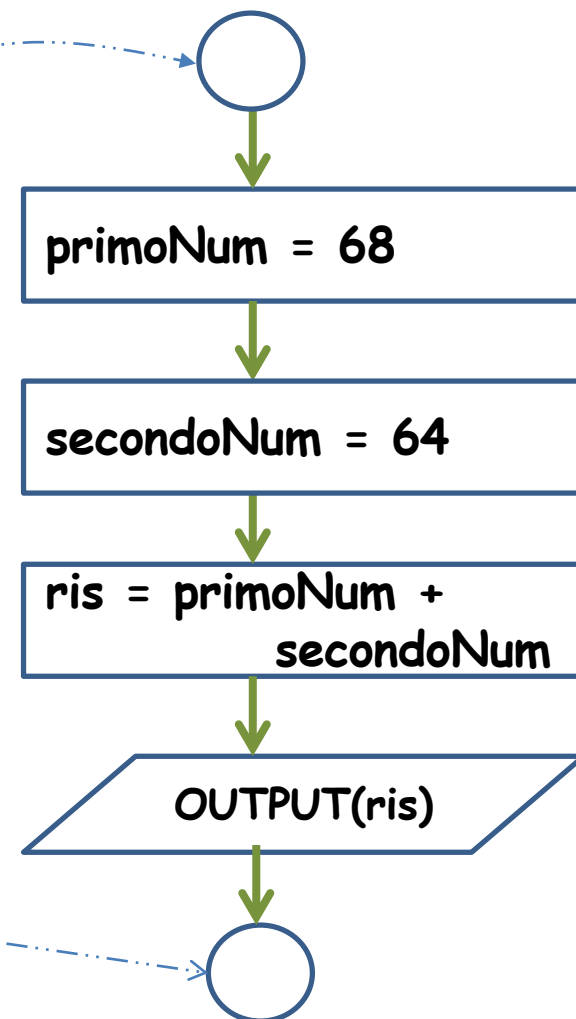
Anche *Diagrammi a Blocchi*. Un formalismo grafico per rappresentare algoritmi (e seguire il flusso di esecuzione)

- 1) Assegna primoNum con 68
- 2) Assegna 64 a secondoNum
- 3) Assegna a ris il risultato di primoNum+secondoNum
- 4) Stampa ris
- 5) Fine

Flusso di esecuzione: 1) → 2) → 3) → 4) → 5)

Questi pallocchi sono un vecchio modo per simboleggiare il punto di entrata e di uscita dal programma

Li ometteremo in seguito



Algoritmo per l'eq. di secondo grado

Dati i coefficienti a, b, c di un'equazione di secondo grado, calcolare le soluzioni reali, supponendo che esistano soluzioni reali.

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi Δ , ... $2 \cdot a, 4 \cdot a \cdot c$...

1) INPUT (a, b, c)

2) ☺

3) $x_1 = \frac{-b + \sqrt{\Delta}}{2 \cdot a}$

4) $x_2 = \frac{-b - \sqrt{\Delta}}{2 \cdot a}$

5) ☺

☺ Scrivere qualcosa al posto di ☺

Algoritmo per l'eq. di secondo grado

Dati i coefficienti a, b, c di un'equazione di secondo grado, calcolare le soluzioni reali, supponendo che esistano soluzioni reali.

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi Δ , $2a, 4ac$...

- 1) INPUT (a, b, c)
- 2) $\Delta = b^2 - 4ac$
- 3) x_1 = formula per la prima soluzione
- 4) x_2 = formula per la seconda soluzione
- 5) OUTPUT(x_1, x_2)

Algoritmo per l'eq. di secondo grado

Dati i coefficienti a, b, c di un'equazione di secondo grado, calcolare le soluzioni reali, supponendo che esistano soluzioni reali.

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi Δ , \dots $2 \cdot a, 4 \cdot a \cdot c \dots$

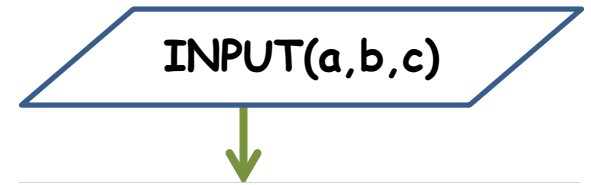
- 1) INPUT (a, b, c)
- 2) $\Delta = b^2 - 4 \cdot a \cdot c$
- 3) x_1 = formula per la prima soluzione
- 4) x_2 = formula per la seconda soluzione
- 5) OUTPUT(x_1, x_2)

☺ Diagramma di flusso? ☺

Algoritmo per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi Δ , $\dots 2*a, 4*a*c \dots$

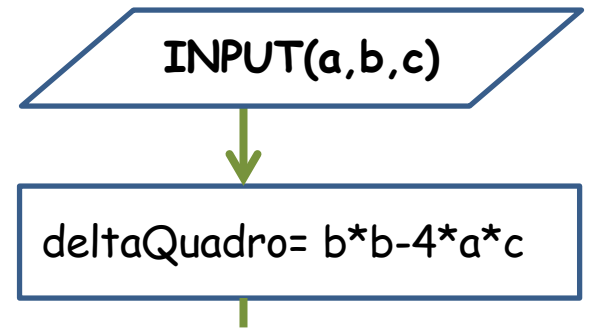
- 1) INPUT (a, b, c)
- 2) $\Delta = b*b - 4*a*c$
- 3) x_1 = formula per la prima soluzione
- 4) x_2 = formula per la seconda soluzione
- 5) OUTPUT(x_1, x_2)



Algoritmo per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi $\text{deltaquadro}, \dots 2*a, 4*a*c \dots$

- 1) INPUT (a, b, c)
- 2) $\text{deltaQuadro} = b*b - 4*a*c$
- 3) x_1 = formula per la prima soluzione
- 4) x_2 = formula per la seconda soluzione
- 5) OUTPUT(x_1, x_2)

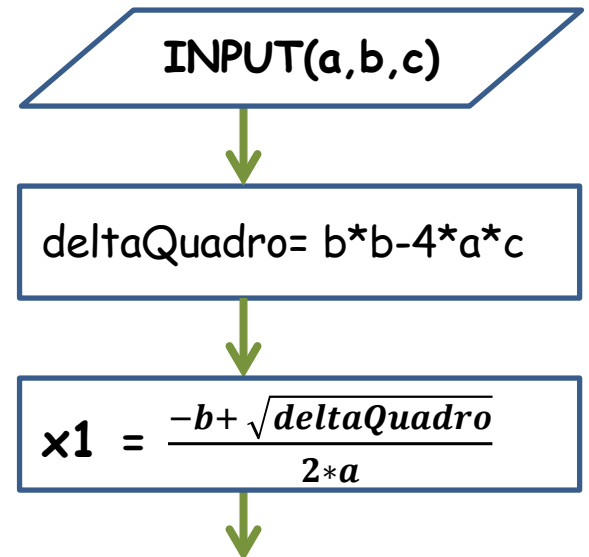


😊...

Algoritmo per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi deltaQuadro , ... $2*a, 4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaQuadro} = b*b - 4*a*c$
- 3) x_1 = formula per la prima soluzione
- 4) x_2 = formula per la seconda soluzione
- 5) OUTPUT(x_1, x_2)

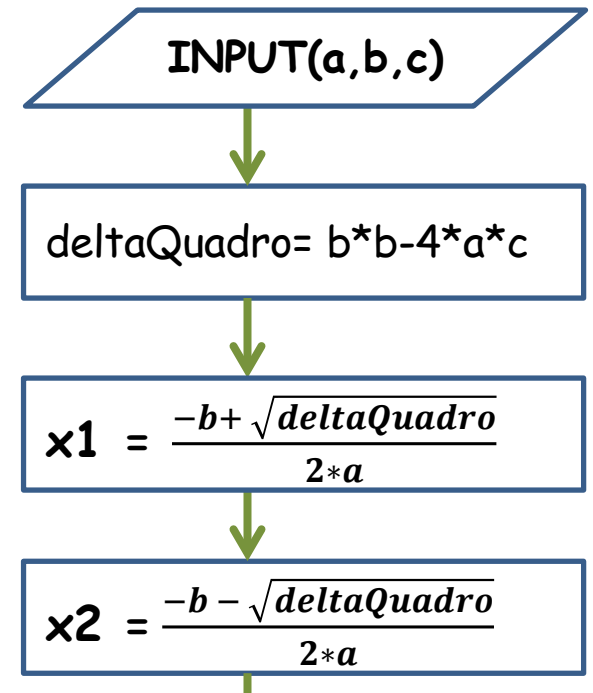


😊 ...

Algoritmo per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi deltaQuadro , ... $2*a, 4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaQuadro} = b*b - 4*a*c$
- 3) x_1 = formula per la prima soluzione
- 4) x_2 = formula per la seconda soluzione
- 5) OUTPUT(x_1, x_2)

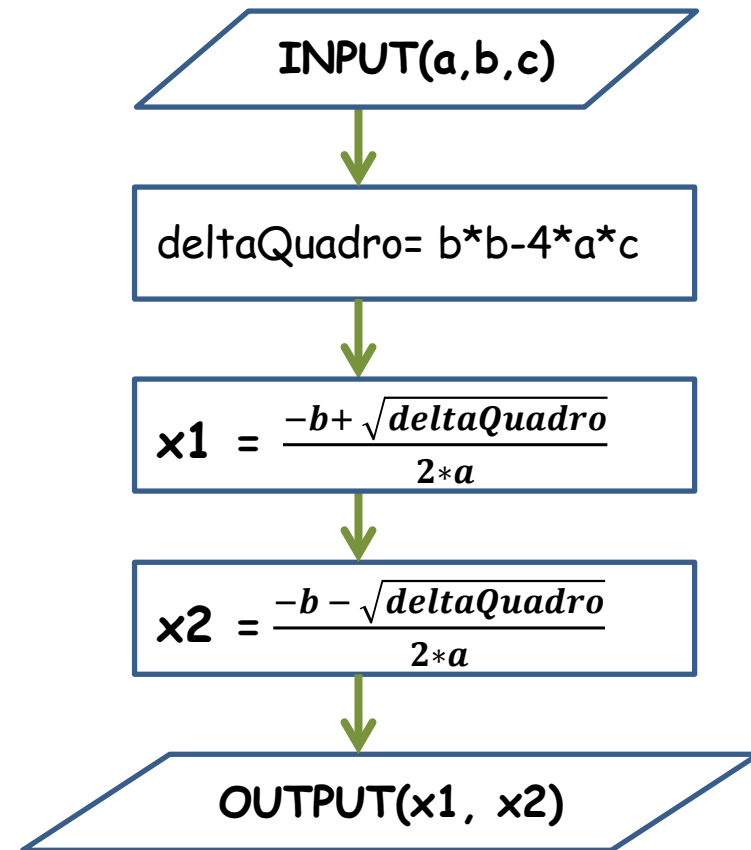


😊 ...

Algoritmo per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi deltaQuadro , ... $2*a, 4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaQuadro} = b*b - 4*a*c$
- 3) $x_1 =$ formula per la prima soluzione
- 4) $x_2 =$ formula per la seconda soluzione
- 5) OUTPUT(x_1, x_2)



☺ Flusso di esecuzione per $a=1, b=4, c=1$? ☺

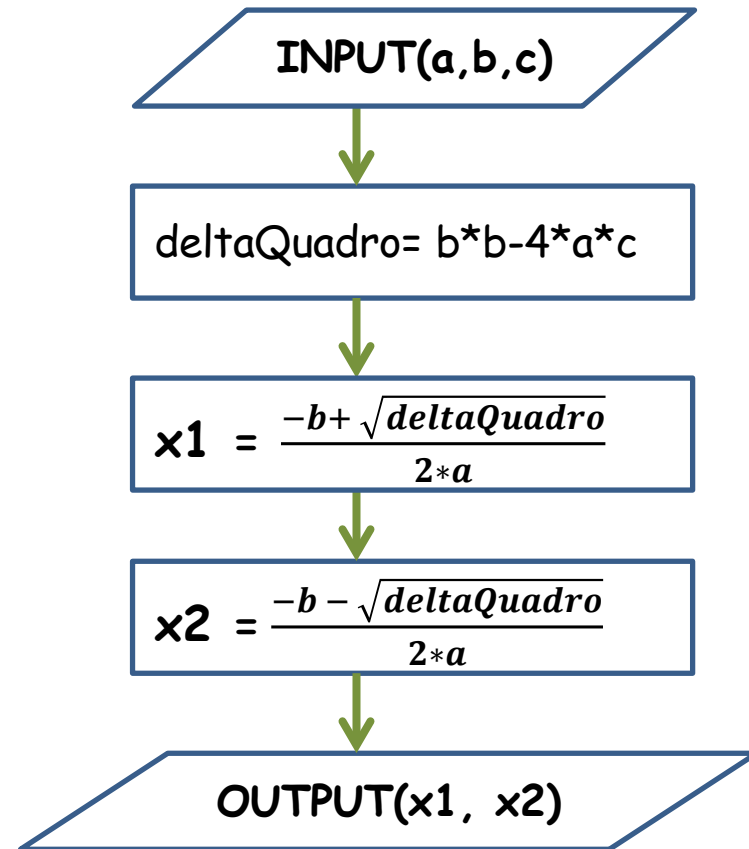
Algoritmo per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi $\text{deltaQuadro}, \dots 2*a, 4*a*c \dots$

- 1) INPUT (a, b, c)
- 2) $\text{deltaQuadro} = b*b - 4*a*c$
- 3) $x_1 =$ formula per la prima soluzione
- 4) $x_2 =$ formula per la seconda soluzione
- 5) OUTPUT(x_1, x_2)

Flusso di esecuzione per $a=1, b=4, c=1$:

1) \rightarrow 2) \rightarrow 3) \rightarrow 4) \rightarrow 5)



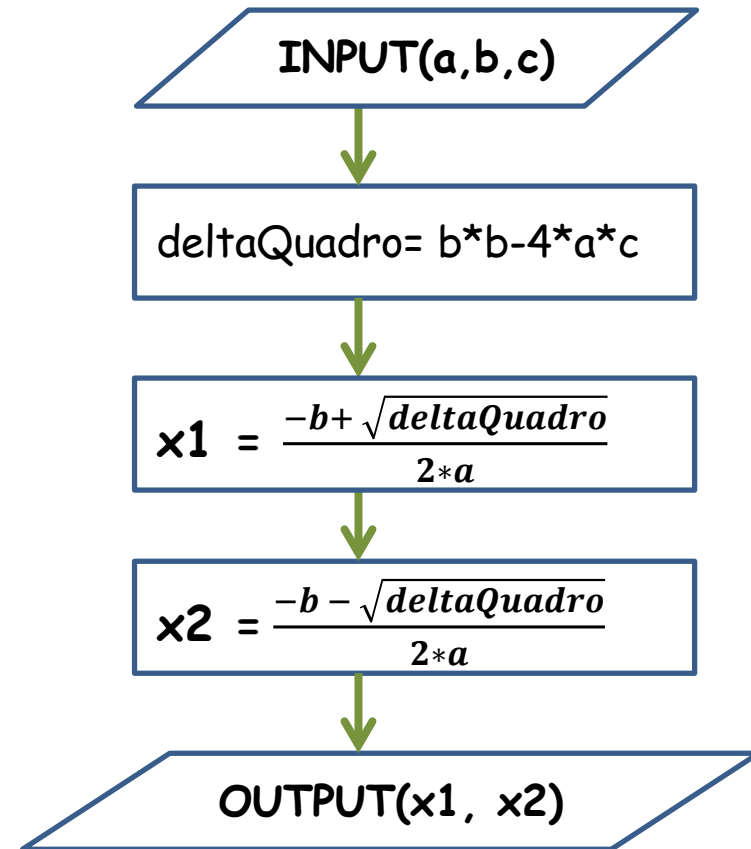
Algoritmo per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi $\text{deltaQuadro}, \dots, 2*a, 4*a*c \dots$

- 1) INPUT (a, b, c)
- 2) $\text{deltaQuadro} = b*b - 4*a*c$
- 3) $x_1 =$ formula per la prima soluzione
- 4) $x_2 =$ formula per la seconda soluzione
- 5) OUTPUT(x_1, x_2)

Flusso di esecuzione per $a=1, b=4, c=1$:

1) \rightarrow 2) \rightarrow 3) \rightarrow 4) \rightarrow 5)



(in realta` , per qualunque istanza il flusso e` sempre quello ...)

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comun Divisore.

BTW, $\text{MCD}(9,81)=9$, $\text{MCD}(37,7)=1$, $\text{MCD}(6,4)=2$;

$\text{MCD}(6,4)=?$

cosa sappiamo di certo su questo numero?

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comun Divisore.

BTW, $\text{MCD}(9,81)=9$, $\text{MCD}(37,7)=1$, $\text{MCD}(6,4)=2$;

$\text{MCD}(6,4)=?$

Sicuramente è un divisore di 4 (non può essere più grande di 4; o è 4 o è un numero più piccolo, al limite 1)

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comune Divisore.
BTW, $MCD(9,81)=9$, $MCD(37,7)=1$, $MCD(6,4)=2$;

$MCD(6,4)=?$

- 4 divide sia 6 che 4? No
- 3? No
- 2? Si'!!

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comune Divisore.
BTW, $MCD(9,81)=9$, $MCD(37,7)=1$, $MCD(6,4)=2$;

$MCD(6,4)=?$

Sicuramente è un divisore di 4 (non può essere più grande di 4; o è 4 o è un numero più piccolo, al limite 1)

- 4 divide sia 6 che 4? No
- 3? No
- 2? Sì!!

0) i dati: n , m (input) e ris (con cui calcoliamo il MCD)

1) INPUT (n , m)

2) ris = 😊

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comune Divisore.
BTW, $MCD(9,81)=9$, $MCD(37,7)=1$, $MCD(6,4)=2$;

$MCD(6,4)=?$

Sicuramente e' un divisore di 4 (non puo' essere piu' grande di 4; o e' 4 o e' un numero piu' piccolo, al limite 1)

- 4 divide sia 6 che 4? No
- 3? No
- 2? Si'!!

0) i dati: n , m (input) e ris (con cui calcoliamo il MCD)

1) INPUT (n , m)

2) $ris = \text{😊}$

3) SE ris «DIVIDE n » E «DIVIDE m » EUREKA!

...

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comune Divisore.
BTW, $MCD(9,81)=9$, $MCD(37,7)=1$, $MCD(6,4)=2$;

$MCD(6,4)=?$

Sicuramente e' un divisore di 4 (non puo' essere piu' grande di 4; o e' 4 o e' un numero piu' piccolo, al limite 1)

- 4 divide sia 6 che 4? No
- 3? No
- 2? Si'!!

0) i dati: n , m (input) e ris (con cui calcoliamo il MCD)

1) INPUT (n , m)

2) $ris = \text{😊}$

3) SE ris «DIVIDE n » E «DIVIDE m » EUREKA!

4) $ris = \overset{\dots}{ris} - 1$

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comune Divisore.
BTW, $MCD(9,81)=9$, $MCD(37,7)=1$, $MCD(6,4)=2$;

$MCD(6,4)=?$

Sicuramente e' un divisore di 4 (non puo' essere piu' grande di 4; o e' 4 o e' un numero piu' piccolo, al limite 1)

- 4 divide sia 6 che 4? No
- 3? No
- 2? Si'!!

0) i dati: n , m (input) e ris (con cui calcoliamo il MCD)

1) INPUT (n , m)

2) $ris = \text{😊}$

3) SE ris «DIVIDE n » E «DIVIDE m » EUREKA!

4) $ris = ris - 1$

E torna al passo 3

5) OUTPUT(ris)

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comune Divisore.
BTW, $MCD(9,81)=9$, $MCD(37,7)=1$, $MCD(6,4)=2$;

$MCD(6,4)=?$

Sicuramente e' un divisore di 4 (non puo' essere piu' grande di 4; o e' 4 o e' un numero piu' piccolo, al limite 1)

- 4 divide sia 6 che 4? No
- 3? No
- 2? Si'!!

0) i dati: n , m (input) e ris (con cui calcoliamo il MCD)

1) INPUT (n , m)

2) $ris = \text{😊😊😊😊😊😊😊😊}$

3) SE ris «DIVIDE n » E «DIVIDE m » EUREKA!

...
4) $ris = ris - 1$

E torna al passo 3

5) OUTPUT(ris)

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comune Divisore.
BTW, $MCD(9,81)=9$, $MCD(37,7)=1$, $MCD(6,4)=2$;

$MCD(6,4)=?$

Sicuramente e' un divisore di 4 (non puo' essere piu' grande di 4; o e' 4 o e' un numero piu' piccolo, al limite 1)

- 4 divide sia 6 che 4? No
- 3? No
- 2? Si'!!

0) i dati: n , m (input) e ris (con cui calcoliamo il MCD)

1) INPUT (n , m)

2) $ris = \text{minimo tra } n \text{ ed } m$

3) SE ris «DIVIDE n » E «DIVIDE m » EUREKA!

4) $ris = ris - 1$

E torna al passo 3

5) OUTPUT(ris)

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comune Divisore.
BTW, $MCD(9,81)=9$, $MCD(37,7)=1$, $MCD(6,4)=2$;

$MCD(6,4)=?$

Sicuramente e' un divisore di 4 (non puo' essere piu' grande di 4, o e' 4 o e' un numero piu' piccolo, al limite 1)

- 4 divide sia 6 che 4? No
- 3? No
- 2? Si'!!

0) i dati: n , m (input) e ris (con cui calcoliamo il MCD)

1) INPUT (n , m)

2) $ris = \min\{n,m\}$

3) SE ris «DIVIDE n » E «DIVIDE m » EUREKA!

4) $ris = ris - 1$

E torna al passo 3

5) OUTPUT(ris)

Eureka che?? Vuoi dire "ris è il MCD dei due numeri". Ok, ma poi non dovremmo finire?

Eeeek, Come finisco?

Come ci arrivo al passo 5)?!

E Torna? Torna?!?!?

Che vuol dire Torna?

... calma

Algoritmo per il MCD

Dati due numeri interi, n , m , maggiori di zero, calcolare il Massimo Comun Divisore.

