



UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

AlgoBuild: Strutture selettive, iterative ed array

Prof. Arcangelo Castiglione

A.A. 2016/17



AlgoBuild

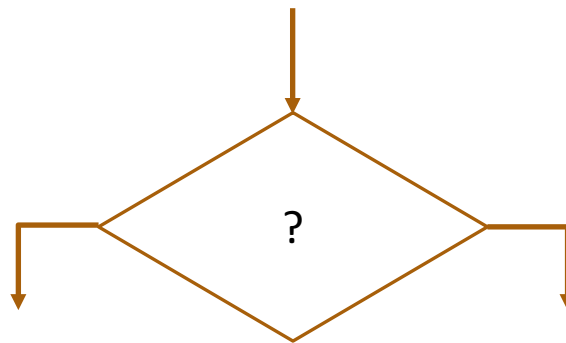
AlgoBuild : Strutture iterative e selettive

OUTLINE

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

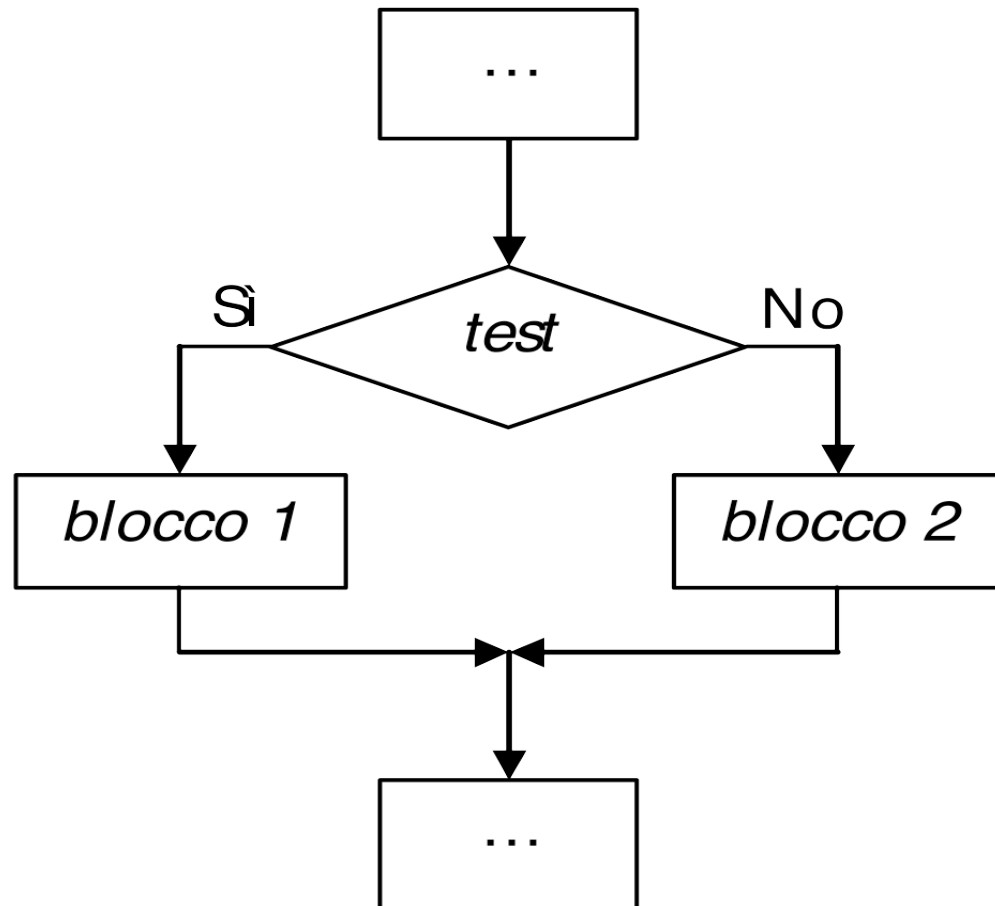
Diagrammi di Flusso: Blocco di Decisione Binaria (o Condizionale)

- 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

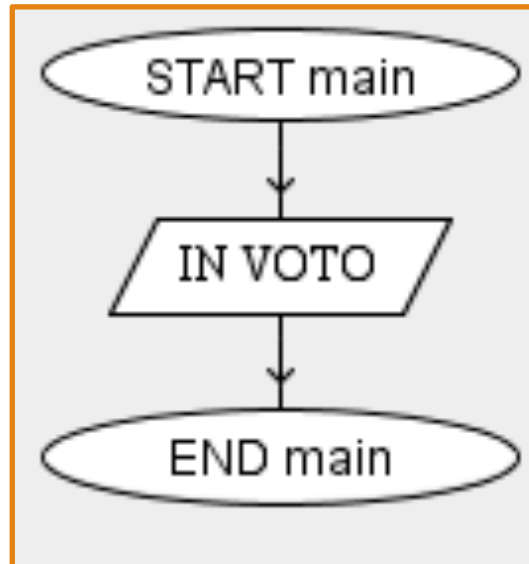


Struttura Selettiva IF – Esempio

- 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

The screenshot displays the AlgoBuild 0.75 testing... application window. The interface includes a menu bar with 'File' and 'Aiuto', a toolbar with icons for file operations and execution, and a main workspace. The workspace contains a flowchart with the following elements:

