Università di Salerno
Dipartimento di
Ingegneria Industriale

UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

AlgoBuild: Strutture selettive, iterative ed array

Prof. Arcangelo Castiglione

A.A. 2016/17

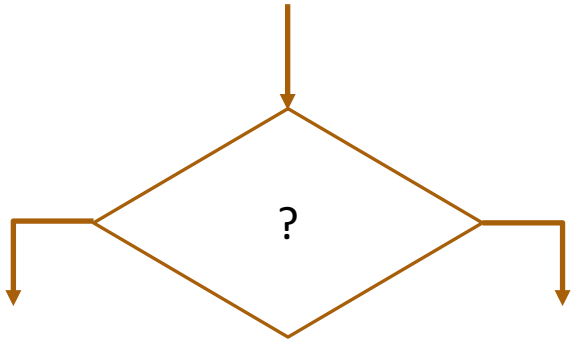
AlgoBuild

AlgoBuild : Strutture iterative e selettive OUTLINE

- Struttura selettiva
  - *Esempi*
- Struttura iterativa FOR (ciclo a condizione iniziale)
  - *Esempi*
- Struttura iterativa DO/WHILE (ciclo a condizione finale)
  - *Esempi*
- Array

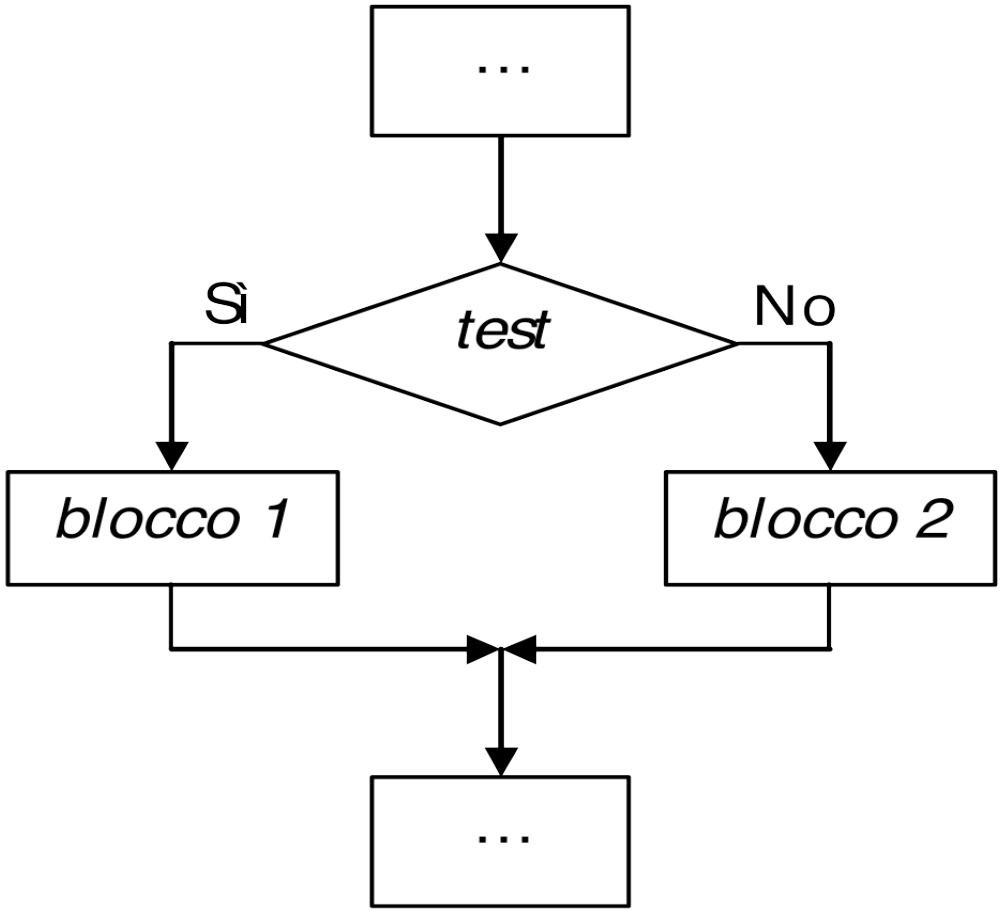
Diagrammi di Flusso: Blocco di Decisione Binaria (o Condizionale)

- Possono essere presenti **istruzioni condizionali**, la cui esecuzione dipende cioè da scelte effettuate in base ai dati
- Concettualmente, possiamo immaginare che il flusso di esecuzione si **ramifichi**
  - In base ad una **condizione** viene deciso se eseguire un'operazione **oppure** un'altra



Diramazione
(condizionale)

Strutture di Controllo: Selezione a Due Vie

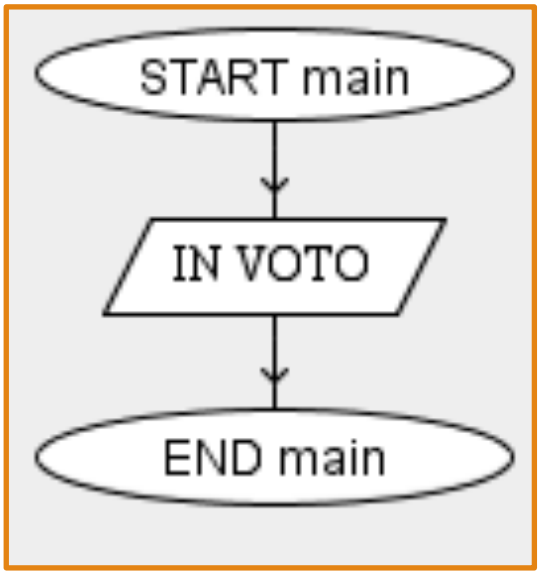


Struttura Selettiva IF – Esempio

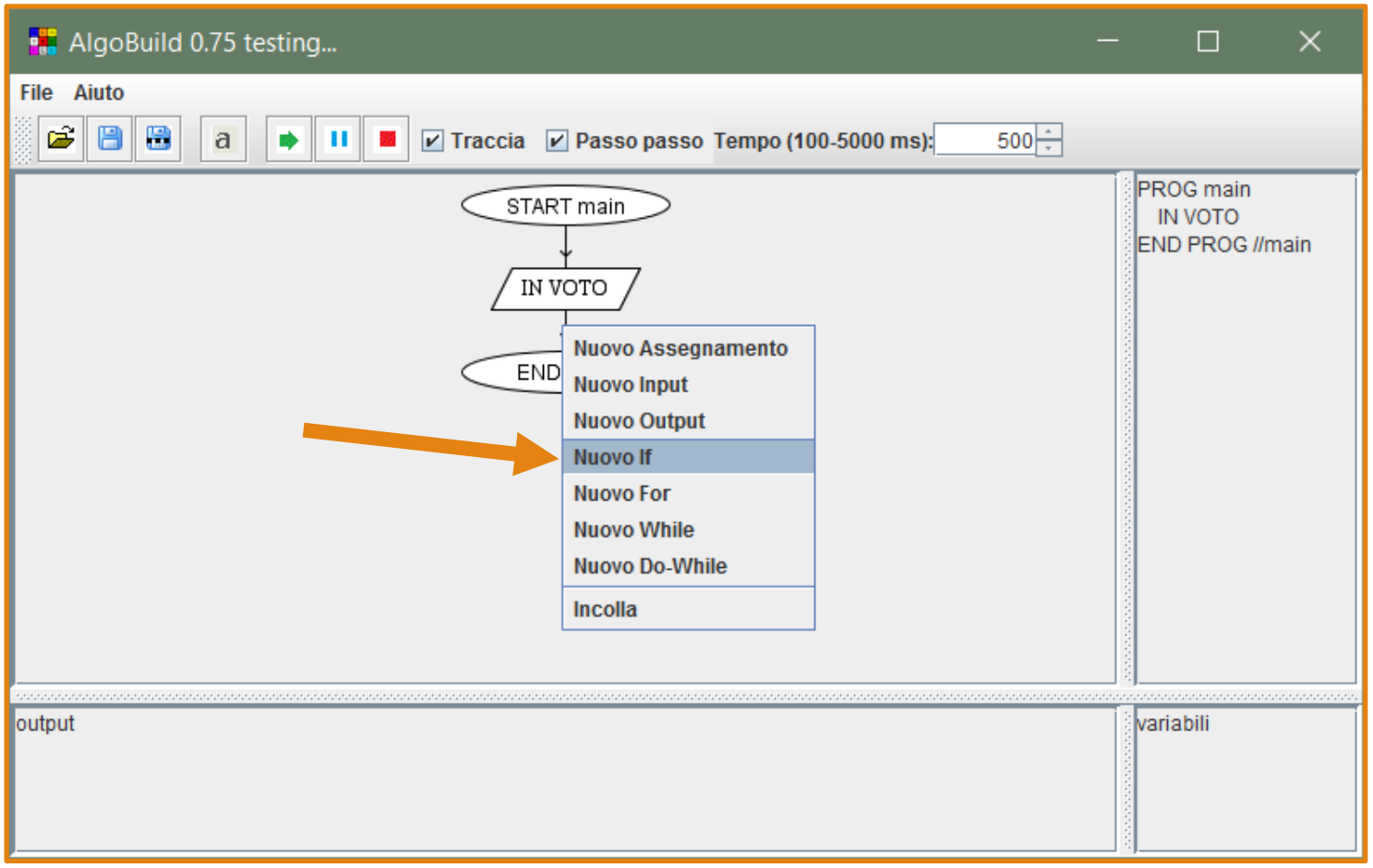
- Definiamo un diagramma di flusso che
  - Prende in **input** la variabile **VOTO** da parte dell'utente
  - Mostra in output **«Superato!»** **se** VOTO è maggiore o uguale di 18
    - **«NON Superato»**, **altrimenti**

Struttura Selettiva IF

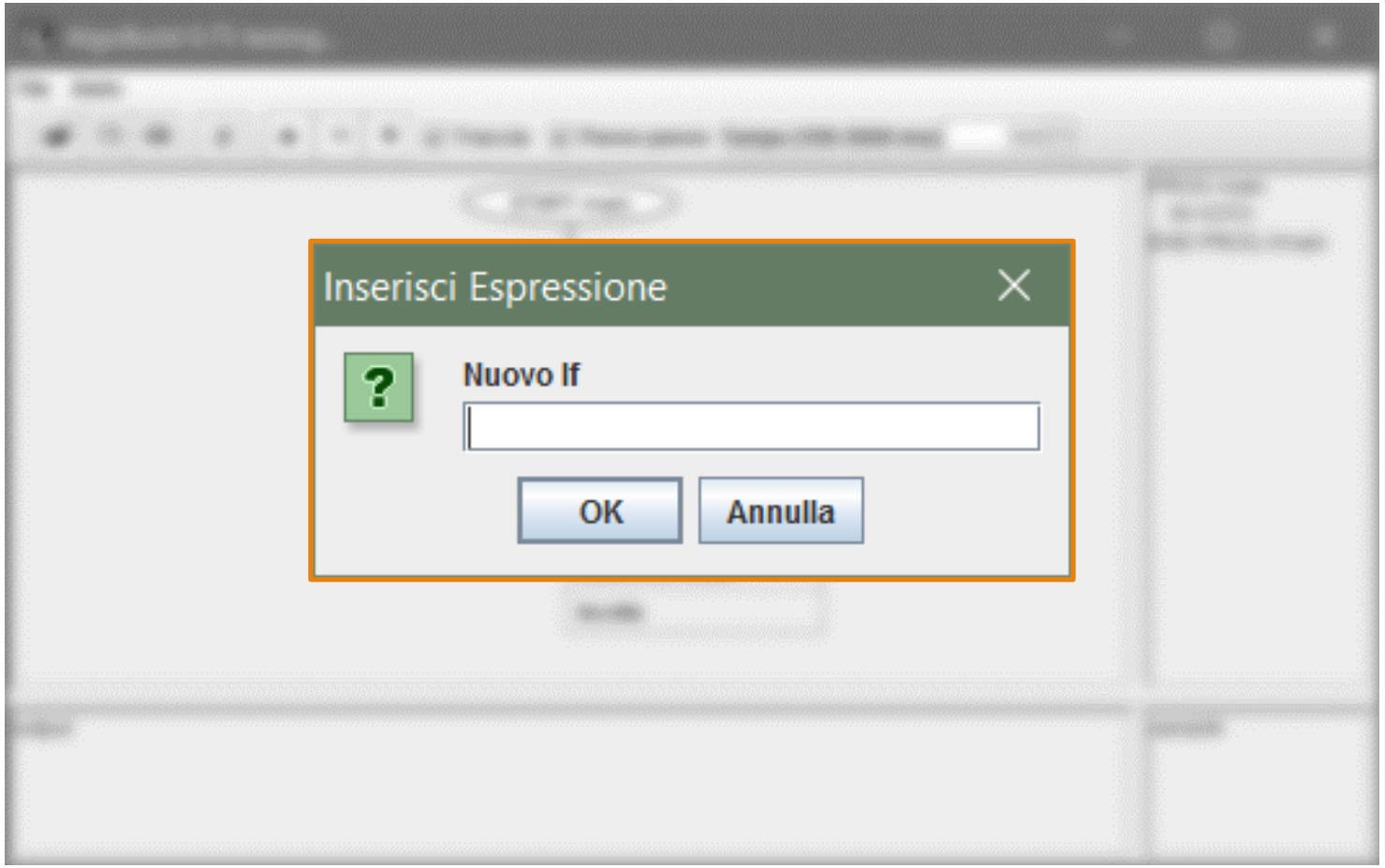
- Iniziamo dal prendere in input la variabile **VOTO**



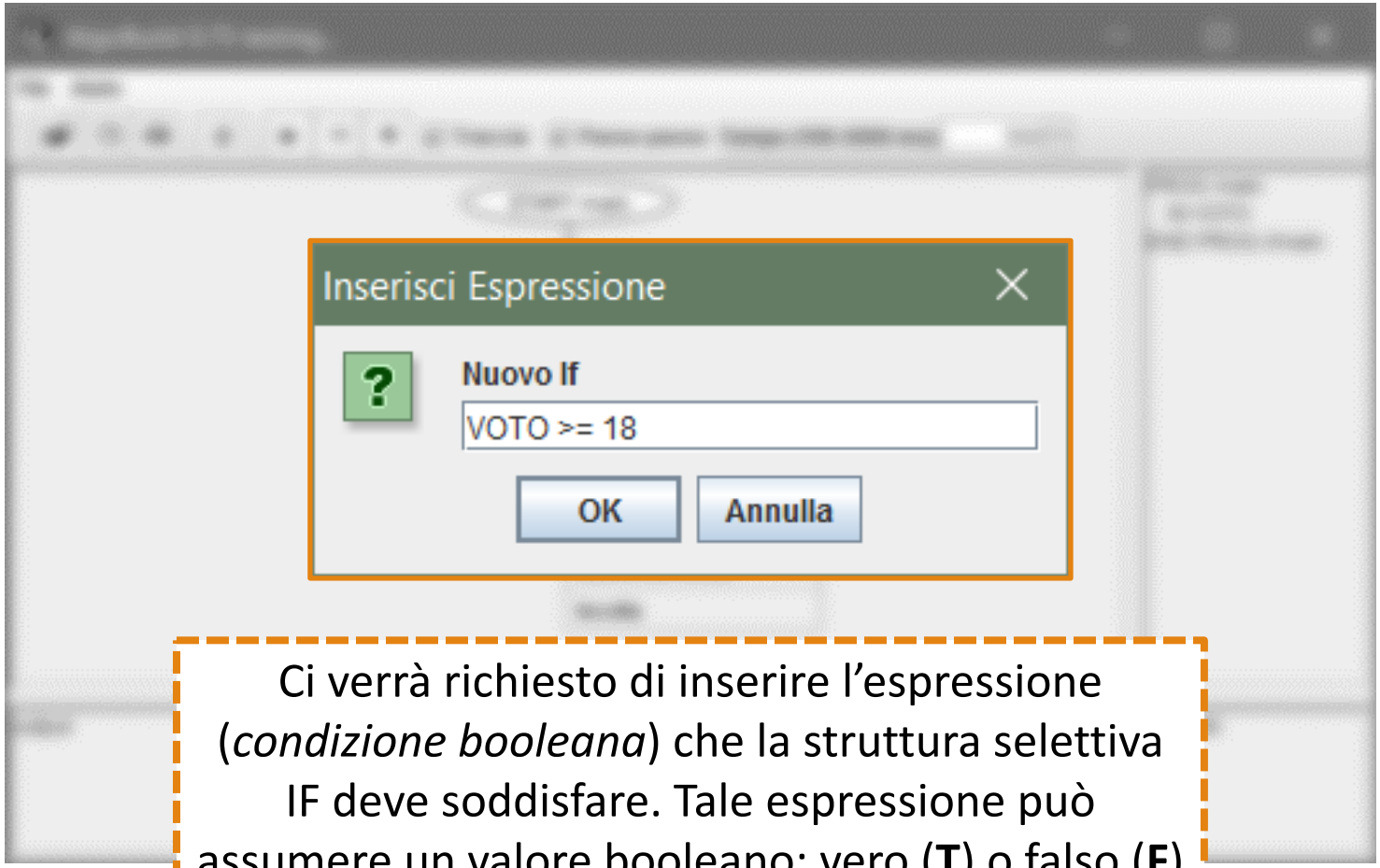
Struttura Selettiva IF



Struttura Selettiva IF

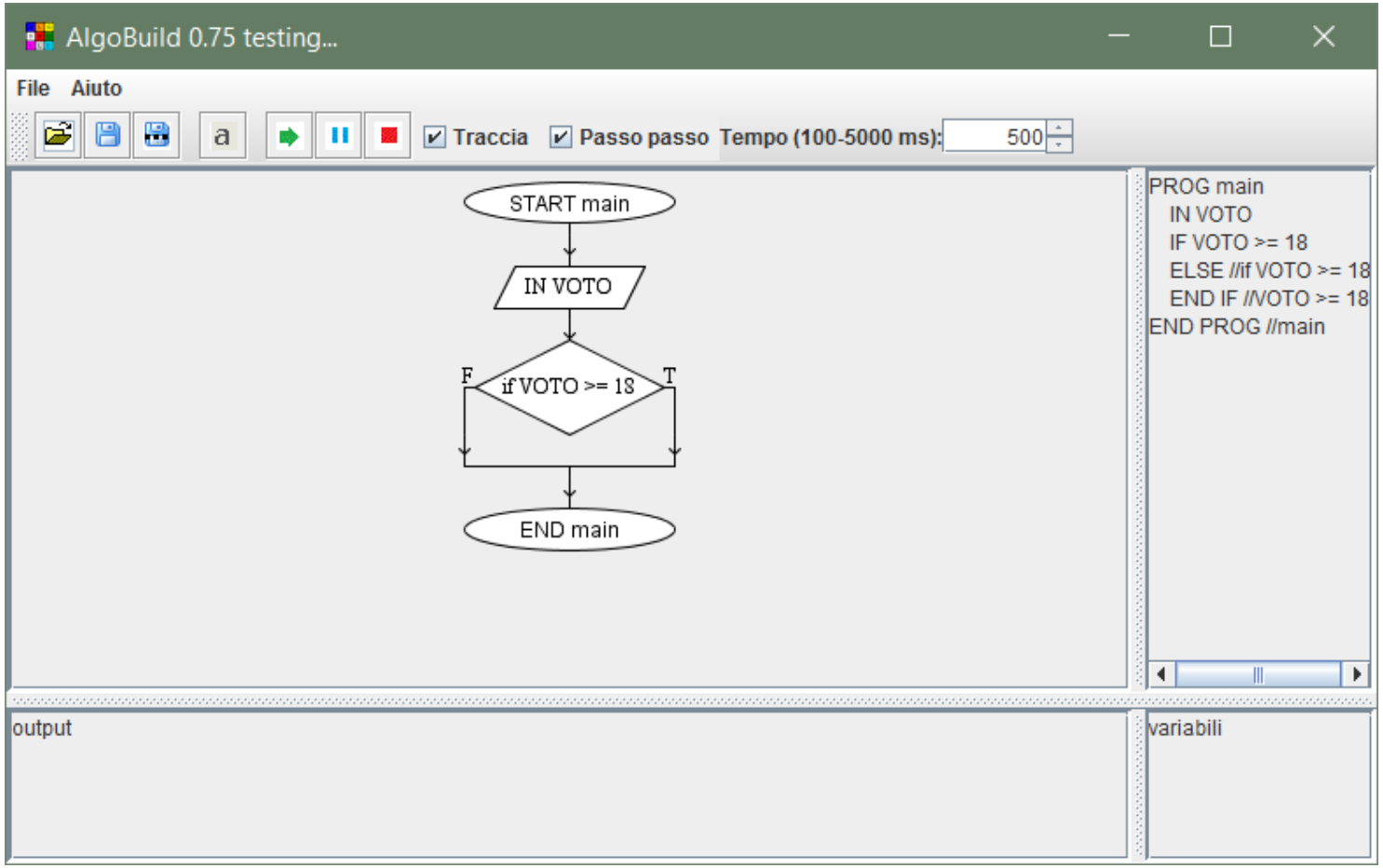


Struttura Selettiva IF

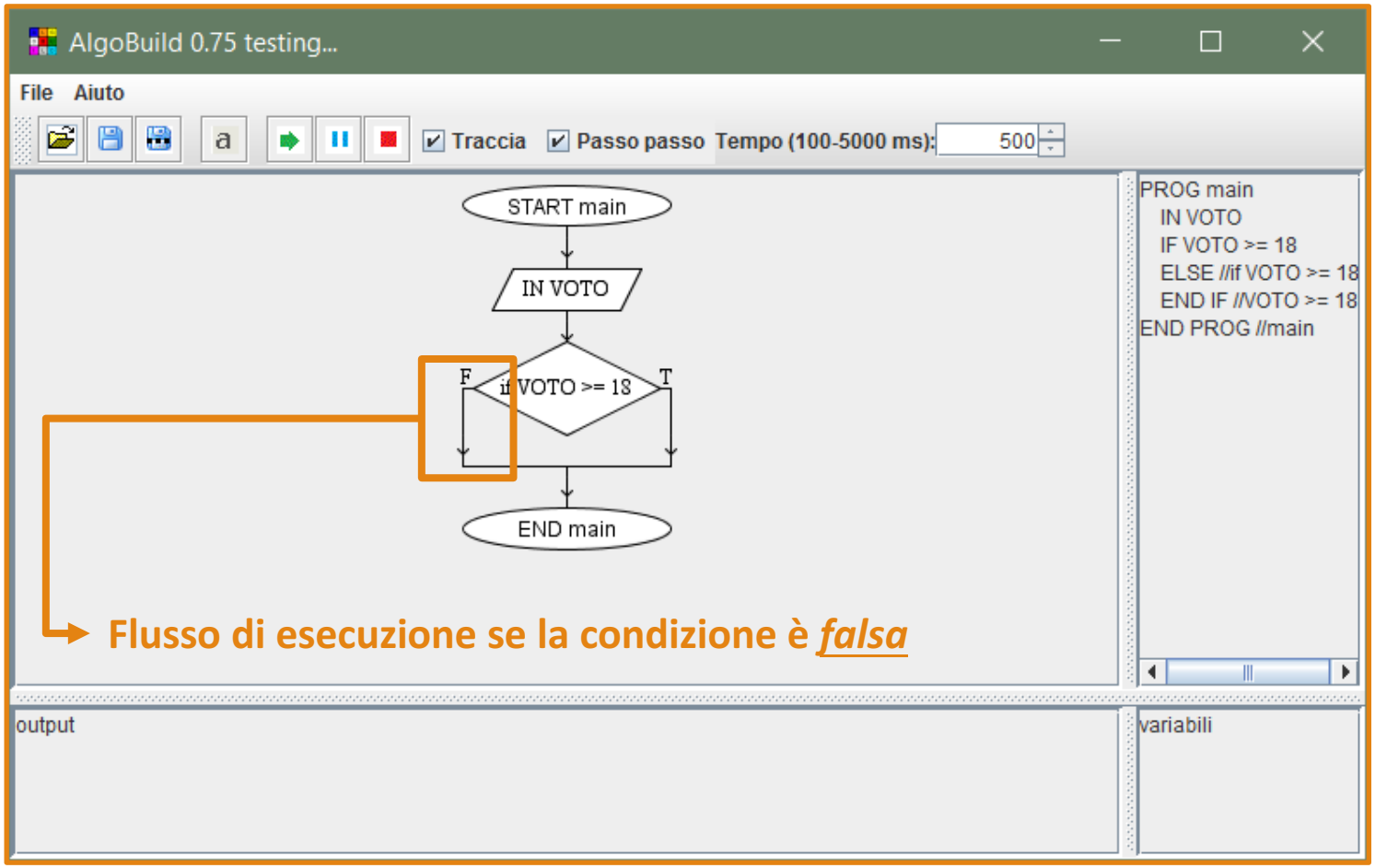


Ci verrà richiesto di inserire l'espressione (*condizione booleana*) che la struttura selettiva IF deve soddisfare. Tale espressione può assumere un valore booleano: vero (**T**) o falso (**F**)