BTW, $MCD(9,81)=9$, $MCD(37,7)=1$, $MCD(6,4)=2$;

$MCD(6,4)=?$

Sicuramente e' un divisore di 4 (non puo' essere piu' grande di 4; o e' 4 o e' un numero piu' piccolo, al limite 1)

- 4 divide sia 6 che 4? No
- 3? No
- 2? Si!!

0) i dati: n , m (input) e ris (con cui calcoliamo il MCD)

1) INPUT (n , m)

2) $ris = \min\{n,m\}$

3) SE ris «DIVIDE n » E «DIVIDE m »

Vai al passo 5) per finire

4) $ris = ris - 1$

E torna al passo 3

5) OUTPUT(ris)

Vai? Torna? Ma il flusso non è sequenziale? Dopo la 3 c'è la 4 o la 5??

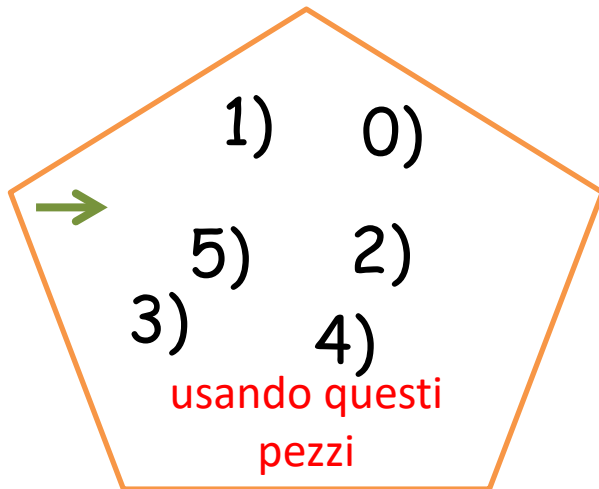
Algoritmo per il MCD (il flusso di esecuzione)

Il flusso di esecuzione puo' dipendere dal verificarsi di condizioni durante i calcoli ...

Flusso di esecuzione per $n=14$, $m=28$:



Flusso di esecuzione per $n=6$, $m=4$:



- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) SE ris «DIVIDE n » E «DIVIDE m »
Vai al passo 5) per finire
- 4) $ris = ris - 1$
E torna al passo 3
- 5) OUTPUT(ris)

Algoritmo per il MCD (il flusso di esecuzione)

Il flusso di esecuzione puo' dipendere dal verificarsi di condizioni durante i calcoli ...

Flusso di esecuzione per $n=14$, $m=28$:

1) → 2) → 3) → 5) e stampa 14

Flusso di esecuzione per $n=6$, $m=4$:

1) → 2) → 3) → 4) → 3) → 4) → 3) → 5)
e stampa 2 perché ris è diventato 2

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \min\{n, m\}$
- 3) SE ris «DIVIDE n » E «DIVIDE m »
Vai al passo 5) per finire
- 4) $ris = ris - 1$
E torna al passo 3
- 5) OUTPUT(ris)

Algoritmo per il MCD (il flusso di esecuzione)

Il flusso di esecuzione puo' dipendere dal verificarsi di condizioni durante i calcoli ...

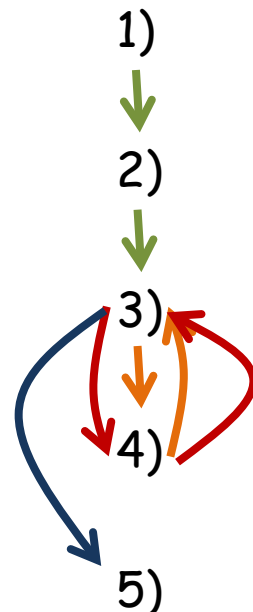
Flusso di esecuzione per $n=14, m=28$:

1) → 2) → 3) → 5)

Flusso di esecuzione per $n=6, m=4$:

1) → 2) → 3) → 4) → 3) → 4) → 3) → 5)

- 0) i dati: n, m (input) e ris
- 1) INPUT (n, m)
- 2) $\text{ris} = \min\{n, m\}$
- 3) SE $\text{ris} \llcorner \text{DIVIDE } n \gg$ E $\llcorner \text{DIVIDE } m \gg$
Vai al passo 5) per finire
- 4) $\text{ris} = \text{ris} - 1$
E torna al passo 3
- 5) OUTPUT(x_1, x_2)



NB

Algoritmo = sequenza di passi progettata per risolvere un problema

Flusso di esecuzione = sequenza dei passi effettivamente eseguiti durante l'esecuzione dell'algoritmo su un'istanza del problema

Algoritmo per il MCD (il flusso di esecuzione)

Il flusso di esecuzione puo' dipendere dal verificarsi di condizioni durante i calcoli ...

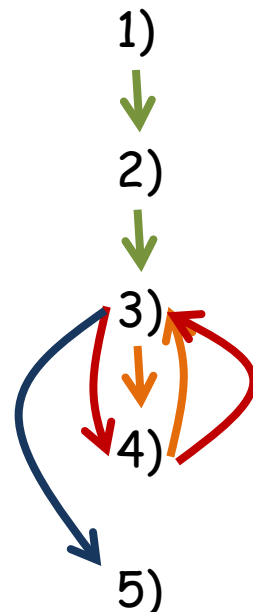
Flusso di esecuzione per $n=14, m=28$:

1) → 2) → 3) → 5)

Flusso di esecuzione per $n=6, m=4$:

1) → 2) → 3) → 4) → 3) → 4) → 3) → 5)

- 0) i dati: n, m (input) e ris
- 1) INPUT (n, m)
- 2) $ris = \min\{n, m\}$
- 3) SE $ris \llcorner \text{DIVIDE } n \gg$ E $\llcorner \text{DIVIDE } m \gg$
Vai al passo 5) per finire
- 4) $ris = ris - 1$
E torna al passo 3
- 5) OUTPUT(x_1, x_2)



NB

Algoritmo = sequenza di passi progettata per risolvere un problema

Flusso di esecuzione = sequenza dei passi effettivamente eseguiti durante l'esecuzione dell'algoritmo su un'istanza del problema

Algoritmo per il MCD (il flusso di esecuzione)

Il flusso di esecuzione puo' dipendere dal verificarsi di condizioni durante i calcoli ...

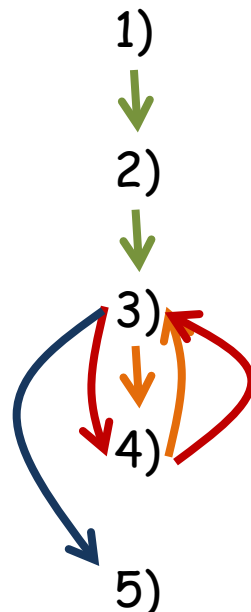
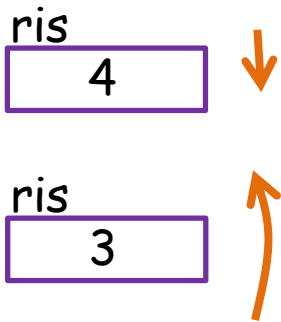
Flusso di esecuzione per $n=14, m=28$:

1) → 2) → 3) → 5)

Flusso di esecuzione per $n=6, m=4$:

1) → 2) → 3) → 4) → 3) → 4) → 3) → 5)

- 0) i dati: n, m (input) e ris
- 1) INPUT (n, m)
- 2) $ris = \min\{n, m\}$
- 3) SE $ris \llcorner \text{DIVIDE } n \gg$ E $\llcorner \text{DIVIDE } m \gg$
Vai al passo 5) per finire
- 4) $ris = ris - 1$
E torna al passo 3
- 5) OUTPUT(x_1, x_2)



NB

Algoritmo = sequenza di passi progettata per risolvere un problema

Flusso di esecuzione = sequenza dei passi effettivamente eseguiti durante l'esecuzione dell'algoritmo su un'istanza del problema

Algoritmo per il MCD (il flusso di esecuzione)

Il flusso di esecuzione puo' dipendere dal verificarsi di condizioni durante i calcoli ...

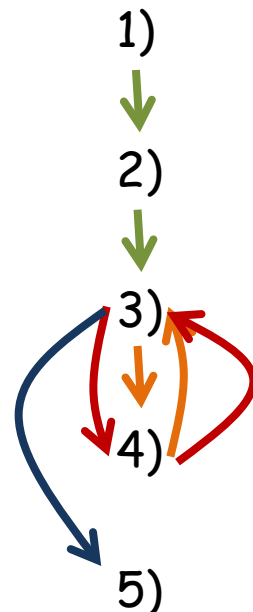
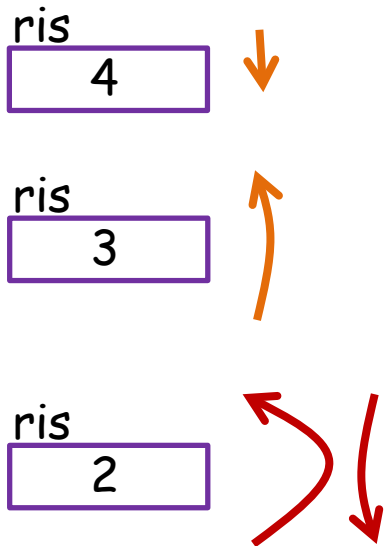
Flusso di esecuzione per $n=14, m=28$:

1) → 2) → 3) → 5)

Flusso di esecuzione per $n=6, m=4$:

1) → 2) → 3) → 4) → 3) → 4) → 3) → 5)

- 0) i dati: n, m (input) e ris
- 1) INPUT (n, m)
- 2) $ris = \min\{n, m\}$
- 3) SE $ris \llcorner \text{DIVIDE } n \gg$ E $\llcorner \text{DIVIDE } m \gg$
Vai al passo 5) per finire
- 4) $ris = ris - 1$
E torna al passo 3
- 5) OUTPUT(x_1, x_2)



NB

Algoritmo = sequenza di passi progettata per risolvere un problema

Flusso di esecuzione = sequenza dei passi effettivamente eseguiti durante l'esecuzione dell'algoritmo su un'istanza del problema

Ok, il flusso di esecuzione e' ...

il risultato del controllo cui viene sottoposta l'esecuzione di un algoritmo

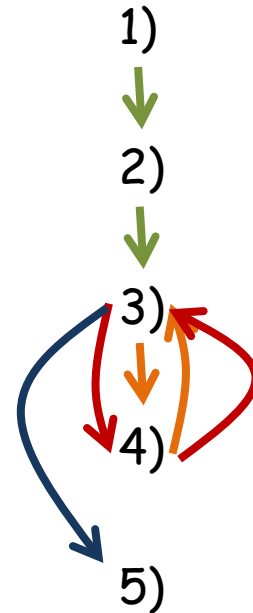
ALGORITMO = SEQUENZA DI PASSI CONTROLLATI

controllo

un passo dopo l'altro, in ordine!
anche attraverso

MA

- SELEZIONE di un'AZIONE nel caso si verifichi una CONDIZIONE
- SALTO ad un passo diverso da quello immediatamente seguente ...

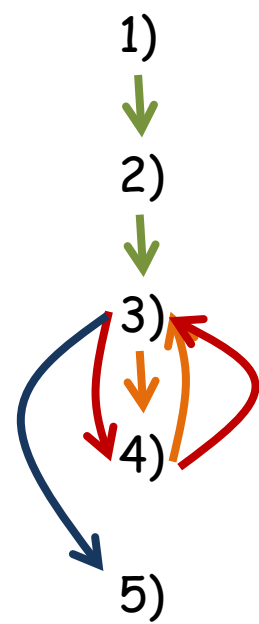


Per gestire (la programmazione di) algoritmi come sequenze di passi controllati si usa la **Programmazione Strutturata**

Ok, il flusso di esecuzione e' ...

Il flusso di esecuzione è il risultato del **controllo** cui viene sottoposta l'esecuzione di un algoritmo (e poi ovviamente del programma corrispondente!!) durante l'esecuzione stessa

ALGORITMO = SEQUENZA DI PASSI CONTROLLATI



Di base il **controllo** consiste nell'eseguire un passo dopo l'altro, in ordine!

MA il **controllo** si estrinseca anche attraverso i meccanismi di base della

- SCELTA di un'AZIONE nel caso si verifichi una CONDIZIONE
- SALTO ad un passo diverso da quello immediatamente seguente ...

"Vai ad altra istruzione" | "go to" | istruzione vintage: goto | non la usiamo

In realtà il salto va gestito diversamente da come abbiamo visto finora (*goto*)

Il salto puo' essere condizionato (SE ... vai a) o incondizionato (vai a ...) ma comunque **NON USEREMO GOTO!** Ma una cosa migliore



Per gestire (la programmazione di) algoritmi come sequenze di passi controllati si usa la **Programmazione Strutturata**

Programmazione Strutturata

Il programma Strutturato è composto da questi tre tipi di **istruzioni**

- **BLOCCO** sequenza di 1 o più istruzioni "non scomponibili" come un'operazione di I/O o un'assegnazione, o una chiamata di printf(), o una chiamata di qualsiasi funzione, o una delle istruzioni dette più giù qui sotto ...)

```
istruzione  
secondaistruzione  
...  
ultimaistruzione
```


Programmazione Strutturata

Il programma Strutturato è composto da questi tre tipi di **istruzioni**

- **BLOCCO** sequenza di 1 o più istruzioni "non scomponibili" come un'operazione di I/O o un'assegnazione, o una chiamata di printf(), o una chiamata di qualsiasi funzione, o una delle istruzioni dette più giù qui sotto ...)

```
istruzione1  
istruzione2  
...
```

- **ISTRUZIONE CONDIZIONALE**
seleziona, in base al **valore di verità** di una **CONDIZIONE**,
l'istruzione da eseguire:

```
SE (CONDIZIONE)  
    istruzione  
ALTRIMENTI  
    altraistruzione
```

Programmazione Strutturata

Il programma Strutturato è composto da questi tre tipi di **istruzioni**

- **BLOCCO** sequenza di 1 o più istruzioni (non scomponibili, come un'operazione di I/O o un'assegnazione, o una chiamata di printf(), o una chiamata di qualsiasi funzione, o una delle istruzioni seguenti ...)

```
istruzione1  
istruzione2  
...
```

- **ISTRUZIONE CONDIZIONALE**
seleziona, in base al valore di verità di una **CONDIZIONE**,
l'istruzione da eseguire:

```
SE (CONDIZIONE)  
    istruzione  
ALTRIMENTI  
    altraistruzione
```

- **ISTRUZIONE DI RIPETIZIONE**
di base, consiste nella ripetizione della medesima istruzione
FINTANTO CHE (**mentre**) una certa **CONDIZIONE** è
VERIFICATA (cioè ha valore di verità TRUE)

```
MENTRE (CONDIZIONE)  
    istruzione
```

Programmazione Strutturata

Il programma Strutturato è composto da questi tre tipi di **istruzioni**

- **BLOCCO** sequenza di 1 o più istruzioni (non scomponibili, come un'operazione di I/O o un'assegnazione, o una chiamata di printf(), o una chiamata di qualsiasi funzione, o una delle istruzioni seguenti ...)

```
istruzione1  
istruzione2  
...
```

- **ISTRUZIONE CONDIZIONALE**
selezione in base al valore di verità di una **CONDIZIONE**,

NB Ogni istruzione è una di quelle tre:
blocco, istr. condizionale, o istr. di
ripetizione.

```
SE (CONDIZIONE)  
    istruzione  
ALTRIMENTI  
    altraistruzione
```

- **ISTRUZIONE DI RIPETIZIONE**
di base, consiste nella ripetizione della medesima istruzione
FINTANTO CHE (**mentre**) una certa **CONDIZIONE** è
VERIFICATA (cioè ha valore di verità TRUE)

```
MENTRE (CONDIZIONE)  
    istruzione
```

Programmazione Strutturata

Il programma Strutturato e' composto da questi tre tipi di **istruzioni**

- **BLOCCO** sequenza di istruzioni elementari (non scomponibili, come un'operazione di I/O o un'assegnazione, o una chiamata di printf(), o una chiamata di qualsiasi funzione)

istr. el.

- **ISTRUZIONE CON SELEZIONE**
selezione
l'istruzione

Teorema di Bohem-Jacopini (~1966):

qualsiasi programma
(funzione calcolabile)

può essere scritto usando solo le strutture di

- sequenza,
- selezione
- e ripetizione.

- **ISTRUZIONE DI RIPETIZIONE**

di base, consiste nella ripetizione della medesima istruzione **FINTANTO CHE** (mentre) una certa **CONDIZIONE** e' **VERIFICATA** (cioè ha valore di verita' **TRUE**)

MENTRE (CONDIZIONE)
istruzione

Algoritmo strutturato per l'eq. di secondo grado

- 0) i dati: a, b, c ; i risultati, x_1, x_2 ; i dati intermedi deltaquadro, $2*a, 4*a*c$...
- 1) INPUT (a, b, c)
 - 2) $\text{deltaQuadro} = b*b - 4*a*c$
 - 3) SE ($\text{deltaQuadro} \geq 0$)
 - 3.1) $x_1 =$ formula per la sol 1
 - 3.2) $x_2 =$ formula per la sol 2
 - 3.3) OUTPUT(x_1, x_2)ALTRIMENTI OUTPUT(«no»)
 - 4) OUTPUT(«fine programma»)

in pratica, ci sono soluzioni reali solo quando il deltaQuadro e' ≤ 0 ;

allora

se il deltaQuadro e' ≥ 0 calcoliamo x_1 e x_2 e le stampiamo, senno` stampiamo NO

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 ; i dati intermedi Δ , $2a, 4ac$

...

- 1) INPUT (a, b, c)
- 2) $\Delta = b^2 - 4ac$
- 3) **SE** ($\Delta \geq 0$)
 - 3.1) x_1 = formula per la sol 1
 - 3.2) x_2 = formula per la sol 2
 - 3.3) OUTPUT(x_1, x_2)
- ALTRIMENTI** OUTPUT(«no»)
- 4) OUTPUT(«fine programma»)

in pratica, ci sono soluzioni reali solo quando il Δ è ≥ 0 ;

allora

se il Δ è ≥ 0 calcoliamo x_1 e x_2 e le stampiamo, senno` stampiamo NO

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 ; i dati intermedi $\text{deltaQuadro}, \dots 2*a, 4*a*c \dots$

1) INPUT (a, b, c)

2) $\text{deltaQuadro} = b*b - 4*a*c$

3) **SE** ($\text{deltaQuadro} \geq 0$)

3.1) $x_1 = \text{formu}$

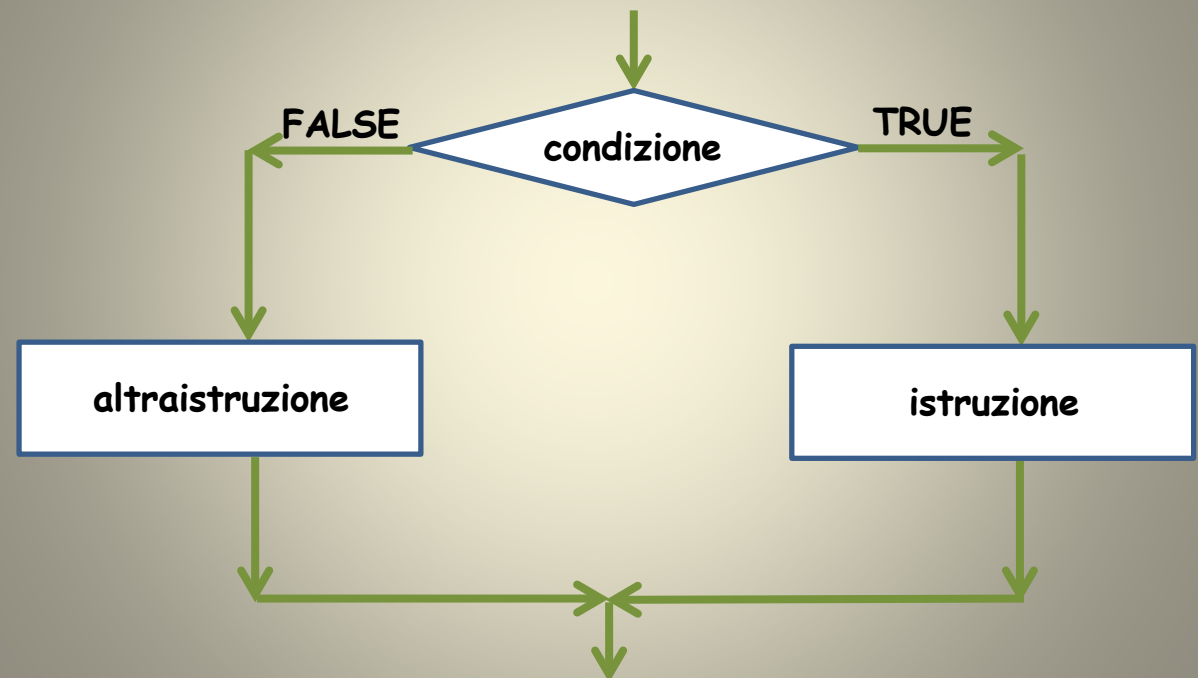
3.2) $x_2 = \text{formu}$

3.3) OUTPUT(x)

ALTRIMENTI OUTPUT

4) OUTPUT(«fine prog

C'è un'istruzione condizionale, che nel diagramma a blocchi è rappresentata dalla forma seguente

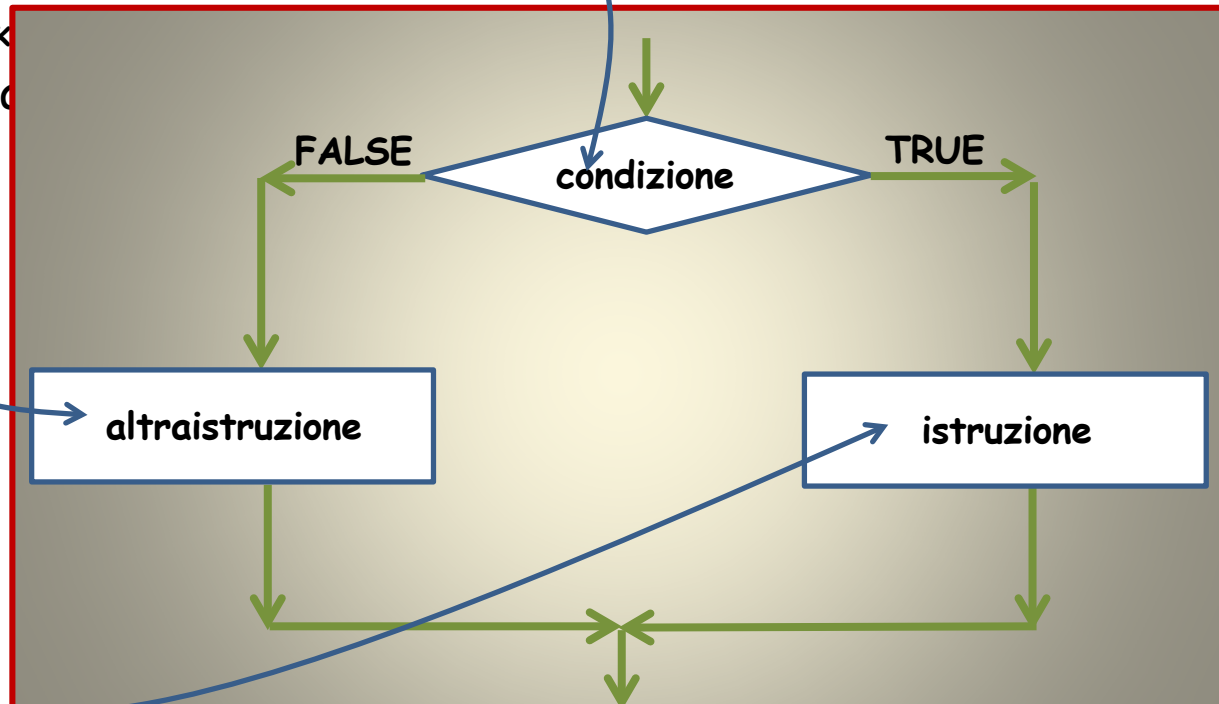


SE (CONDIZIONE)
istruzione
ALTRIMENTI
altraistruzione

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2; i dati intermedi deltaQuadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaQuadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaQuadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)**ALTRIMENTI** OUTPUT(«
- 4) OUTPUT(«fine programma»)



istruzione1
istruzione2
...