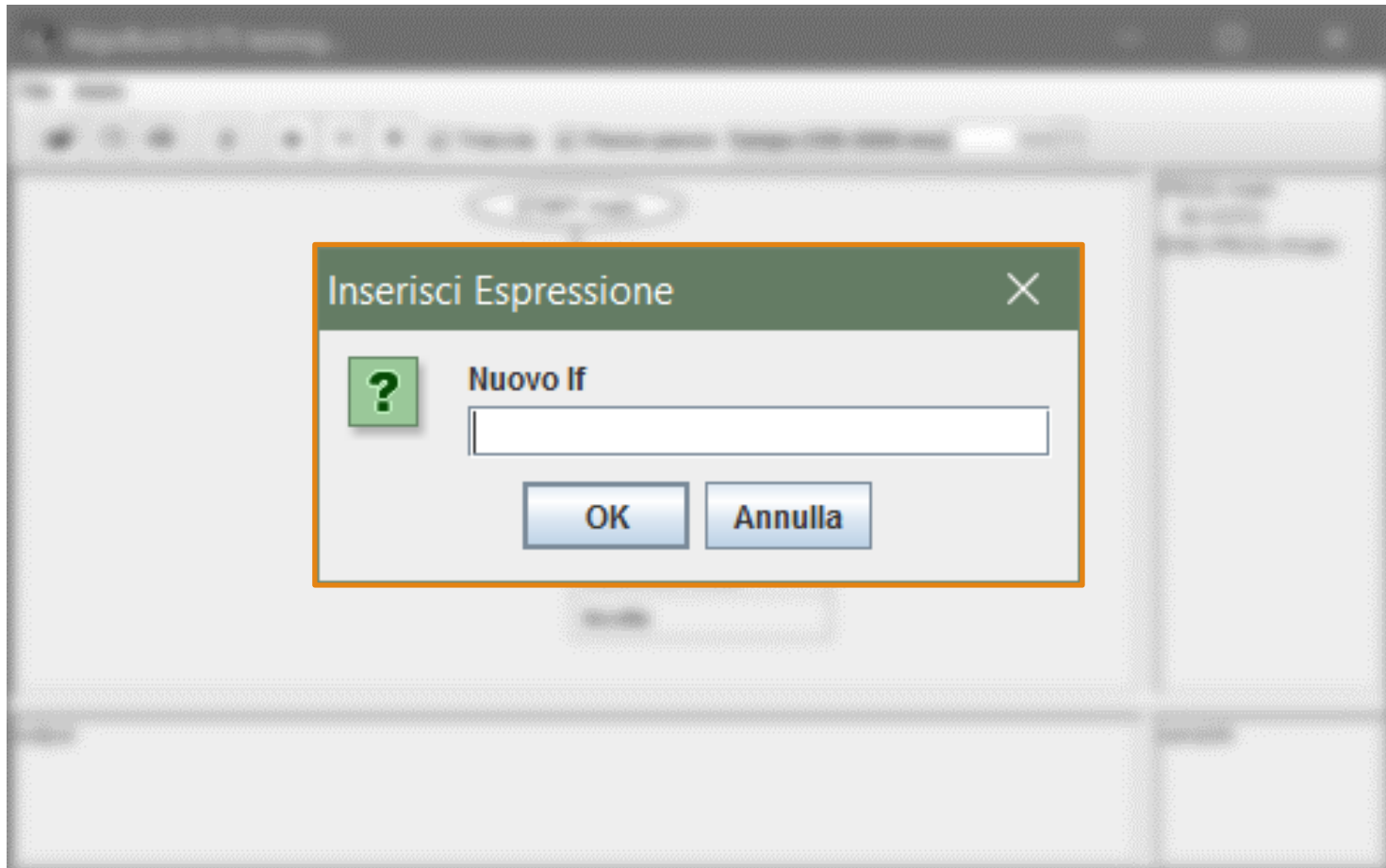
- An oval labeled 'START main' at the top.
- A downward arrow connecting 'START main' to a parallelogram labeled 'IN VOTO'.
- Another downward arrow connecting 'IN VOTO' to an oval labeled 'END'.

A context menu is open over the 'END' node, listing the following options:

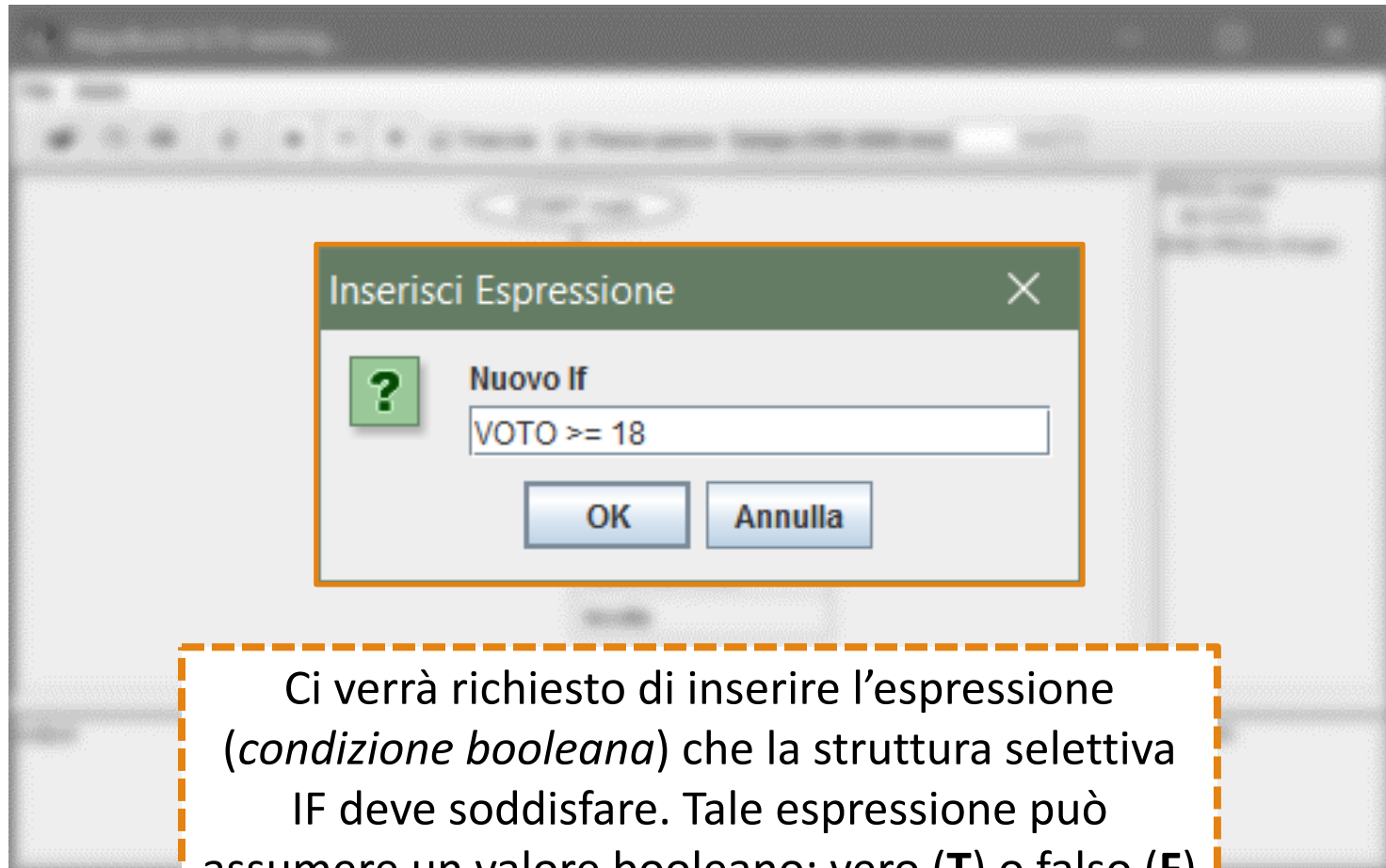
- Nuovo Assegnamento
- Nuovo Input
- Nuovo Output
- Nuovo If** (highlighted)
- Nuovo For
- Nuovo While
- Nuovo Do-While
- Incolla

An orange arrow points from the left towards the 'Nuovo If' option in the menu. On the right side of the workspace, there is a text area containing the code: 'PROG main', 'IN VOTO', and 'END PROG //main'. At the bottom of the window, there are two empty text areas labeled 'output' and 'variabili'.