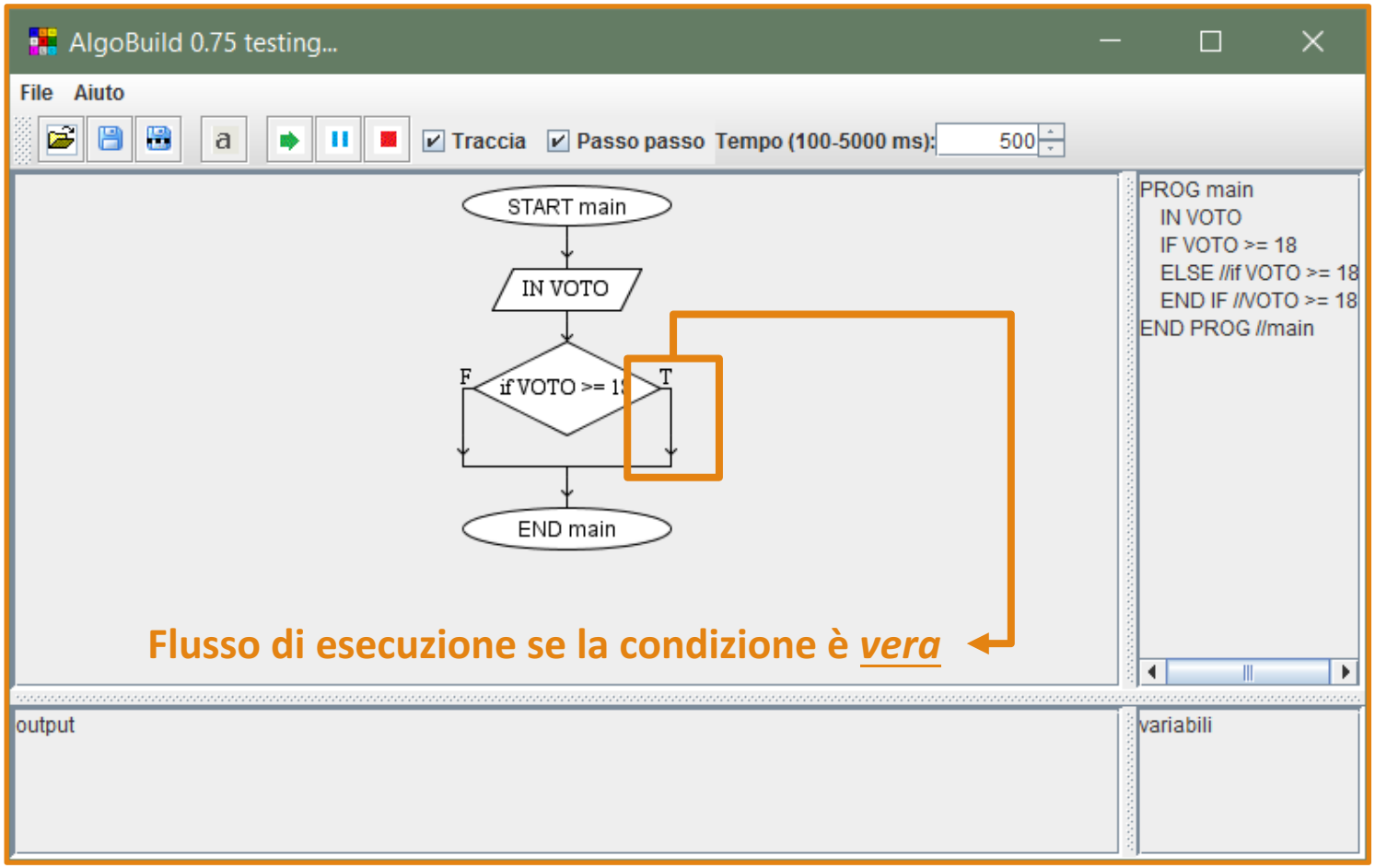
Struttura Selettiva IF



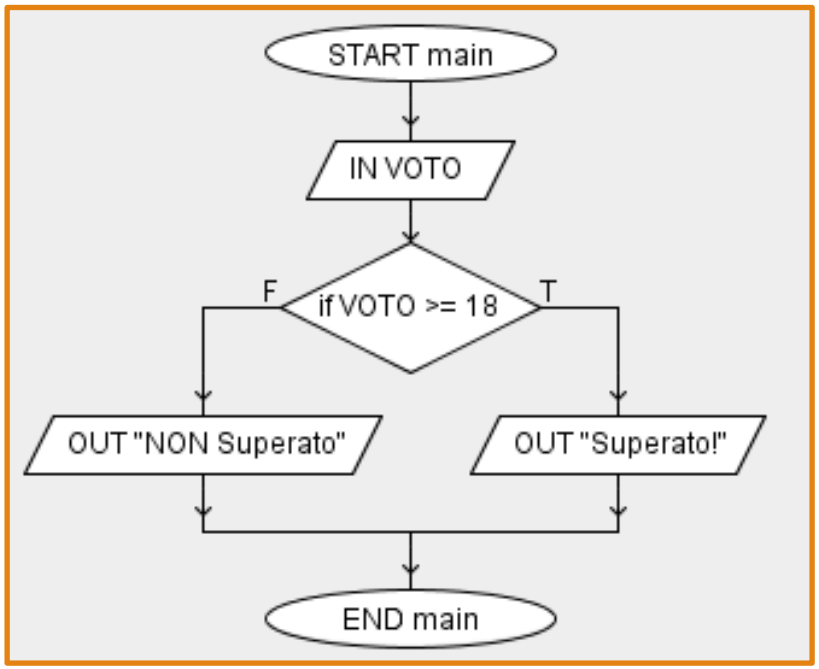
Struttura Selettiva IF



Struttura Selettiva IF



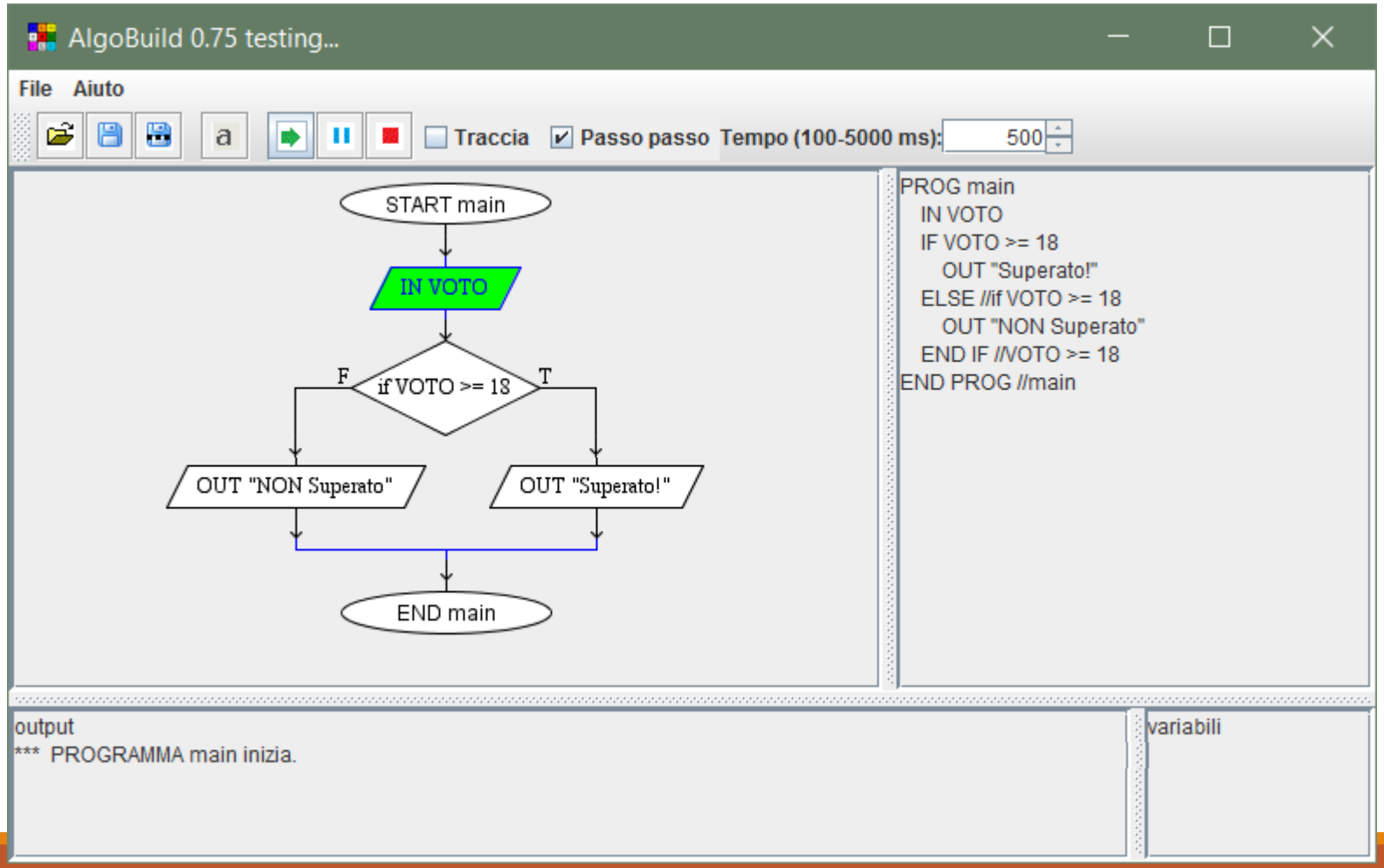
Struttura Selettiva IF



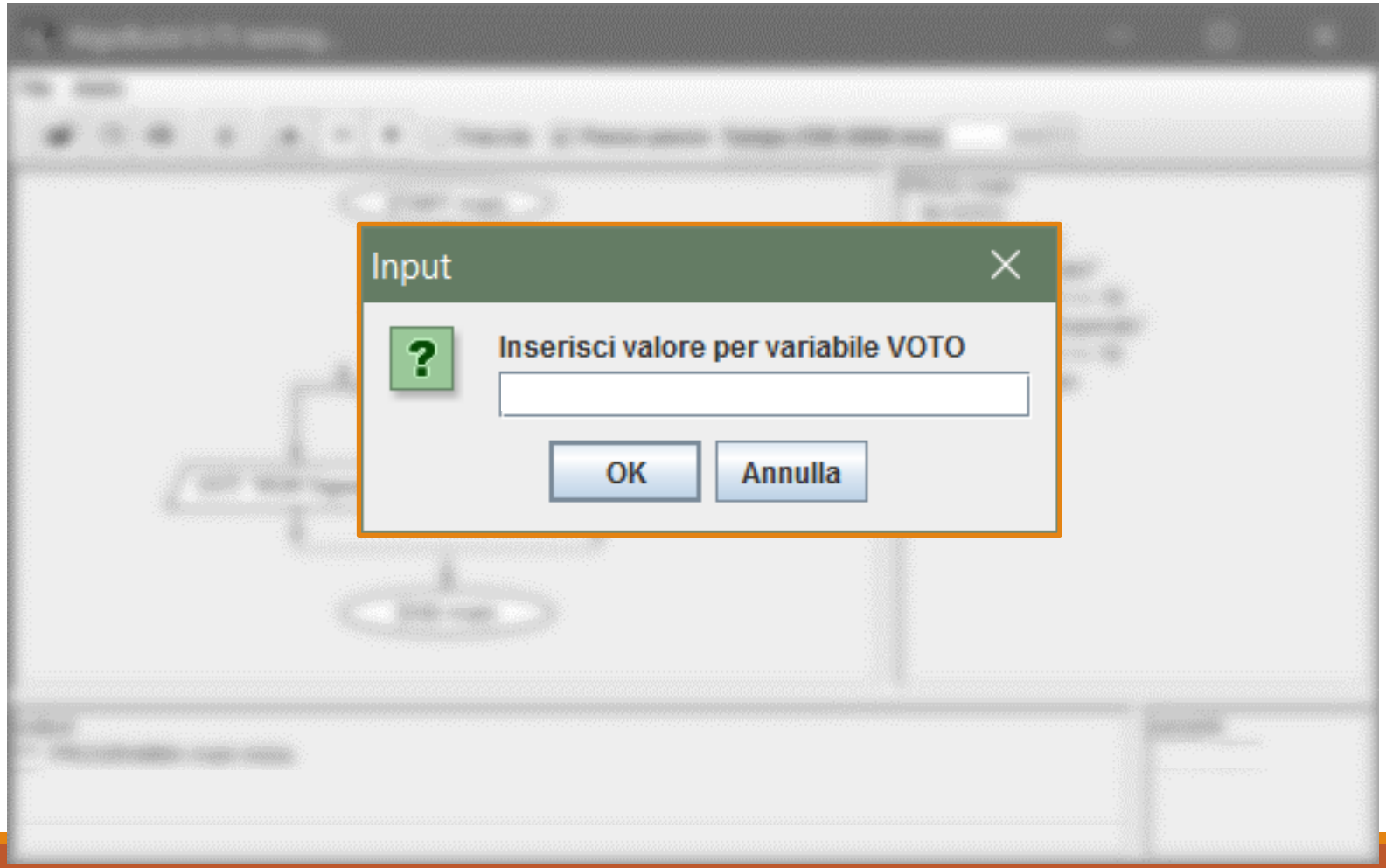
PSEUDO-CODICE

```
PROG main
   IN VOTO
   IF VOTO >= 18
      OUT "Superato!"
   ELSE //if VOTO >= 18
      OUT "NON Superato"
   END IF //VOTO >= 18
END PROG //main
```