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 ; i dati intermedi $\text{deltaQuadro}, \dots 2*a, 4*a*c \dots$

1) INPUT (a, b, c)

2) $\text{deltaQuadro} = b*b - 4*a*c$

3) **SE** ($\text{deltaQuadro} \geq 0$)

3.1) $x_1 =$ formula per la sol 1

3.2) $x_2 =$ formula per la sol 2

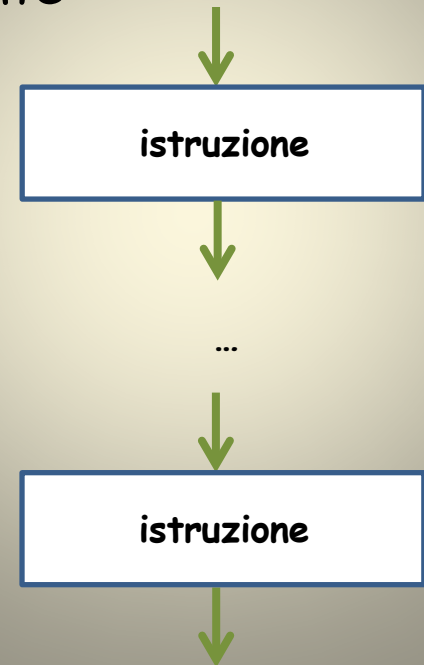
3.3) OUTPUT(x_1, x_2)

ALTRIMENTI OUTPUT(«no»)

4) OUTPUT(«fine programma»)



Vabbè, c'è anche un **BLOCCO**, che nel diagramma a blocchi è rappresentato abbastanza ovviamente



Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 ; i dati intermedi $\text{deltaQuadro}, \dots 2*a, 4*a*c \dots$

- 1) INPUT (a, b, c)
- 2) $\text{deltaQuadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaQuadro} \geq 0$)
 - 3.1) $x_1 = \text{formula per la sol 1}$
 - 3.2) $x_2 = \text{formula per la sol 2}$
 - 3.3) OUTPUT(x_1, x_2)

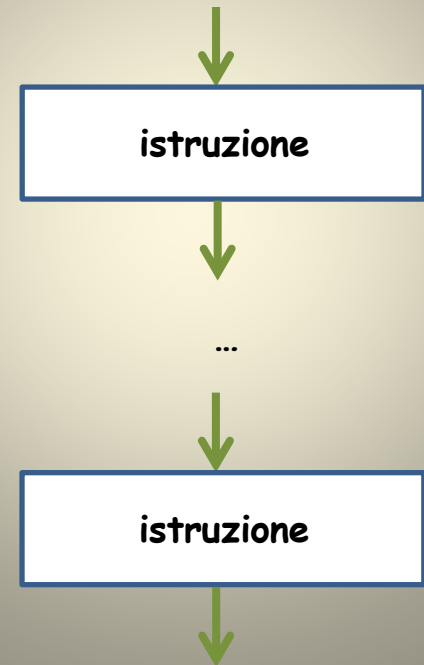
ALTRIMENTI

OUTPUT(«no»)

- 4) OUTPUT(«fine programma»)



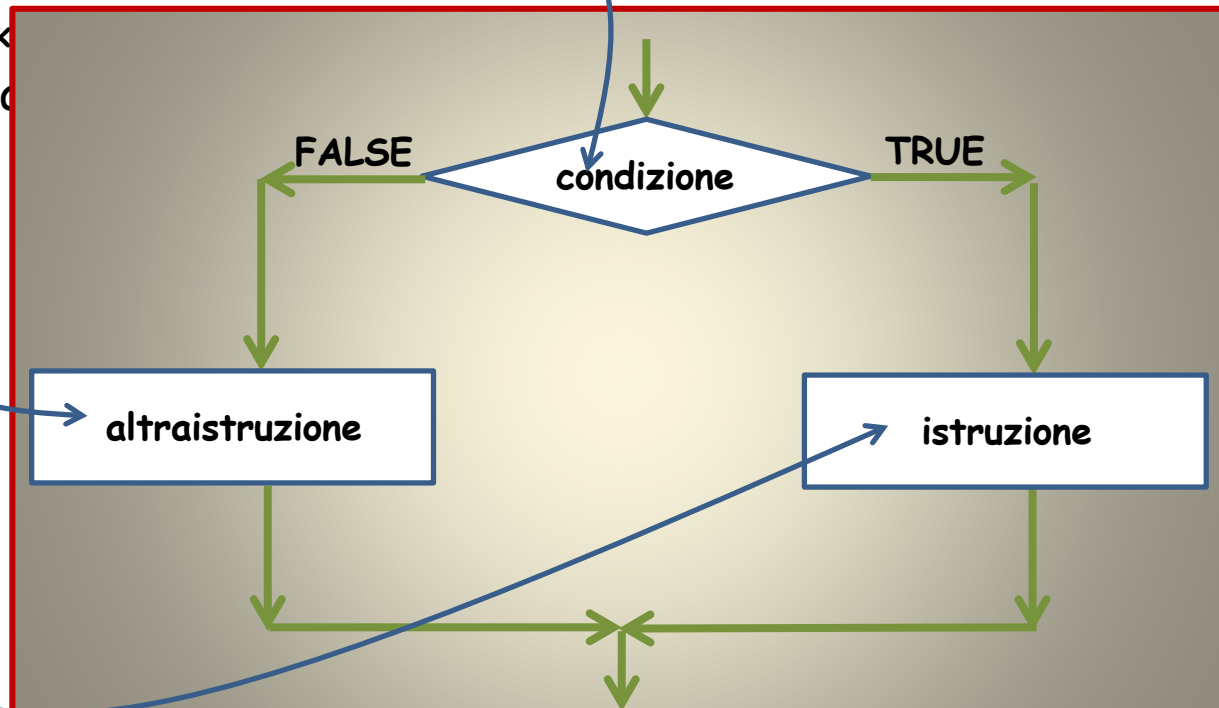
Vari blocchi, actually



ricordiamo

0) i dati: a, b, c ; i risultati, x_1, x_2 ; i dati intermedi Δ , ... $2a, 4ac$...

- 1) INPUT (a, b, c)
- 2) $\Delta = b^2 - 4ac$
- 3) **SE** ($\Delta \geq 0$)
 - 3.1) x_1 = formula per la sol 1
 - 3.2) x_2 = formula per la sol 2
 - 3.3) OUTPUT(x_1, x_2)
- ALTRIMENTI** OUTPUT(«
- 4) OUTPUT(«fine programma»)



Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c ; i risultati, x_1, x_2 (supponiamo che esistano reali); i dati intermedi $\text{deltaQuadro}, \dots 2*a, 4*a*c \dots$

1) INPUT (a, b, c)

2) $\text{deltaQuadro} = b*b - 4*a*c$

3) **SE** ($\text{deltaQuadro} \geq 0$)

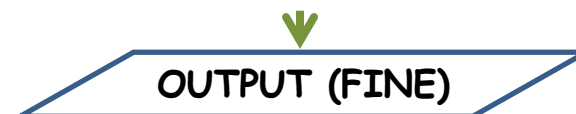
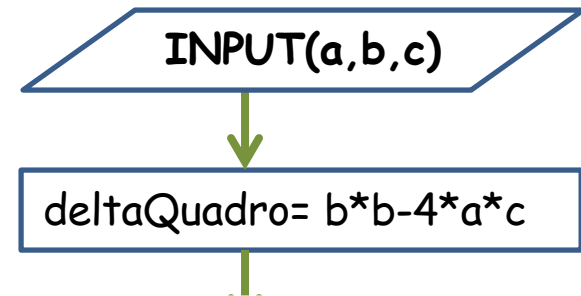
3.1) $x_1 =$ formula per la sol 1

3.2) $x_2 =$ formula per la sol 2

3.3) OUTPUT(x_1, x_2)

ALTRIMENTI OUTPUT(«no»)

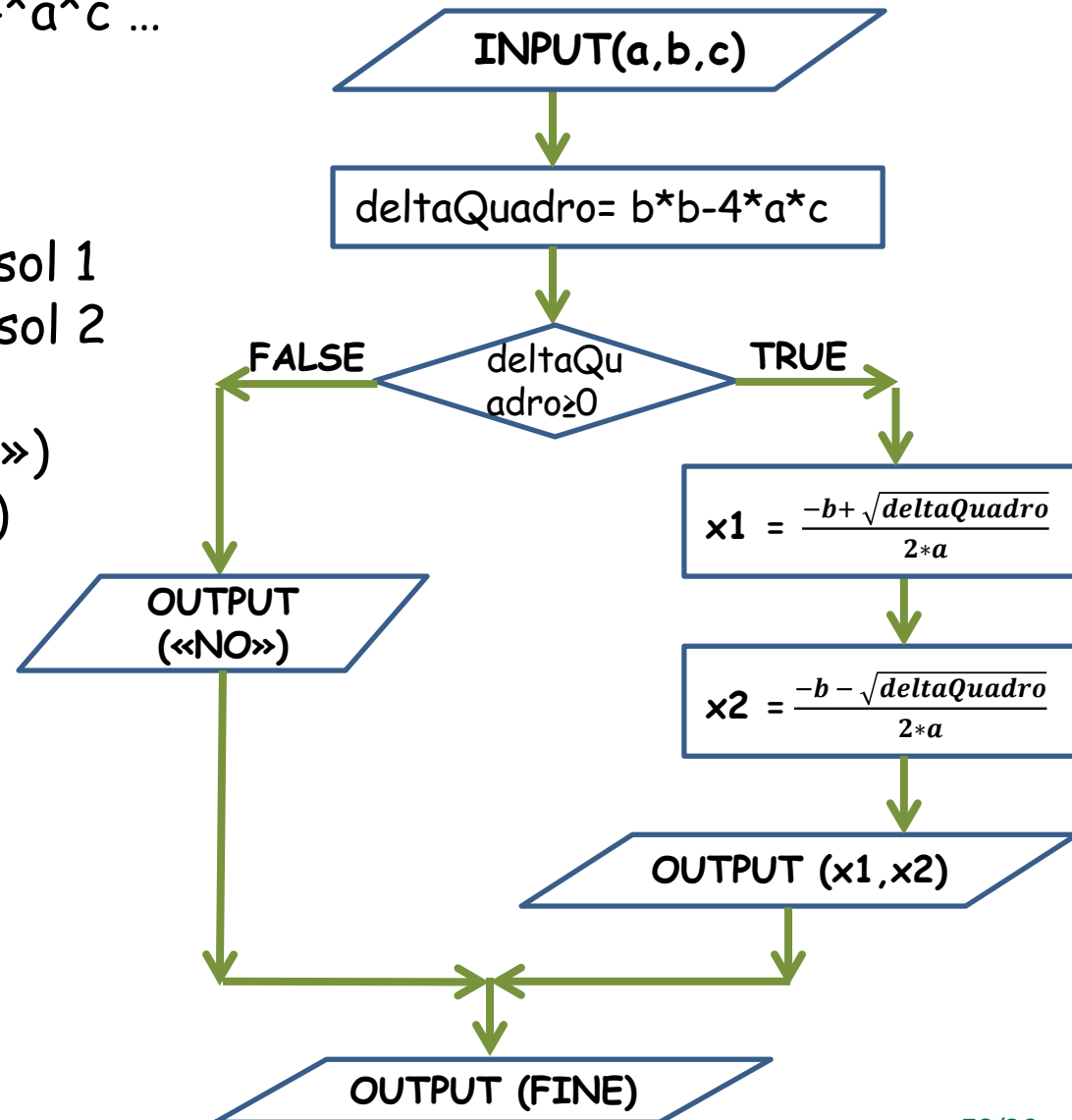
4) OUTPUT(«fine programma»)



Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaQuadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaQuadro} = b*b - 4*a*c$
- 3) SE ($\text{deltaQuadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- ALTRIMENTI OUTPUT(«no»)
- 4) OUTPUT(«fine programma»)

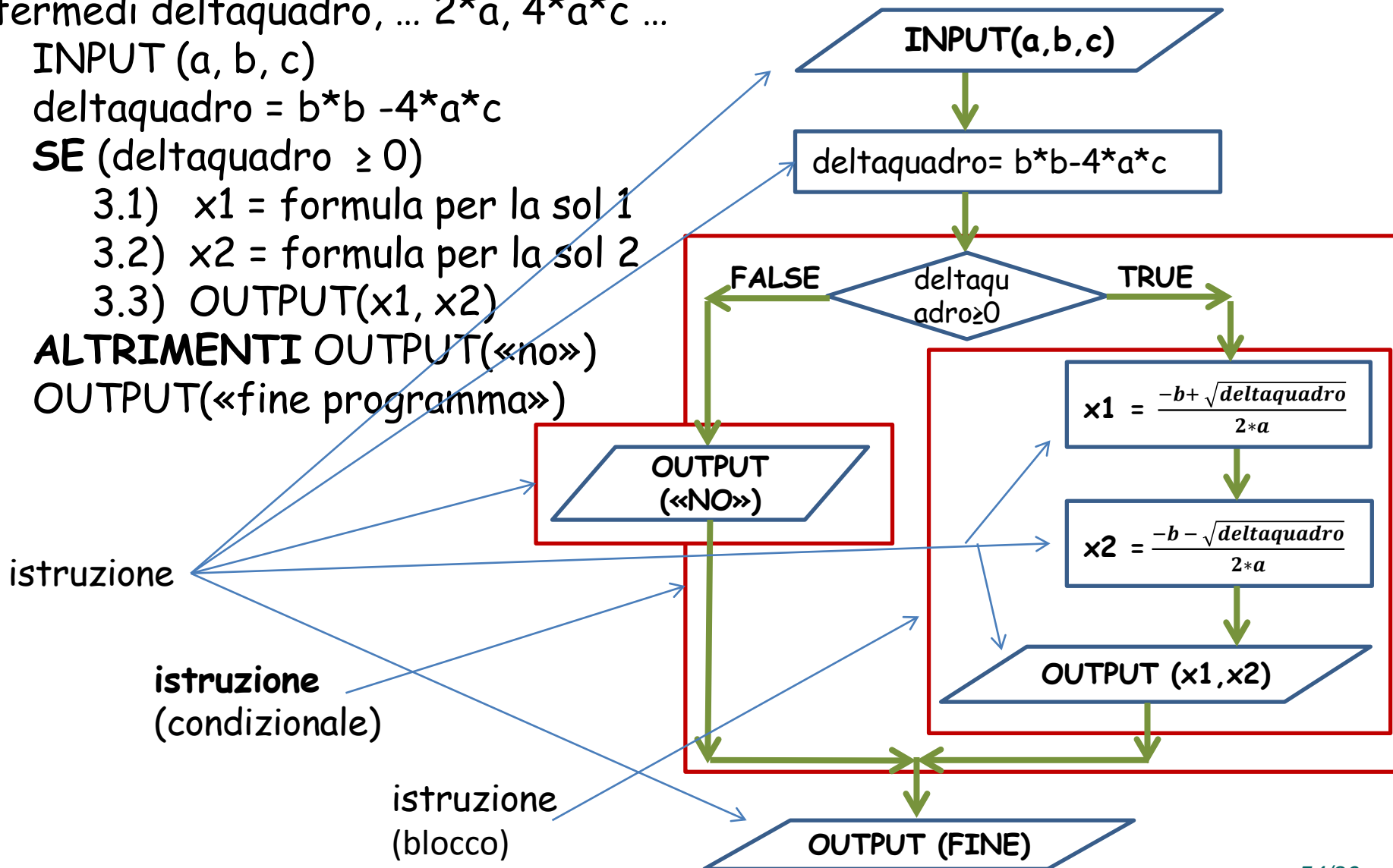


Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) SE ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- 4) OUTPUT(«fine programma»)

ALTRIMENTI OUTPUT(«no»)

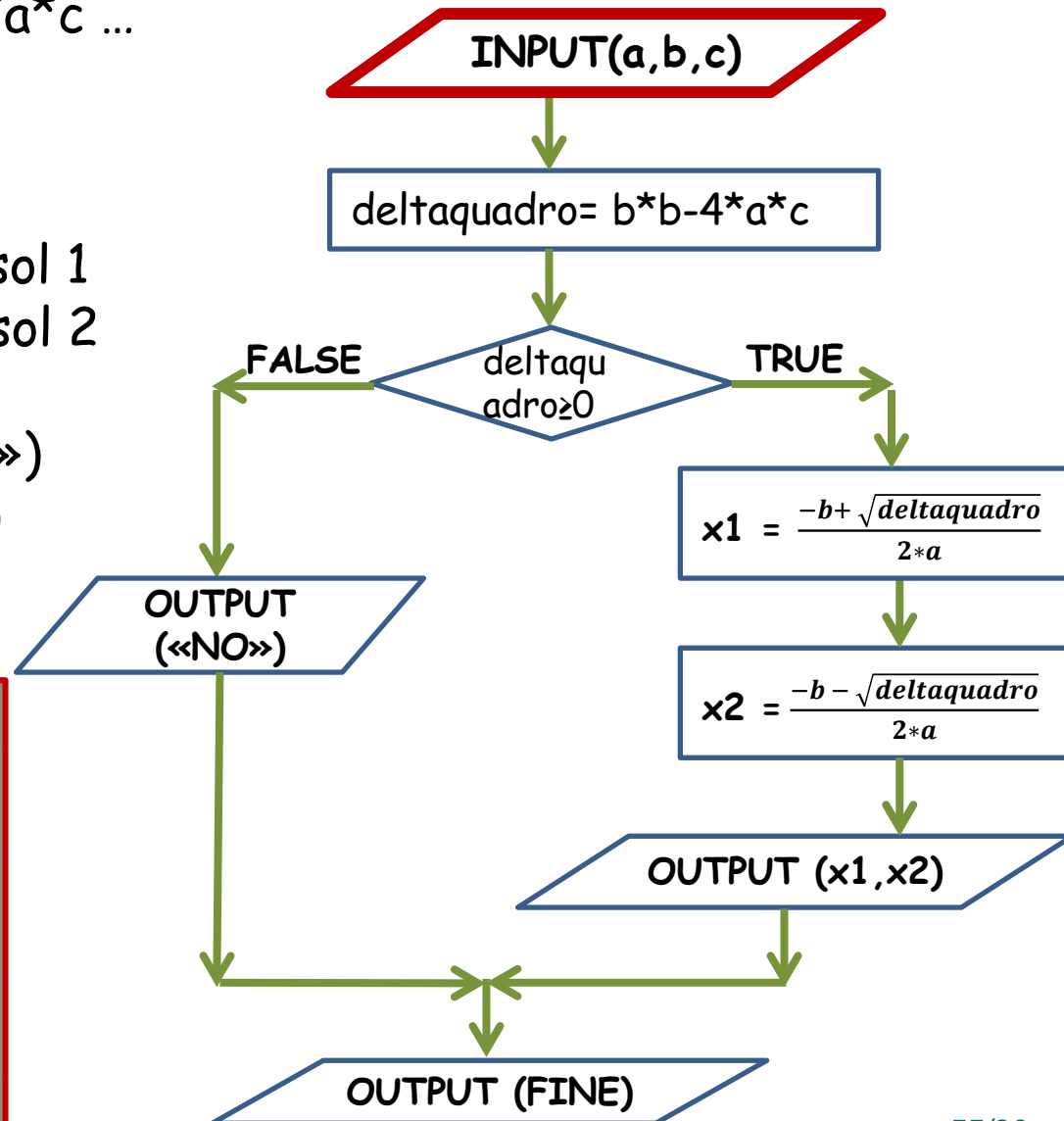


Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- 4) OUTPUT(«fine programma»)

ALTRIMENTI OUTPUT(«no»)