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

The screenshot displays the AlgoBuild 0.75 testing environment. The main workspace contains a flowchart for an algorithm named 'main'. The flowchart starts with an oval labeled 'START main', followed by a parallelogram labeled 'IN VOTO'. Below this is a diamond-shaped decision node labeled 'if VOTO >= 18'. The 'F' (False) path loops back to the start of the decision diamond, while the 'T' (True) path proceeds to an oval labeled 'END main'. The right-hand panel shows the corresponding code in a text editor:

```
PROG main
  IN VOTO
  IF VOTO >= 18
  ELSE //if VOTO >= 18
  END IF //VOTO >= 18
END PROG //main
```

The interface also includes a menu bar with 'File' and 'Aiuto', a toolbar with icons for file operations and execution control, and a status bar with checkboxes for 'Traccia' and 'Passo passo', and a 'Tempo (100-5000 ms):' field set to 500. At the bottom, there are sections for 'output' and 'variabili'.

Struttura Selettiva IF

AlgoBuild 0.75 testing...

File Aiuto

Traccia Passo passo Tempo (100-5000 ms): 500

```
graph TD; Start([START main]) --> Input[/IN VOTO/]; Input --> Decision{if VOTO >= 18}; Decision -- F --> End([END main]); Decision -- T --> End;
```

```
PROGRAM main
INPUT VOTO
IF VOTO >= 18
ELSE //if VOTO >= 18
END IF //VOTO >= 18
END PROGRAM //main
```

output

variabili

Flusso di esecuzione se la condizione è falsa

Struttura Selettiva IF

The screenshot displays the AlgoBuild 0.75 testing environment. The main workspace shows a flowchart for a program named 'main'. The flow starts with an oval labeled 'START main', followed by a parallelogram labeled 'IN VOTO'. A diamond-shaped decision node contains the condition 'if VOTO >= 18'. The 'F' (False) path leads down to an oval labeled 'END main'. The 'T' (True) path leads right and then loops back to the entry point of the decision diamond. An orange box highlights the 'T' path, and an orange arrow points from the text 'Flusso di esecuzione se la condizione è vera' to this path.

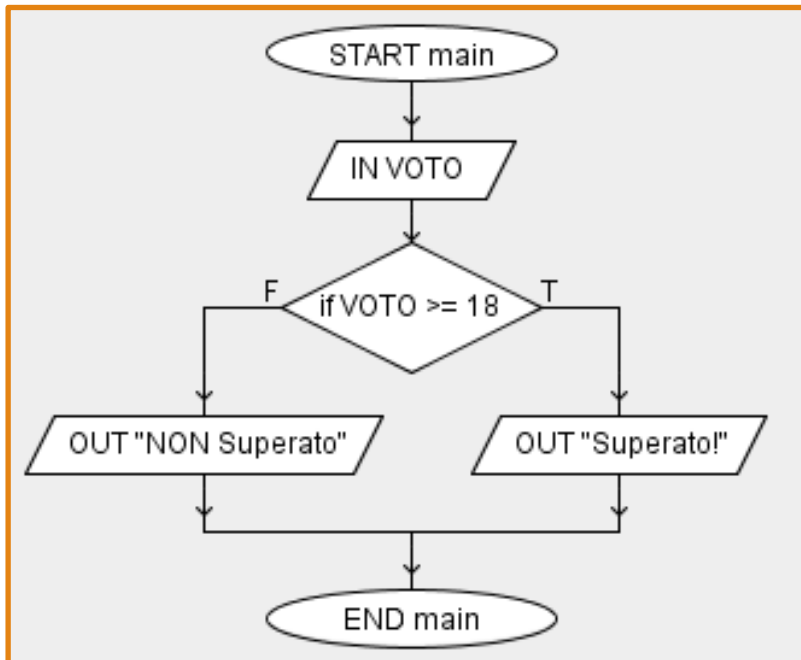
Below the flowchart, there are two empty panels: 'output' on the left and 'variabili' on the right.

The right-hand pane shows the following code:

```
PROG main
  IN VOTO
  IF VOTO >= 18
  ELSE //if VOTO >= 18
  END IF //VOTO >= 18
END PROG //main
```

The top toolbar includes icons for file operations, execution control (run, pause, stop), and checkboxes for 'Traccia' and 'Passo passo'. A 'Tempo (100-5000 ms):' field is set to 500.