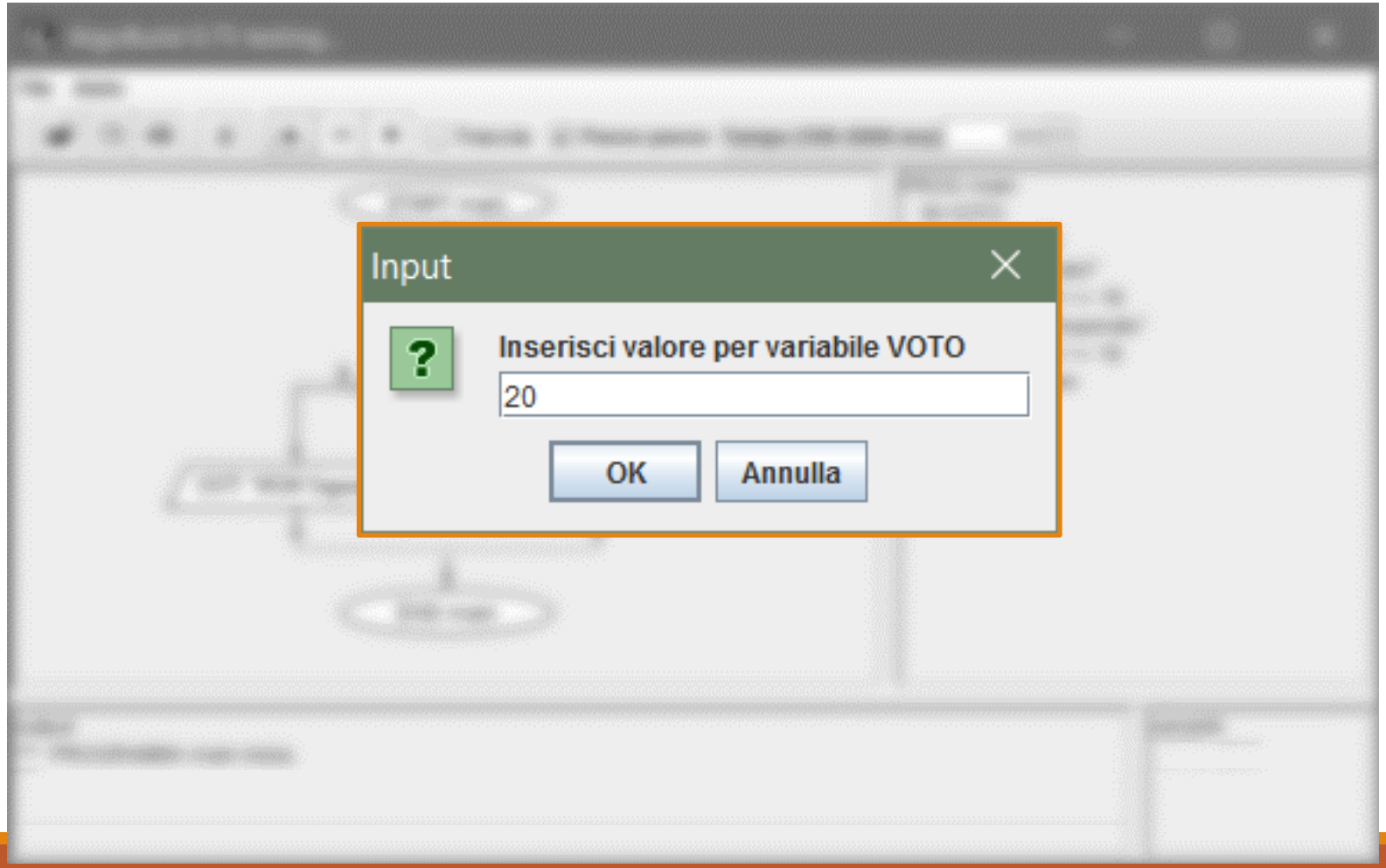
Struttura Selettiva IF – DEMO



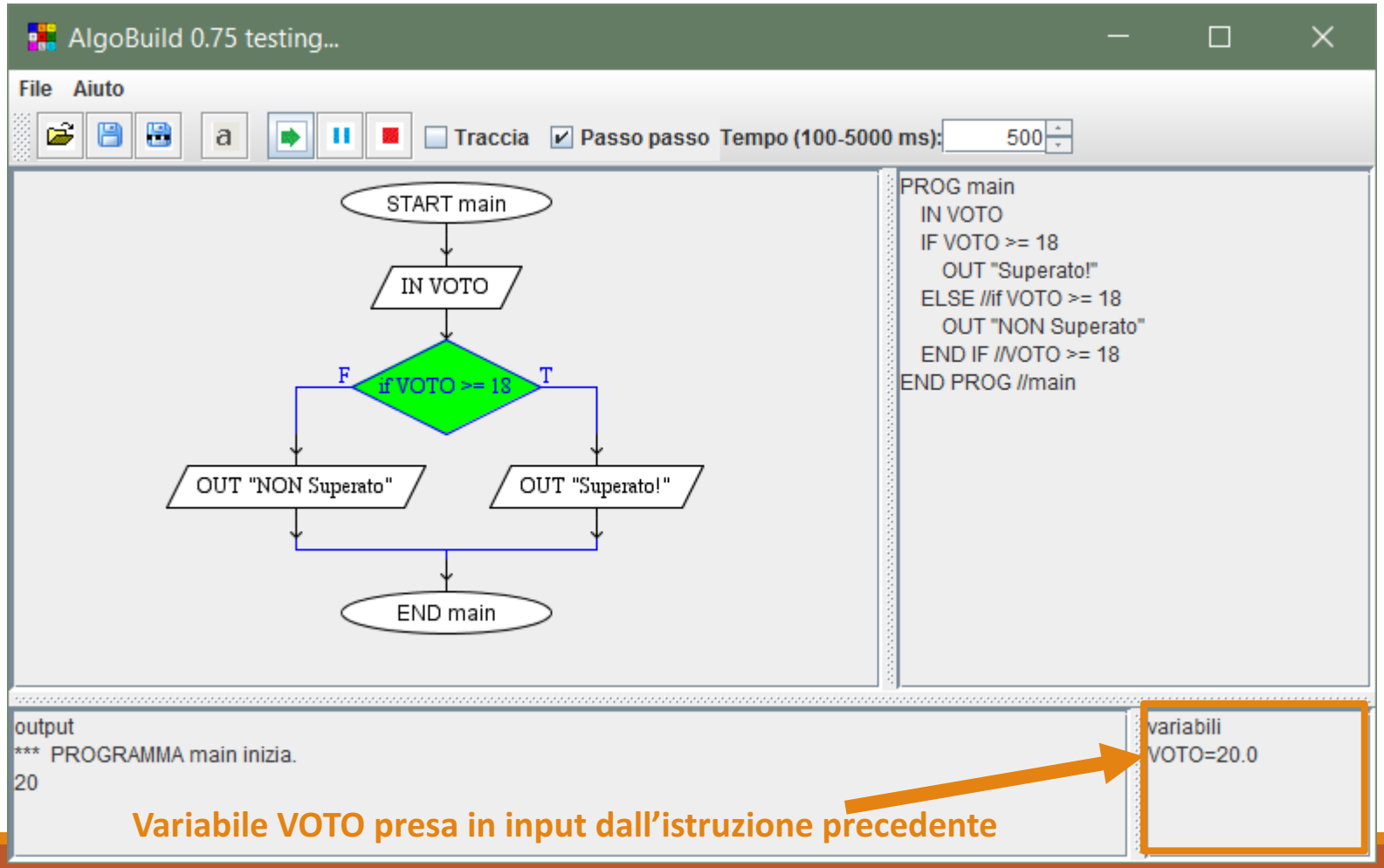
Struttura Selettiva IF – DEMO



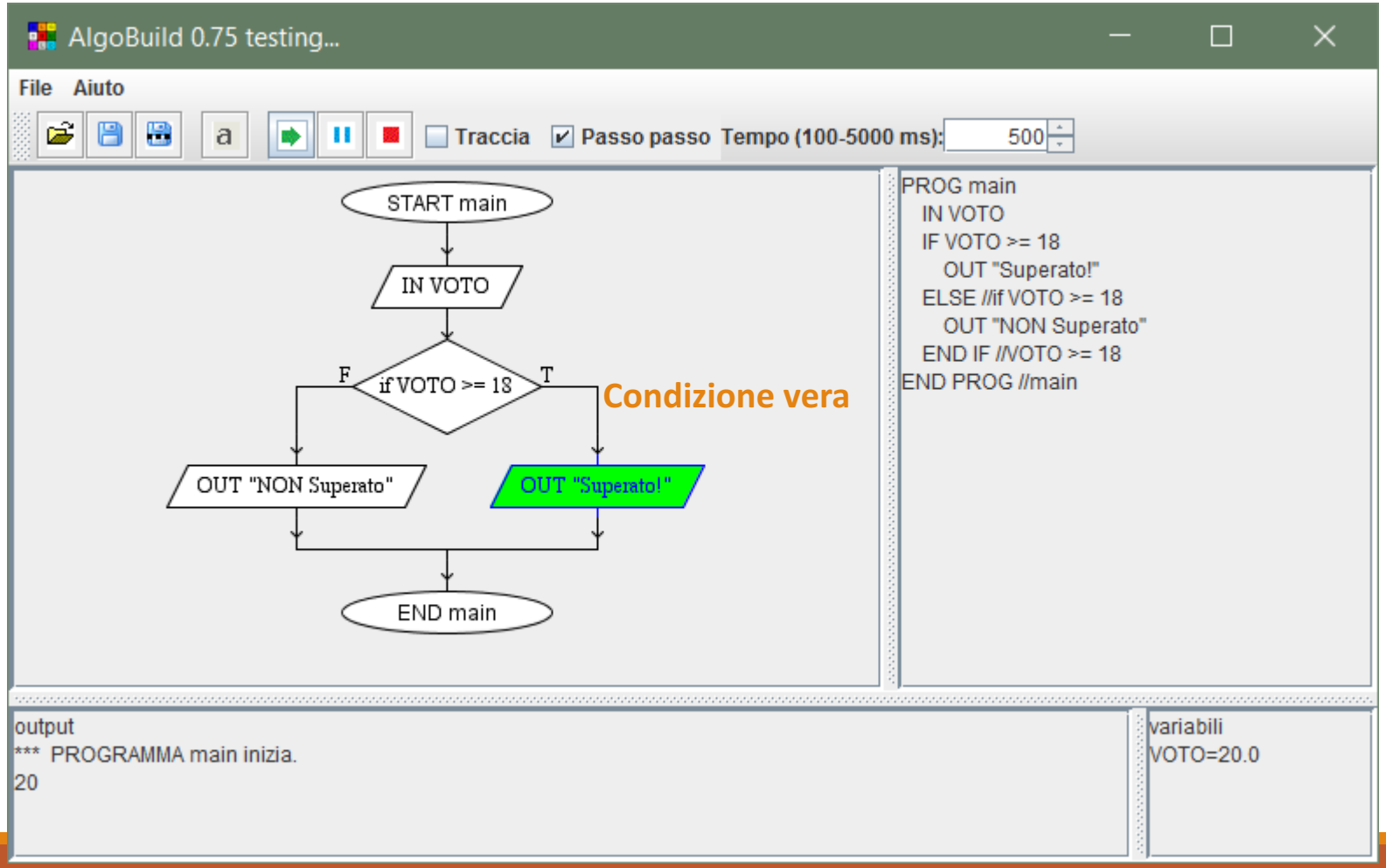
Struttura Selettiva IF – DEMO



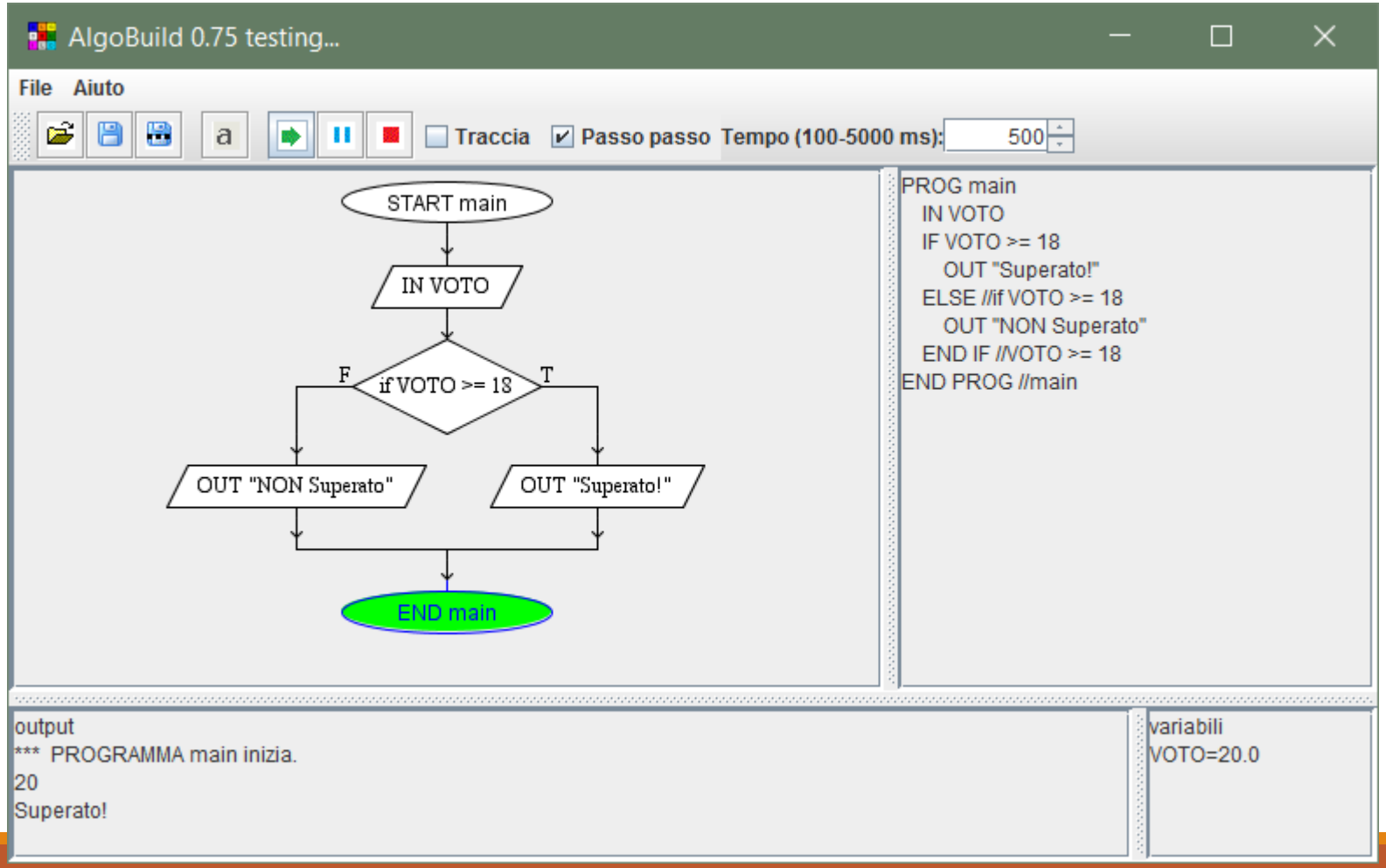
Struttura Selettiva IF – DEMO



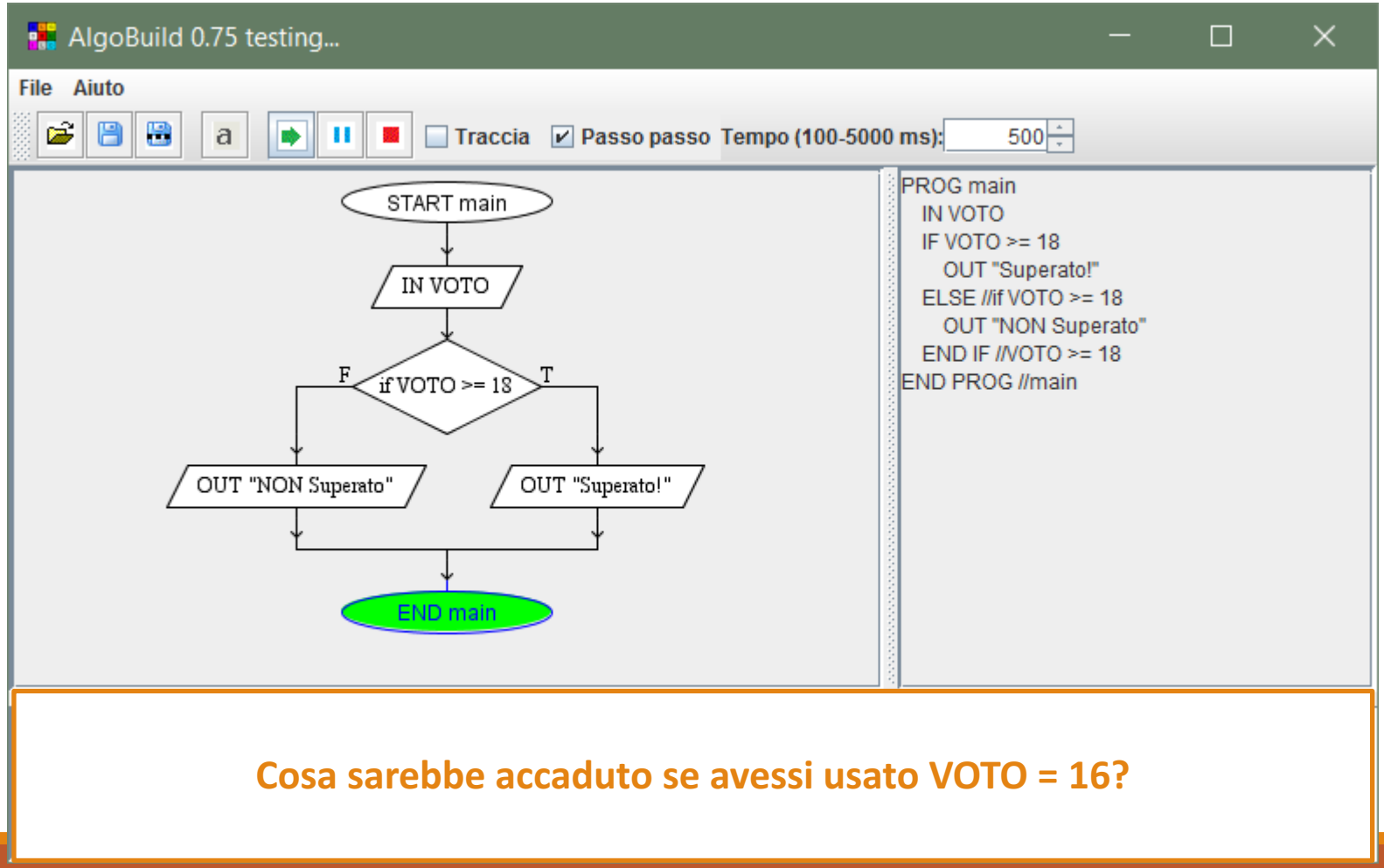
Struttura Selettiva IF – DEMO



Struttura Selettiva IF – DEMO



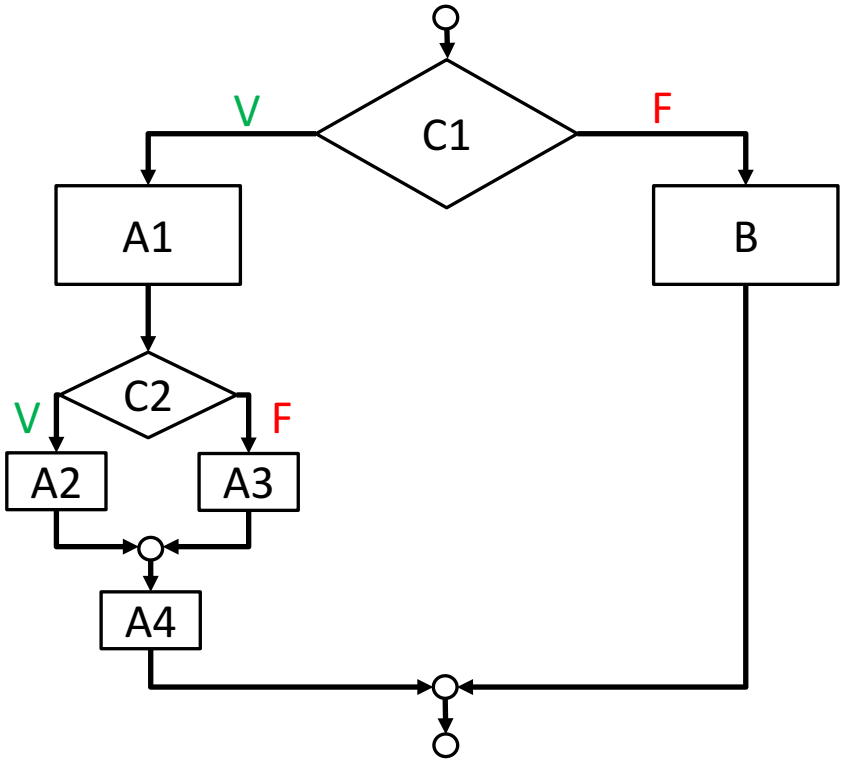
Struttura Selettiva IF – DEMO



Cosa sarebbe accaduto se avessi usato VOTO = 16?

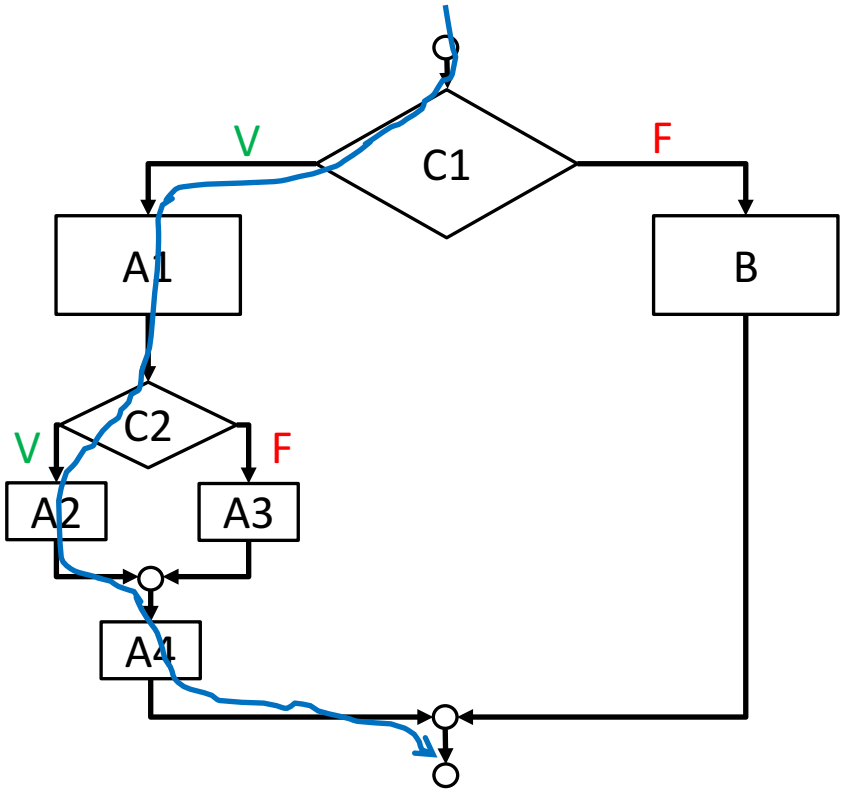
Struttura Selettiva IF (Annidata)

- Nelle istruzioni del blocco "Vero" o del blocco "Falso", è possibile inserire altri blocchi di scelta
- In tal caso si dice che la seconda scelta risulta **annidata** all'interno della prima



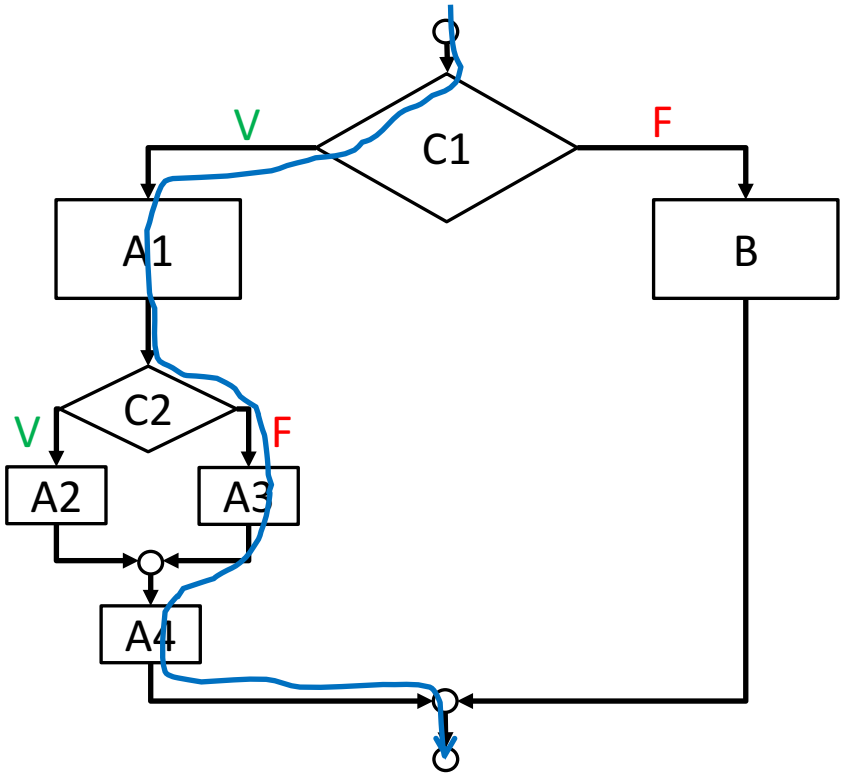
Struttura Selettiva IF (Annidata)

- C1 Vero, C2 Vero
  - Istruzioni eseguite: A1, A2, A4



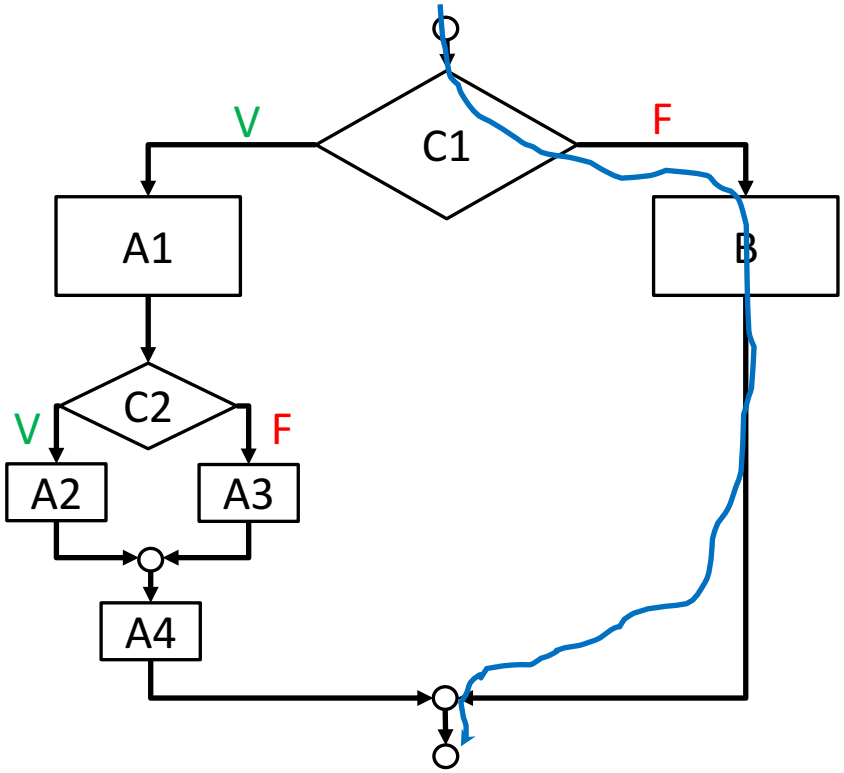
Struttura Selettiva IF (Annidata)

- C1 Vero, C2 Falso
  - Istruzioni eseguite: A1, A3, A4



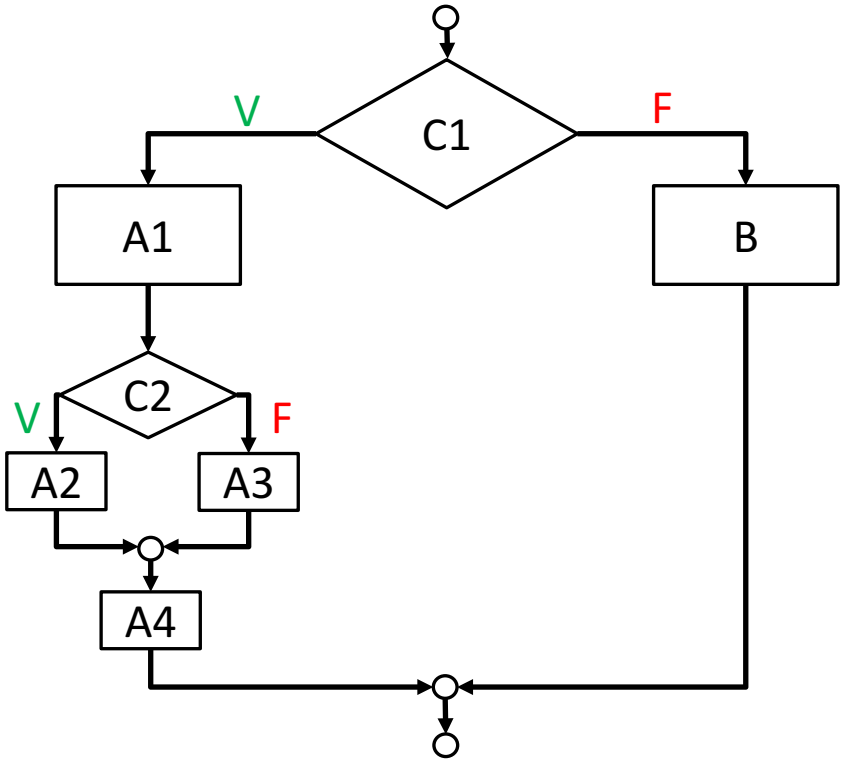
Struttura Selettiva IF (Annidata)

- C1 Falso
  - Istruzioni eseguite: B



Struttura Selettiva IF (Annidata)

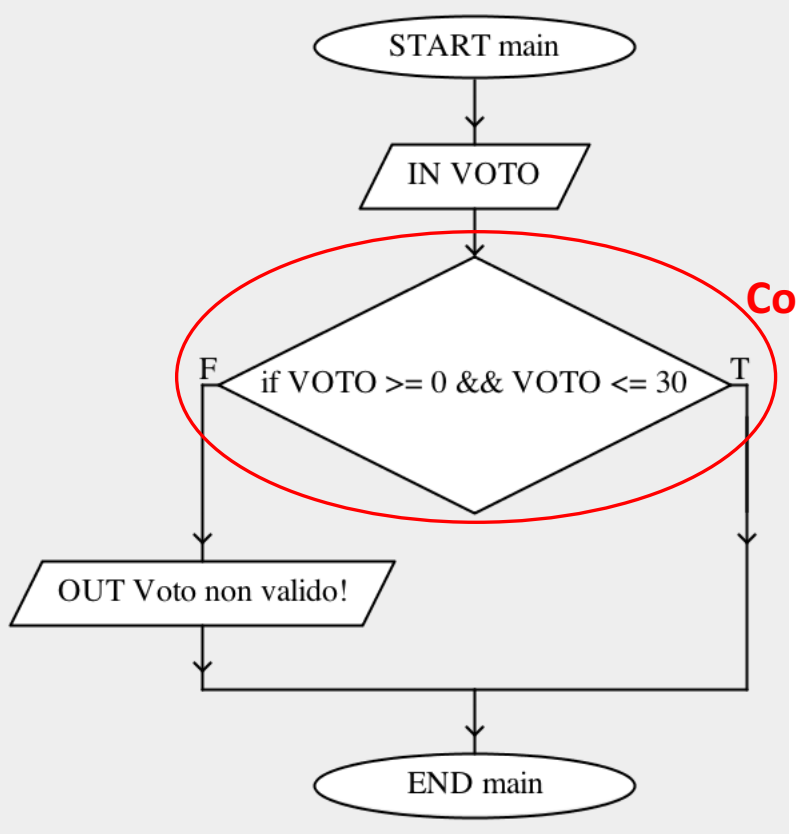
- L'intero blocco di scelta più interno (dalla condizione fino al ricongiungimento) deve essere completamente contenuto all'interno di uno dei rami del blocco più esterno



Struttura Selettiva IF (Annidata): Esempio 1

- L'esito dell'esame di uno studente può assumere una delle seguenti valutazioni
  - Sufficiente: se il voto è compreso tra 18 e 21
  - Buono: se il voto è compreso tra 22 e 25
  - Ottimo: se il voto è compreso tra 26 e 28
  - Eccellente: se il voto è 29 o 30
- Un voto è valido se è compreso tra 0 e 30

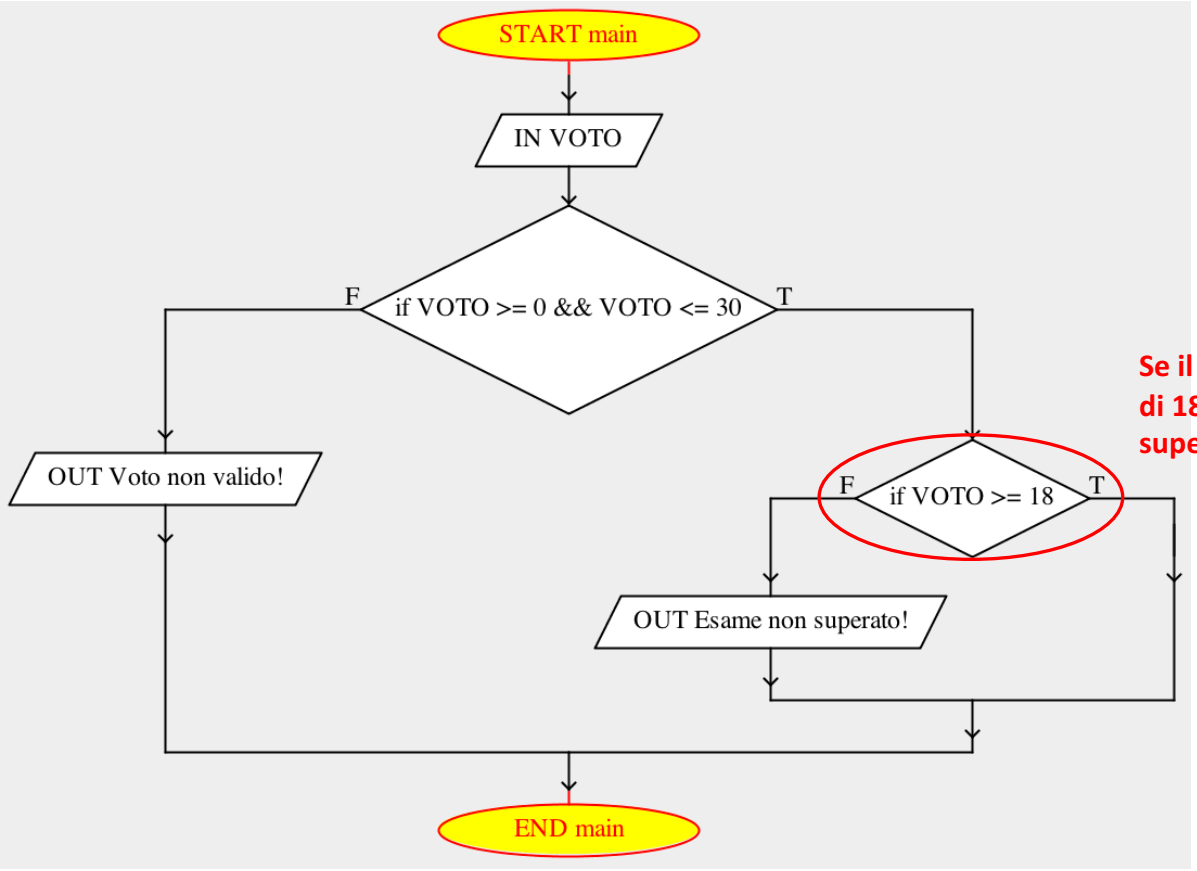
Struttura Selettiva IF (Annidata): Esempio 1



Controllo che il voto sia valido

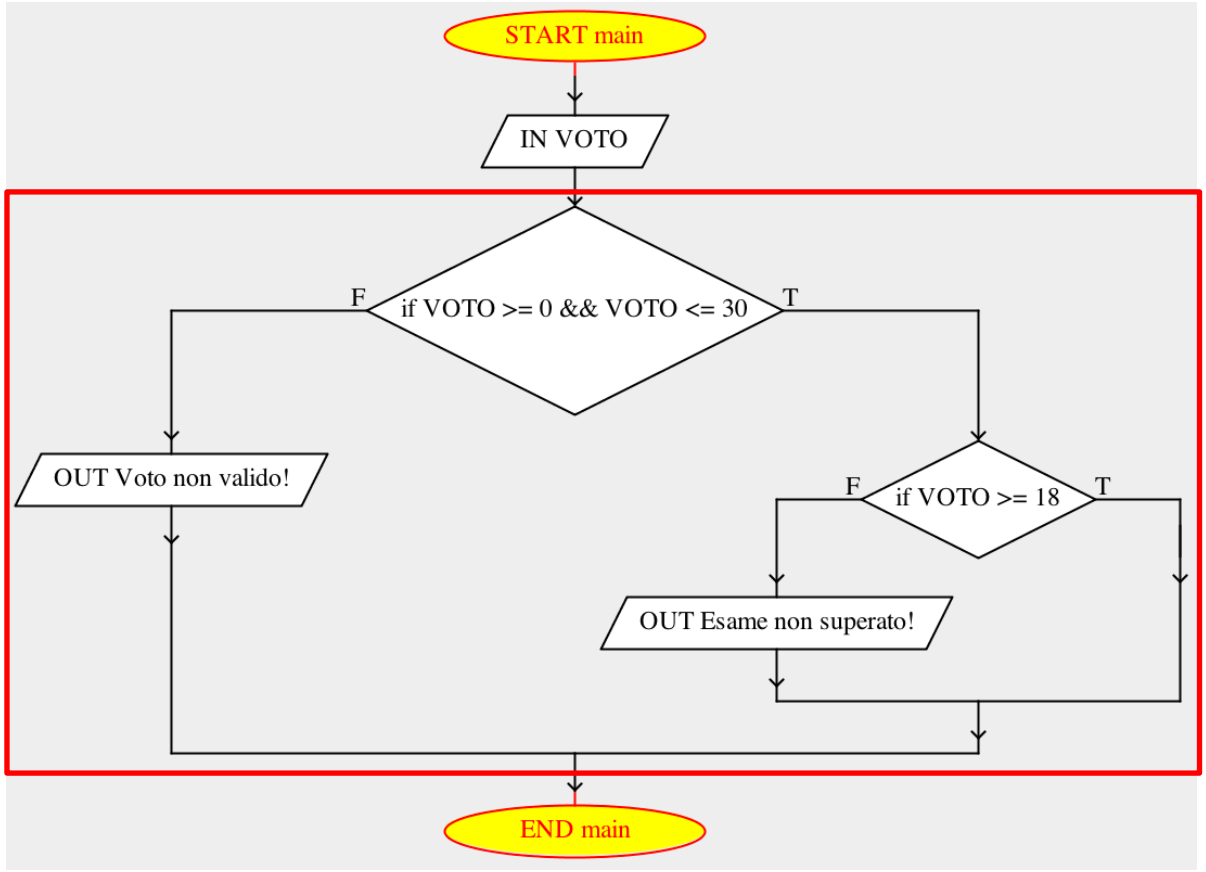
```
PROG main
   IN VOTO
   IF VOTO >= 0 && VOTO <= 30
   ELSE //if VOTO >= 0 && VOTO <= 30
      OUT Voto non valido!
   END IF //VOTO >= 0 && VOTO <= 30
END PROG //main
```