Esecuzione, per a=1, b=4, c=1

a b c

deltaQuadro

x1

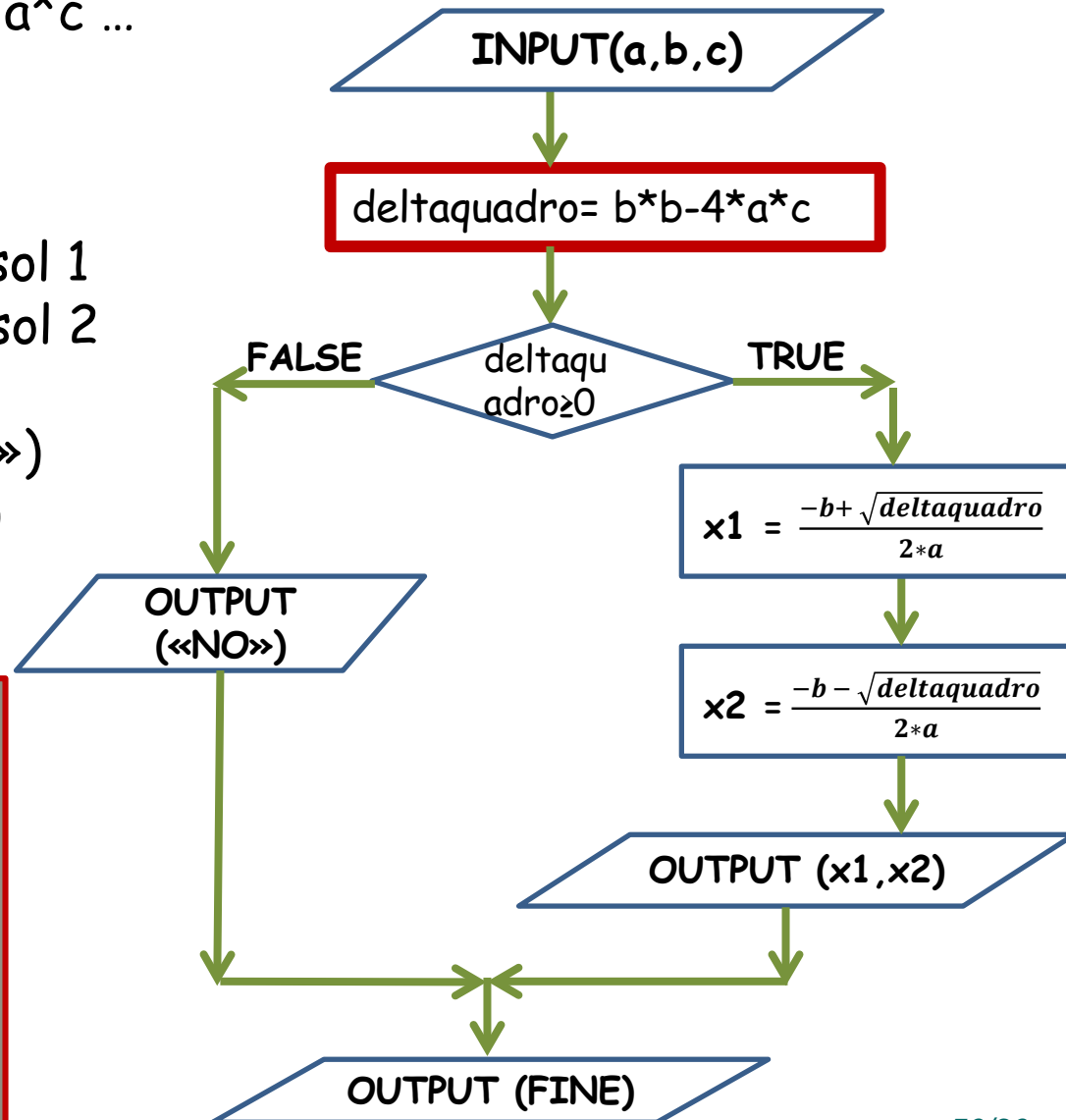
x2

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) SE ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- 4) OUTPUT(«fine programma»)

ALTRIMENTI OUTPUT(«no»)



Esecuzione, per a=1, b=4, c=1

a b c

deltaQuadro

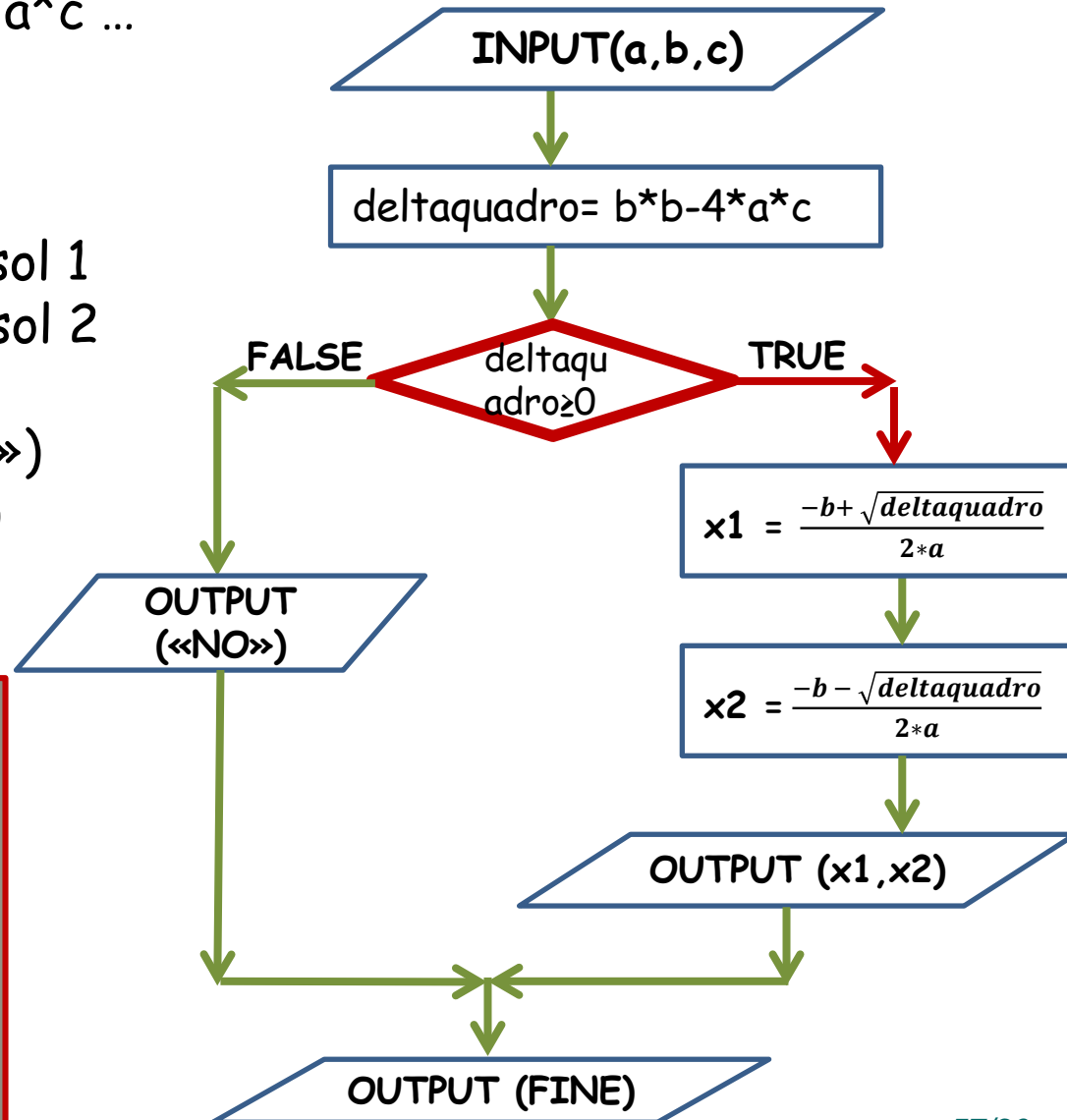
x1

x2

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- ALTRIMENTI OUTPUT(«no»)
- 4) OUTPUT(«fine programma»)



Esecuzione, per a=1, b=4, c=1

a b c

deltaQuadro

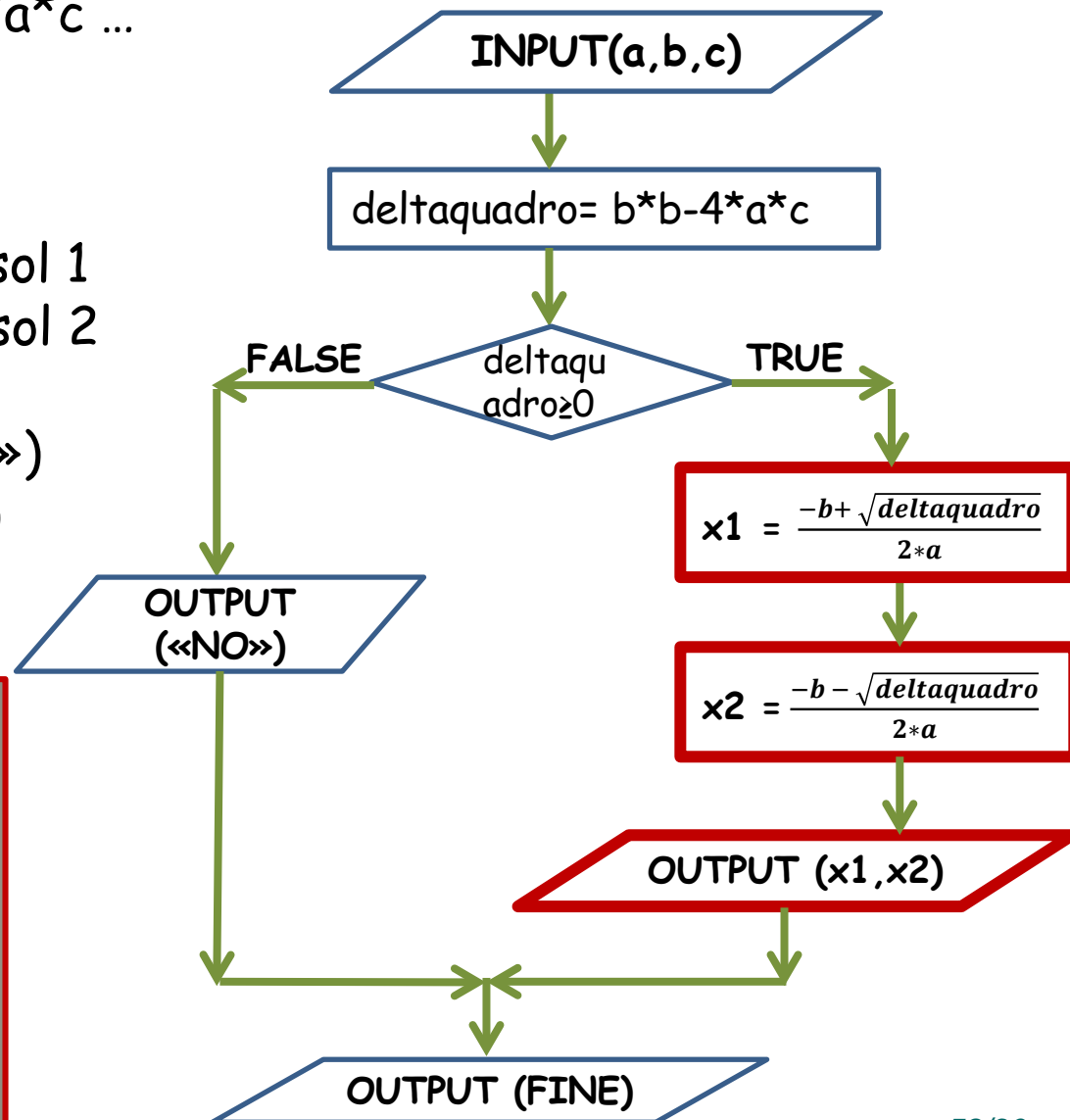
x1

x2

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- ALTRIMENTI OUTPUT(«no»)
- 4) OUTPUT(«fine programma»)



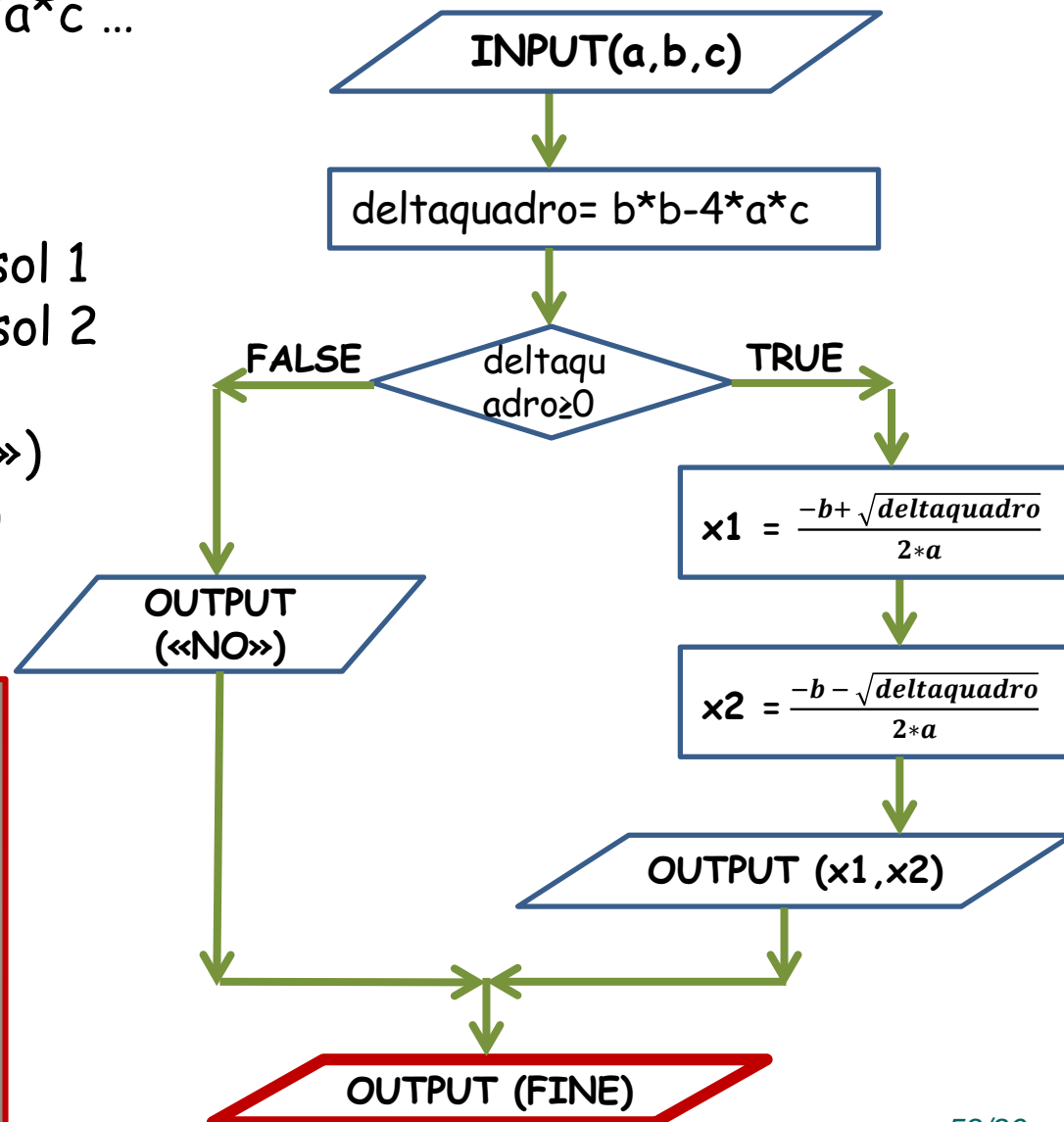
Esecuzione, per a=1, b=4, c=1

| | | | | | |
|---------------|--------|---|---|---|---|
| a | 1 | b | 4 | c | 1 |
| deltaQuadro | 12 | | | | |
| output x1, x2 | x1 ... | | | | |
| | x2 ... | | | | |

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- ALTRIMENTI OUTPUT(«no»)
- 4) OUTPUT(«fine programma»)



Esecuzione, per a=1, b=4, c=1

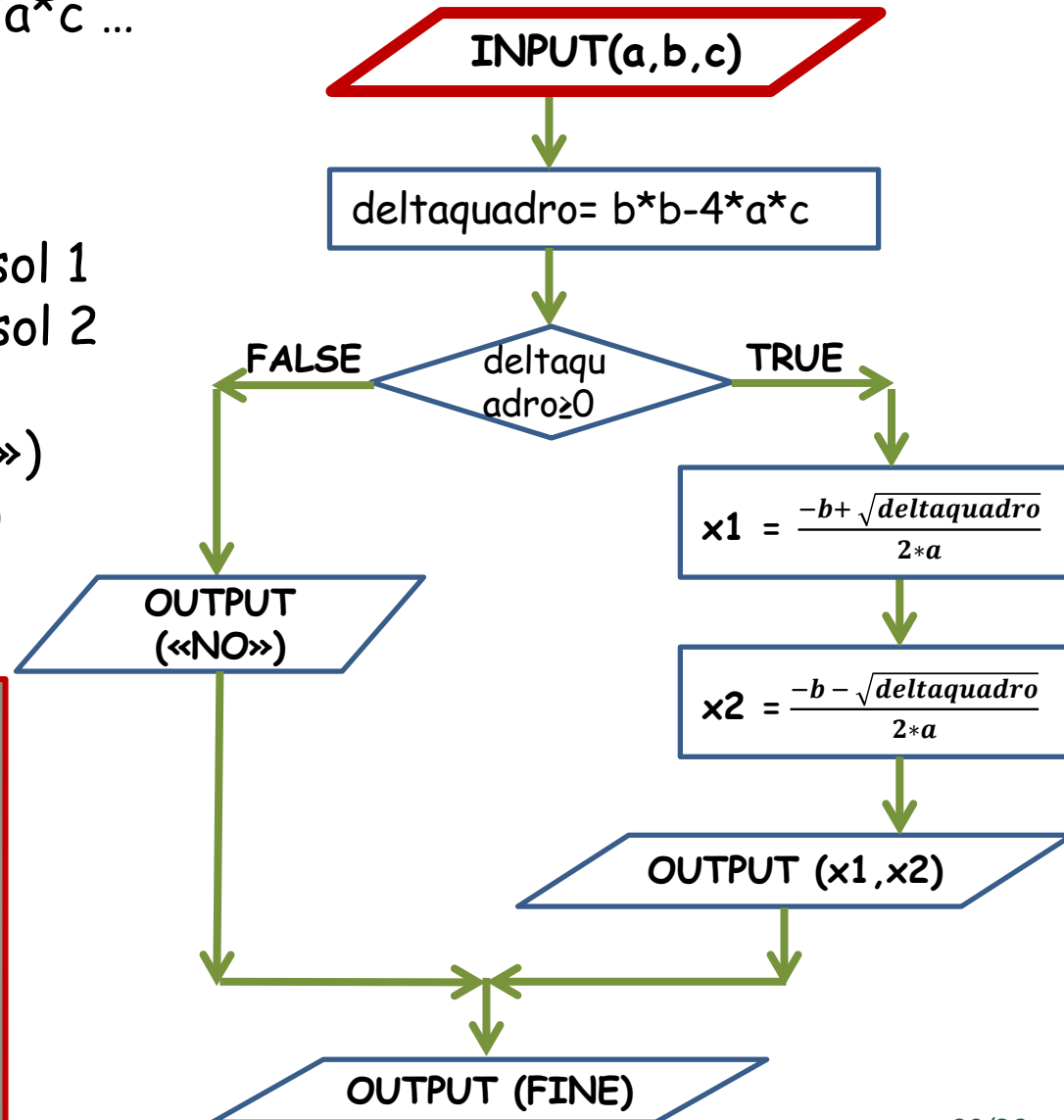
| | | | | | |
|---------------|------|---|---|---|---|
| a | 1 | b | 4 | c | 1 |
| deltaQuadro | 12 | | | | |
| output x1, x2 | FINE | | | | |
| x1 | ... | | | | |
| x2 | ... | | | | |

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- 4) OUTPUT(«fine programma»)

ALTRIMENTI OUTPUT(«no»)



Esecuzione, per a=4, b=4, c=4

a b c

deltaQuadro

x1

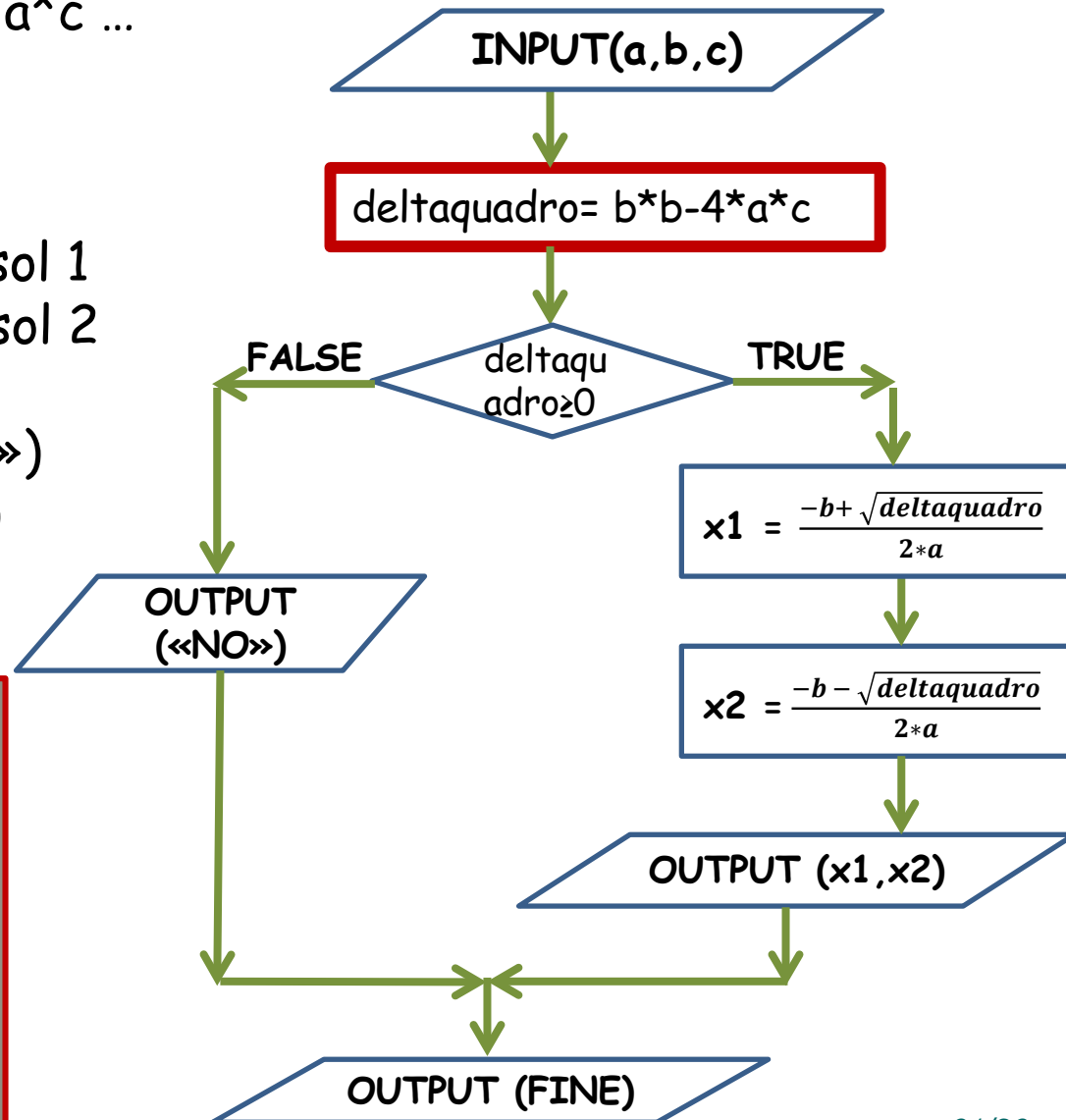
x2

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- 4) OUTPUT(«fine programma»)