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
```

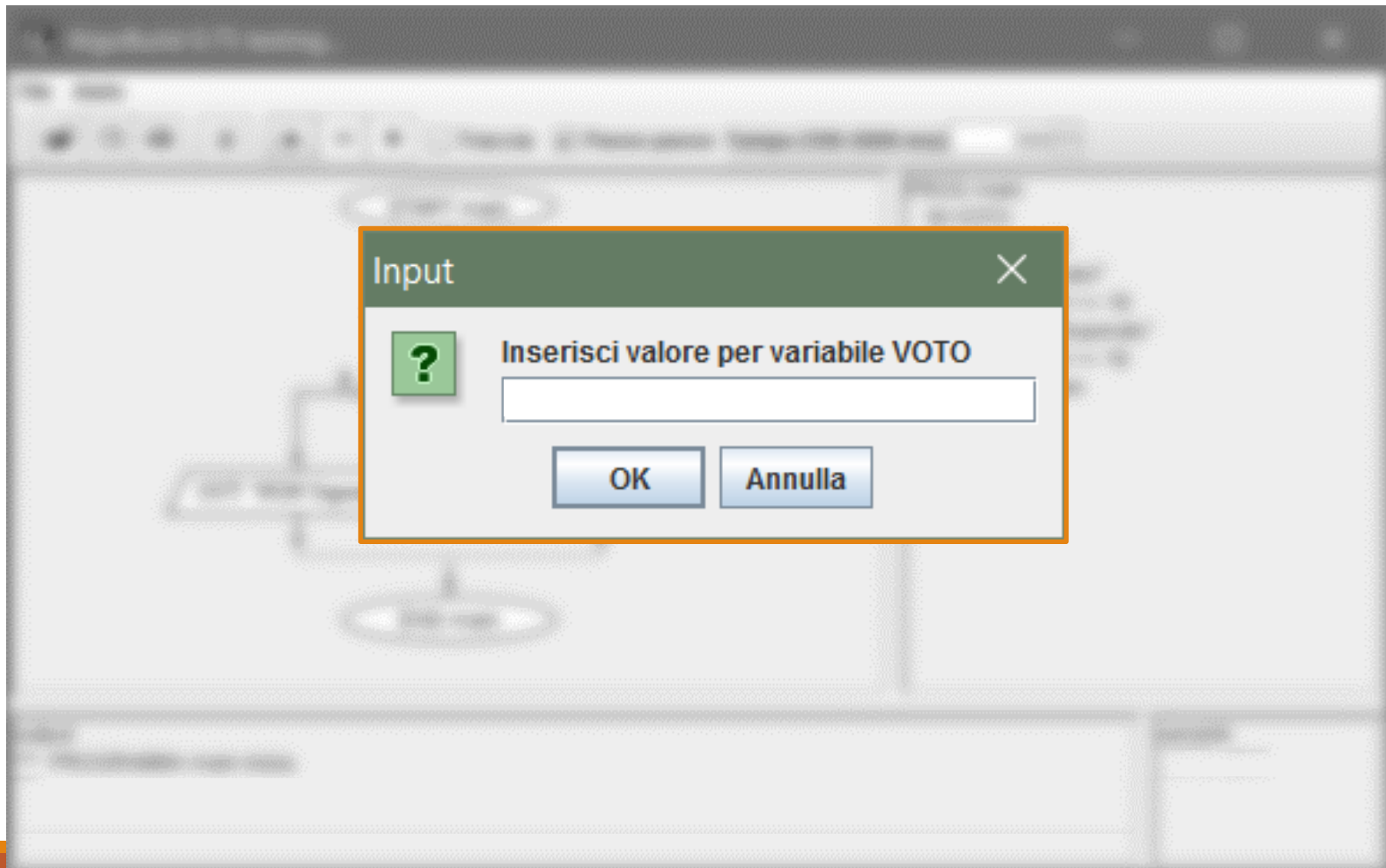
Struttura Selettiva IF – DEMO

The screenshot displays the AlgoBuild 0.75 testing environment. The main window contains a flowchart on the left and a code editor on the right. The flowchart starts with an oval labeled "START main", followed by a green parallelogram labeled "IN VOTO". A decision diamond contains the text "if VOTO >= 18". The "F" (False) path leads to a parallelogram labeled "OUT 'NON Superato'", and the "T" (True) path leads to a parallelogram labeled "OUT 'Superato!'". Both paths merge and lead to an oval labeled "END main". The code editor on the right contains the following code:

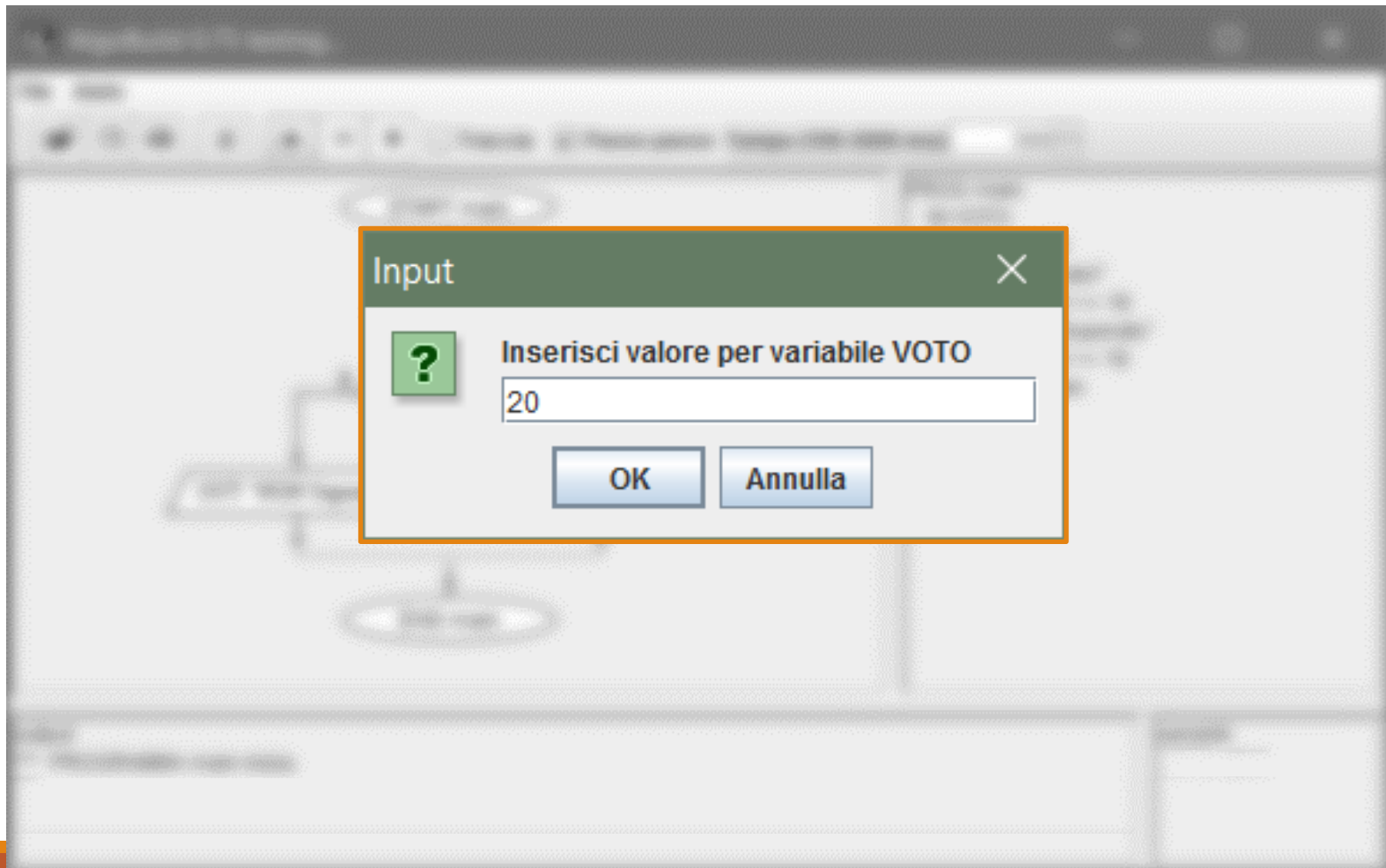
```
PROG main
IN VOTO
IF VOTO >= 18
  OUT "Superato!"
ELSE //if VOTO >= 18
  OUT "NON Superato"
END IF //VOTO >= 18
END PROG //main
```

The bottom of the window features two panels: "output" on the left, which shows the text "*** PROGRAMMA main inizia.", and "variabili" on the right, which is currently empty.

Struttura Selettiva IF – DEMO



Struttura Selettiva IF – DEMO



Struttura Selettiva IF – DEMO

The screenshot displays the AlgoBuild 0.75 testing environment. The main workspace is divided into three sections:

- Flowchart:** A flowchart starting with an oval labeled "START main". It leads to a parallelogram labeled "IN VOTO". This is followed by a green diamond decision box labeled "if VOTO >= 18". The "F" (False) path leads to a parallelogram labeled "OUT 'NON Superato'", and the "T" (True) path leads to a parallelogram labeled "OUT 'Superato!'". Both paths merge and lead to an oval labeled "END main".
- Code Editor:** Contains the following code:

```
PROG main
IN VOTO
IF VOTO >= 18
  OUT "Superato!"
ELSE //if VOTO >= 18
  OUT "NON Superato"
END IF //VOTO >= 18
END PROG //main
```
- Output and Variables:** The "output" window shows:*** PROGRAMMA main inizia.
20
The "variabili" window shows:VOTO=20.0

An orange arrow points from the text "Variabile VOTO presa in input dall'istruzione precedente" to the "variabili" window.

Variabile VOTO presa in input dall'istruzione precedente

Struttura Selettiva IF – DEMO

The screenshot displays the AlgoBuild 0.75 testing environment. The main window is titled "AlgoBuild 0.75 testing...". The interface includes a menu bar with "File" and "Aiuto", a toolbar with icons for file operations and execution, and a control panel with "Traccia" (checked), "Passo passo" (checked), and a "Tempo (100-5000 ms):" field set to 500.

The central workspace shows a flowchart for a program named "main". It starts with an oval "START main", followed by a parallelogram "IN VOTO". A decision diamond contains the condition "if VOTO >= 18". The "F" (False) path leads to a parallelogram "OUT 'NON Superato'", and the "T" (True) path leads to a green parallelogram "OUT 'Superato!'". Both paths merge and lead to an oval "END main". An orange annotation "Condizione vera" points to the "T" path of the decision diamond.

To the right of the flowchart, the code for the program is displayed:

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

At the bottom of the window, there are two panels. The "output" panel on the left shows the text: "output", "*** PROGRAMMA main inizia.", and "20". The "variabili" panel on the right shows "VOTO=20.0".

Struttura Selettiva IF – DEMO

The screenshot displays the AlgoBuild 0.75 testing environment. The interface includes a menu bar (File, Aiuto), a toolbar with icons for file operations and execution control, and a main workspace divided into three sections: a flowchart, a code editor, and a status/output area.

Flowchart: The flowchart starts with an oval labeled "START main". It proceeds to a parallelogram labeled "IN VOTO". A decision diamond follows, containing the text "if VOTO >= 18". The "F" (False) branch leads to a parallelogram labeled "OUT 'NON Superato'", and the "T" (True) branch leads to a parallelogram labeled "OUT 'Superato!'". Both branches merge and lead to a final oval labeled "END main".

```
graph TD; Start([START main]) --> InVoto[/IN VOTO/]; InVoto --> Decision{if VOTO >= 18}; Decision -- F --> OutNonSuperato[/OUT "NON Superato"/]; Decision -- T --> OutSuperato[/OUT "Superato!"/]; OutNonSuperato --> EndMain([END main]); OutSuperato --> EndMain;
```

Code Editor: The code editor shows the following code:

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

Output and Variables: The bottom-left section shows the output of the program execution:

```
output
*** PROGRAMMA main inizia.
20
Superato!
```

The bottom-right section shows the current state of variables:

```
variabili
VOTO=20.0
```

Struttura Selettiva IF – DEMO

The screenshot displays the AlgoBuild 0.75 testing environment. The window title is "AlgoBuild 0.75 testing...". The menu bar includes "File" and "Aiuto". The toolbar contains icons for file operations (open, save, print), a search icon, and execution controls (run, pause, stop). A checkbox for "Traccia" is unchecked, and "Passo passo" is checked. The "Tempo (100-5000 ms)" is set to 500.

The central workspace shows a flowchart for a program named "main". It starts with an oval "START main", followed by a parallelogram "IN VOTO". A diamond decision node asks "if VOTO >= 18". The "F" (False) path leads to a parallelogram "OUT 'NON Superato'", and the "T" (True) path leads to a parallelogram "OUT 'Superato!'". Both paths merge and lead to an oval "END main".

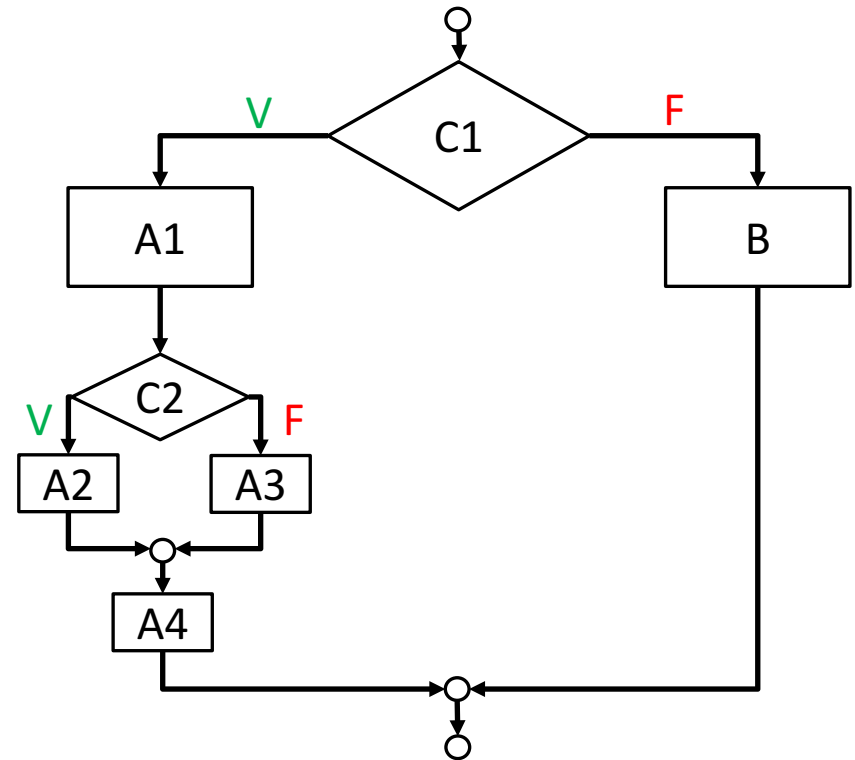
The right-hand pane shows the corresponding code:

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

Cosa sarebbe accaduto se avessi usato VOTO = 16?