Struttura Selettiva IF (Annidata): Esempio 1



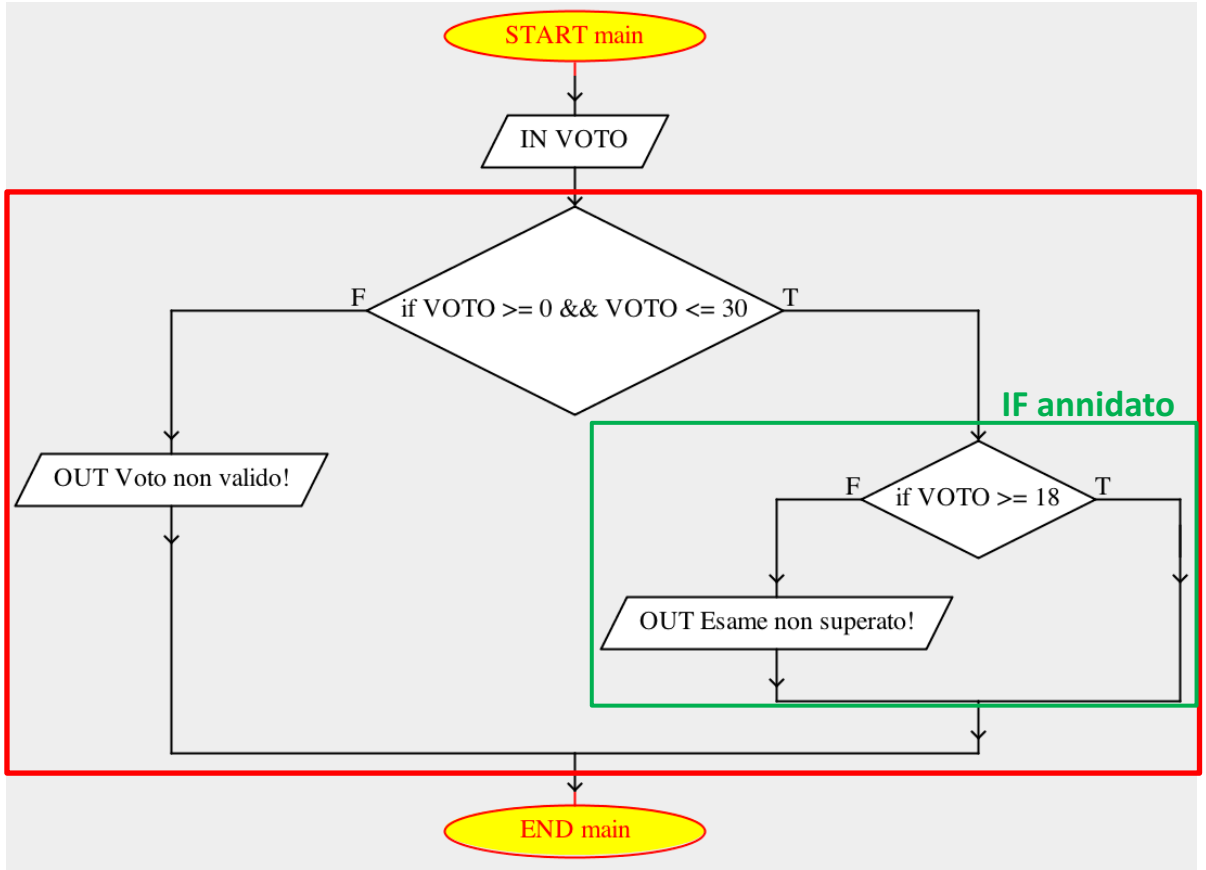
Se il voto è minore di 18, l'esame non è superato

Struttura Selettiva IF (Annidata): Esempio 1



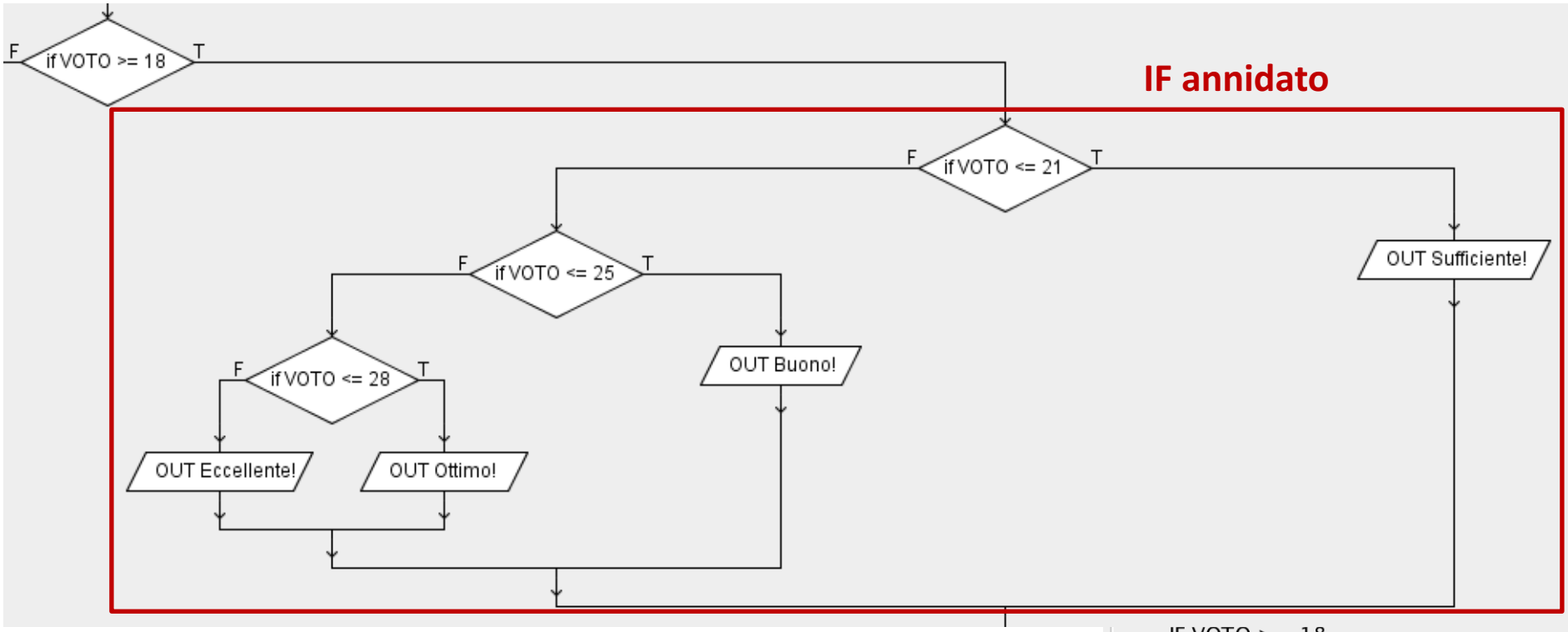

```
PROG main
   IN VOTO
   IF VOTO >= 0 && VOTO <= 30
      IF VOTO >= 18
      ELSE //if VOTO >= 18
         OUT Esame non superato!
      END IF //VOTO >= 18
   ELSE //if VOTO >= 0 && VOTO <= 30
      OUT Voto non valido!
   END IF //VOTO >= 0 && VOTO <= 30
END PROG //main
```

Struttura Selettiva IF (Annidata): Esempio 1

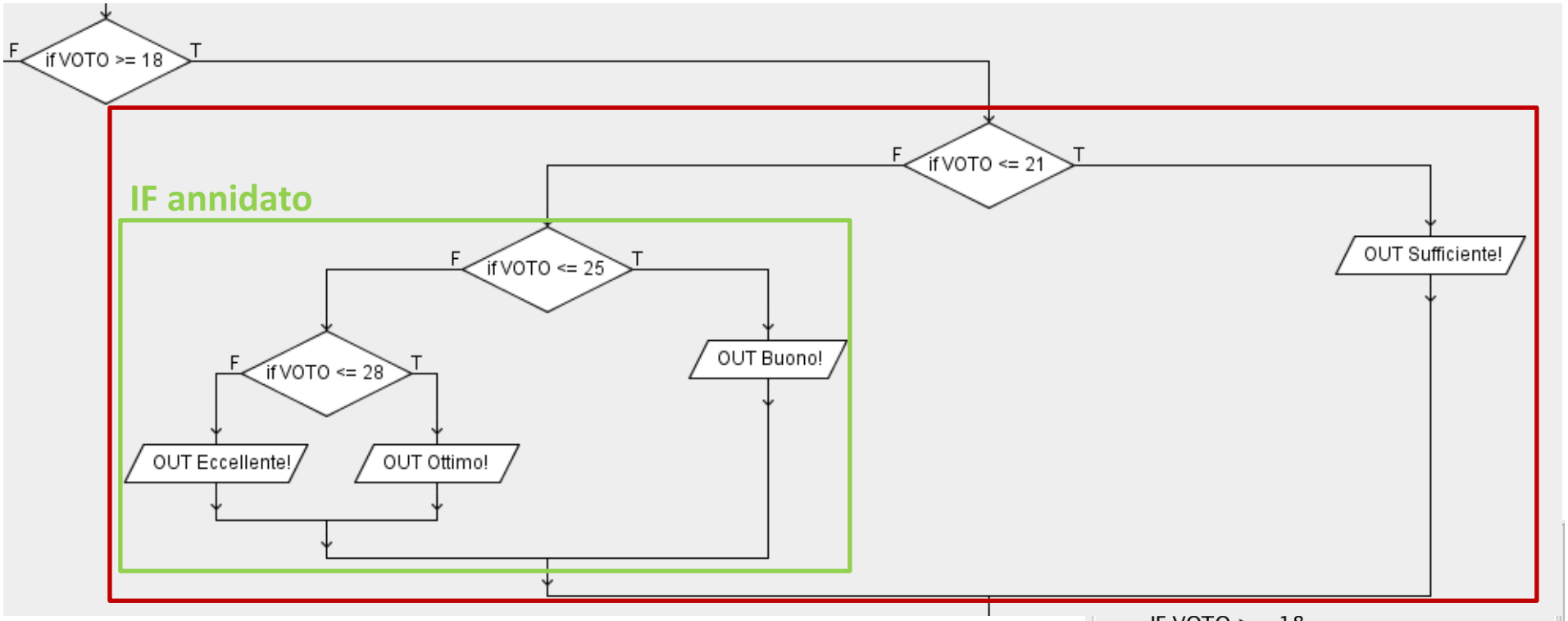



```
PROG main
   IN VOTO
   IF VOTO >= 0 && VOTO <= 30
      IF VOTO >= 18
      ELSE //if VOTO >= 18
         OUT Esame non superato!
      END IF //VOTO >= 18
   ELSE //if VOTO >= 0 && VOTO <= 30
      OUT Voto non valido!
   END IF //VOTO >= 0 && VOTO <= 30
END PROG //main
```

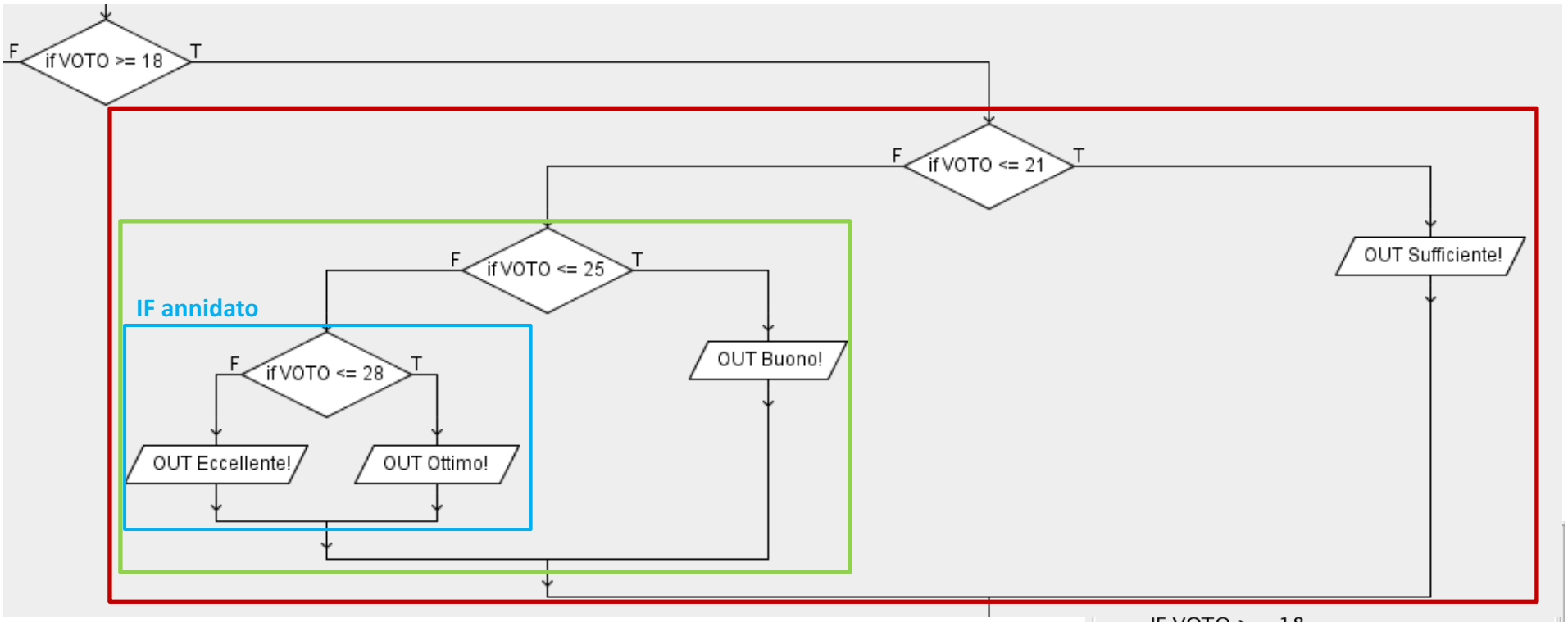
IF annidato




```
IF VOTO >= 18
   IF VOTO <= 21
      OUT Sufficiente!
   ELSE //if VOTO <= 21
      IF VOTO <= 25
         OUT Buono!
      ELSE //if VOTO <= 25
         IF VOTO <= 28
            OUT Ottimo!
         ELSE //if VOTO <= 28
            OUT Eccellente!
         END IF //VOTO <= 28
      END IF //VOTO <= 25
   END IF //VOTO <= 21
ELSE //if VOTO >= 18
   OUT Esame non superato!
END IF //VOTO >= 18
ELSE //if VOTO >= 0 && VOTO <= 30
   OUT Voto non valido!
END IF //VOTO >= 0 && VOTO <= 30
END PROG //main
```



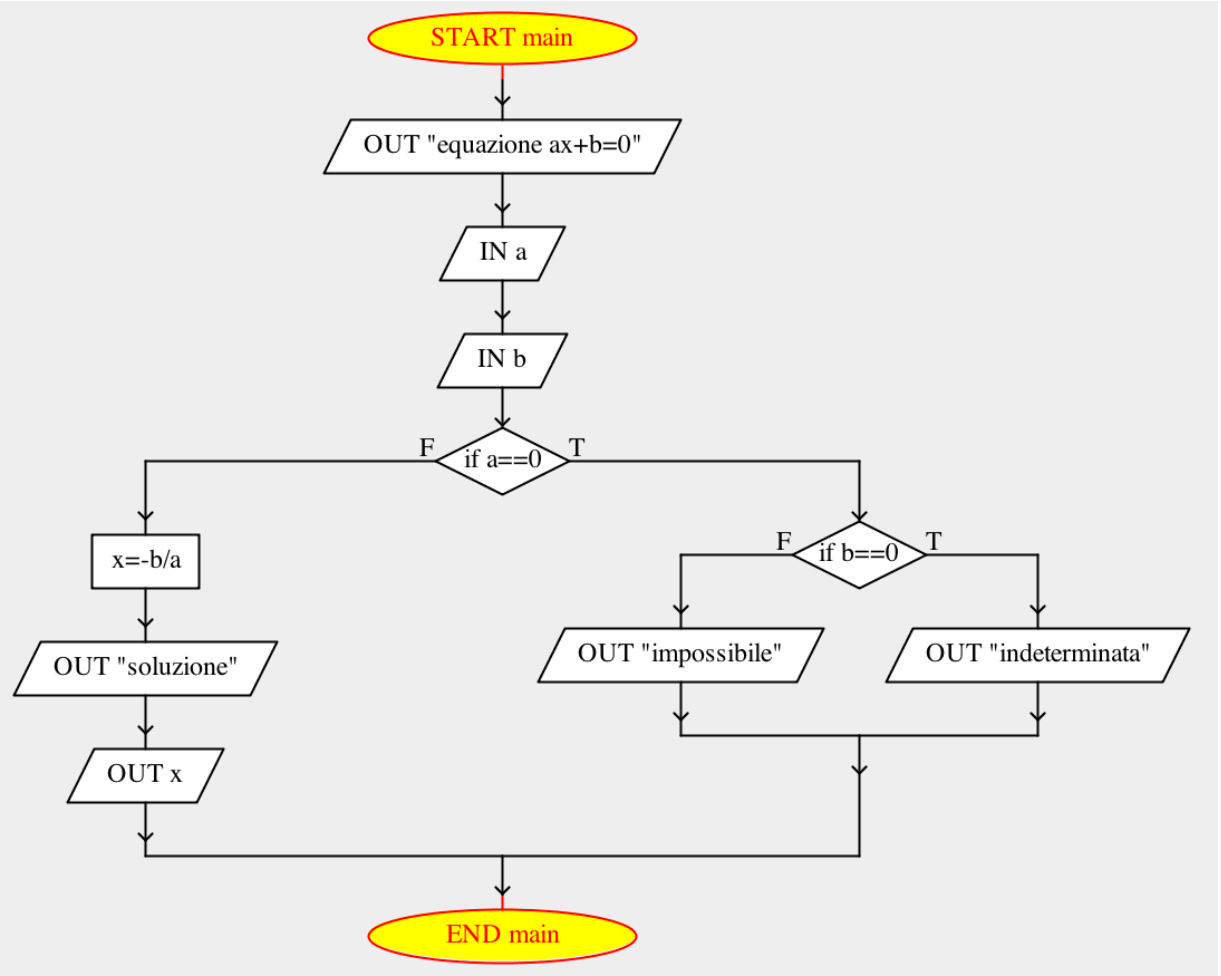

```
IF VOTO >= 18
   IF VOTO <= 21
      OUT Sufficiente!
   ELSE //if VOTO <= 21
      IF VOTO <= 25
         OUT Buono!
      ELSE //if VOTO <= 25
         IF VOTO <= 28
            OUT Ottimo!
         ELSE //if VOTO <= 28
            OUT Eccellente!
         END IF //VOTO <= 28
      END IF //VOTO <= 25
   END IF //VOTO <= 21
ELSE //if VOTO >= 18
   OUT Esame non superato!
END IF //VOTO >= 18
ELSE //if VOTO >= 0 && VOTO <= 30
   OUT Voto non valido!
END IF //VOTO >= 0 && VOTO <= 30
END PROG //main
```



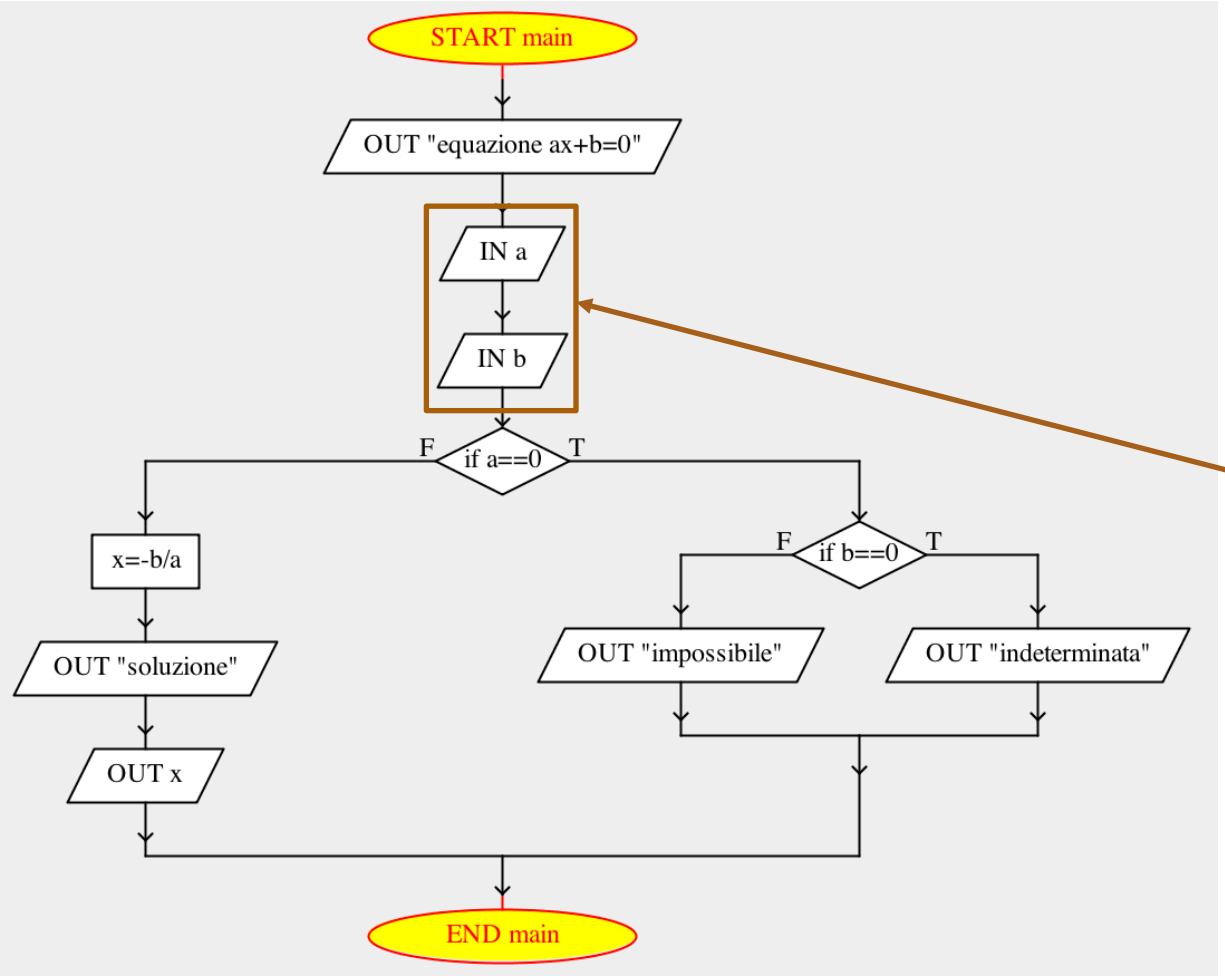

```
IF VOTO >= 18
   IF VOTO <= 21
      OUT Sufficiente!
   ELSE //if VOTO <= 21
      IF VOTO <= 25
         OUT Buono!
      ELSE //if VOTO <= 25
         IF VOTO <= 28
            OUT Ottimo!
         ELSE //if VOTO <= 28
            OUT Eccellente!
         END IF //VOTO <= 28
      END IF //VOTO <= 25
   END IF //VOTO <= 21
ELSE //if VOTO >= 18
   OUT Esame non superato!
END IF //VOTO >= 18
ELSE //if VOTO >= 0 && VOTO <= 30
   OUT Voto non valido!
END IF //VOTO >= 0 && VOTO <= 30
END PROG //main
```