ALTRIMENTI OUTPUT(«no»)



Esecuzione, per a=4, b=4, c=4

a b c

deltaQuadro

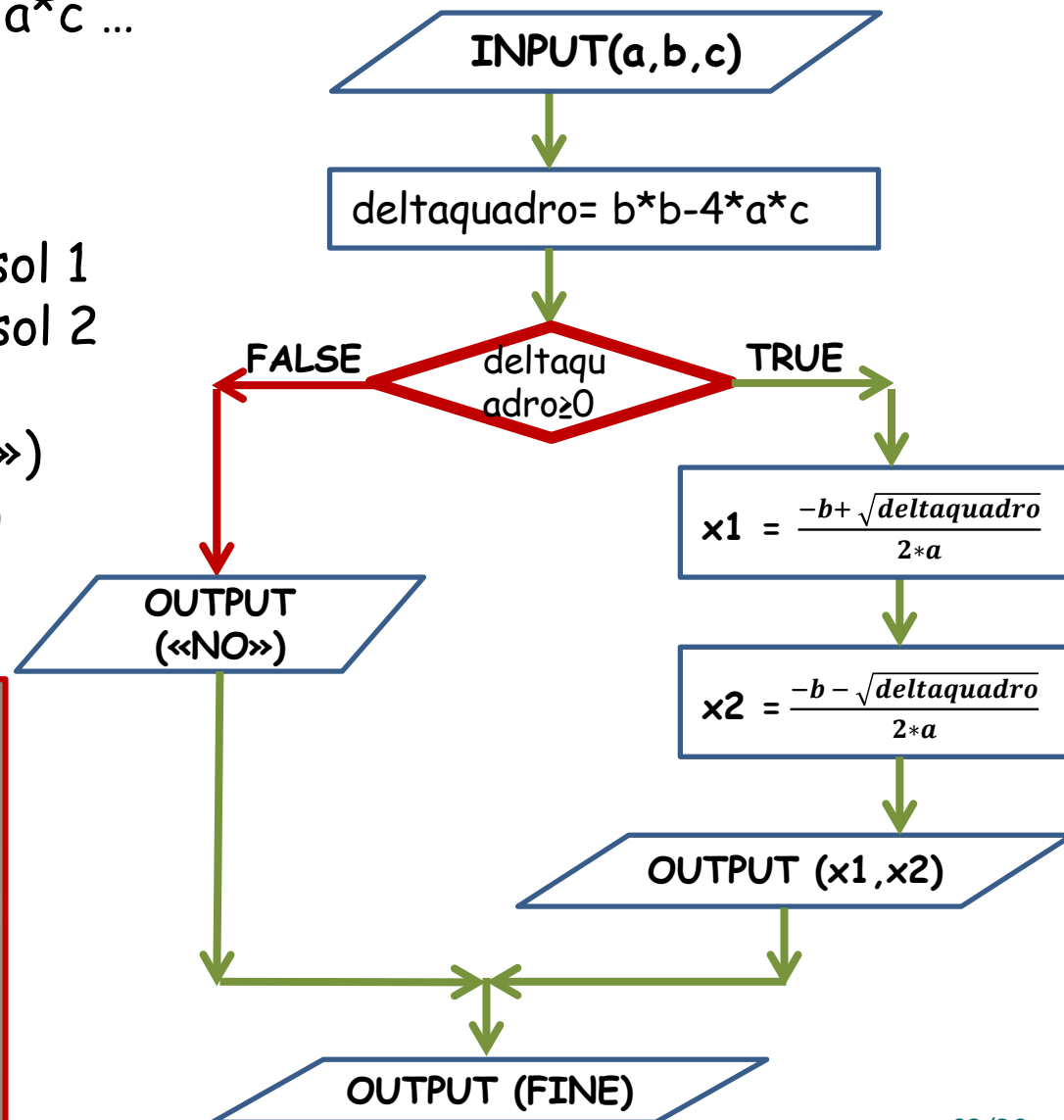
x1

x2

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- ALTRIMENTI OUTPUT(«no»)
- 4) OUTPUT(«fine programma»)



Esecuzione, per a=4, b=4, c=4

4 4 4

deltaQuadro -48

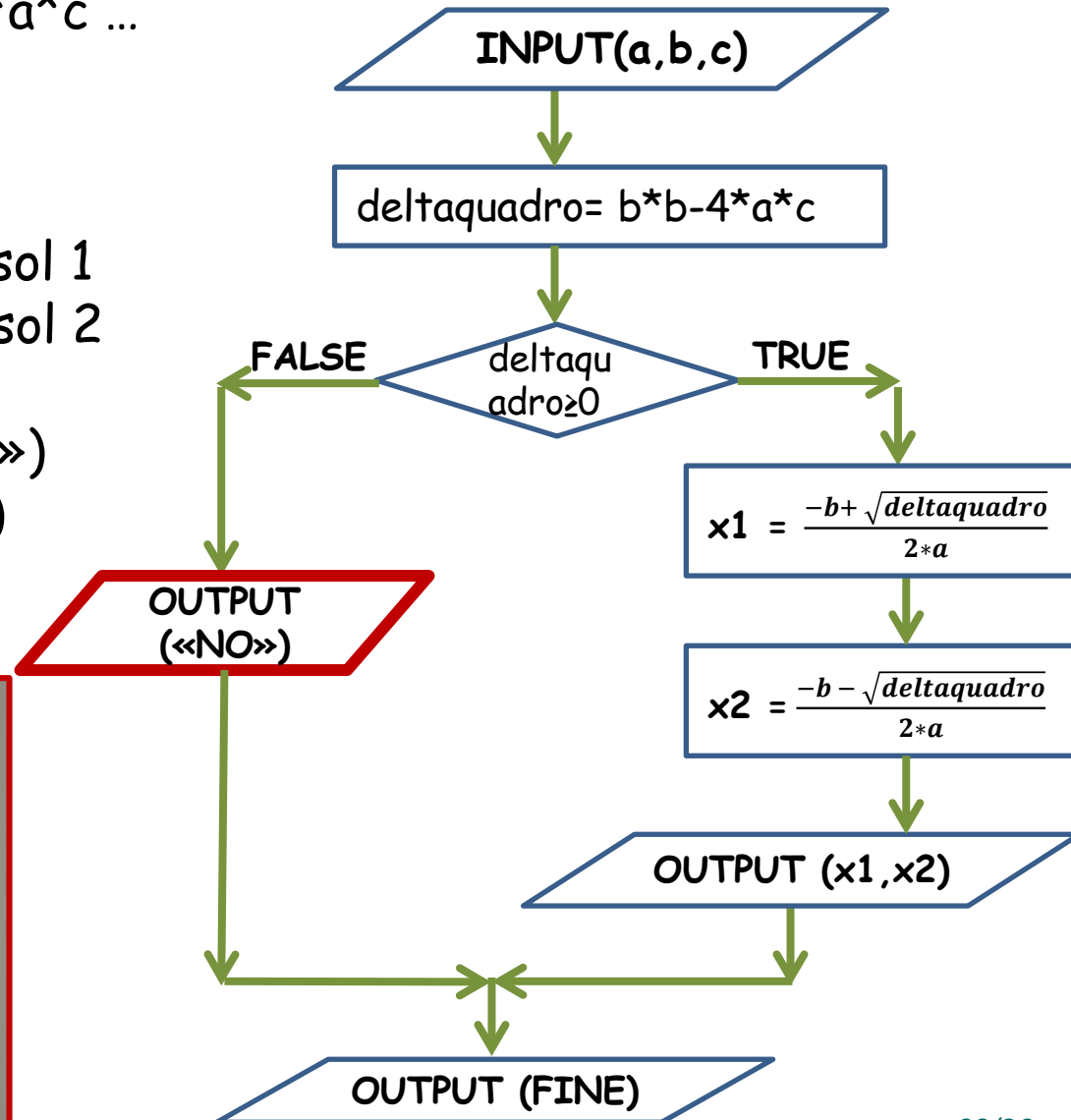
x1

x2

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- ALTRIMENTI OUTPUT(«no»)
- 4) OUTPUT(«fine programma»)



Esecuzione, per a=4, b=4, c=4

4 4 4

deltaQuadro -48

NO

x1

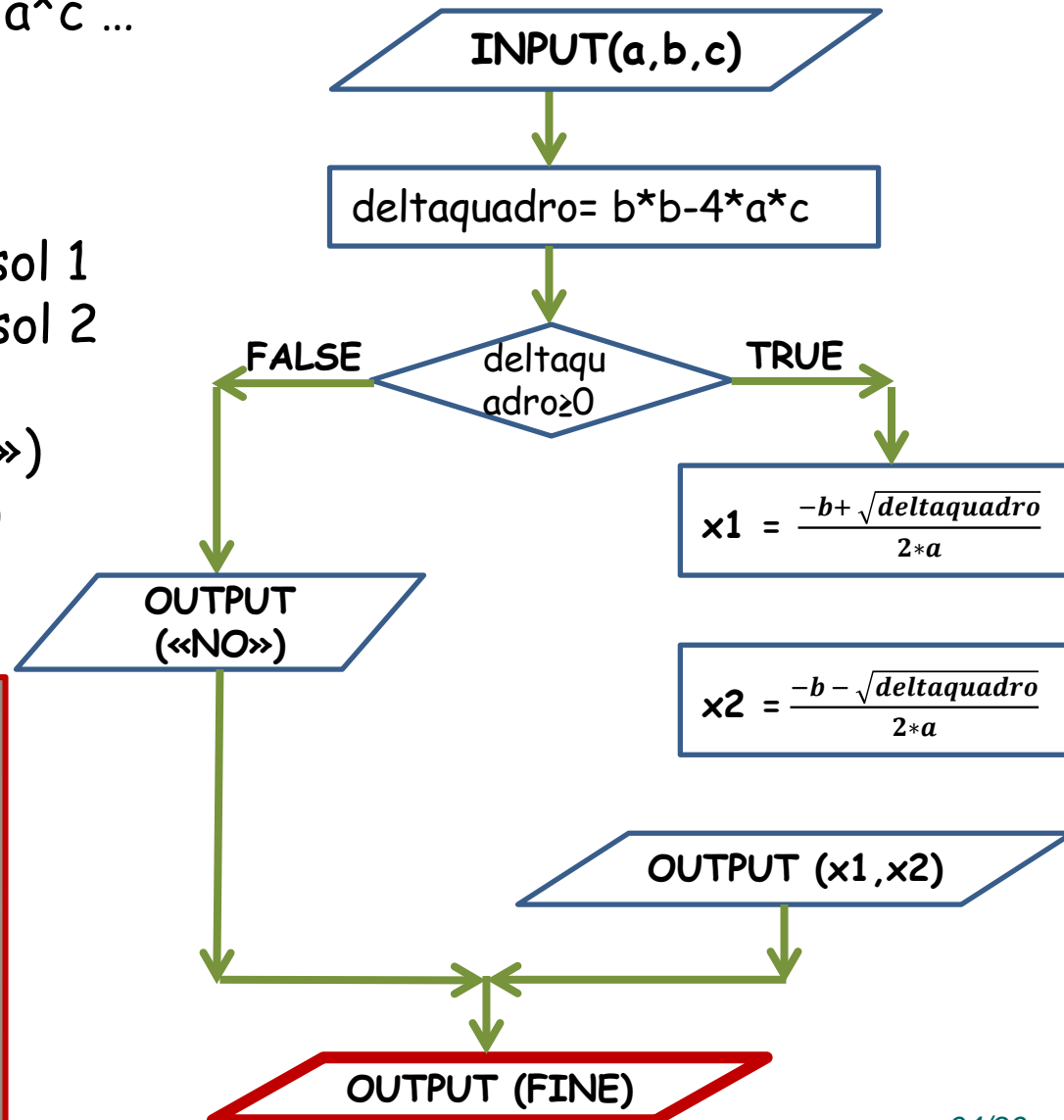
x2

Algoritmo strutturato per l'eq. di secondo grado

0) i dati: a, b, c; i risultati, x1, x2 (supponiamo che esistano reali); i dati intermedi deltaquadro, ... $2*a$, $4*a*c$...

- 1) INPUT (a, b, c)
- 2) $\text{deltaquadro} = b*b - 4*a*c$
- 3) **SE** ($\text{deltaquadro} \geq 0$)
 - 3.1) x1 = formula per la sol 1
 - 3.2) x2 = formula per la sol 2
 - 3.3) OUTPUT(x1, x2)
- 4) OUTPUT(«fine programma»)

ALTRIMENTI OUTPUT(«no»)



Esecuzione, per a=4, b=4, c=4

deltaQuadro

NO
FINE

x1

x2

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \min\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) oppure (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \min\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) oppure (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)

NB

Nello scrivere gli algoritmi abbiamo usato una certa libertà espressiva perché non stiamo programmando in un linguaggio di programmazione e non abbiamo troppi vincoli (a parte quello della precisione, non ambiguità ed univocità di quel che scriviamo ... già, roba da poco ...).

A volte usiamo per brevità scritte che sembrano linee di codice di programma: stiamo usando **PSEUDOCODICE**

NB2

Confronta questo algoritmo con quello precedente visto per il MCD.

Anche qui facciamo un test e, se è il caso, eseguiamo $ris = ris - 1$

e poi torniamo ad eseguire il test (cioè torniamo al passo 3). Però la logica del test è inversa rispetto a quello precedente. È stato necessario invertirla per poter scrivere questa istruzione strutturata (di ripetizione).

[Dopo si vede che questo test si chiama "condizione di ripetizione".]

Se il test fallisce, il passo 3.1 non viene eseguito, non si torna ad eseguire il passo 3) e invece si passa ad eseguire il passo 4) ... che è un'azione di output, come il passo 5 dell'algoritmo precedente.

Algoritmo strutturato per il MCD

0) i dati: n , m (input) e ris

1) INPUT (n , m)

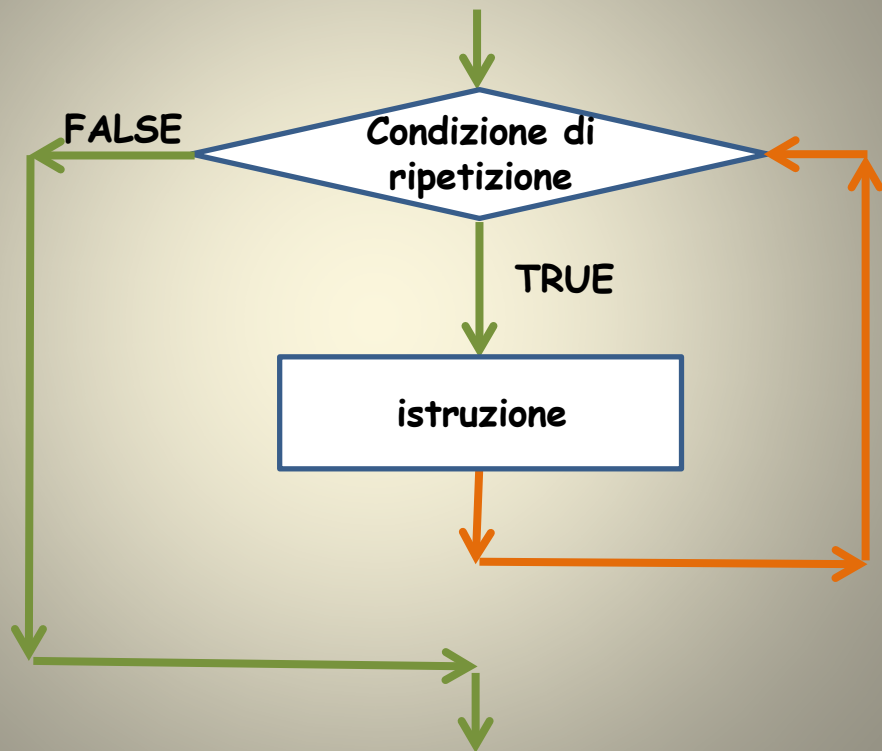
2) $ris = \text{minimo}\{n,m\}$

3) FINTANTO CHE (ris NON DIVIDE n) OPPURE (ris NON DIVIDE m)

3.1) $ris = ris - 1$

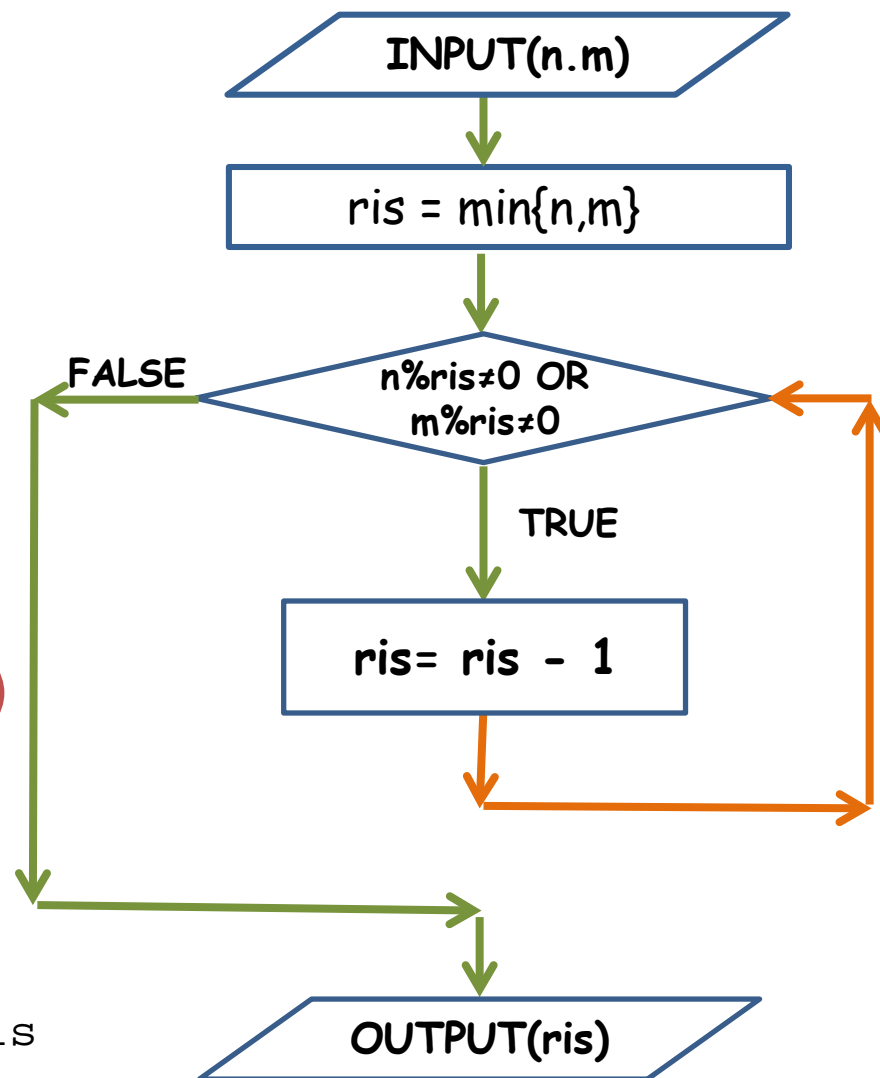
4) OUTPUT(ris)

C'è un'istruzione di ripetizione, che nel diagramma a blocchi è rappresentata dalla forma seguente



Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \min\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OPPURE (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