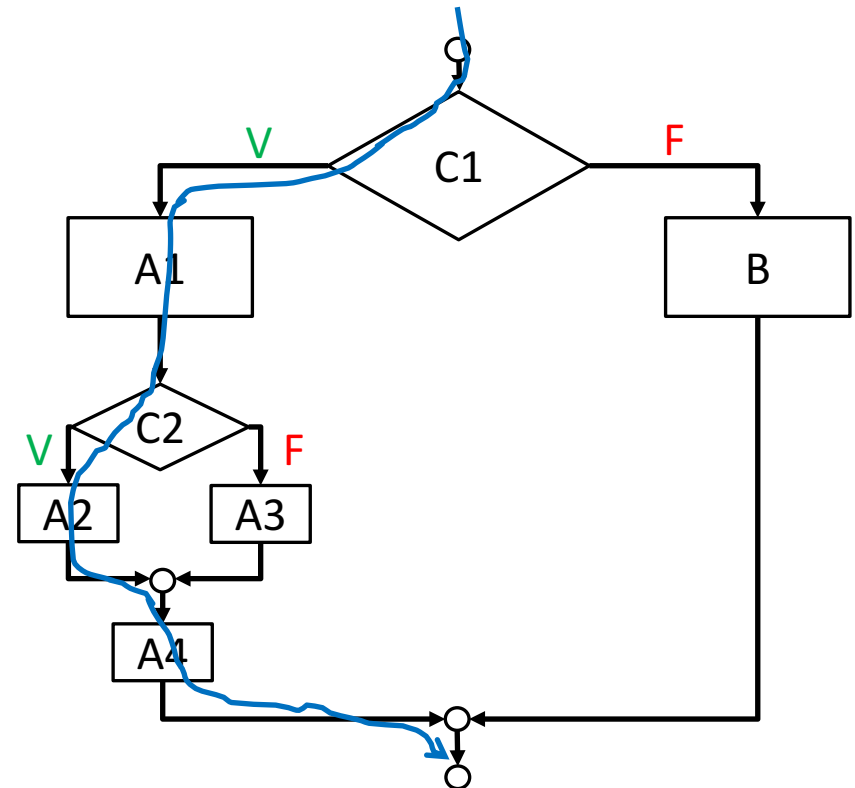
Struttura Selettiva IF (Annidata)

- 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



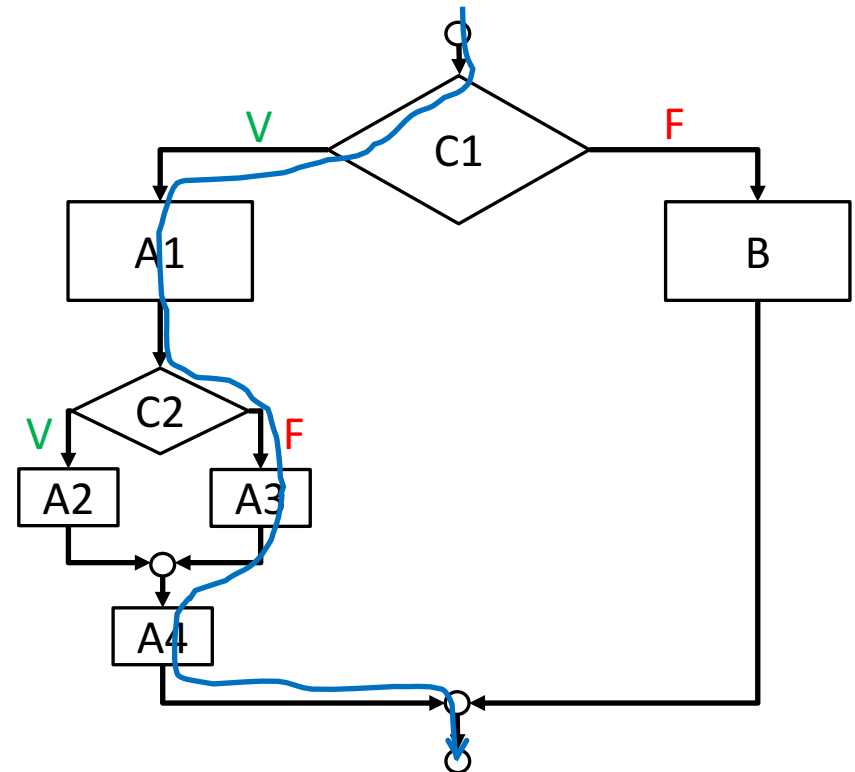
Struttura Selettiva IF (Annidata)

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



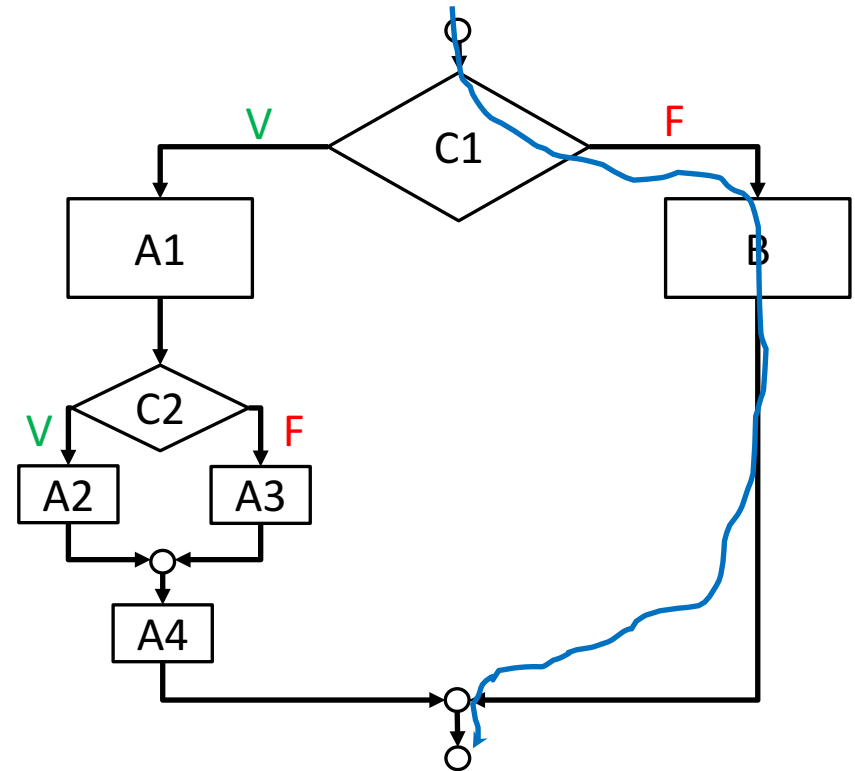
Struttura Selettiva IF (Annidata)

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



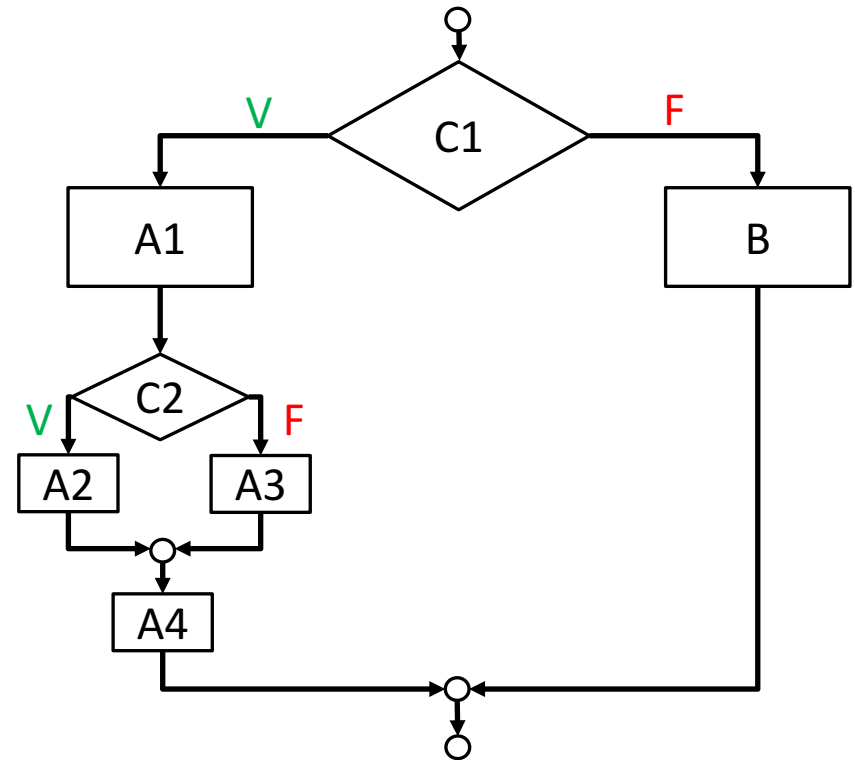
Struttura Selettiva IF (Annidata)

- C1 **Falso**
 - Istruzioni eseguite: B



Struttura Selettiva IF (Annidata)

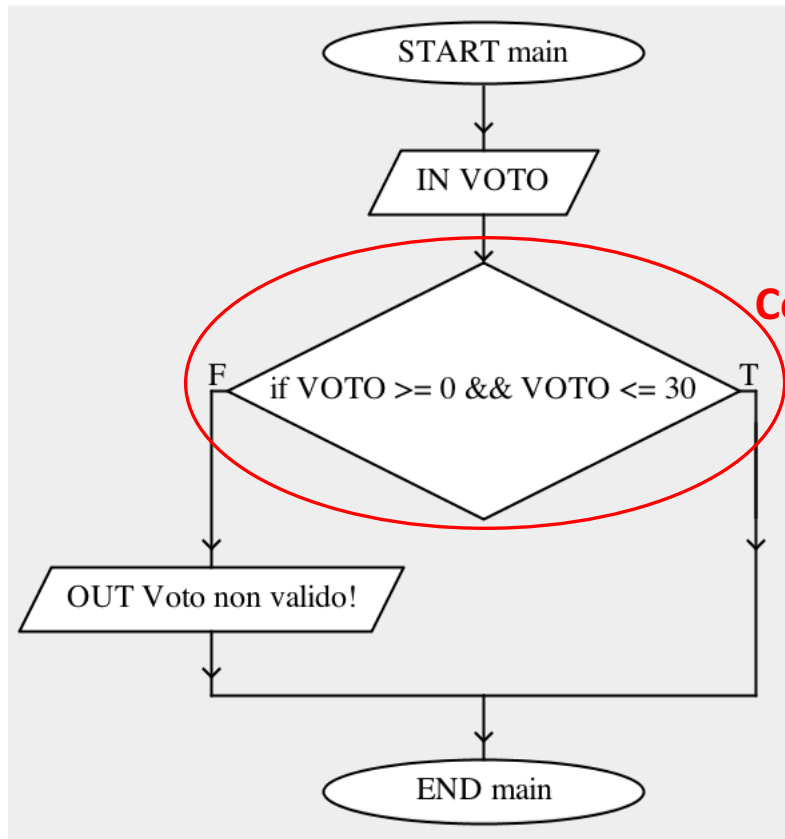
- L'intero blocco di scelta più interno (dalla condizione fino al ricongiungimento) deve essere completamente contenuto all'interno di uno dei rami del blocco più esterno



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

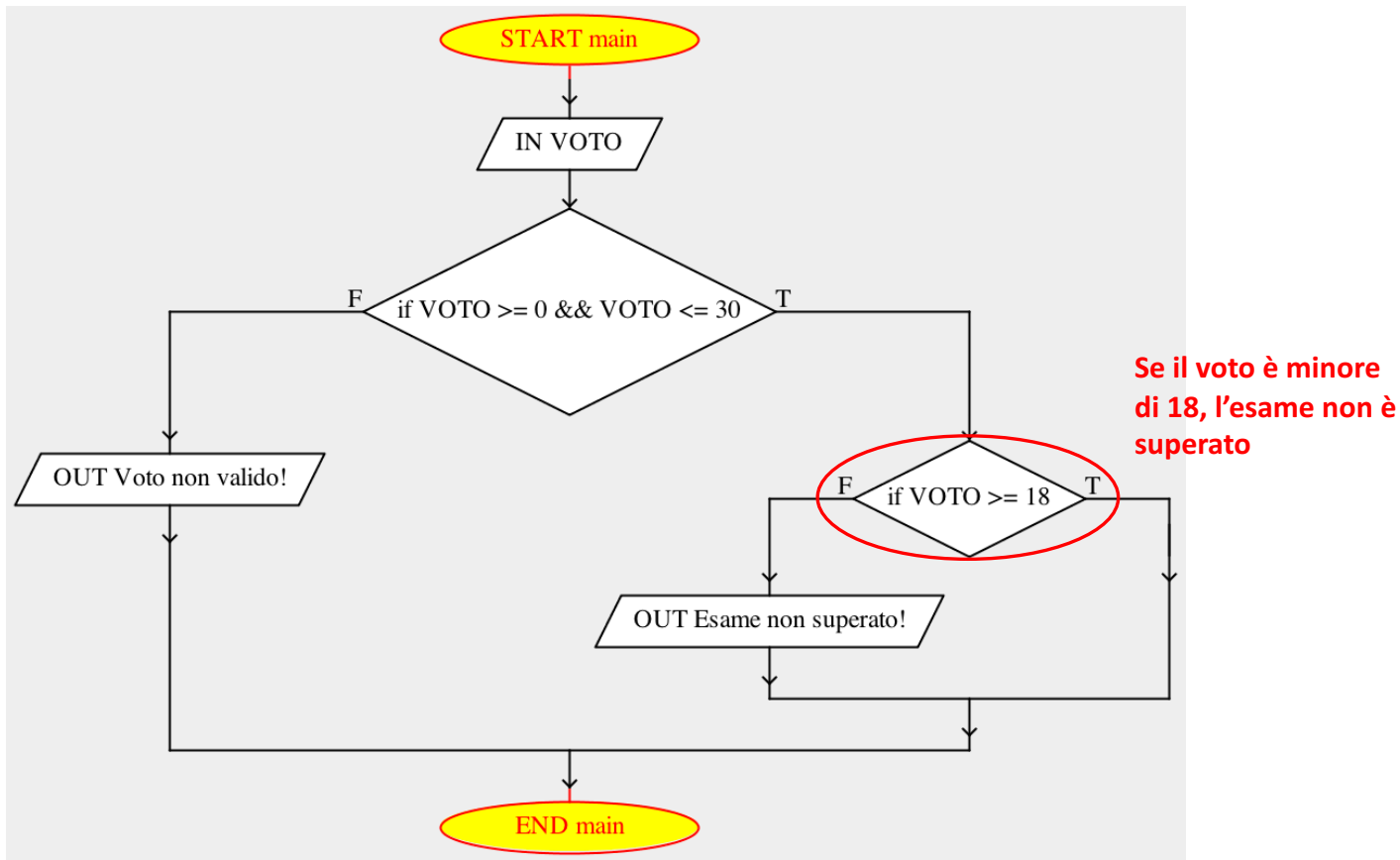
Struttura Selettiva IF (Annidata): Esempio 1



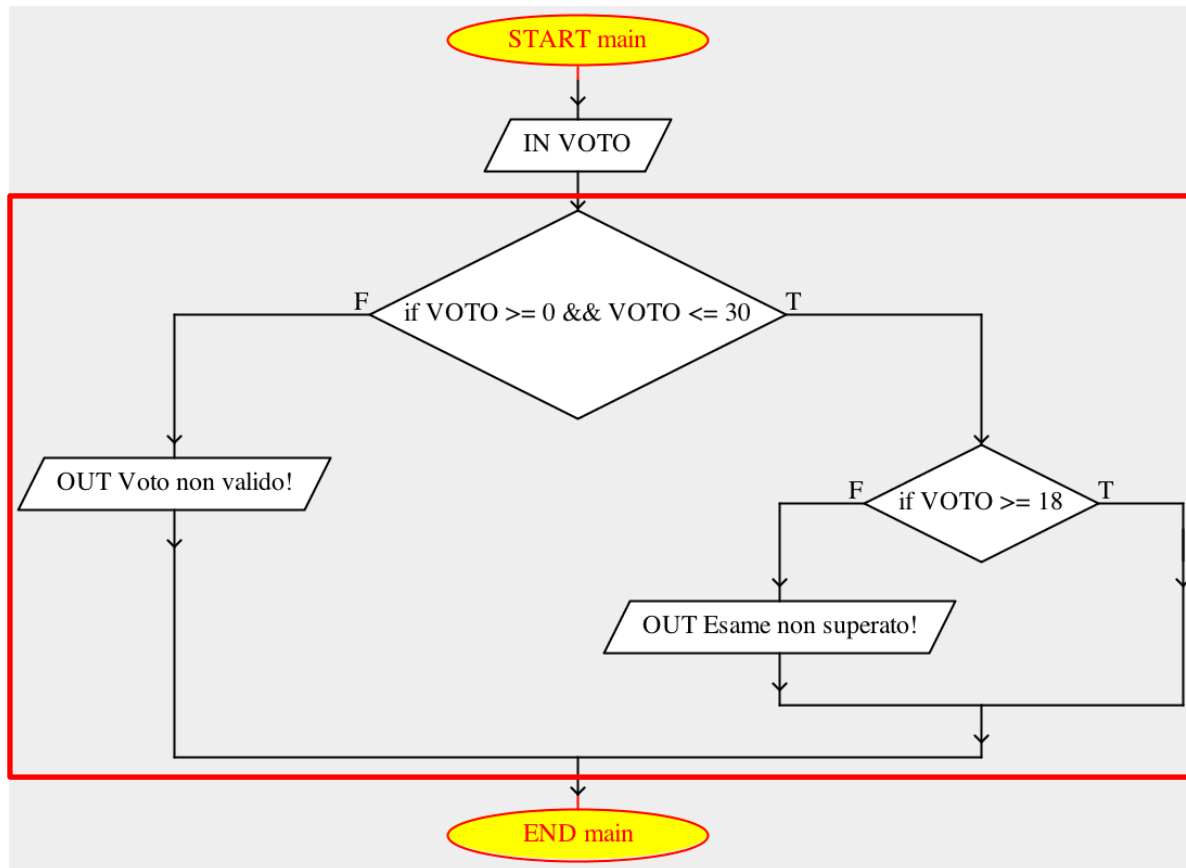
Controllo che il voto sia valido