Esempio: Risoluzione di equazioni di primo grado

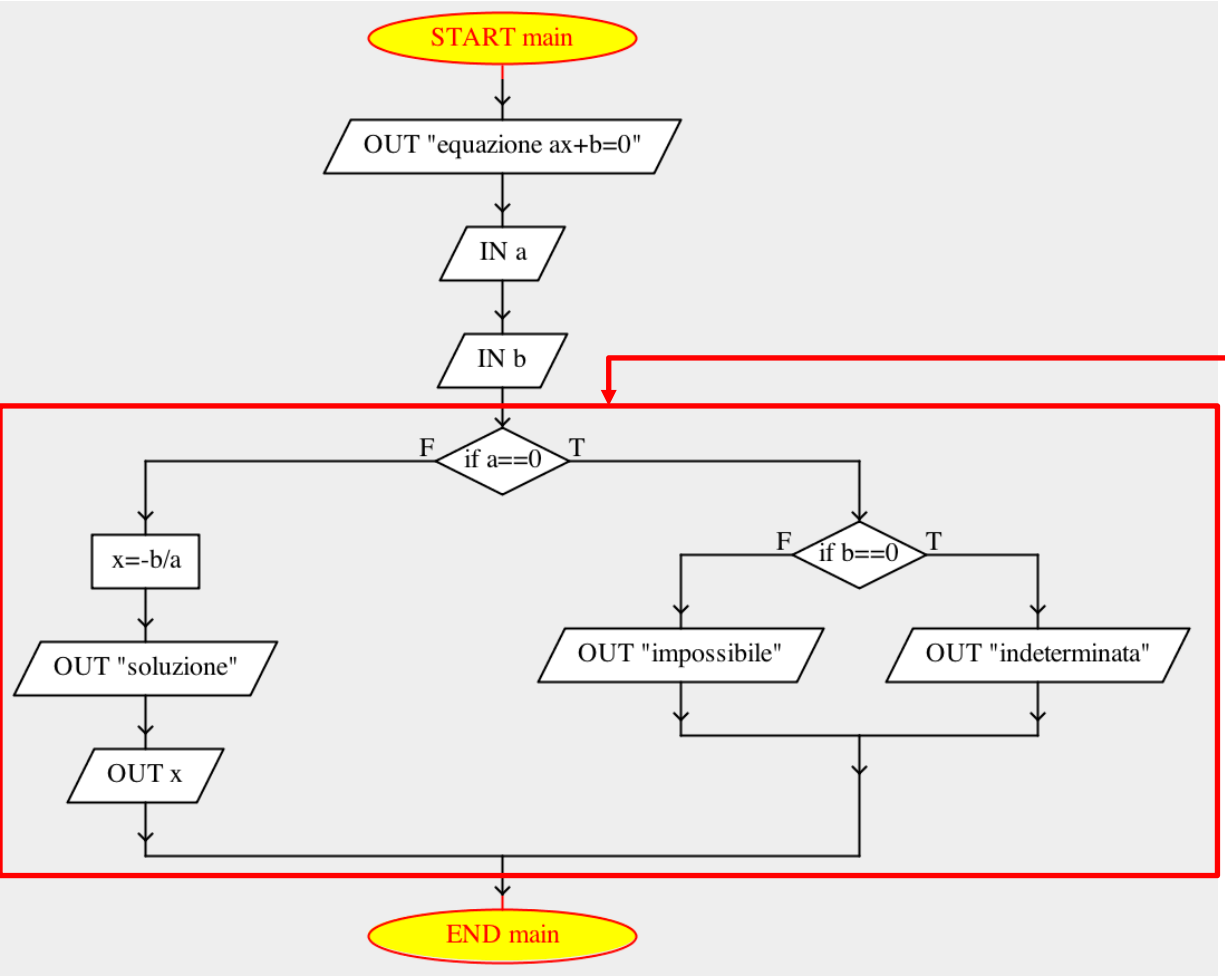
- $a\,x\ +\ b\ =\ 0$
- La soluzione è:
  - $x\ =\ -\ b\ /\ a$
    - Solo se $a\ \neq 0$
  - $x\ =\ \ indeterminato\ (infinite\ soluzioni)$
    - Se $a = 0\ e\ b = 0$
  - $x\ =\ \ impossibile\ (nessuna\ soluzione)$
    - Se $a = 0\ e$ b $\neq 0$



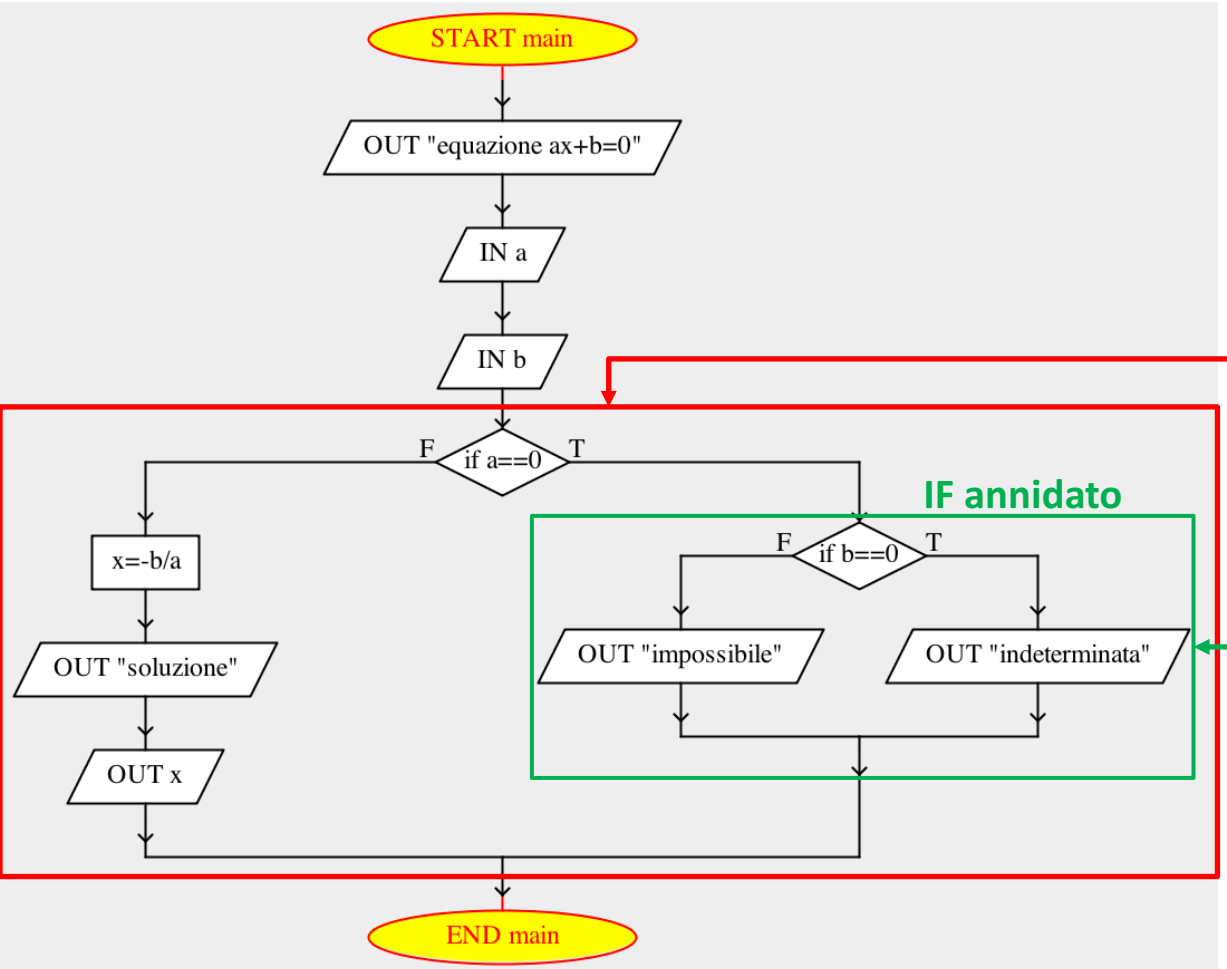

```
PROG main
   OUT "equazione ax+b=0"
   IN a
   IN b
   IF a==0
      IF b==0
         OUT "indeterminata"
      ELSE //if b==0
         OUT "impossibile"
      END IF //b==0
   ELSE //if a==0
      ASSIGN x=-b/a
      OUT "soluzione"
      OUT x
   END IF //a==0
END PROG //main
```



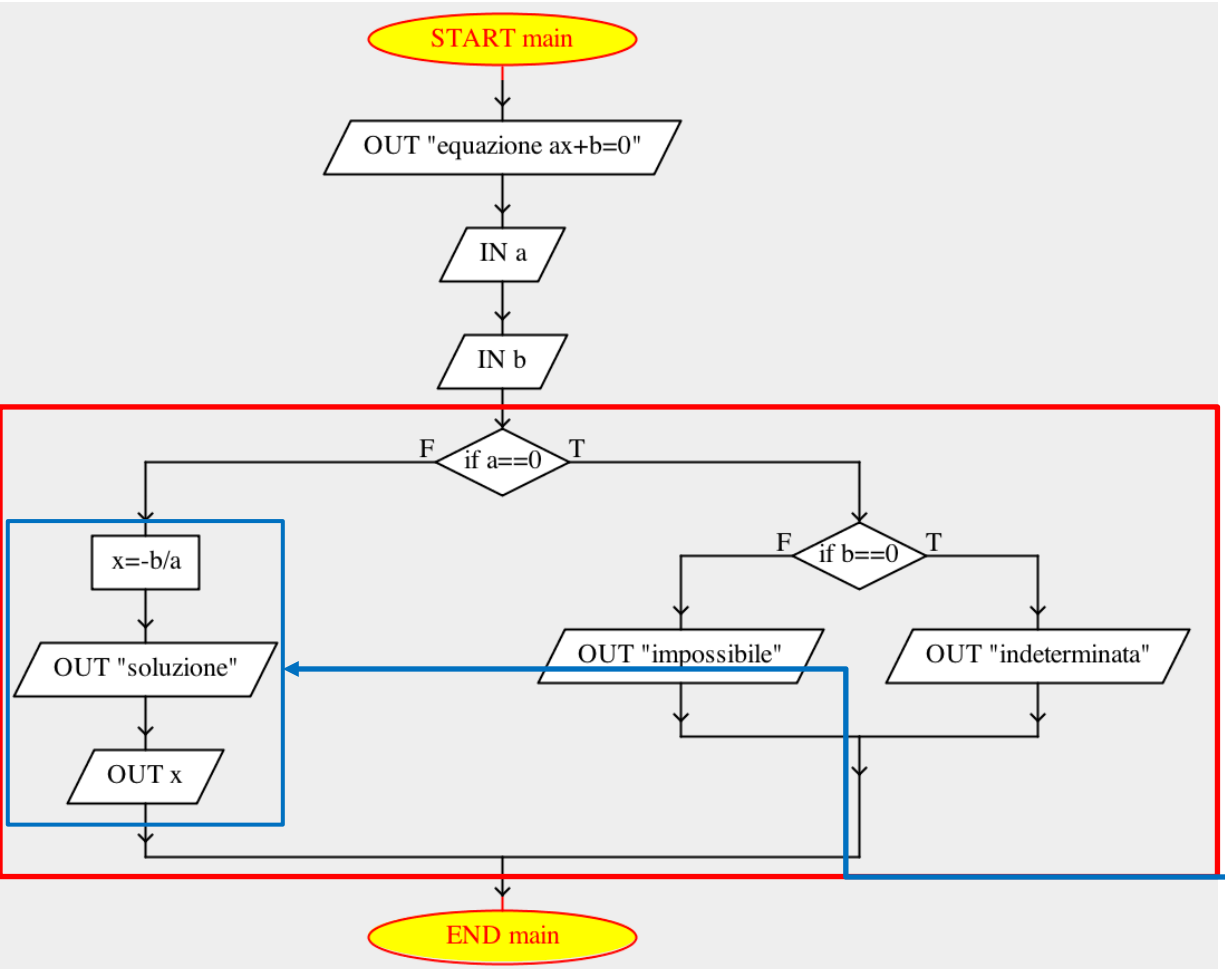

```
PROG main
   OUT "equazione ax+b=0"
   IN a
   IN b
   IF a==0
      IF b==0
         OUT "indeterminata"
      ELSE //if b==0
         OUT "impossibile"
      END IF //b==0
   ELSE //if a==0
      ASSIGN x=-b/a
      OUT "soluzione"
      OUT x
   END IF //a==0
END PROG //main
```




```
PROG main
   OUT "equazione ax+b=0"
   IN a
   IN b
   IF a==0
      IF b==0
         OUT "indeterminata"
      ELSE //if b==0
         OUT "impossibile"
      END IF //b==0
   ELSE //if a==0
      ASSIGN x=-b/a
      OUT "soluzione"
      OUT x
   END IF //a==0
END PROG //main
```

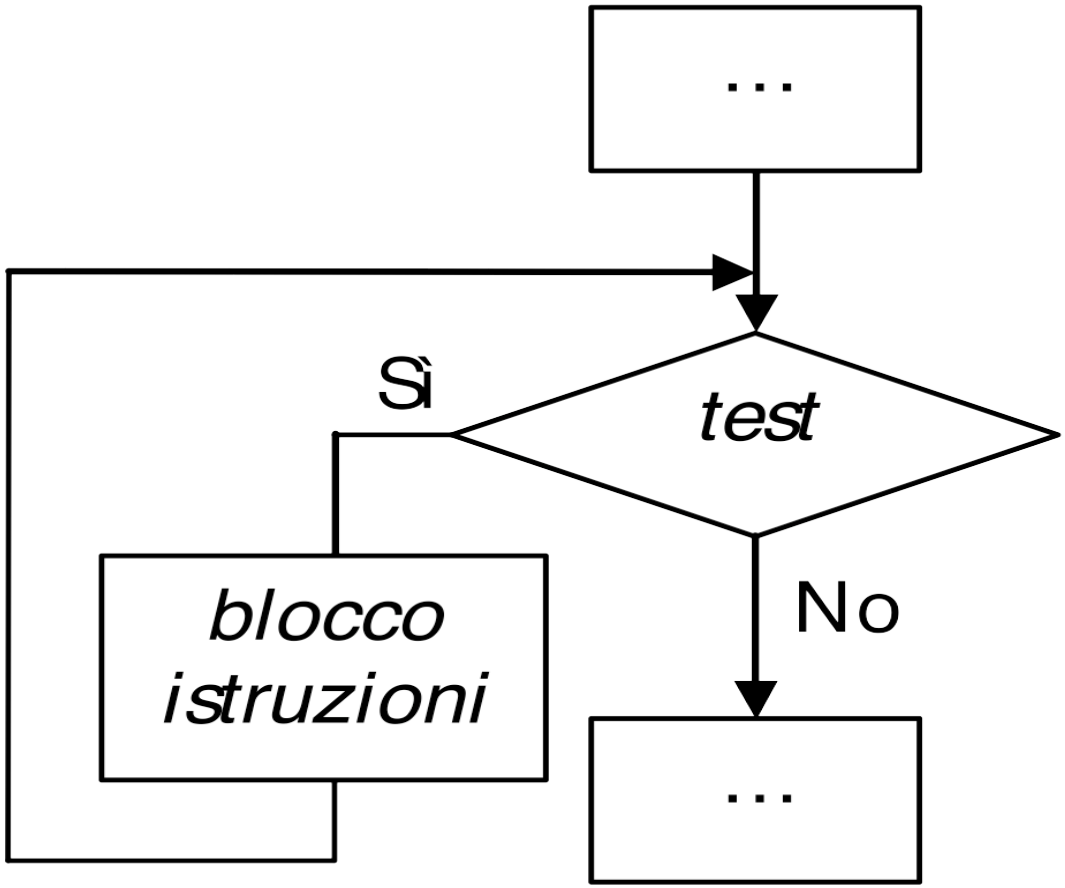



```
PROG main
   OUT "equazione ax+b=0"
   IN a
   IN b
   IF a==0          IF annidato
      IF b==0
         OUT "indeterminata"
      ELSE //if b==0
         OUT "impossibile"
      END IF //b==0
   ELSE //if a==0
      ASSIGN x=-b/a
      OUT "soluzione"
      OUT x
   END IF //a==0
END PROG //main
```




```
PROG main
   OUT "equazione ax+b=0"
   IN a
   IN b
   IF a==0
      IF b==0
         OUT "indeterminata"
      ELSE //if b==0
         OUT "impossibile"
      END IF //b==0
   ELSE //if a==0
      ASSIGN x=-b/a
      OUT "soluzione"
      OUT x
   END IF //a==0
END PROG //main
```

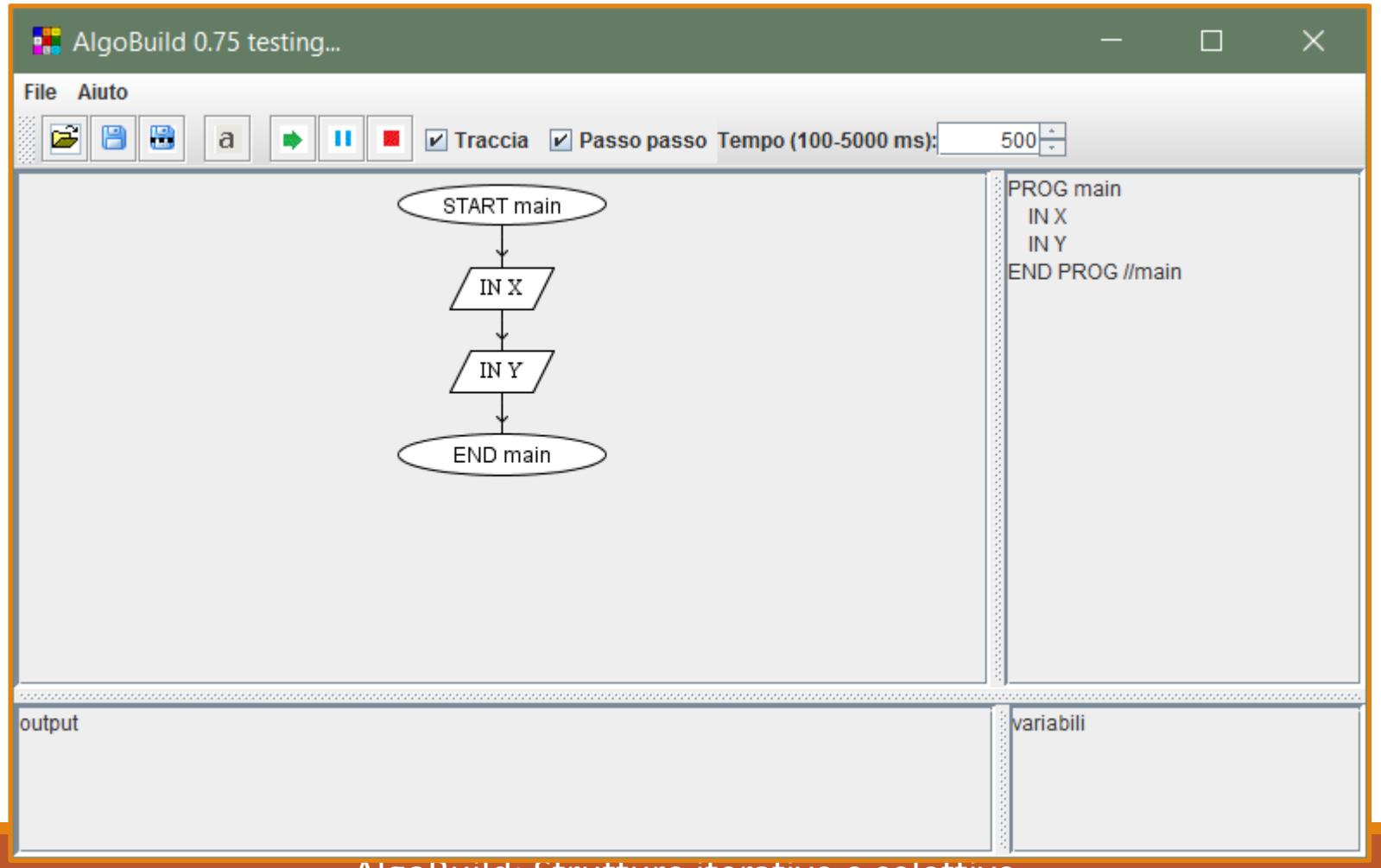
Strutture di Controllo: Ciclo a Condizione Iniziale



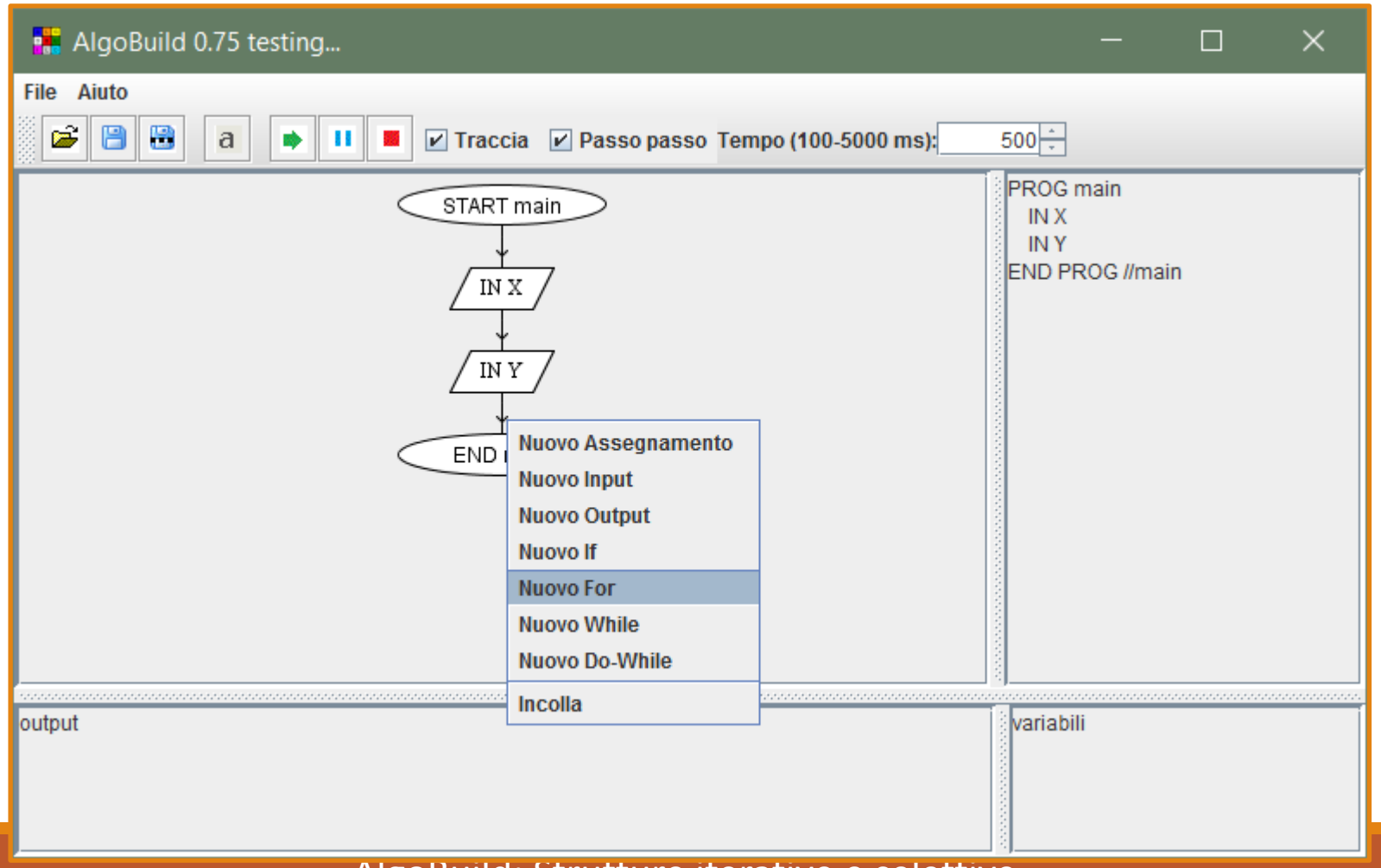
Struttura Iterativa FOR – Esempio 1

- Definiamo un diagramma di flusso che
  - Prende in **input** le seguenti due variabili
    - **X**
    - **Y**
  - Memorizza e (mostra in output) una variabile
    - $\mathbf{R} = \mathbf{X}^{\mathbf{Y}}$
    - **N.B.** Il calcolo di $\mathbf{X}^{\mathbf{Y}}$ deve essere eseguito con il metodo moltiplicativo
      - $\underbrace{\mathbf{X} \times \mathbf{X} \times \ldots \times \mathbf{X}}_{\mathbf{Y}\text{ volte}}$