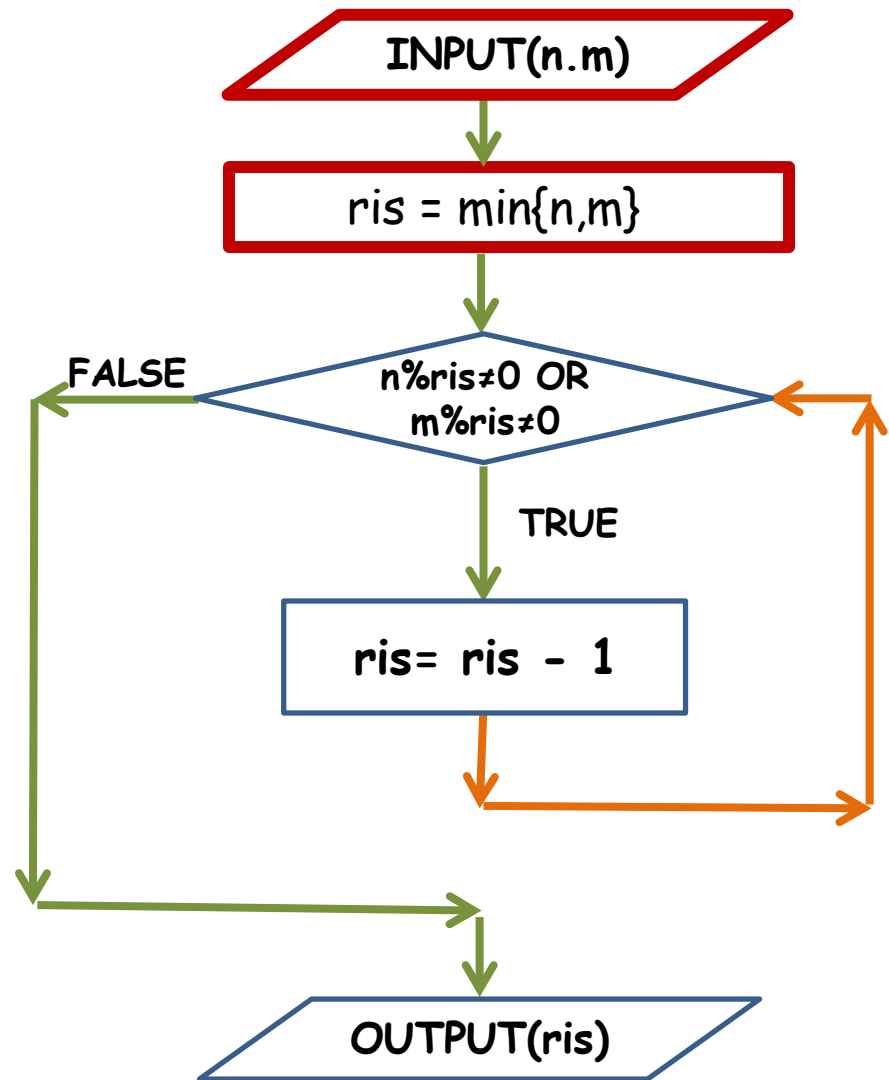
NB
 ris divide n se $n \% ris$ è zero
o anche
 ris non divide n se $n \% ris$ è diverso da zero

$n \% ris = \langle n \text{ modulo } ris \rangle$
resto della divisione di n per ris

la condizione vale se almeno una tra $(n \% ris \neq 0)$ e $(m \% ris \neq 0)$ è vera

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OPPURE (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



Esecuzione, per $n=6$, $m=21$

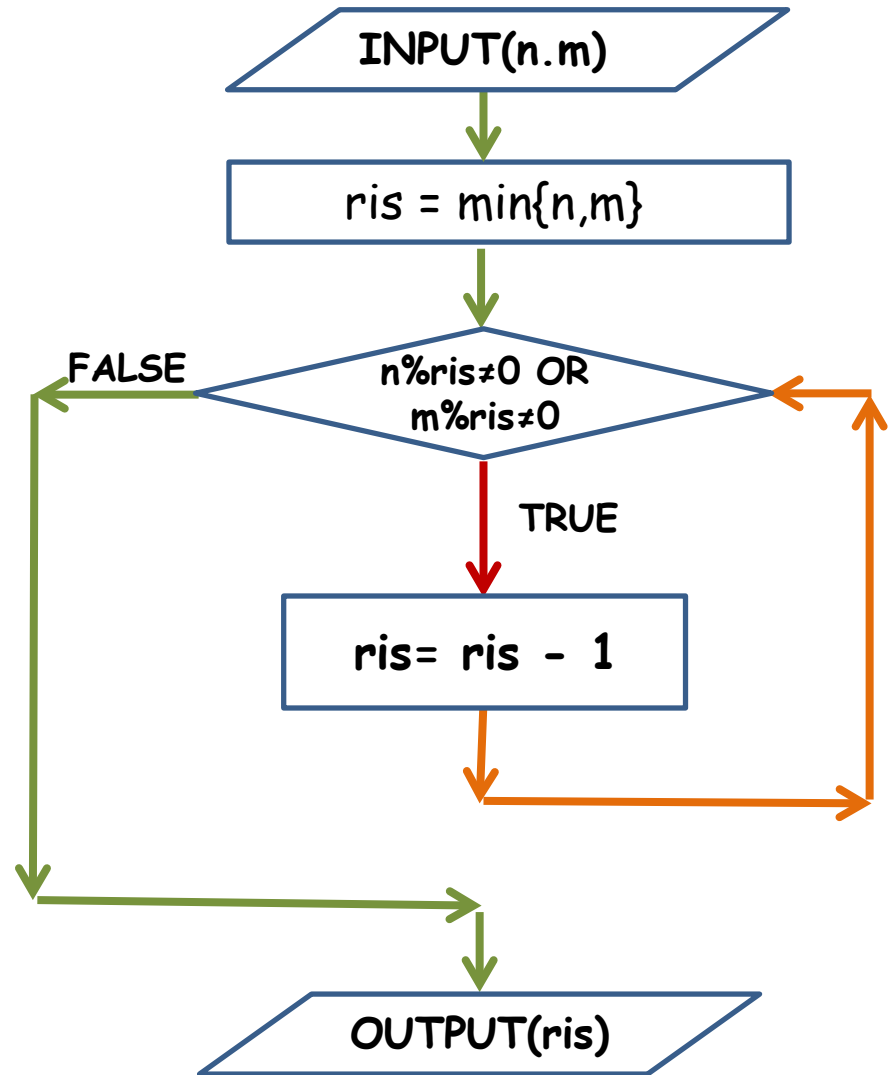
i primi due passi

ris

6

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



Esecuzione, per $n=6$, $m=21$

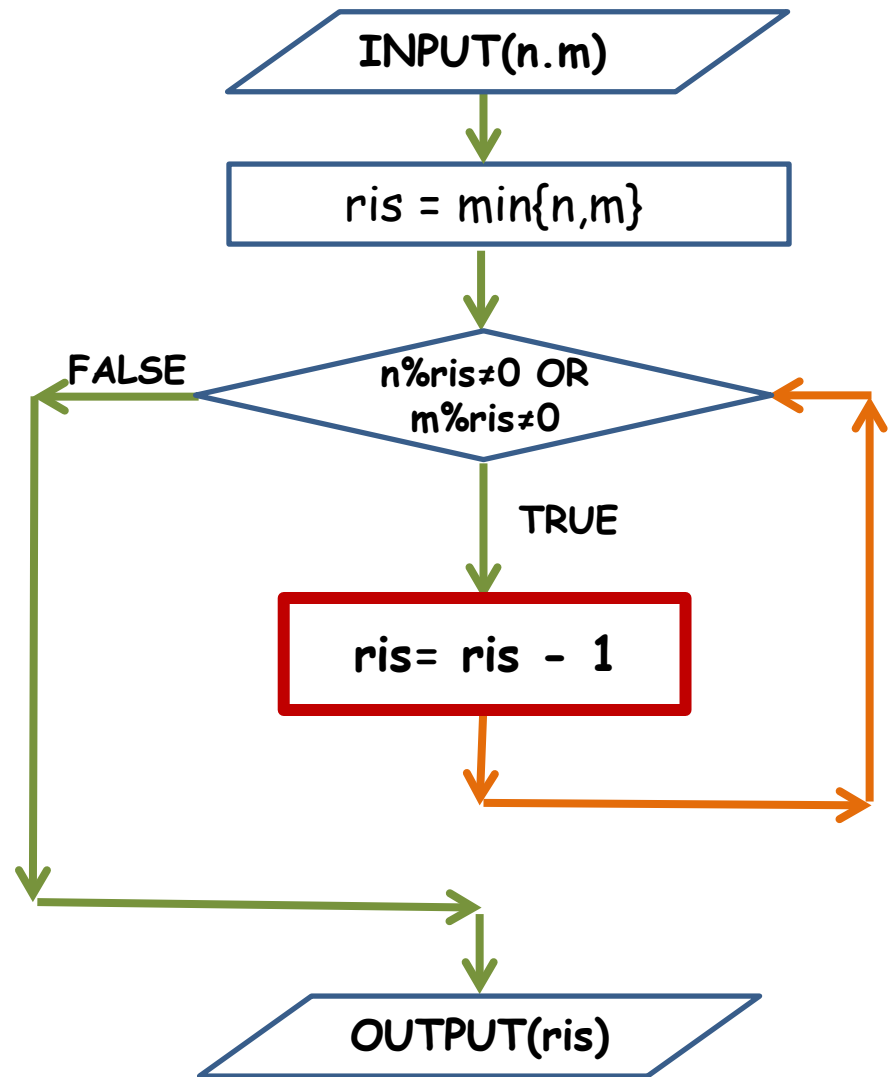
ris **6**

verifica condizione di ripetizione per la prima iterazione

($ris \% n$ diverso da zero non è vero; ma l'altra è vera; quindi almeno una è vera; quindi la condizione è vera)