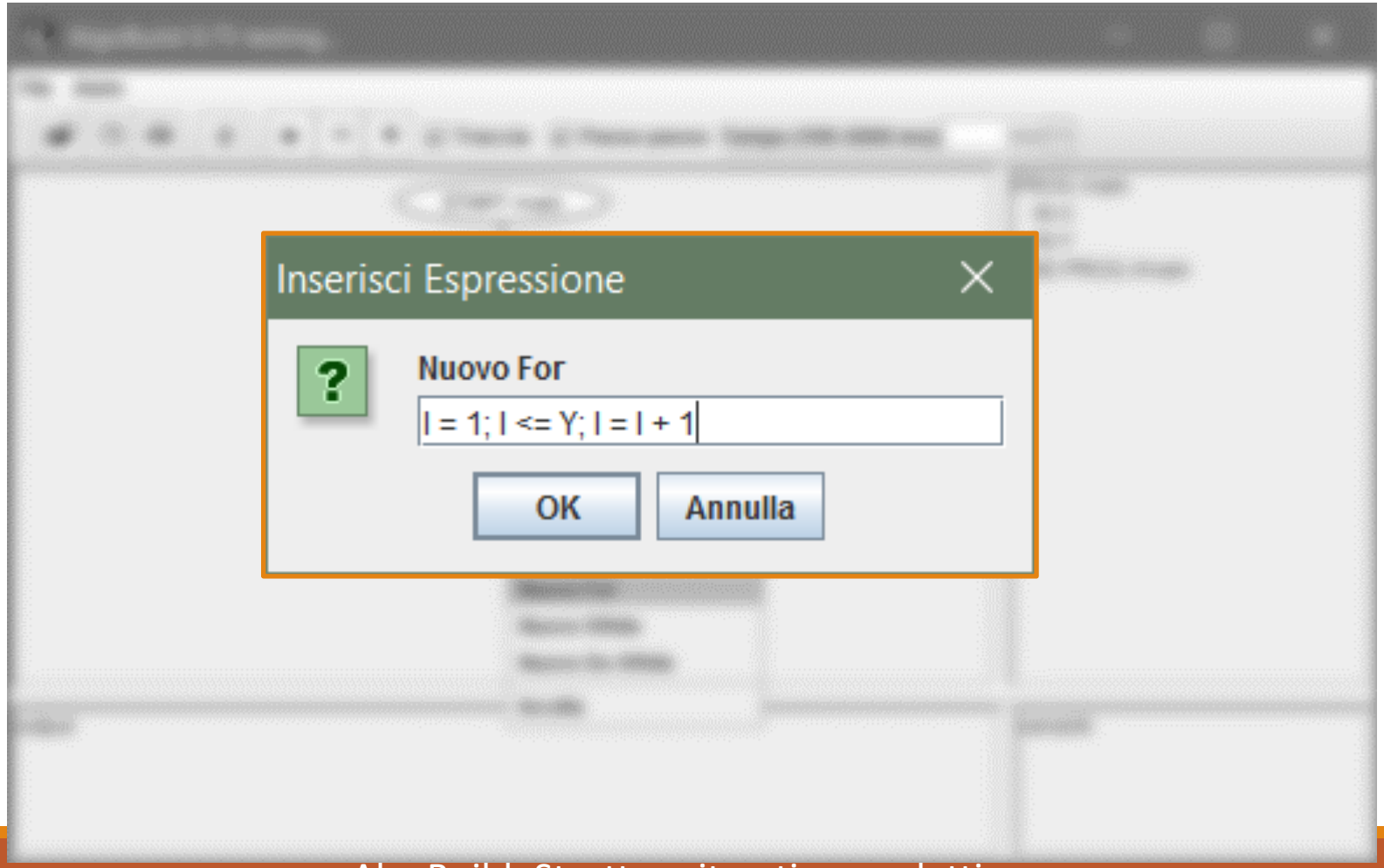
Struttura Iterativa FOR – Esempio 1



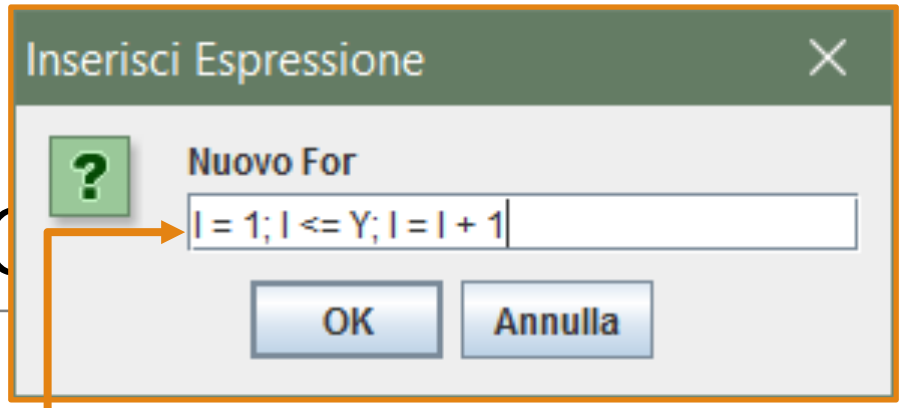
Struttura Iterativa FOR – Esempio 1



Struttura Iterativa FOR – Esempio 1

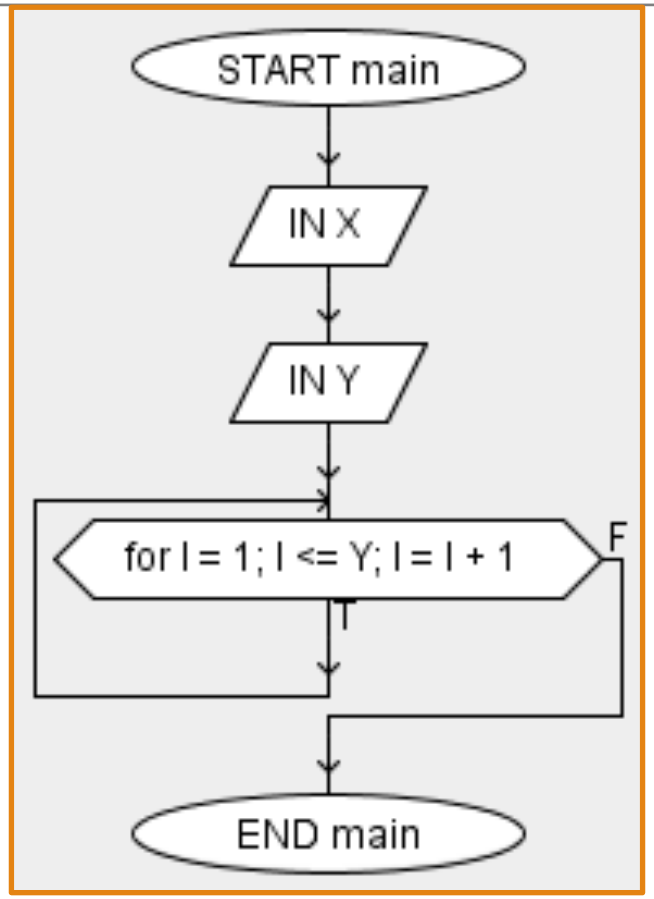


Struttura Iterativa FO



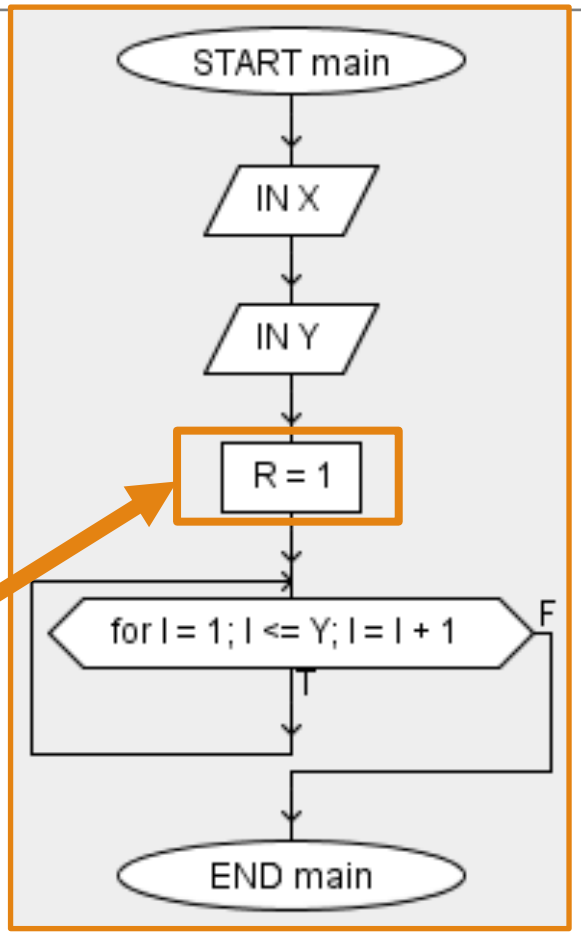
- I = 1; I <= Y; I = I + 1
  - I = 1
    - *Inizializzazione* del ciclo FOR (alla prima iterazione I = 1)
  - I <= Y
    - *Condizione di uscita* dal ciclo FOR (appena I sarà maggiore di Y, il ciclo FOR si concluderà)
  - I = I + 1
    - *Incremento*: ad ogni iterazione la variabile I verrà incrementata *automaticamente* di 1
- In totale il ciclo verrà eseguito **Y volte**

Struttura Iterativa FOR – Esempio 1

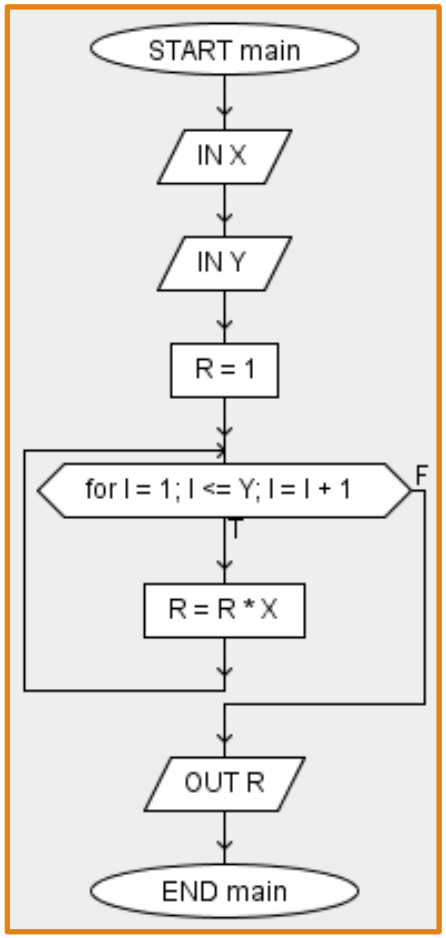


Struttura Iterativa FOR – Esempio 1

R denota la variabile in cui verrà memorizzato di volta in volta l'output. Tale variabile è inizializzata ad 1



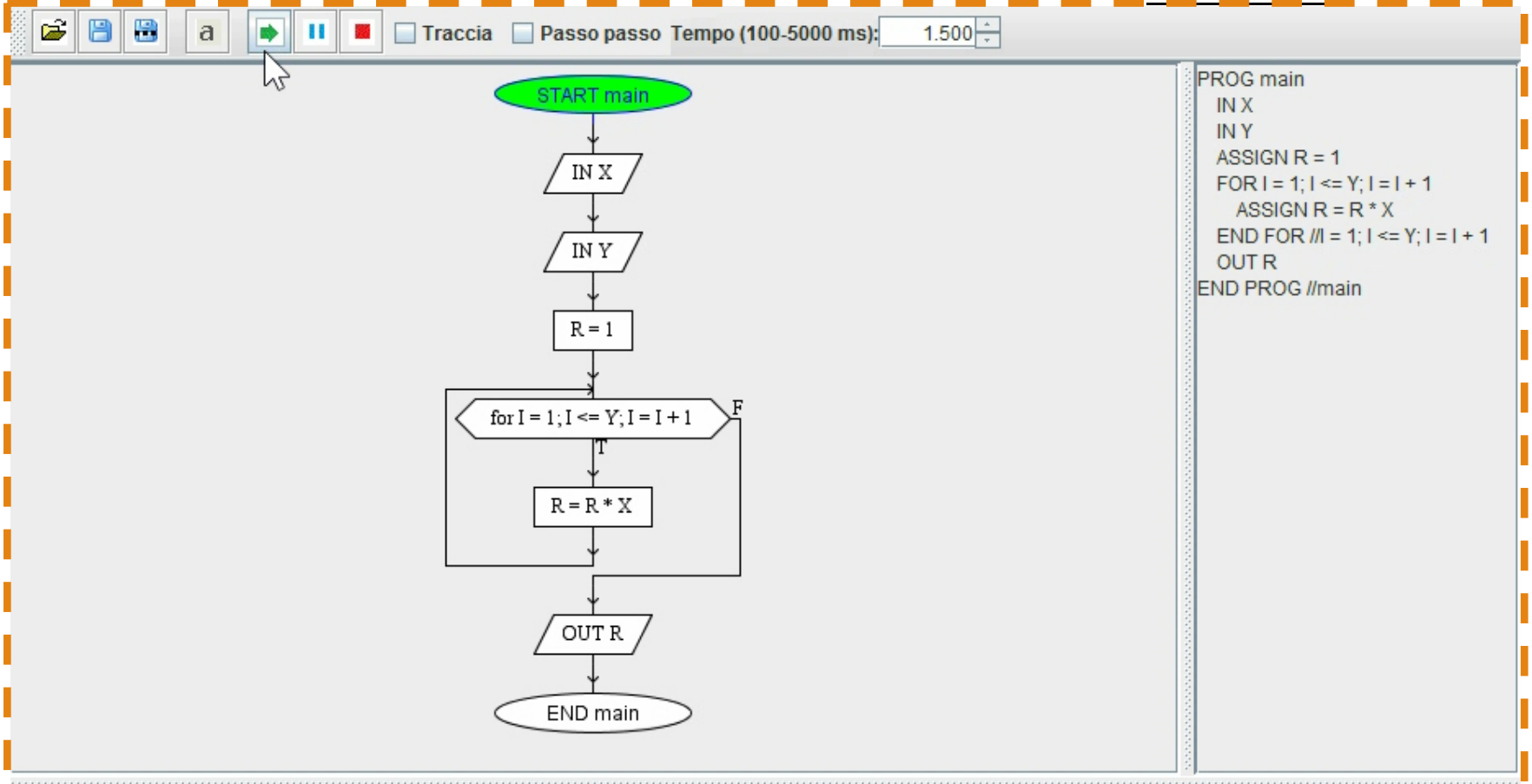
Struttura Iterativa FOR – Esempio 1



PSEUDO-CODICE

```
PROG main
   IN X
   IN Y
   ASSIGN R = 1
   FOR I = 1; I <= Y; I = I + 1
      ASSIGN R = R * X
   END FOR //I = 1; I <= Y; I = I + 1
   OUT R
END PROG //main
```

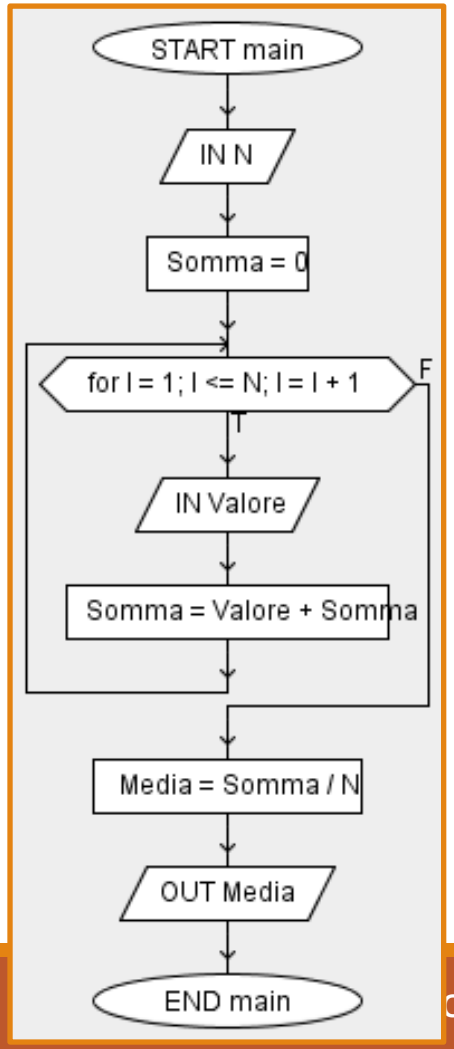
Struttura Iterativa FOR – DEMO (**Video**)



Struttura Iterativa FOR – Esempio 2

- Definiamo un diagramma di flusso che rappresenta il calcolo della media aritmetica di **N** numeri presi in input, dove
  - **N** è una variabile presa in input

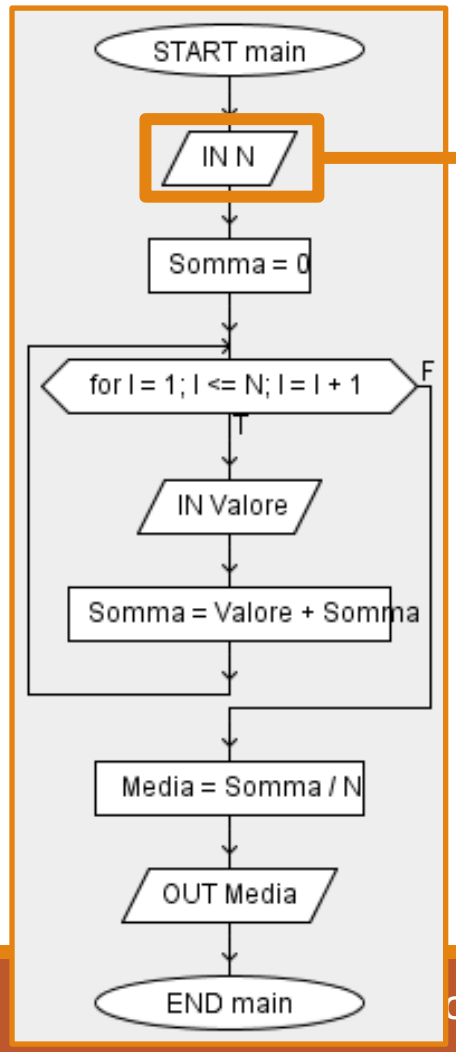
Struttura Iterativa FOR – Esempio 2



PSEUDO-CODICE

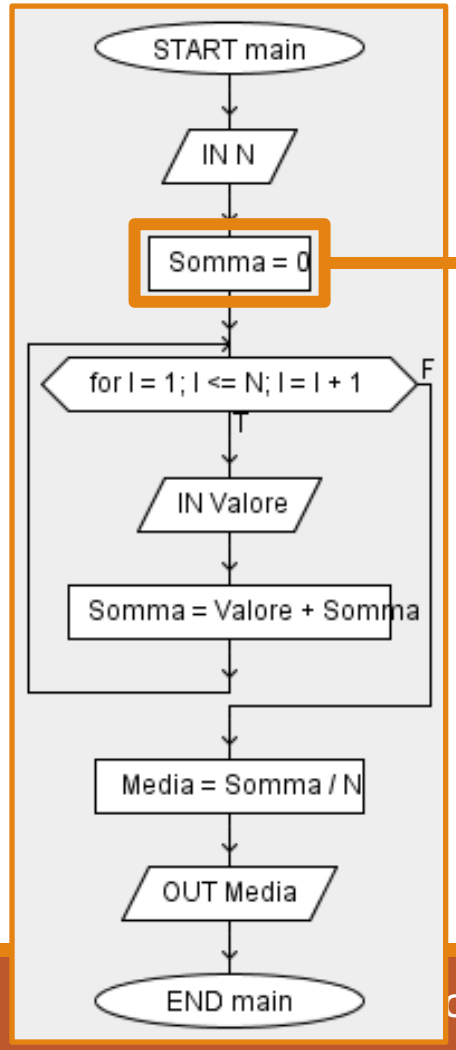
```
PROG main
   IN N
   ASSIGN Somma = 0
   FOR I = 1; I <= N; I = I + 1
      IN Valore
      ASSIGN Somma = Valore + Somma
   END FOR //I = 1; I <= N; I = I + 1
   ASSIGN Media = Somma / N
   OUT Media
END PROG //main
```

Struttura Iterativa FOR – Esempio 2



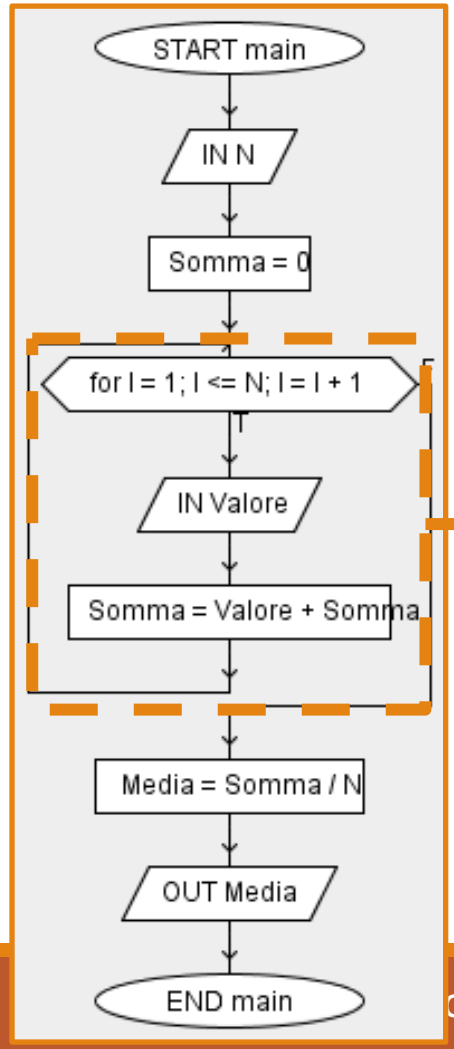
Richiedo **N** all'utente (Input)

Struttura Iterativa FOR – Esempio 2



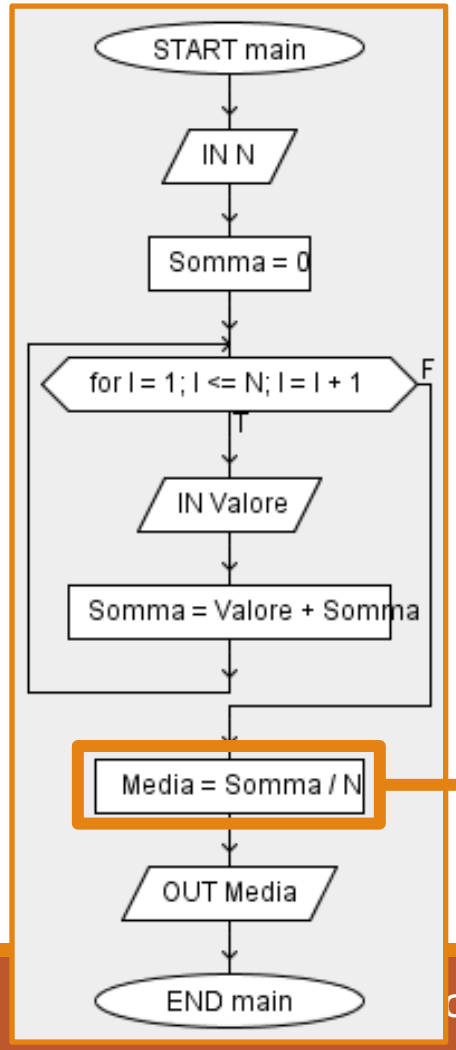
- Imposto **Somma** a zero
  - Ciò mi servirà per tener traccia della somma degli N numeri

Struttura Iterativa FOR – Esempio 2



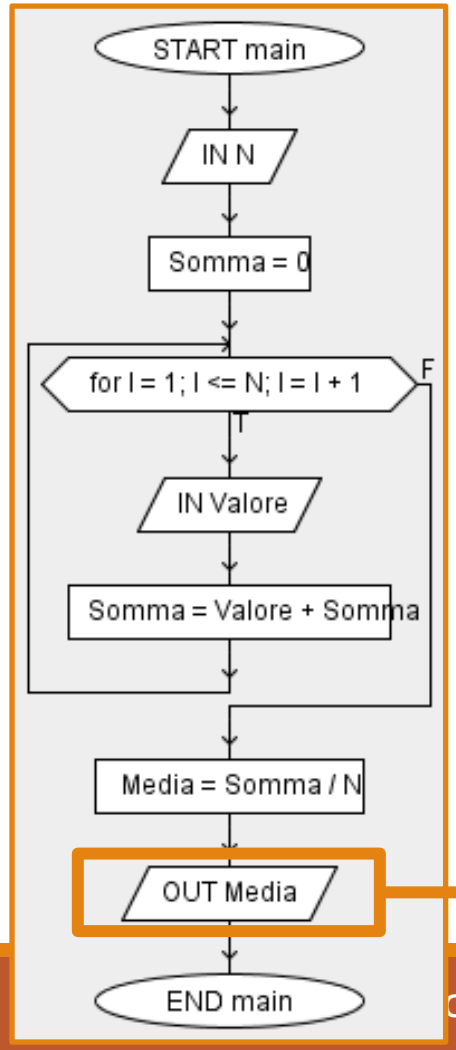
- Ciclo **for** che va da 1 a **N**
  - Ad ogni iterazione
    - Viene richiesto in input un numero (**IN Valore**)
    - Viene aggiornata la variabile **Somma** che verrà utilizzata successivamente per il calcolo della media aritmetica
      - **Somma = *Somma* + Valore**

Struttura Iterativa FOR – Esempio 2



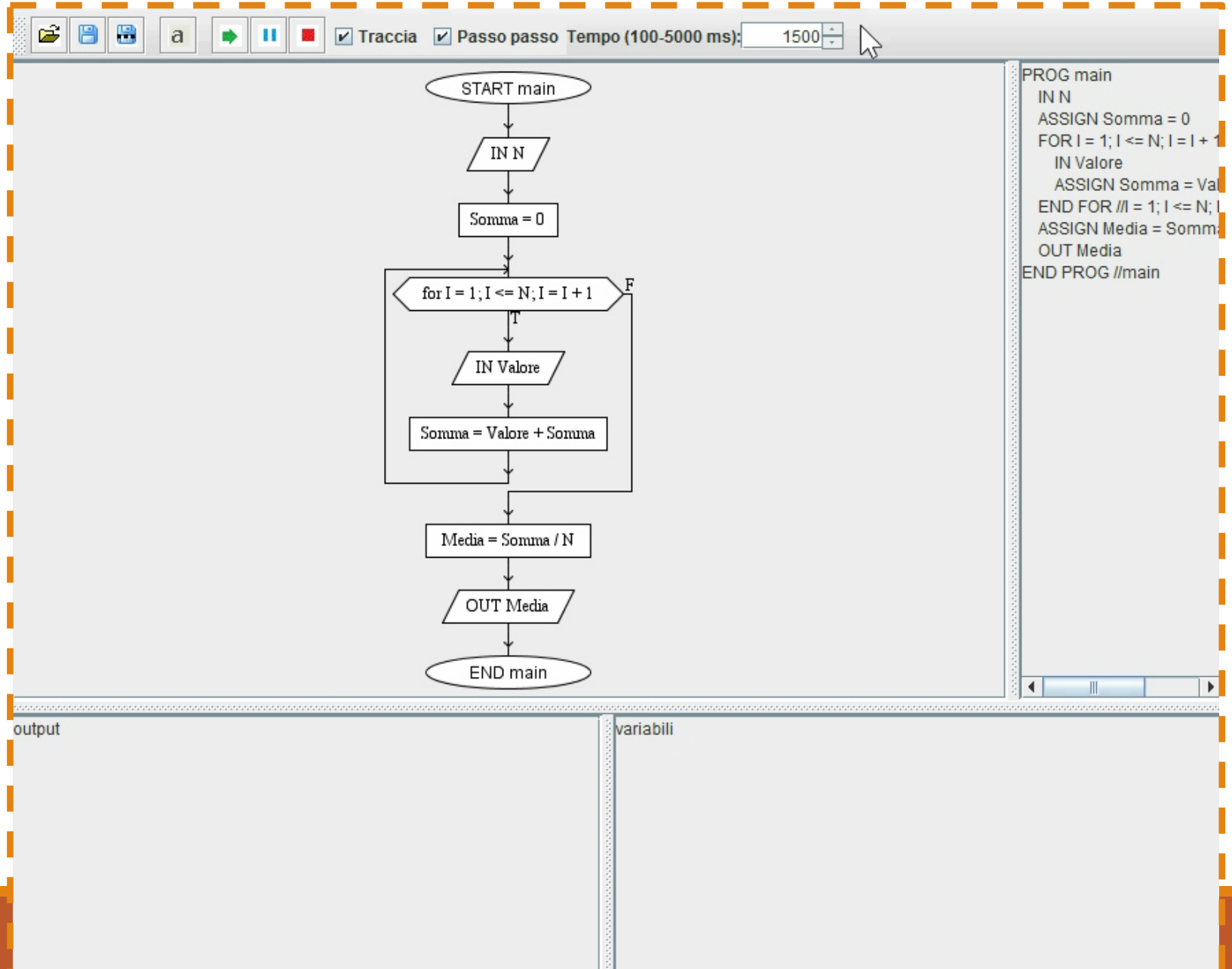
- Calcolo la media
  - Dividendo la somma (variabile **Somma**) per gli **N** numeri presi in input

Struttura Iterativa FOR – Esempio 2

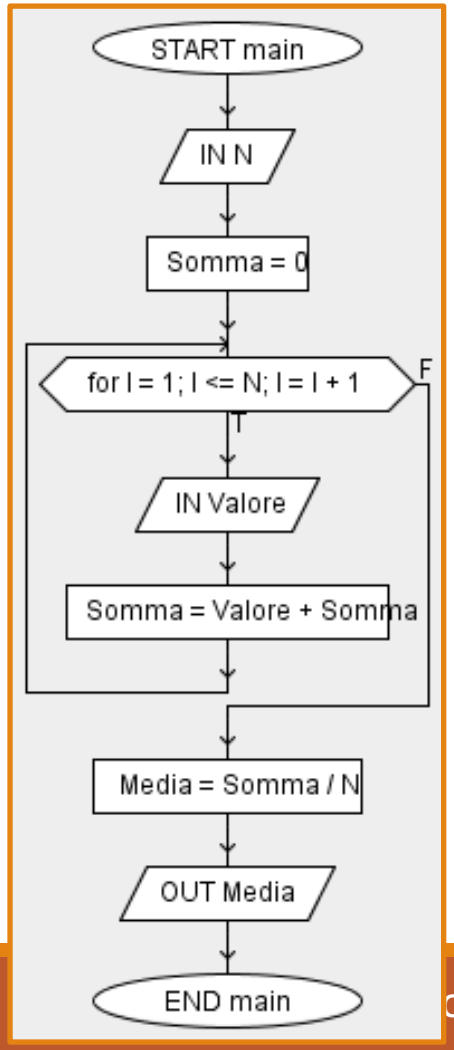


Il contenuto della variabile **Media** viene mostrato in output

Struttura Iterativa FOR – Demo (**Video**)



Struttura Iterativa FOR – Esempio 2

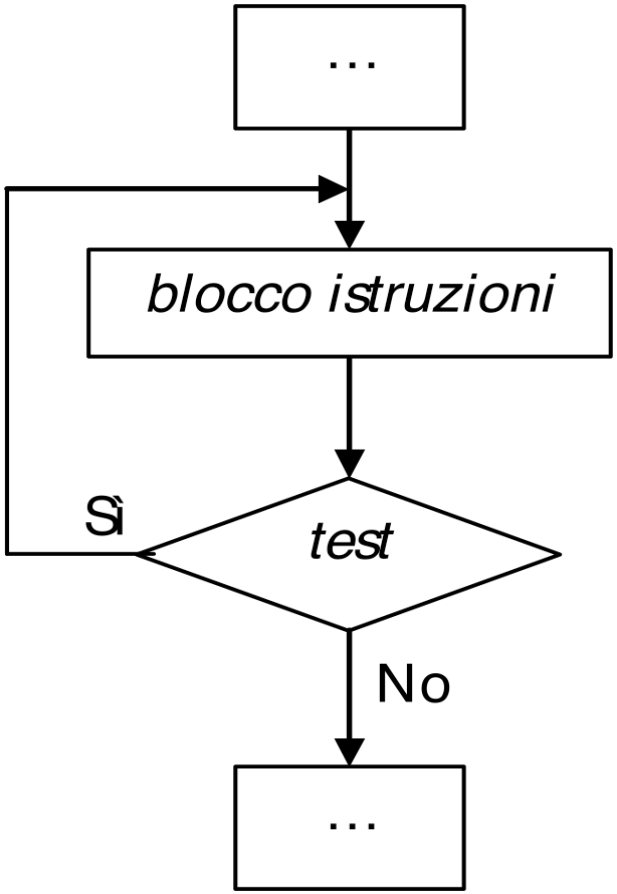


OSSERVAZIONE

È necessario fornire preventivamente al nostro *programma* il numero di valori di cui si dovrà calcolare la media…

…è possibile fare la media di un certo numero (non noto a priori) di valori presi in input?

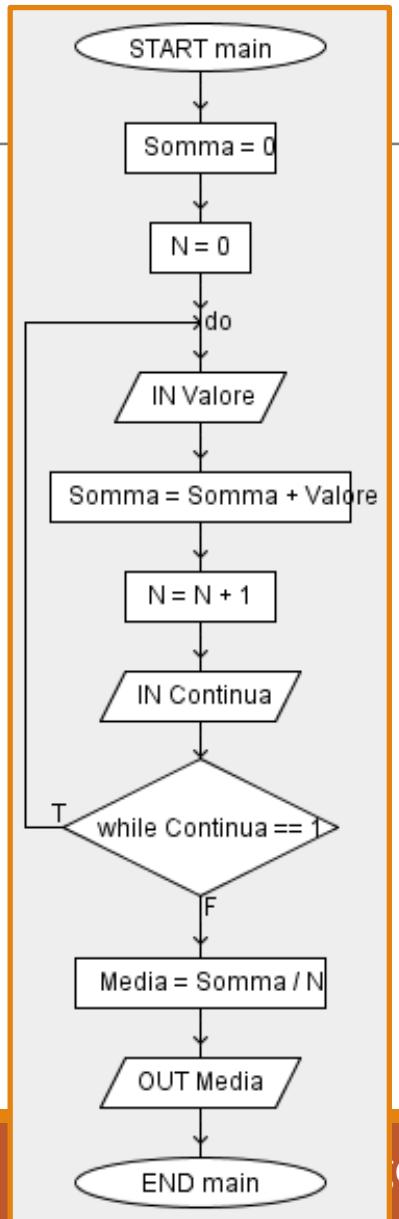
Strutture di Controllo: Ciclo a Condizione Finale



Struttura Iterativa DO/WHILE – Esempio

- Definiamo un diagramma di flusso che rappresenta il calcolo della media aritmetica di un certo numero (non noto a priori) di valori presi in input

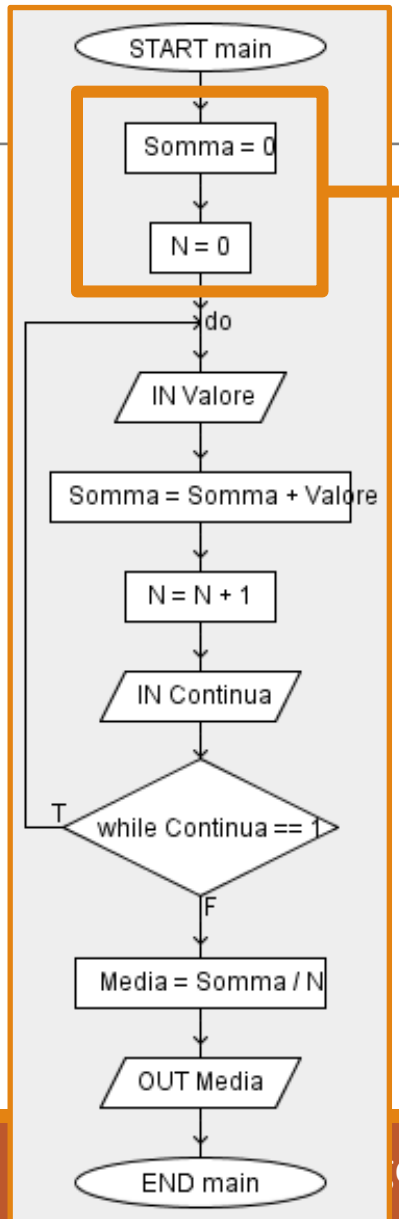
Struttura Iterativa DO/WHILE – Esempio



PSEUDO-CODICE

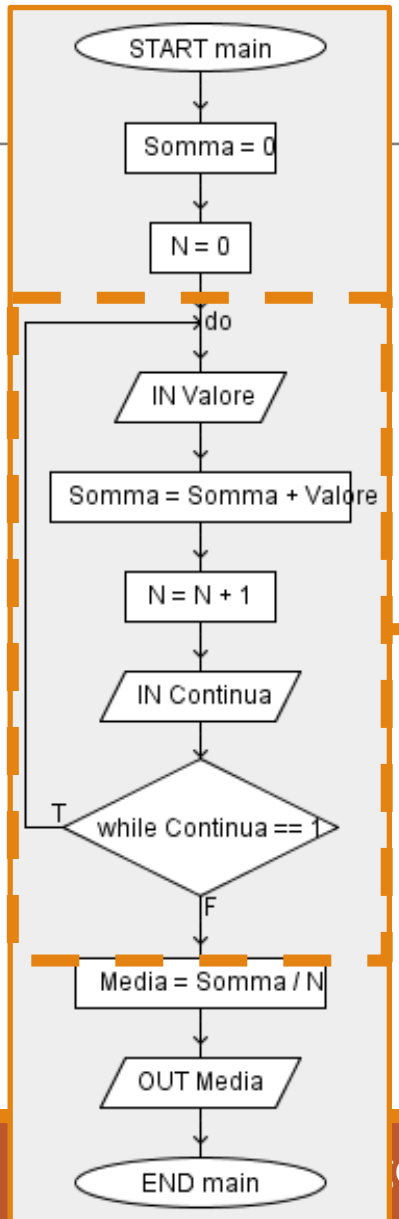
```
PROG main
   ASSIGN Somma = 0
   ASSIGN N = 0
   DO_WHILE //Continua == 1
      IN Valore
      ASSIGN Somma = Somma + Valore
      ASSIGN N = N + 1
      IN Continua
   END DO_WHILE Continua == 1
   ASSIGN Media = Somma / N
   OUT Media
END PROG //main
```

Struttura Iterativa DO/WHILE – Esempio



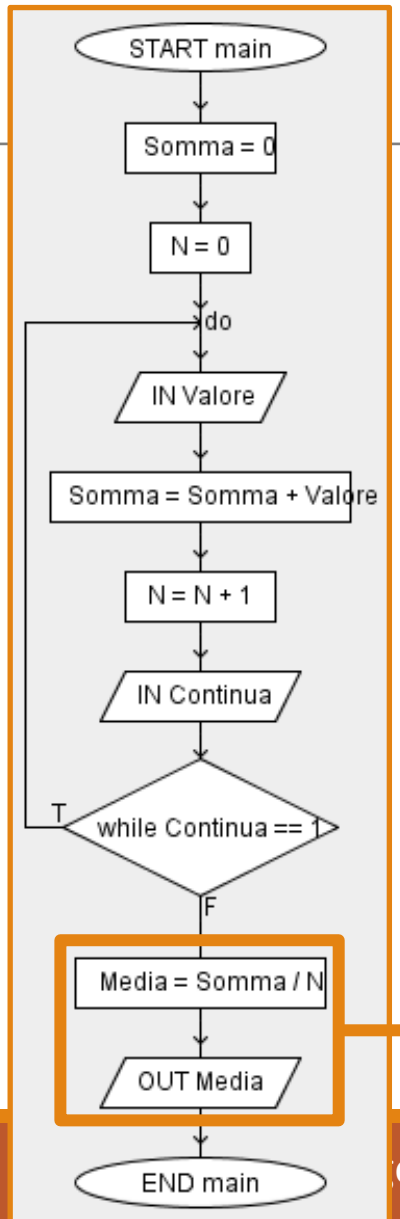
- Inizializzazione a 0 delle variabili **Somma** ed **N**
  - Non è stato preso ancora in input alcun valore (quindi **N** = 0)
  - Non è stata eseguita ancora la somma di alcun valore (quindi **Somma** = 0)

Struttura Iterativa DO/WHILE – Esempio



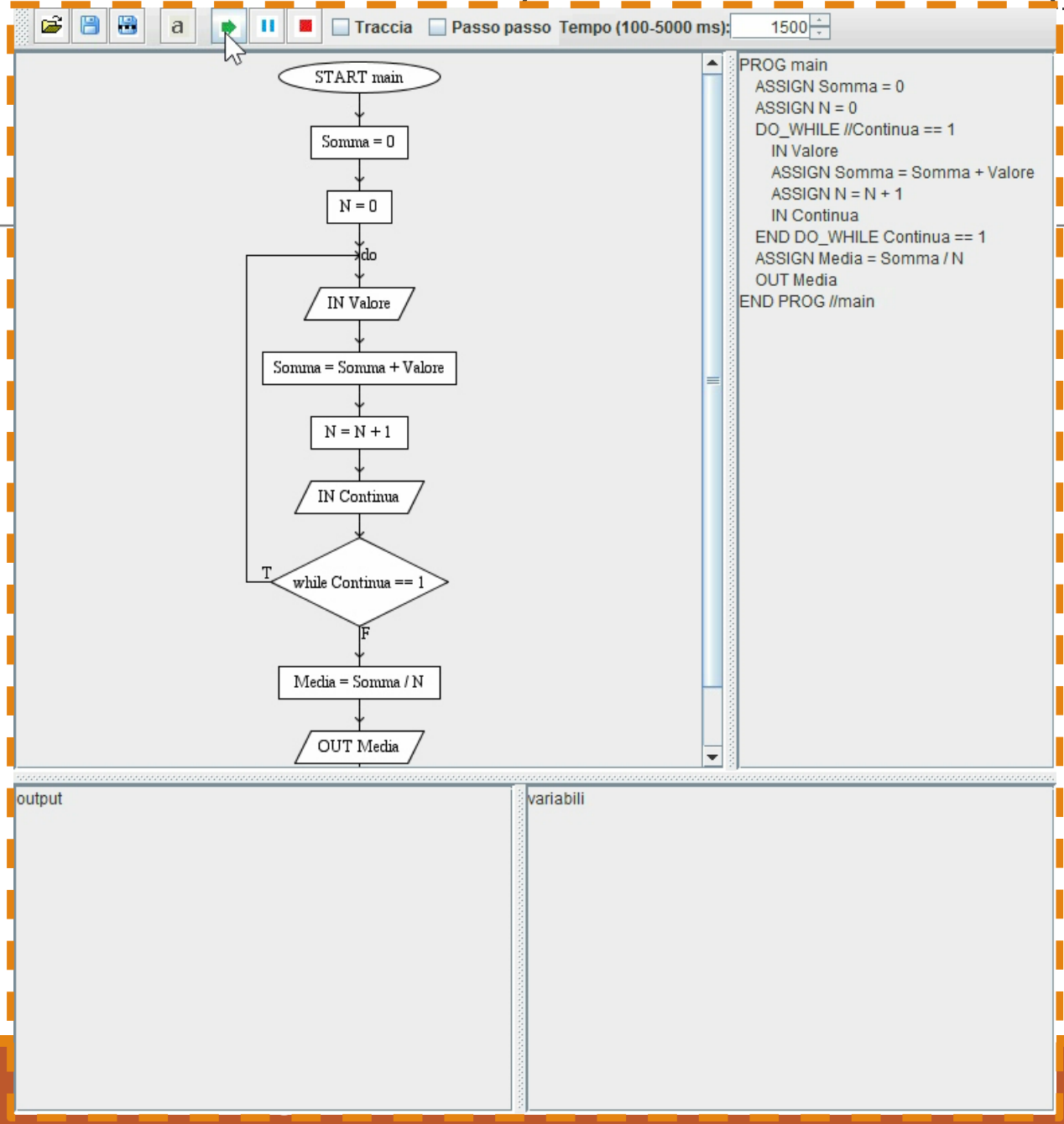
- Ciclo **DO/WHILE**
  - Richiede in input un valore (**IN Valore**)
  - Aggiorna la variabile **Somma**, aggiungendogli la variabile **Valore**
    - **Somma = Somma + Valore**
  - Aggiorna il valore di **N** (dato che è stato preso in input un nuovo valore)
    - **N = N + 1**
  - Chiede all'utente se vuole *continuare* (quindi un nuovo valore verrà preso in input)
    - Se il valore della variabile **Continua** sarà diverso da 1, allora il ciclo *while* si interromperà
    - Altrimenti si proseguirà con un'altra iterazione

Struttura Iterativa DO/WHILE – Esempio



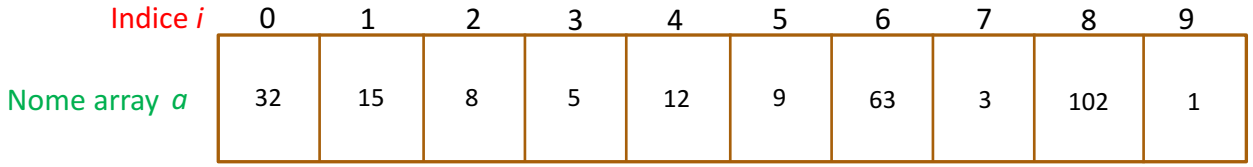
- Calcola la media aritmetica dividendo **Somma** per **N**
- La media aritmetica è memorizzata nella variabile **Media**
  - Che viene poi mostrata in output

Struttura Iterativa DO/WHILE – DEMO – (**Video**)



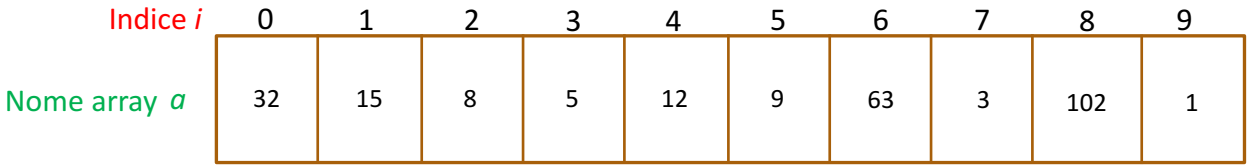
Array (o vettore)

- È una **variabile strutturata** dove è possibile memorizzare più valori tutti dello stesso tipo
- Un **array monodimensionale** o **vettore** può essere immaginato come un contenitore suddiviso in tanti scomparti quanti sono i dati che vi si vogliono memorizzare
  - Un **array bidimensionale** è noto come **matrice**
- Ognuno di questi scomparti, detto **elemento del vettore**, contiene un unico dato ed è individuato da un numero progressivo, detto **indice**, che specifica la posizione dell'elemento all'interno del vettore stesso
- Il numero complessivo degli elementi del vettore viene detto **lunghezza**



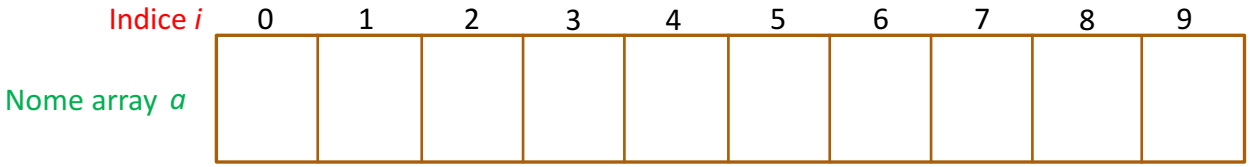
Array (o vettore)

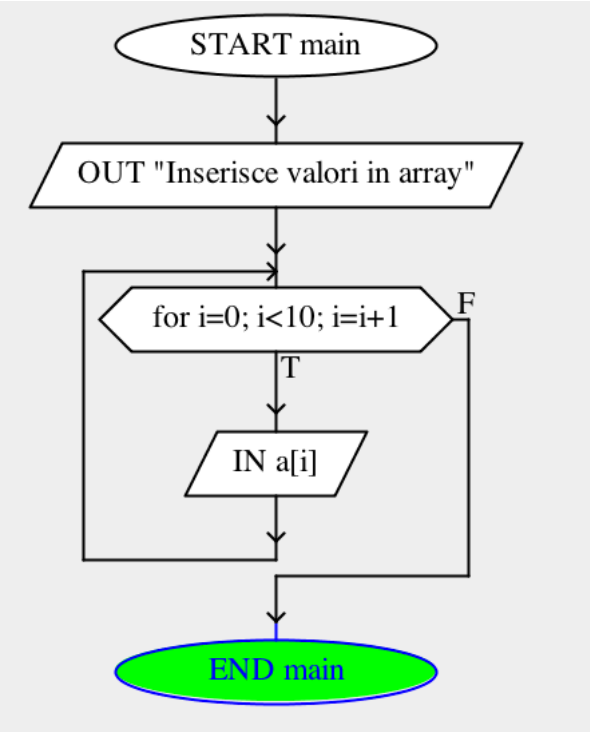
- In AlgoBuild, per accedere ad un singolo elemento di un array, si deve specificare il nome dell'array seguito dall'indice dell'elemento posto tra parentesi quadre: *a*[*i*]
- Es: *a*[*1*] restituisce l'elemento 15, *a*[*3*] restituisce l'elemento 5



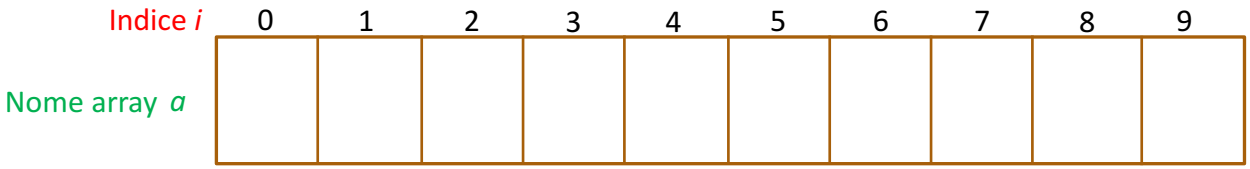
Esempio: Inserire gli elementi in un Array e stamparli

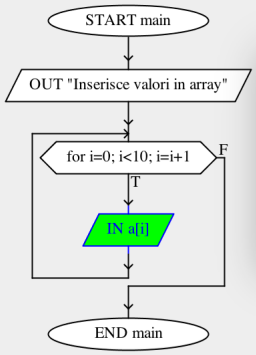
- Vediamo come inserire gli elementi (valori) in un vettore di lunghezza 10
  - Al termine dell'inserimento, gli elementi verranno poi stampati

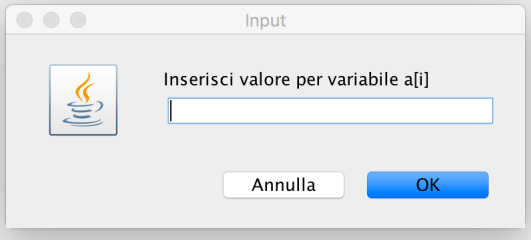





```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```



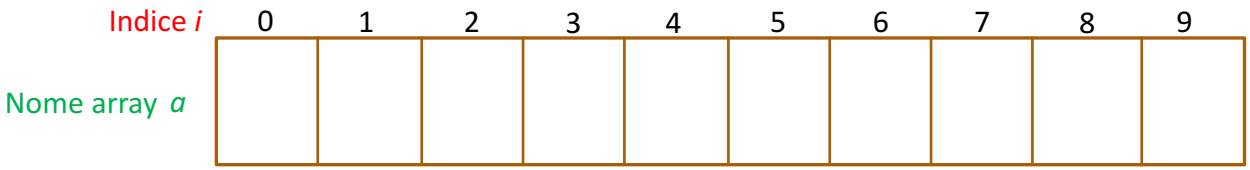


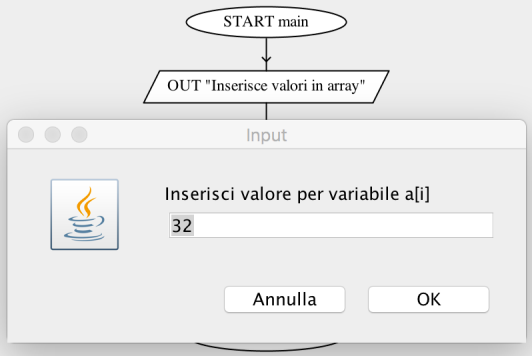



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
output
***  PROGRAMMA main inizia.
OUTPUT: Inserisce valori in array
   FOR (Inizializzazione): i <- 0.0
   var:  | i=0.0 |
   FOR i=0; i<10; i=i+1  iterazione: 1
```

```
variabili
i=0.0
```

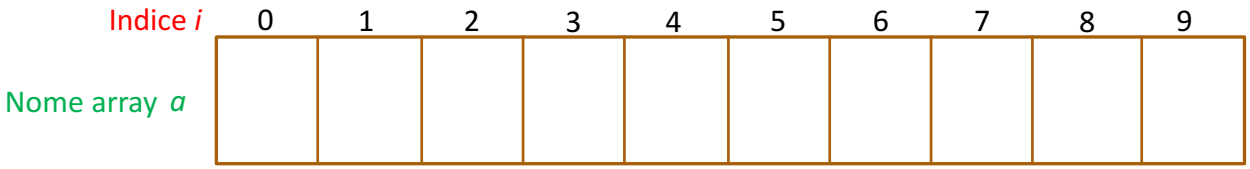


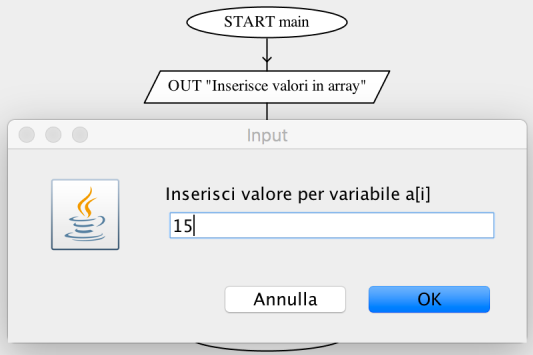



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
output
***  PROGRAMMA main inizia.
OUTPUT: Inserisce valori in array
   FOR (Inizializzazione): i <- 0.0
   var:  | i=0.0 |
   FOR i=0; i<10; i=i+1  iterazione: 1
```