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



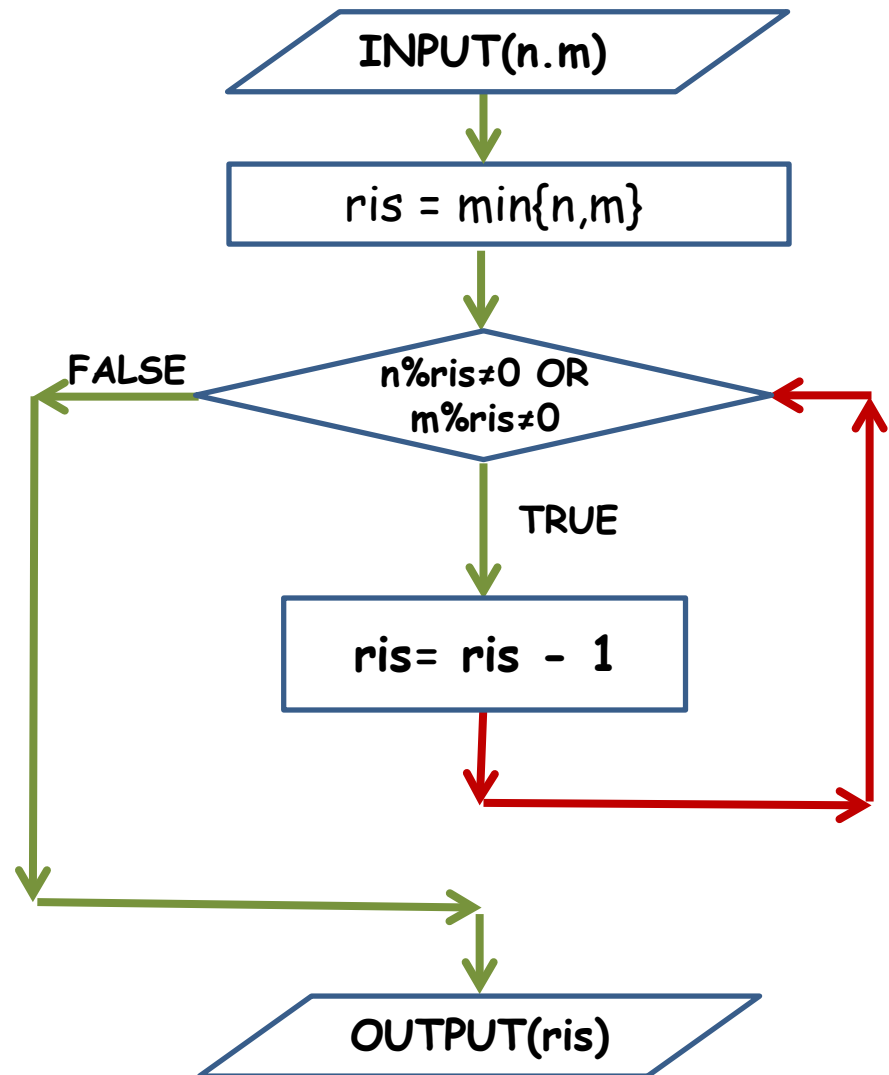
Esecuzione, per $n=6$, $m=21$

ris **5**

prima iterazione

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



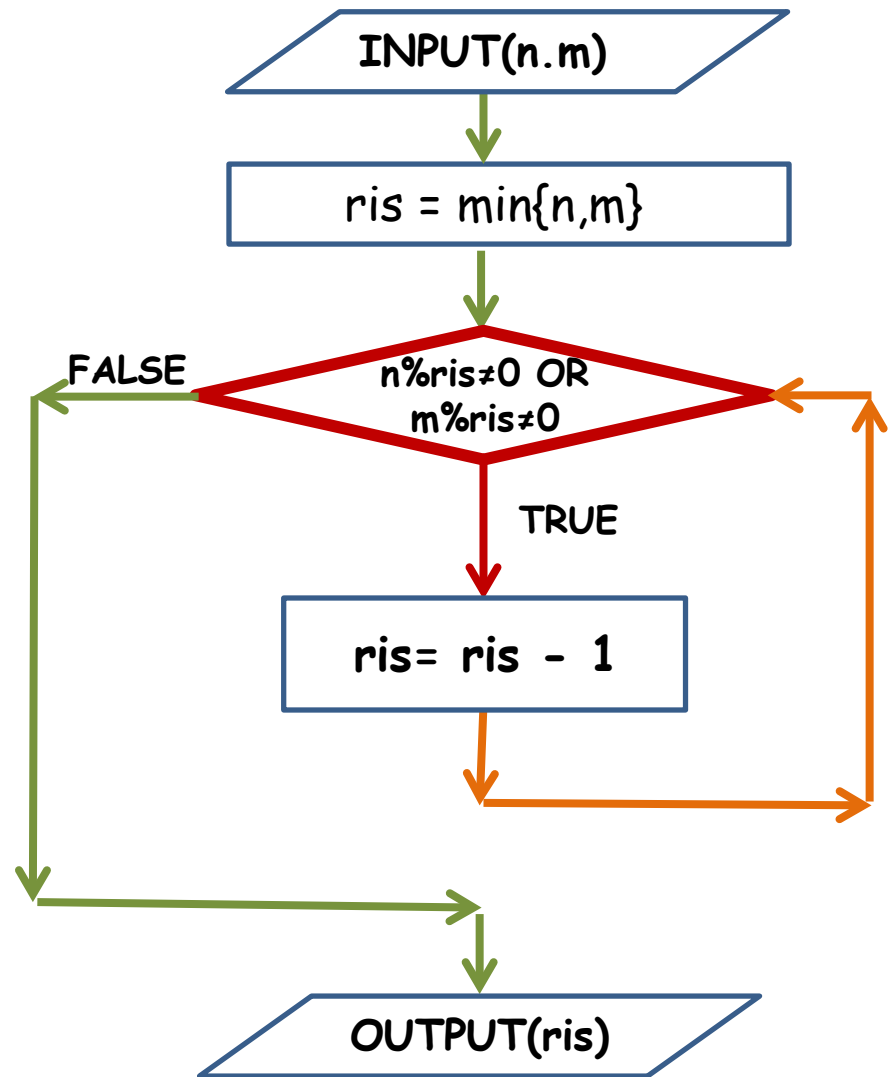
Esecuzione, per $n=6$, $m=21$

ris **5**

prima iterazione finita

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



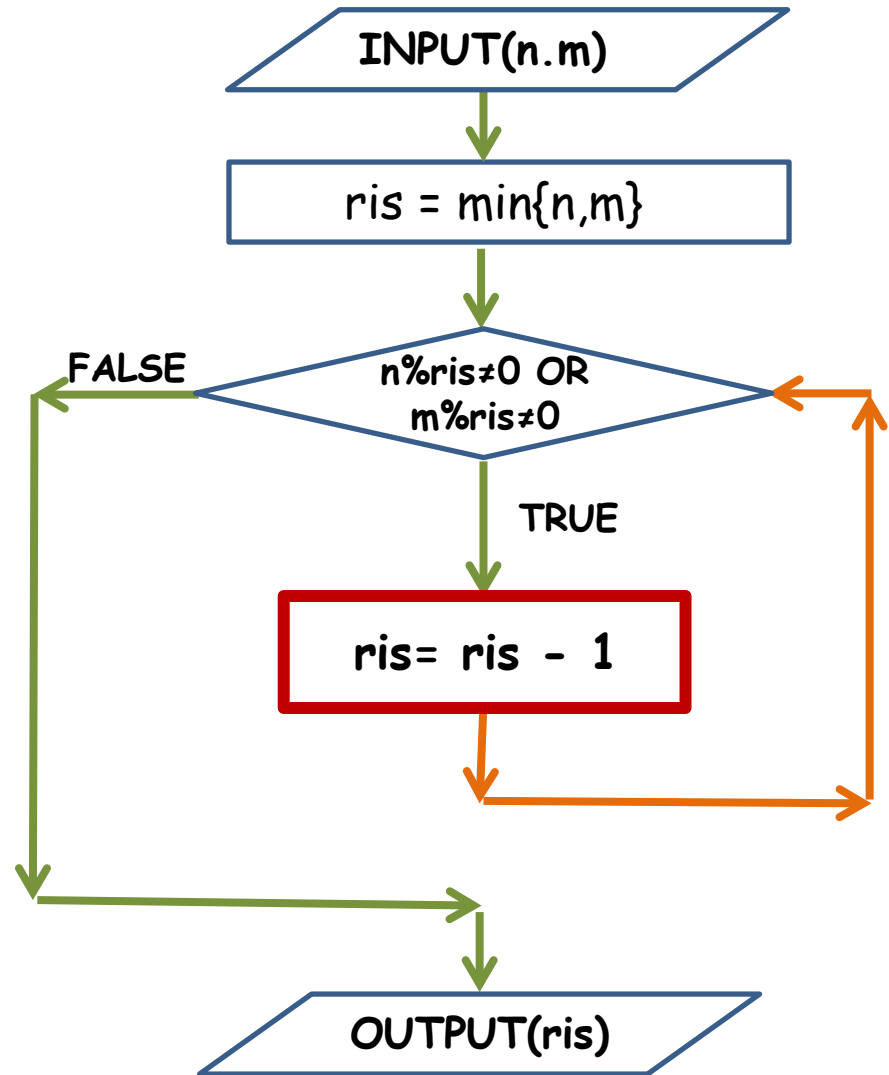
Esecuzione, per $n=6$, $m=21$

ris **5**

verifica condizione di ripetizione

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



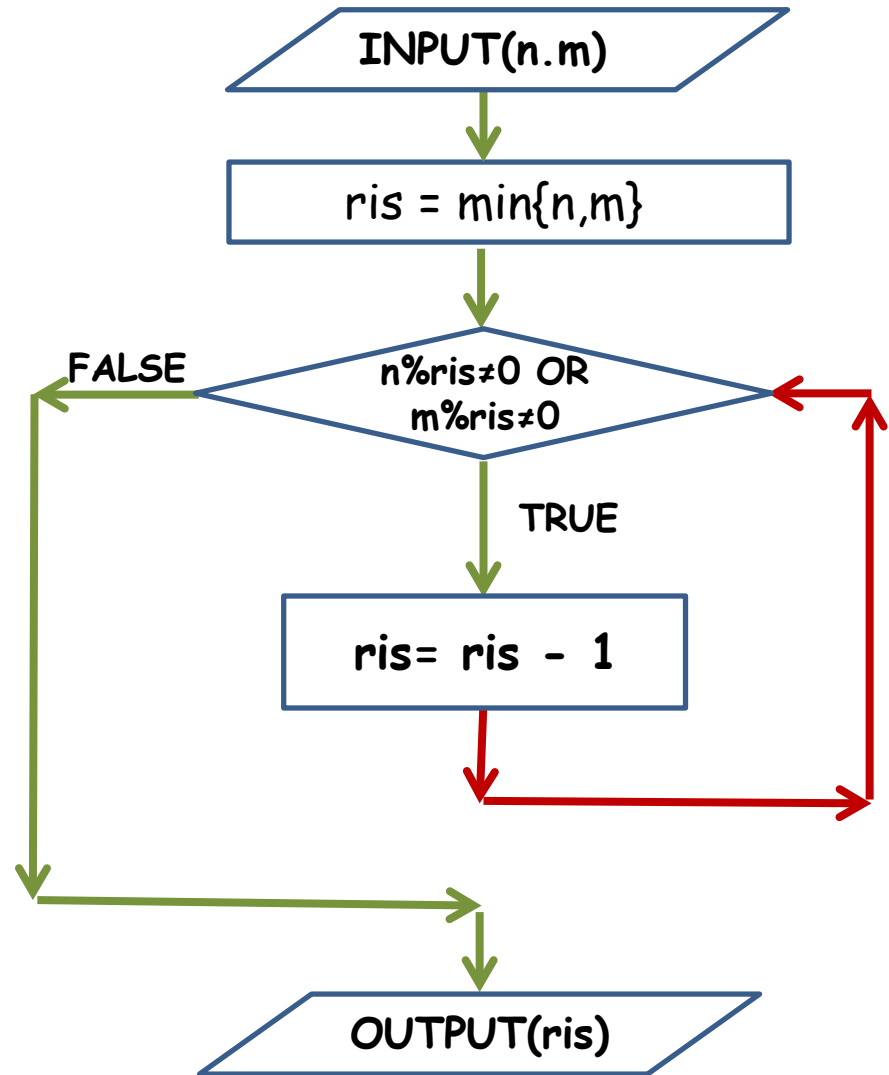
Esecuzione, per $n=6$, $m=21$

ris **4**

seconda iterazione

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



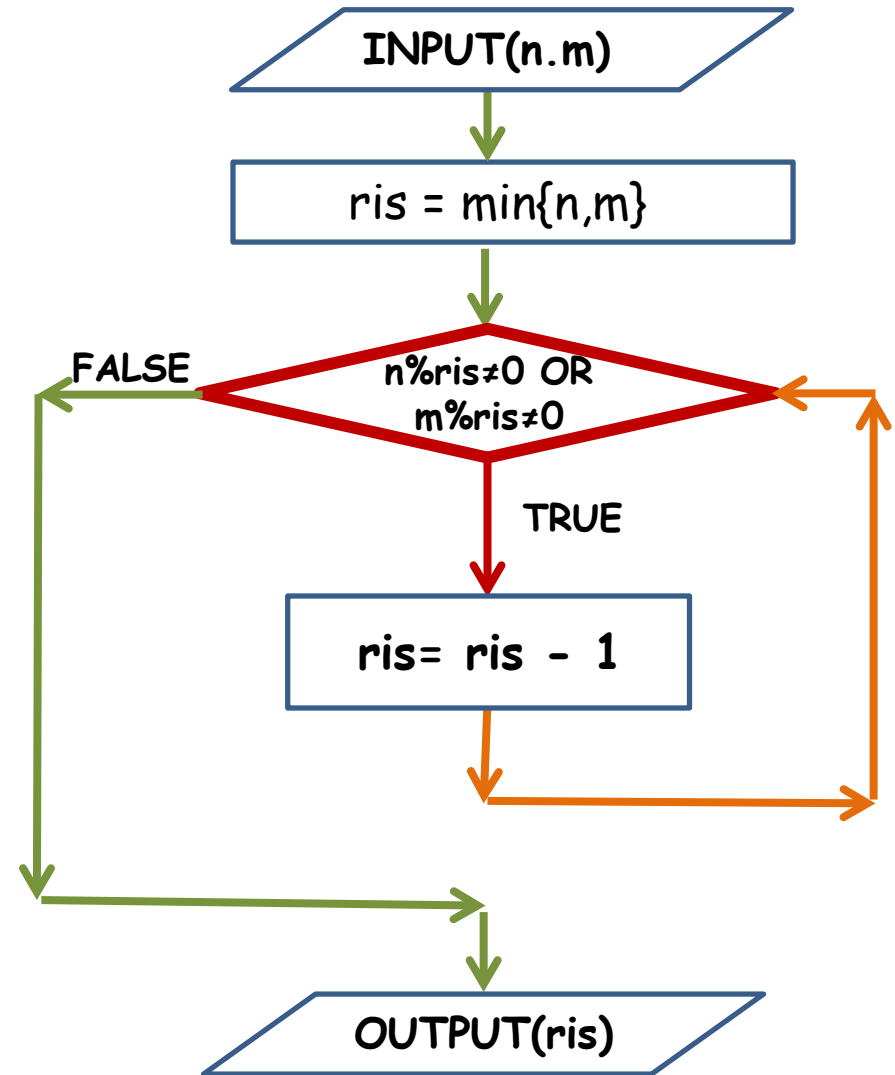
Esecuzione, per $n=6$, $m=21$

ris 4

seconda iterazione finita

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



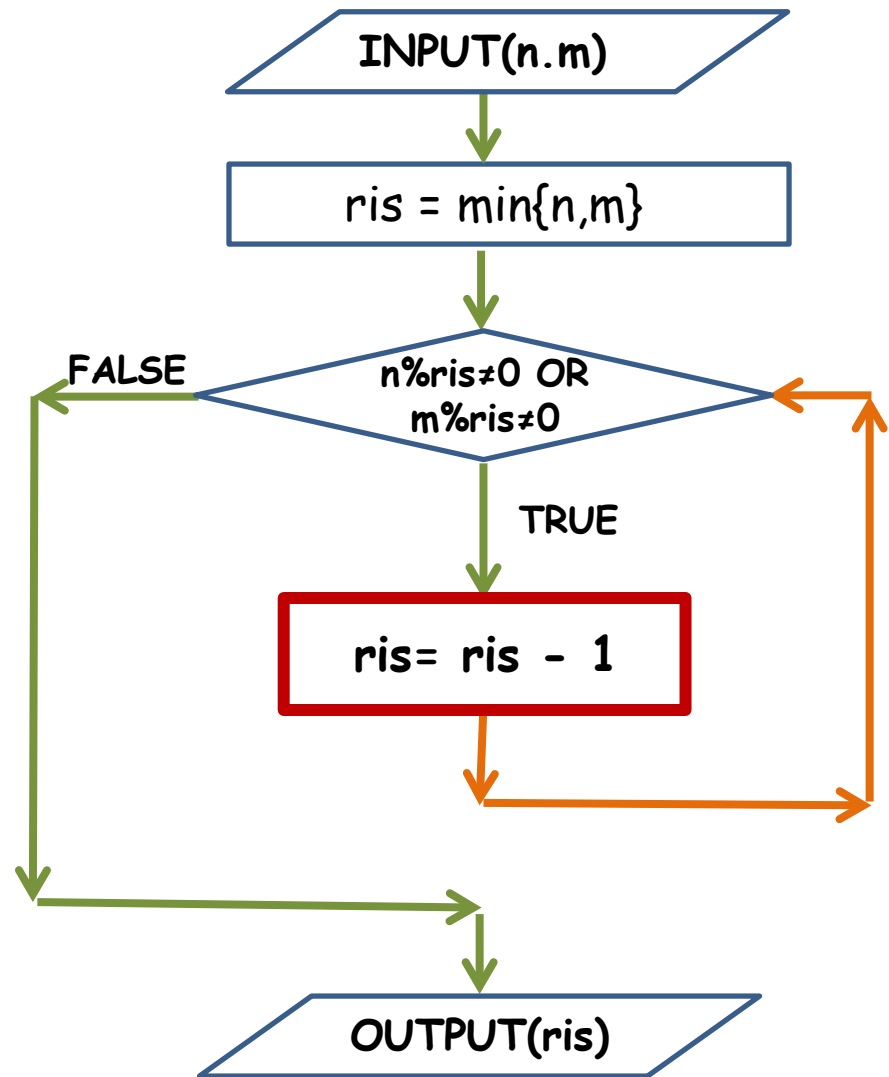
Esecuzione, per $n=6$, $m=21$

ris **4**

verifica condizione di ripetizione

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



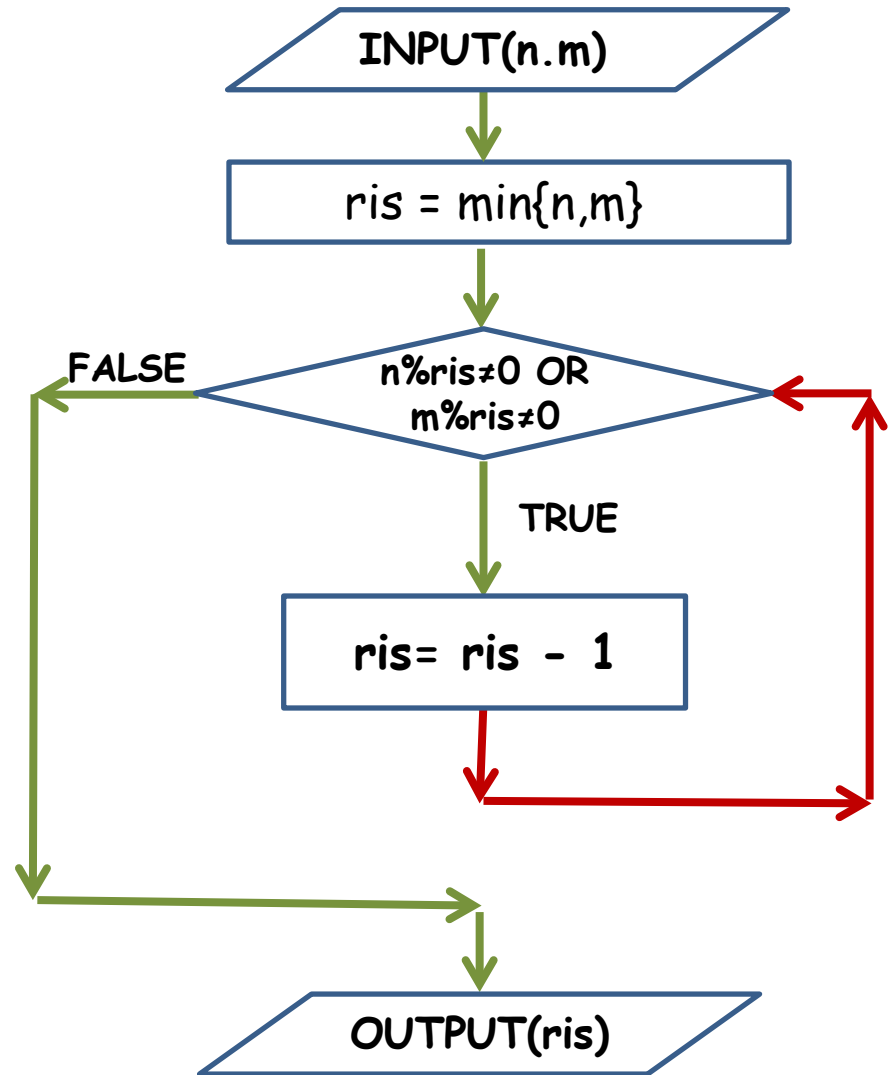
Esecuzione, per $n=6$, $m=21$

ris **3**

terza iterazione

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n,m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



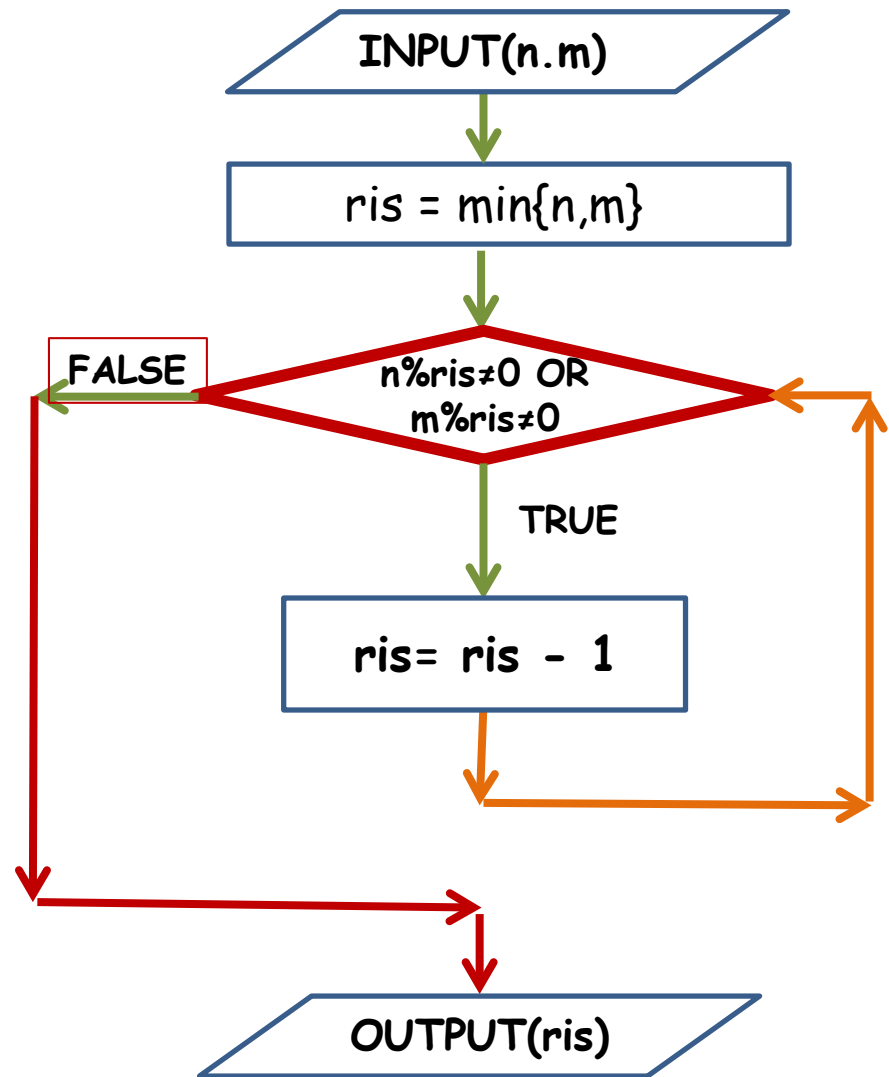
Esecuzione, per $n=6$, $m=21$

ris **3**

terza iterazione finita

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



Esecuzione, per $n=6$, $m=21$

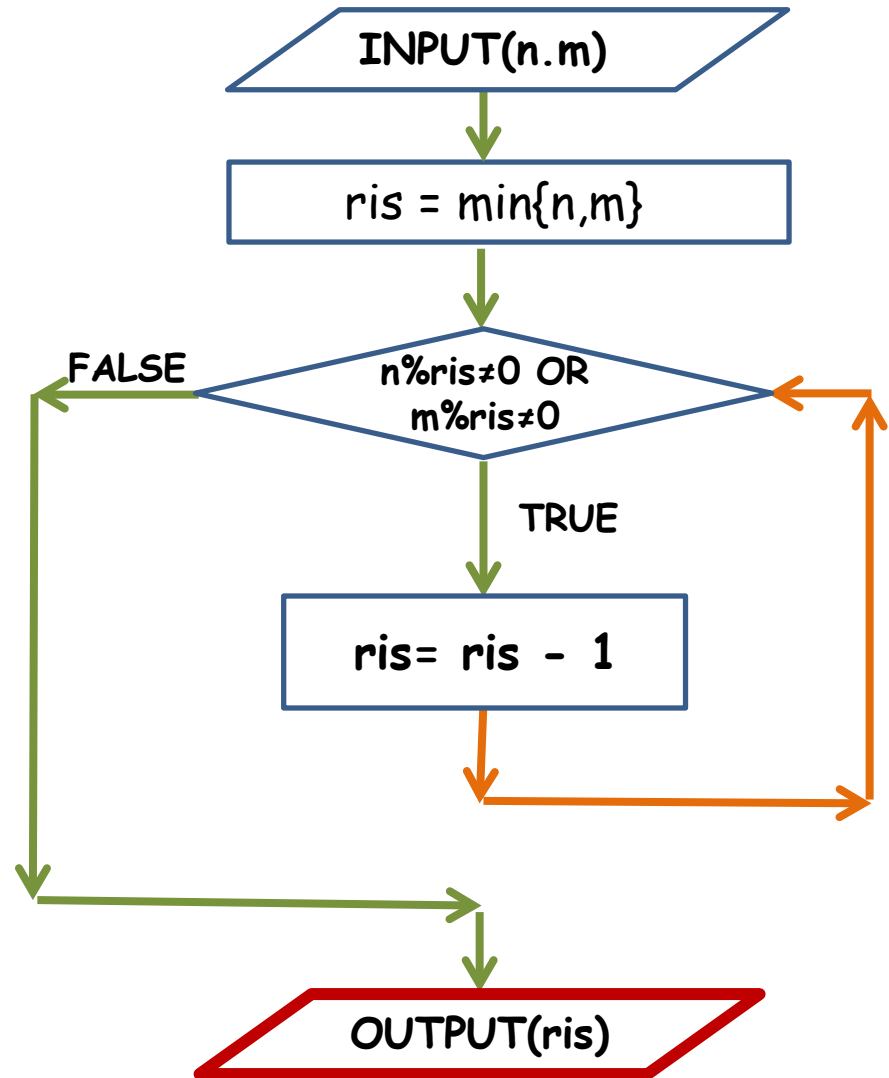
ris **3**

verifica condizione di ripetizione
ehi! fallisce!

Ma e' grave fallire?

Algoritmo strutturato per il MCD

- 0) i dati: n , m (input) e ris
- 1) INPUT (n , m)
- 2) $ris = \text{minimo}\{n, m\}$
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) $ris = ris - 1$
- 4) OUTPUT(ris)



Esecuzione, per $n=6$, $m=21$

ris **3**

OUTPUT MCD è 3

Algoritmo strutturato per il MCD (osservazione)

0) i dati: n , m (input) e ris

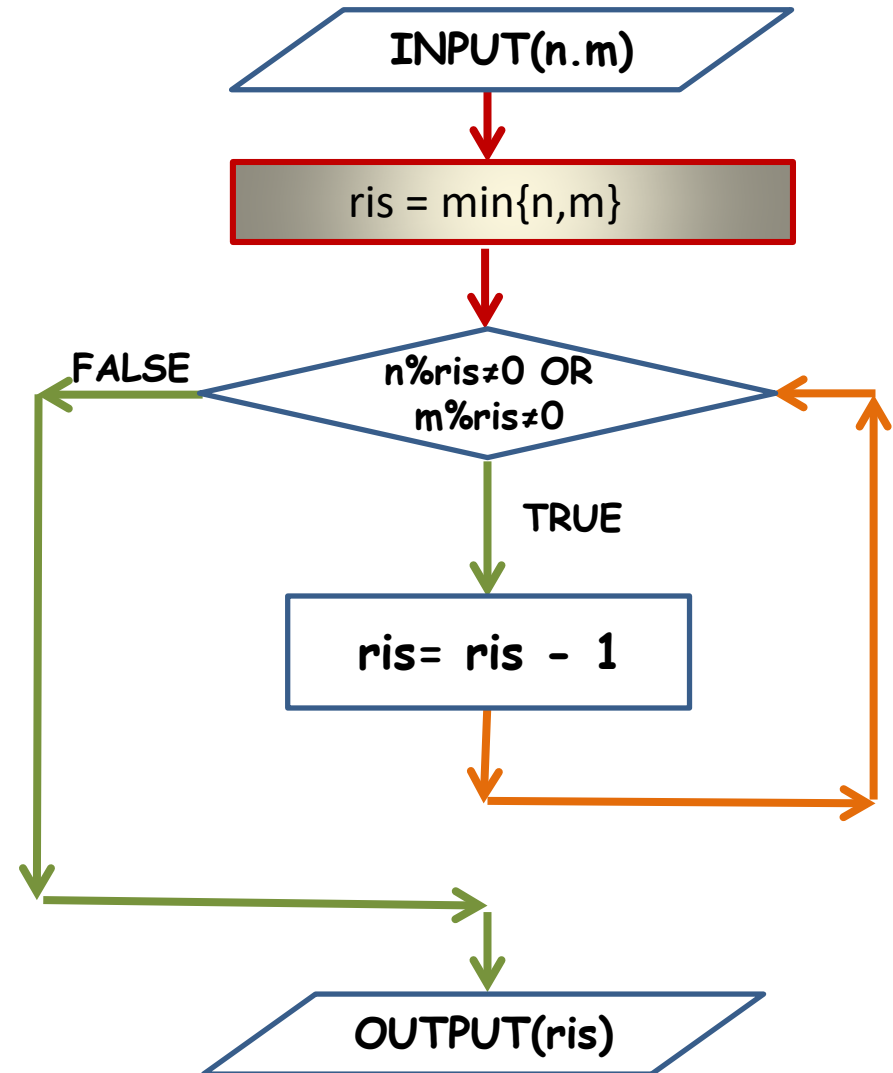
1) INPUT (n , m)

2) $ris = \text{minimo}\{n,m\}$

3)

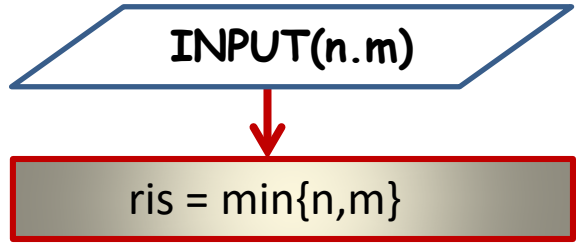


4) questa non è un'istruzione elementare! Ci vuole un algoritmo ad hoc per calcolare, dati due numeri, qual è il minimo tra i due. Qual è l'algoritmo? Con quale diagramma si può esprimere? (psst ... Serve un'istruzione condizionale ...)



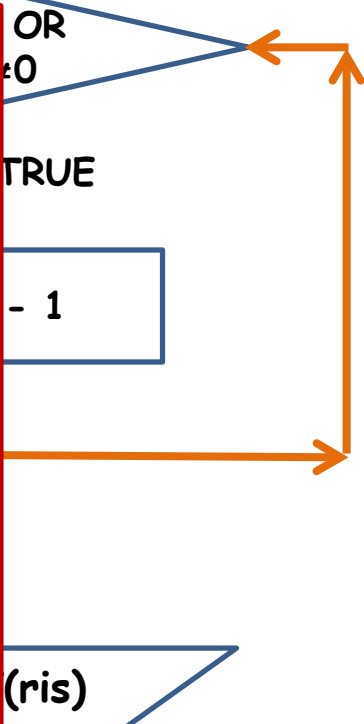
Algoritmo strutturato per il MCD (consiglio)

- 0) i dati: n, m (input) e ris
- 1) INPUT (n, m)
- 2) ris = minimo{n,m}
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
- 3.1) ris = ris - 1



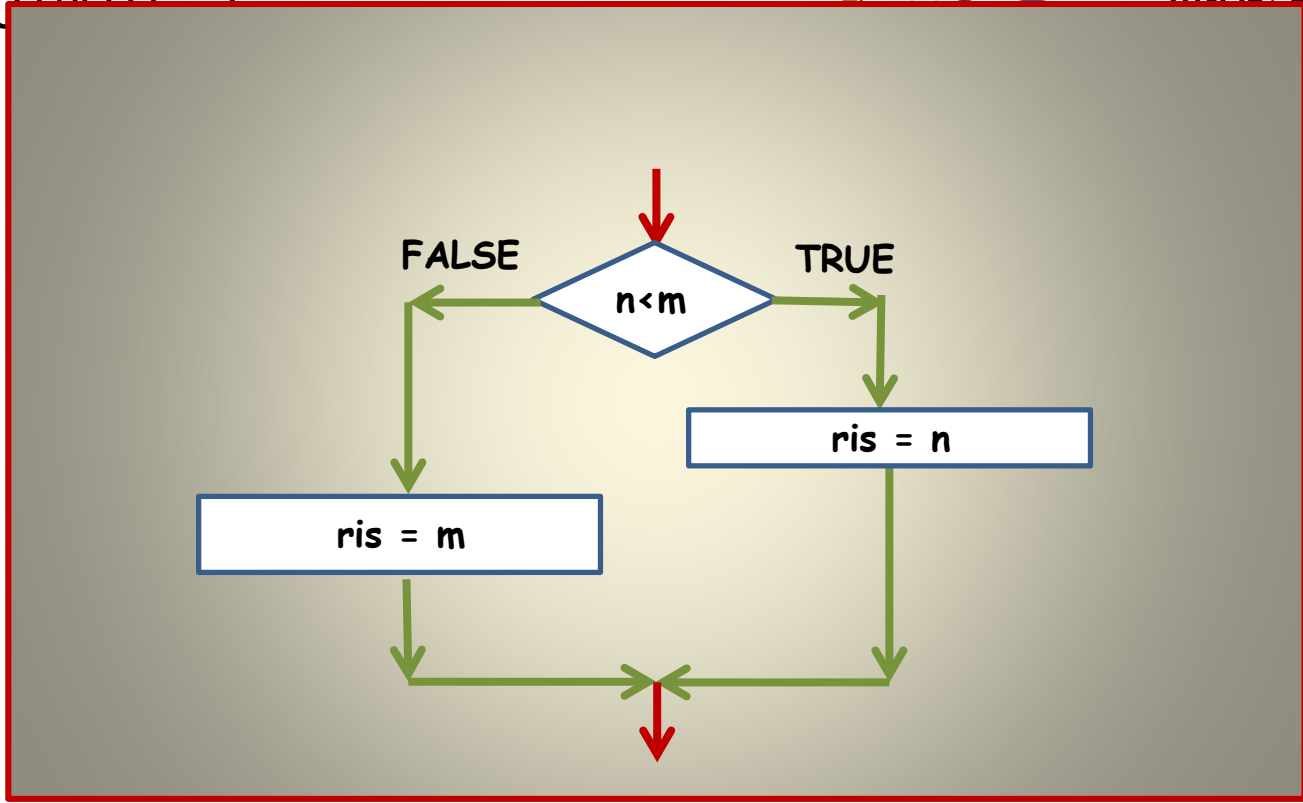
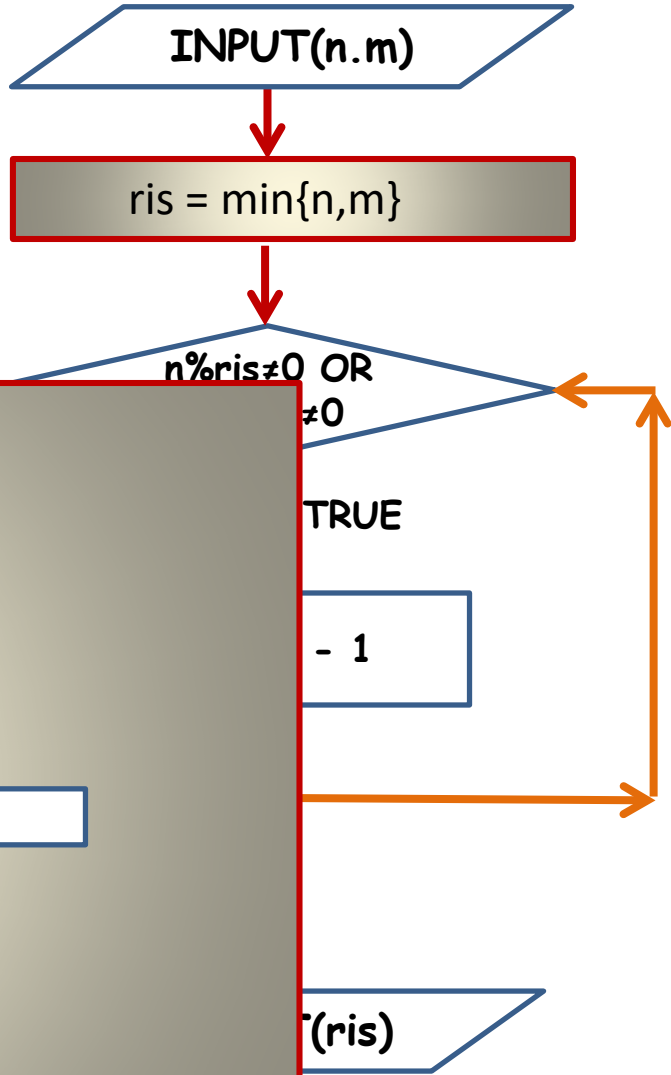
4) OU

😊 abbiamo appena letto n ed m; ora bisogna assegnare a ris il minimo valore tra i due
 al posto di ris = min{n,m} va messo il diagramma corrispondente all' algoritmo per il calcolo del minimo tra due numeri. **Riempirlo!**



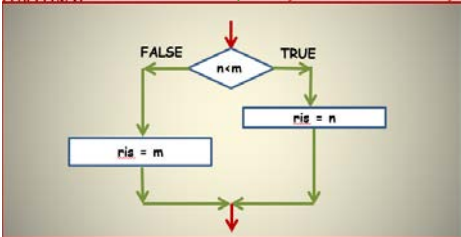
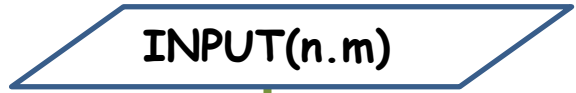
Algoritmo strutturato per il MCD (soluzione)

- 0) i dati: n, m (input) e ris
- 1) INPUT (n, m)
- 2) ris = minimo{n,m}
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
- 3.1) ris = ris - 1
- 4) OUTPUT(ris)

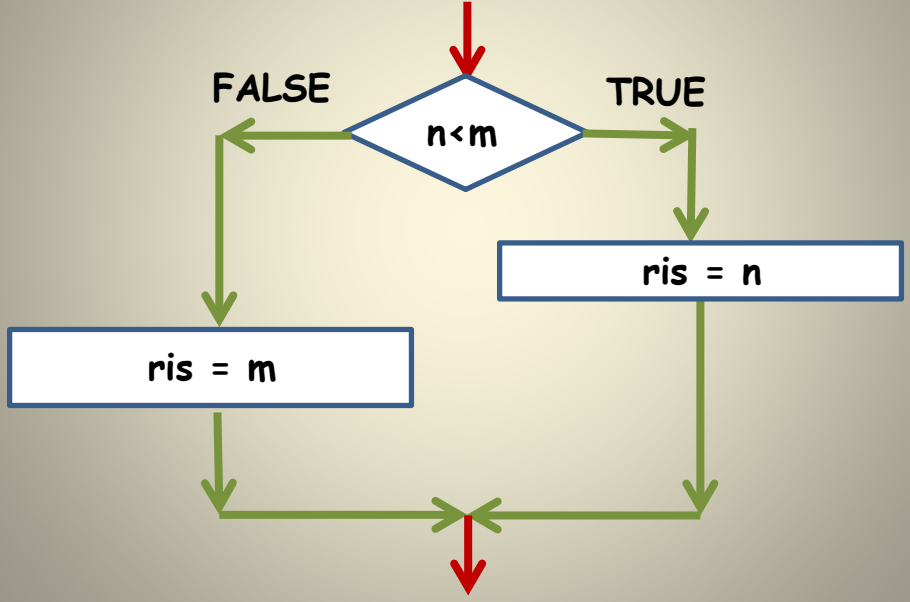


Algoritmo strutturato per il MCD (ancora)

- 0) i dati: n, m (input) e ris
- 1) INPUT (n, m)
- 2) ris = minimo{n,m}
- 3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)
 - 3.1) ris = ris - 1
- 4) OUTPUT(ris)



😊 riscrivere il passo 2



Algoritmo strutturato per il MCD (diagramma completo)

0) i dati: n , m (input) e ris

1) INPUT (n , m)

2) SE $n < m$

$ris = n$

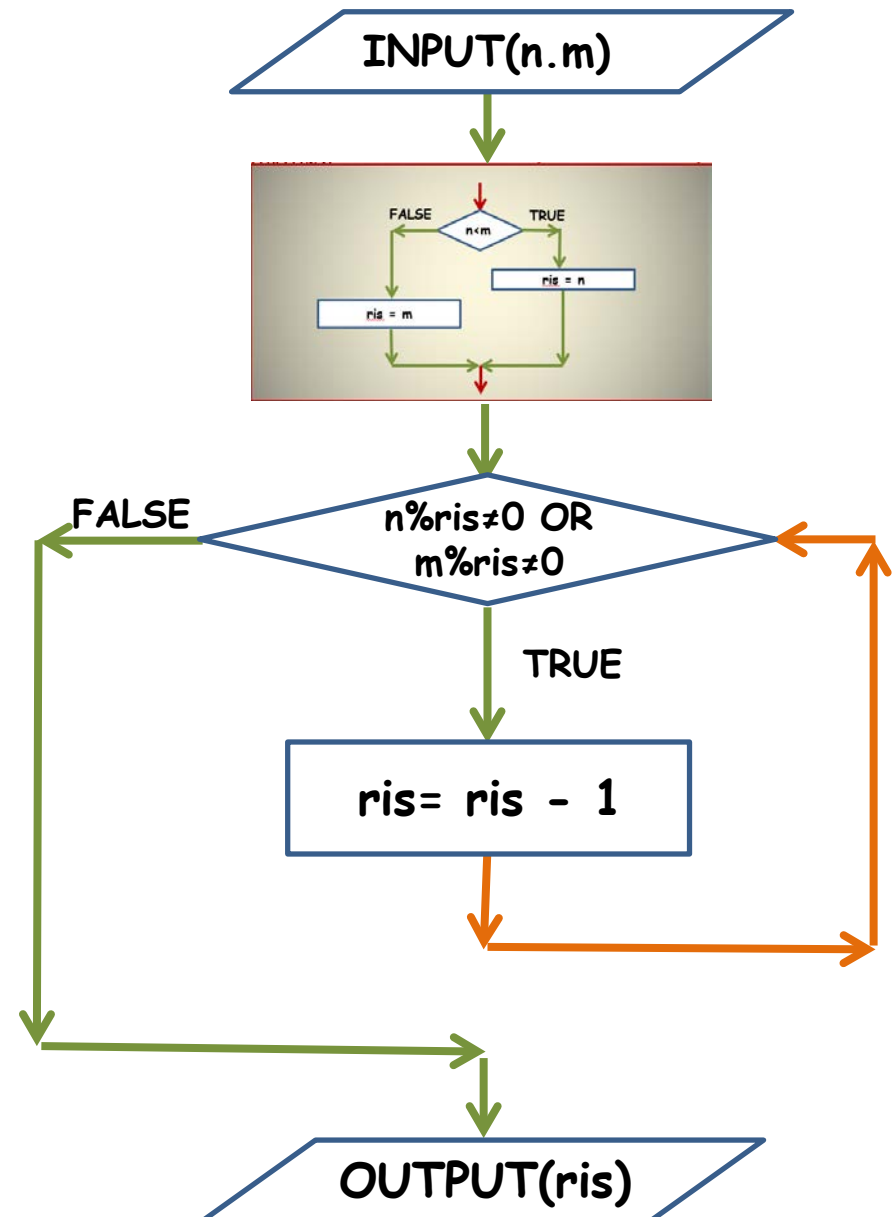
altrimenti

$ris = m$

3) FINTANTO CHE (ris NON DIVIDE n) OR (ris NON DIVIDE m)

3.1) $ris = ris - 1$

4) OUTPUT(ris)



Algoritmo strutturato per stampare il massimo tra tre numeri dati in input

- 0) i dati: n_1, n_2, n_3 (input), e max
- 1) INPUT (n_1, n_2, n_3)
- 2)
- 3)
- 4)



Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (consiglio)

- 0) i dati: n_1, n_2, n_3 (input), e max
- 1) INPUT (n_1, n_2, n_3)
- 2)
- 3)
- 4)

controlliamo i dati, più o meno uno per volta, e cerchiamo di ricordare, dopo ogni controllo, chi è il più grande tra quelli che abbiamo visto ... e poi continuiamo con il prossimo dato da controllare.

Però la prima volta controlliamo i primi due dati ...

Ogni volta ... cioè ad ogni controllo, usiamo **max** per contenere il massimo tra i dati che abbiamo appena controllato.

Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (consiglio2)

- 0) i dati: n_1, n_2, n_3 (input), e max
- 1) INPUT (n_1, n_2, n_3)
- 2) ... $max = \text{massimo}\{n_1, n_2\}$
- 3)
- 4)

controlliamo i dati, più o meno uno per volta, e cerchiamo di ricordare, dopo ogni controllo, chi è il più grande tra quelli che abbiamo visto ... e poi continuiamo con il prossimo dato da controllare.

Però la prima volta controlliamo i primi due dati ...

Ogni volta ... cioè ad ogni controllo, usiamo max per contenere il massimo tra i dati che abbiamo appena controllato.

All'inizio max è il più grande tra n_1 ed n_2 . Questa è l'**inizializzazione** di max .

poi potrebbe cambiare: bisogna controllare

Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (consiglio3)

- 0) i dati: $n1, n2, n3$ (input), e max
- 1) INPUT ($n1, n2, n3$)
- 2) ... $max = \text{massimo}\{n1, n2\}$
- 3)
- 4)

controlliamo i dati, più o meno uno per volta, e cerchiamo di ricordare, dopo ogni controllo, chi è il più grande tra quelli che abbiamo visto ... e poi continuiamo con il prossimo dato da controllare.

Però la prima volta controlliamo i primi due dati ...

Ogni volta ... cioè ad ogni controllo, usiamo max per contenere il massimo tra i dati che abbiamo appena controllato.

All'inizio max è il più grande tra $n1$ ed $n2$. Questa è l'**inizializzazione** di max .

poi potrebbe cambiare: bisogna controllare



come è fatto il passo 2?

Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (passo 2 a posto)

0) i dati: $n1$, $n2$, $n3$ (input), e max

1) INPUT ($n1$, $n2$, $n3$)

2) Inizializzazione di max :

SE $n1 > n2$

$max = n1$

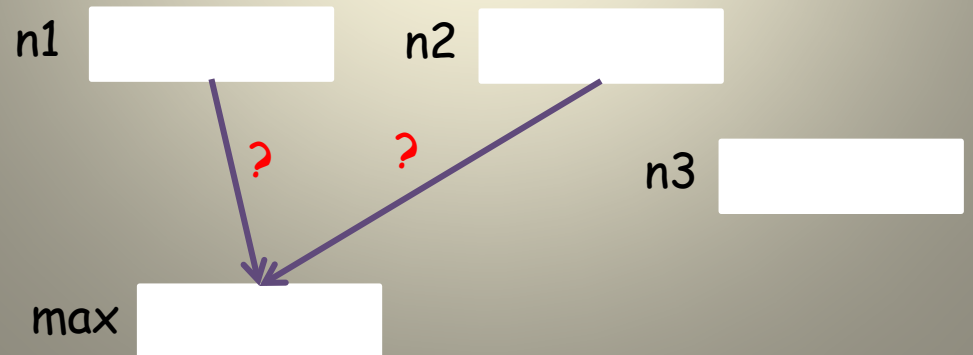
ALTRIMENTI

$max = n2$

3) ? e poi? 😊

4)

All'inizio max è inizializzato con il più grande tra $n1$ ed $n2$



Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (consiglio criptico)

0) i dati: $n1$, $n2$, $n3$ (input), e max

1) INPUT ($n1$, $n2$, $n3$)

2) Inizializzazione di max :

SE $n1 > n2$

$max = n1$

ALTRIMENTI

$max = n2$

3) ?

4)

All'inizio max e' il piu' grande tra $n1$ ed $n2$

poi potrebbe cambiare: bisogna controllare

$n1$

$n2$

$n3$

max

?



Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (passo 3 a posto)

0) i dati: $n1$, $n2$, $n3$ (input), e max

1) INPUT ($n1$, $n2$, $n3$)

2) Inizializzazione di max :

SE $n1 > n2$

$max = n1$

ALTRIMENTI

$max = n2$

3) SE $n3 > max$

$max = n3$

4) e poi? 😊

poi potrebbe cambiare: bisogna controllare

$n1$

$n2$

$n3$

max



?

Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (passo 4 a posto)

- 0) i dati: $n1, n2, n3$ (input), e max
- 1) INPUT ($n1, n2, n3$)
- 2) Inizializzazione di max :
SE $n1 > n2$
 $max = n1$
ALTRIMENTI
 $max = n2$
- 3) SE $n3 > max$
 $max = n3$
- 4) OUTPUT(max)

fine ...

abbiamo controllato tutti i numeri;

per ogni numero controllato, se era più grande di quello in max , lo sostituiva in max ;

alla fine in max c'è il numero che non è mai stato sconfitto da altri ...

$n1$

$n2$

$n3$

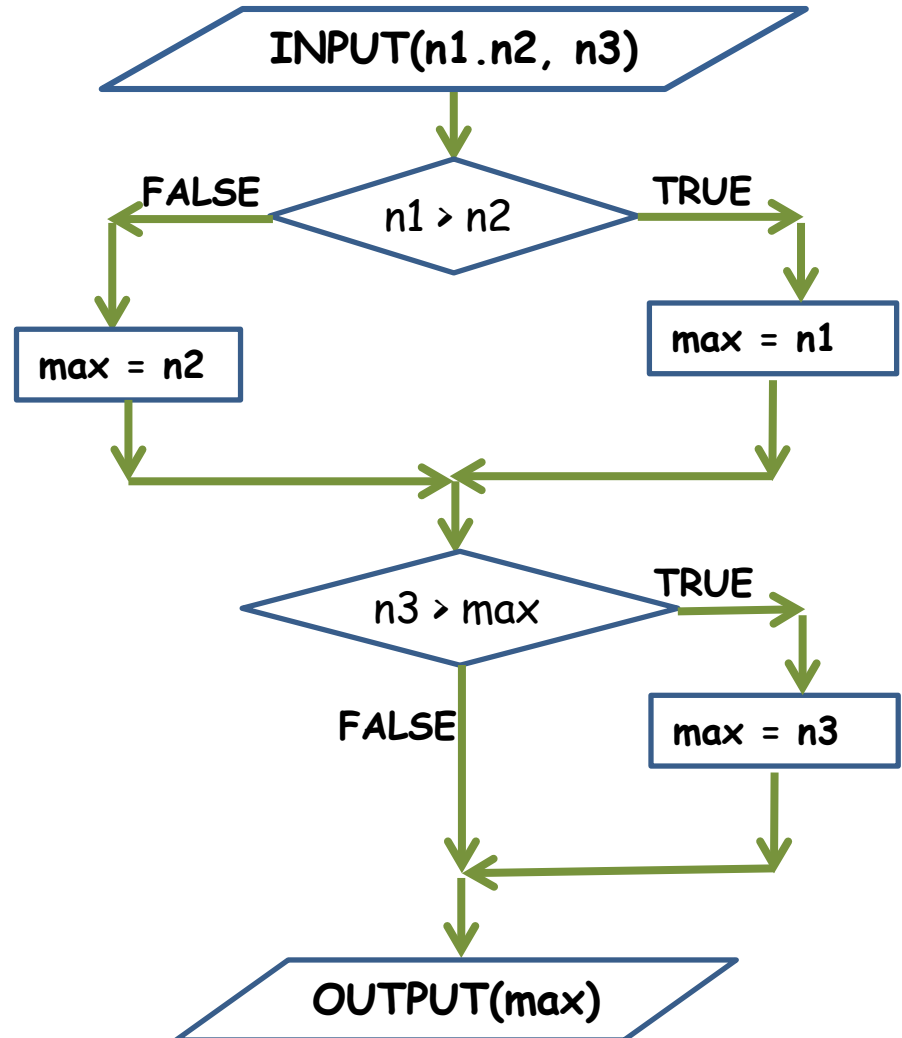
max



Diagramma di flusso?

Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (resa grafica)

- 0) i dati: $n1$, $n2$, $n3$ (input), e max
- 1) INPUT ($n1$, $n2$, $n3$)
- 2) Inizializzazione di max :
SE $n1 > n2$
 $max = n1$
ALTRIMENTI
 $max = n2$
- 3) SE $n3 > max$
 $max = n3$
- 4) OUTPUT(max)



Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (ESERCIZIO)

eseguire l'algoritmo qui sotto per **diverse istanze** del problema (20, 10, 9), (234, 1234, 61) ...

Quante istanze dobbiamo definire per essere sicuri di provare l'algoritmo coprendo tutti i casi possibili? **Provare a definire un numero minimo di istanze, scrivendo una spiegazione del perché si è scelta ciascuna di loro.**

0) i dati: n_1 , n_2 , n_3 (input), e max

1) INPUT (n_1 , n_2 , n_3)

2) Inizializzazione di max :

SE $n_1 > n_2$

$max = n_1$

ALTRIMENTI

$max = n_2$

3) SE $n_3 > max$

$max = n_3$

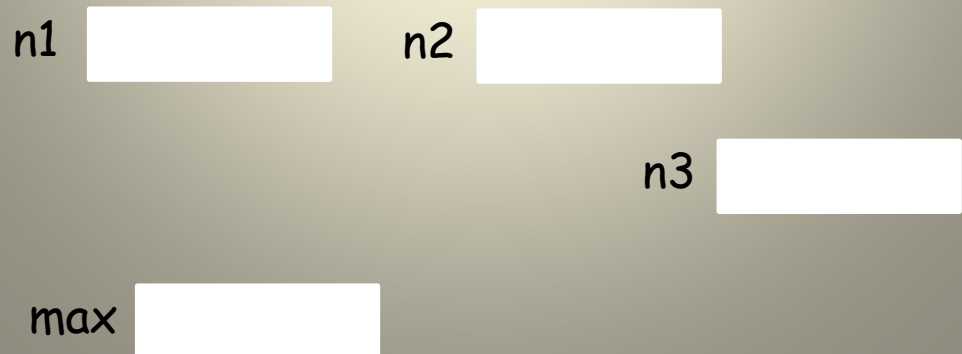
4) OUTPUT(max)

controlliamo i dati, più o meno uno per volta, e cerchiamo di ricordare, dopo ogni controllo, chi è il più grande tra quelli che abbiamo visto ... e poi continuiamo con il prossimo dato da controllare.

Però la prima volta controlliamo i primi due dati ...

Ogni volta ... cioè ad ogni controllo, usiamo **max** per contenere il massimo tra i dati che abbiamo appena controllato.

poi potrebbe cambiare: bisogna controllare



Algoritmo strutturato per stampare il massimo tra tre numeri dati in input (ESERCIZIO)

- ☺ eseguire l'algoritmo usando il diagramma di flusso per diverse istanze del problema ... quelle di prima.
- ☺ Ad ogni esecuzione evidenziare quale istruzione viene eseguita e come prosegue il flusso

n1 n2

n3

max