```
variabili
i=0.0
```

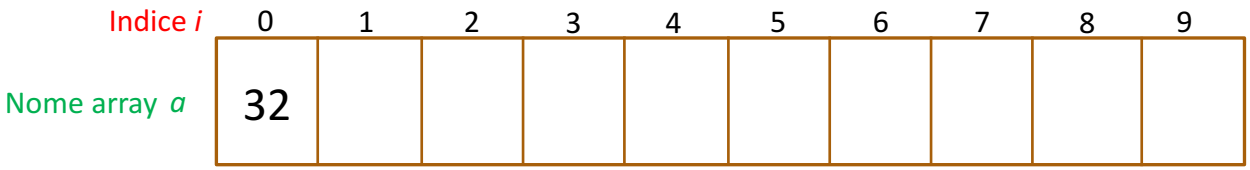


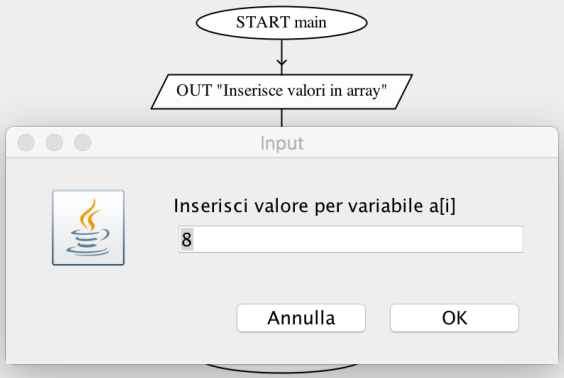



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
output
***  PROGRAMMA main inizia.
OUTPUT: Inserisce valori in array
   FOR (Inizializzazione): i <- 0.0
   var:  | i=0.0 |
   FOR i=0; i<10; i=i+1 iterazione: 1
INPUT: a[i]=32
   var:  | a={32.0} | i=0.0 |
   FOR (Aggiornamento): i <- 1.0
   var:  | a={32.0} | i=1.0 |
   FOR i=0; i<10; i=i+1 iterazione: 2
```

```
variabili
a={32.0}
i=1.0
```

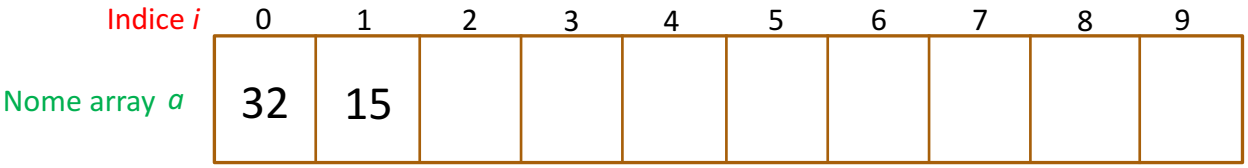


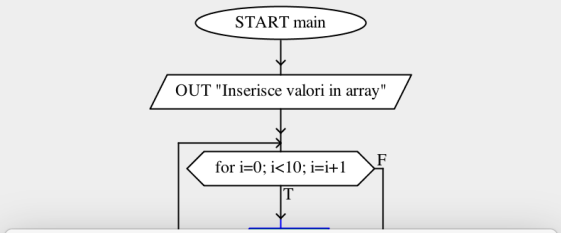


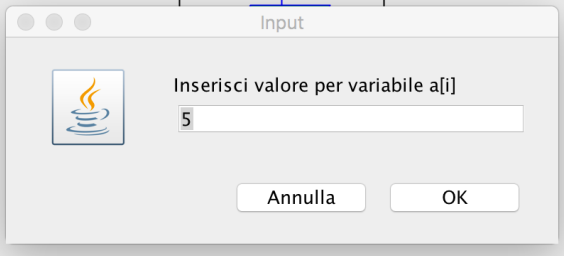

```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
OUTPUT: Inserisce valori in array
   FOR (Inizializzazione): i <- 0.0
   var:  | i=0.0 |
   FOR i=0; i<10; i=i+1 iterazione: 1
INPUT: a[i]=32
   var:  | a={32.0} | i=0.0 |
   FOR (Aggiornamento): i <- 1.0
   var:  | a={32.0} | i=1.0 |
   FOR i=0; i<10; i=i+1 iterazione: 2
INPUT: a[i]=15
   var:  | a={32.0,15.0} | i=1.0 |
   FOR (Aggiornamento): i <- 2.0
   var:  | a={32.0,15.0} | i=2.0 |
   FOR i=0; i<10; i=i+1 iterazione: 3
```

```
variabili
a={32.0,15.0}
i=2.0
```



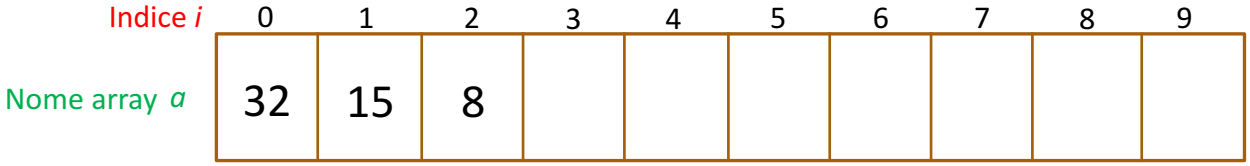


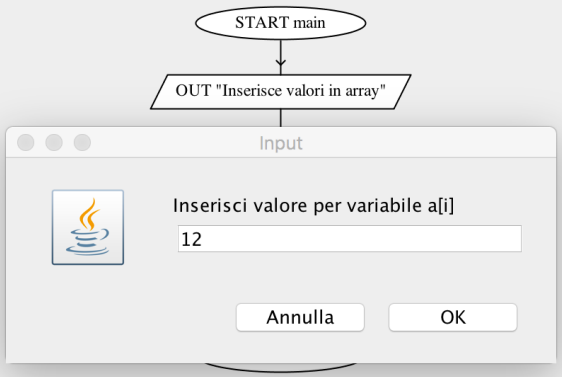



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
   var:  | a={32.0} | i=0.0 |
   FOR (Aggiornamento): i <- 1.0
   var:  | a={32.0} | i=1.0 |
   FOR i=0; i<10; i=i+1 iterazione: 2
INPUT: a[i]=15
   var:  | a={32.0,15.0} | i=1.0 |
   FOR (Aggiornamento): i <- 2.0
   var:  | a={32.0,15.0} | i=2.0 |
   FOR i=0; i<10; i=i+1 iterazione: 3
INPUT: a[i]=8
   var:  | a={32.0,15.0,8.0} | i=2.0 |
   FOR (Aggiornamento): i <- 3.0
   var:  | a={32.0,15.0,8.0} | i=3.0 |
   FOR i=0; i<10; i=i+1 iterazione: 4
```

```
variabili
a={32.0,15.0,8.0}
i=3.0
```



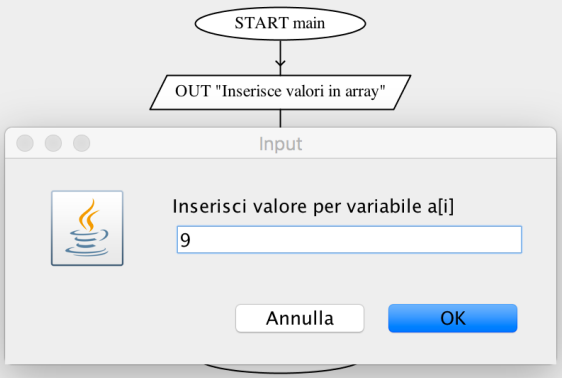



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
   var:  | a={32.0,15.0} | i=1.0 |
   FOR (Aggiornamento): i <- 2.0
   var:  | a={32.0,15.0} | i=2.0 |
   FOR i=0; i<10; i=i+1 iterazione: 3
INPUT: a[i]=8
   var:  | a={32.0,15.0,8.0} | i=2.0 |
   FOR (Aggiornamento): i <- 3.0
   var:  | a={32.0,15.0,8.0} | i=3.0 |
   FOR i=0; i<10; i=i+1 iterazione: 4
INPUT: a[i]=5
   var:  | a={32.0,15.0,8.0,5.0} | i=3.0 |
   FOR (Aggiornamento): i <- 4.0
   var:  | a={32.0,15.0,8.0,5.0} | i=4.0 |
   FOR i=0; i<10; i=i+1 iterazione: 5
```

```
variabili
a={32.0,15.0,8.0,5.0}
i=4.0
```

Indice *i*	0	1	2	3	4	5	6	7	8	9
Nome array *a*	32	15	8	5						

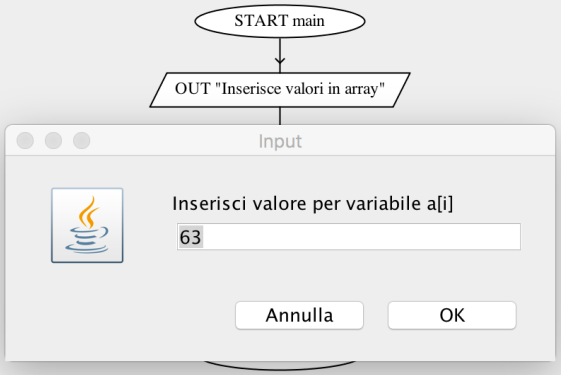



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
   var:  | a={32.0,15.0,8.0} | i=2.0 |
   FOR (Aggiornamento): i <- 3.0
   var:  | a={32.0,15.0,8.0} | i=3.0 |
   FOR i=0; i<10; i=i+1 iterazione: 4
INPUT: a[i]=5
   var:  | a={32.0,15.0,8.0,5.0} | i=3.0 |
   FOR (Aggiornamento): i <- 4.0
   var:  | a={32.0,15.0,8.0,5.0} | i=4.0 |
   FOR i=0; i<10; i=i+1 iterazione: 5
INPUT: a[i]=12
   var:  | a={32.0,15.0,8.0,5.0,12.0} | i=4.0 |
   FOR (Aggiornamento): i <- 5.0
   var:  | a={32.0,15.0,8.0,5.0,12.0} | i=5.0 |
   FOR i=0; i<10; i=i+1 iterazione: 6
```

```
variabili
a={32.0,15.0,8.0,5.0,12.0}
i=5.0
```

Indice *i*	0	1	2	3	4	5	6	7	8	9
Nome array *a*	32	15	8	5	12					

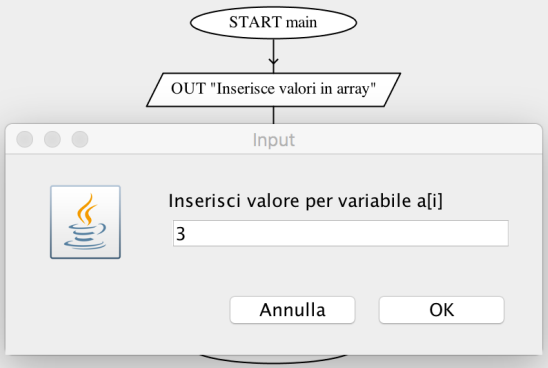



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
   var:  | a={32.0,15.0,8.0,5.0} | i=3.0 |
   FOR (Aggiornamento): i <- 4.0
   var:  | a={32.0,15.0,8.0,5.0} | i=4.0 |
   FOR i=0; i<10; i=i+1 iterazione: 5
INPUT: a[i]=12
   var:  | a={32.0,15.0,8.0,5.0,12.0} | i=4.0 |
   FOR (Aggiornamento): i <- 5.0
   var:  | a={32.0,15.0,8.0,5.0,12.0} | i=5.0 |
   FOR i=0; i<10; i=i+1 iterazione: 6
INPUT: a[i]=9
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0} | i=5.0 |
   FOR (Aggiornamento): i <- 6.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0} | i=6.0 |
   FOR i=0; i<10; i=i+1 iterazione: 7
```

```
variabili
a={32.0,15.0,8.0,5.0,12.0,9.0}
i=6.0
```

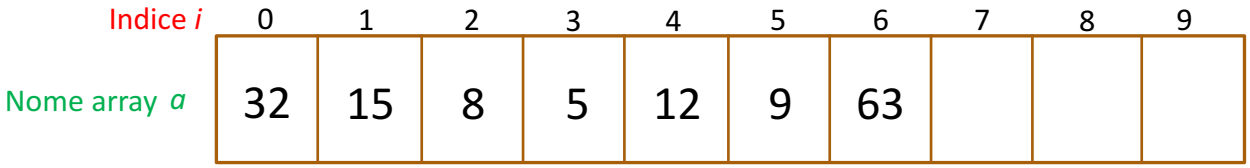
Indice *i*	0	1	2	3	4	5	6	7	8	9
Nome array *a*	32	15	8	5	12	9				

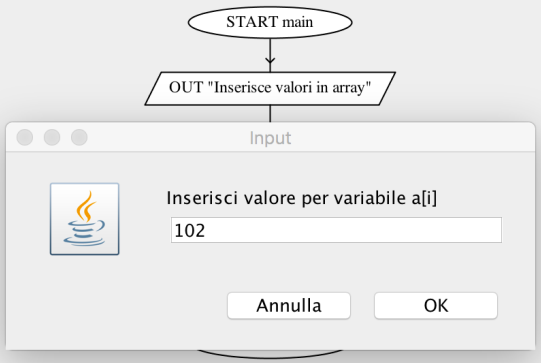



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
   var:  | a={32.0,15.0,8.0,5.0,12.0} | i=4.0 |
   FOR (Aggiornamento): i <- 5.0
   var:  | a={32.0,15.0,8.0,5.0,12.0} | i=5.0 |
   FOR i=0; i<10; i=i+1 iterazione: 6
INPUT: a[i]=9
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0} | i=5.0 |
   FOR (Aggiornamento): i <- 6.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0} | i=6.0 |
   FOR i=0; i<10; i=i+1 iterazione: 7
INPUT: a[i]=63
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0} | i=6.0 |
   FOR (Aggiornamento): i <- 7.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0} | i=7.0 |
   FOR i=0; i<10; i=i+1 iterazione: 8
```

```
variabili
a={32.0,15.0,8.0,5.0,12.0,9.0,63.0}
i=7.0
```



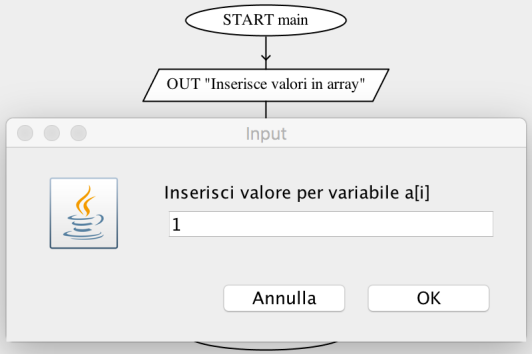



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0} | i=5.0 |
   FOR (Aggiornamento): i <- 6.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0} | i=6.0 |
   FOR i=0; i<10; i=i+1 iterazione: 7
INPUT: a[i]=63
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0} | i=6.0 |
   FOR (Aggiornamento): i <- 7.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0} | i=7.0 |
   FOR i=0; i<10; i=i+1 iterazione: 8
INPUT: a[i]=3
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0} | i=7.0 |
   FOR (Aggiornamento): i <- 8.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0} | i=8.0 |
   FOR i=0; i<10; i=i+1 iterazione: 9
```

```
variabili
a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0}
i=8.0
```

Indice *i*	0	1	2	3	4	5	6	7	8	9
Nome array *a*	32	15	8	5	12	9	63	3		

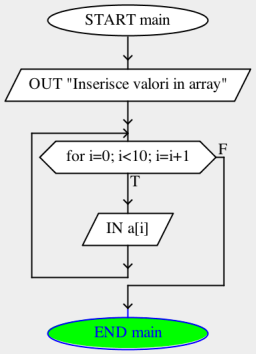



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0} | i=6.0 |
   FOR (Aggiornamento): i <- 7.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0} | i=7.0 |
   FOR i=0; i<10; i=i+1 iterazione: 8
INPUT: a[i]=3
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0} | i=7.0 |
   FOR (Aggiornamento): i <- 8.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0} | i=8.0 |
   FOR i=0; i<10; i=i+1 iterazione: 9
INPUT: a[i]=102
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0,102.0} | i=8.0 |
   FOR (Aggiornamento): i <- 9.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0,102.0} | i=9.0 |
   FOR i=0; i<10; i=i+1 iterazione: 10
```

```
variabili
a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0,102
i=9.0
```

Indice *i*	0	1	2	3	4	5	6	7	8	9
Nome array *a*	32	15	8	5	12	9	63	3	102	

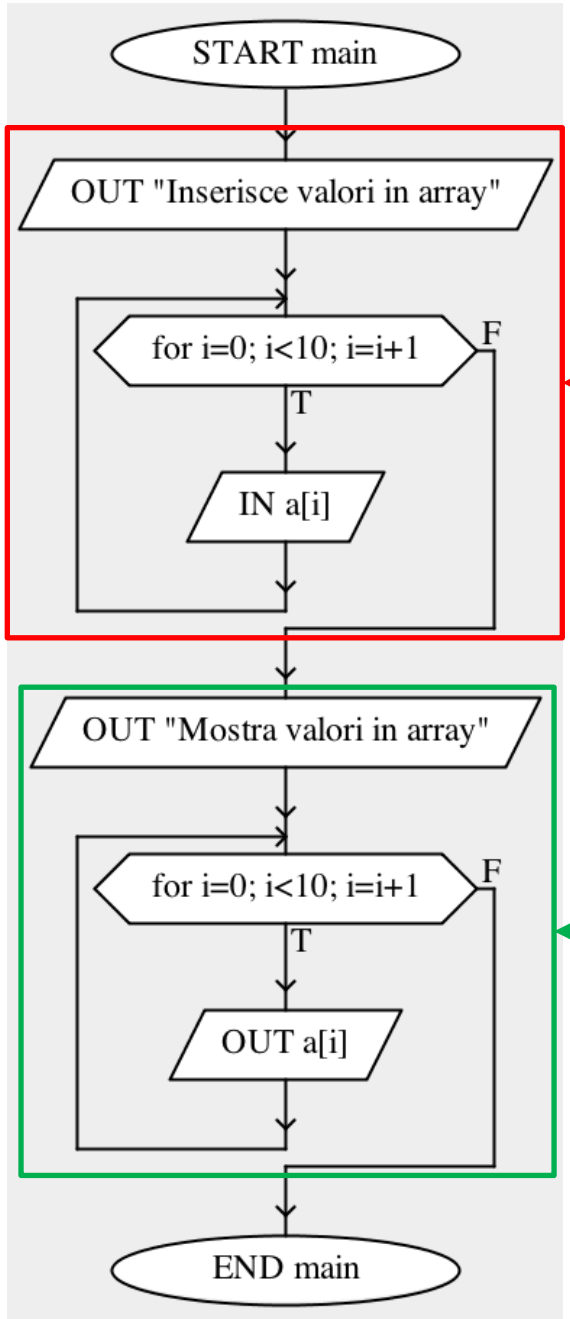



```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
END PROG //main
```

```
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0} | i=7.0 |
   FOR (Aggiornamento): i <- 8.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0} | i=8.0 |
   FOR i=0; i<10; i=i+1 iterazione: 9
INPUT: a[i]=102
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0,102.0} | i=8.0 |
   FOR (Aggiornamento): i <- 9.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0,102.0} | i=9.0 |
   FOR i=0; i<10; i=i+1 iterazione: 10
INPUT: a[i]=1
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0,102.0,1.0} | i=9.0 |
   FOR (Aggiornamento): i <- 10.0
   var:  | a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0,102.0,1.0} | i=10.0 |
   FOR i=0; i<10; i=i+1 termina. Eseguite 11 iterazioni
```

```
variabili
a={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0,102.0,1.0}
i=10.0
```

Indice *i*	0	1	2	3	4	5	6	7	8	9
Nome array *a*	32	15	8	5	12	9	63	3	102	1



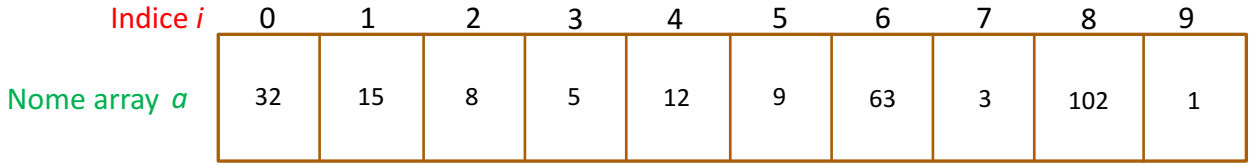
Prende valori in input

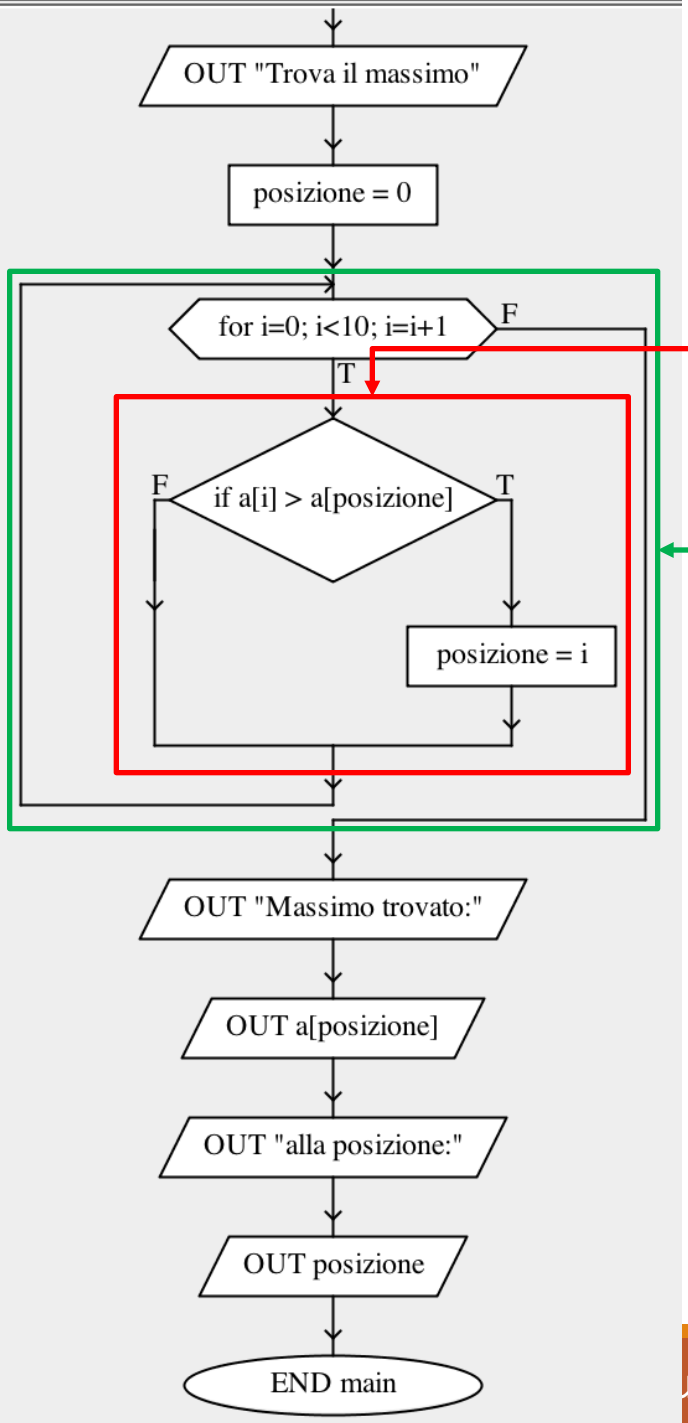
Restituisce valori in output

```
PROG main
    OUT "Inserisce valori in array"
    FOR i=0; i<10; i=i+1
        IN a[i]
    END FOR //i=0; i<10; i=i+1
    OUT "Mostra valori in array"
    FOR i=0; i<10; i=i+1
        OUT a[i]
    END FOR //i=0; i<10; i=i+1
END PROG //main
```

Esempio: Determinare il massimo elemento di un array e stamparne la posizione

```
INIZIO ALGORITMO trovaMax
posizione = 0
Per i che va da 0 a 9
        Se a[i] > a[posizione]
                posizione = i  //Istruzione eseguita se A(i) > max
        Incrementa i
restituisci a[posizione], posizione
FINE ALGORITMO trovaMax
```






```
PROG main
   OUT "Inserisce valori in array"
   FOR i=0; i<10; i=i+1
      IN a[i]
   END FOR //i=0; i<10; i=i+1
   OUT "Trova il massimo"
   ASSIGN posizione = 0
   FOR i=0; i<10; i=i+1
      IF a[i] > a[posizione]
         ASSIGN posizione = i
      ELSE //if a[i] > a[posizione]
      END IF //a[i] > a[posizione]
   END FOR //i=0; i<10; i=i+1
   OUT "Massimo trovato:"
   OUT a[posizione]
   OUT "alla posizione:"
   OUT posizione
END PROG //main
```

Modificare un'Istruzione

- Cliccando con il tasto sinistro su una specifica istruzione apparirà il *menu contestuale* che consente di
  - **Modificare** l'istruzione selezionata
  - **Copiare** l'istruzione
  - **Tagliare** l'istruzione (utile per spostarla da una parte ad un'altra, utilizzando i comandi *Taglia* e *Incolla*)