```
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

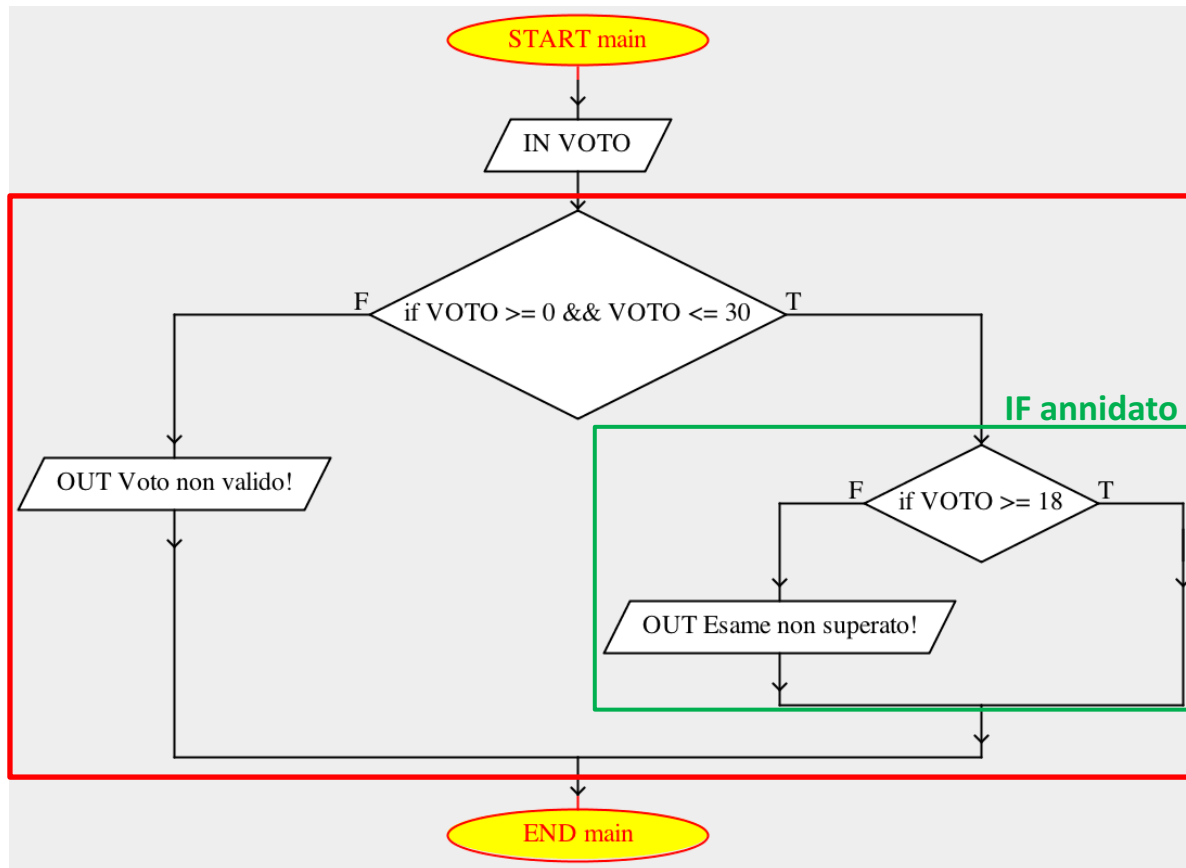


Struttura Selettiva IF (Annidata): Esempio 1

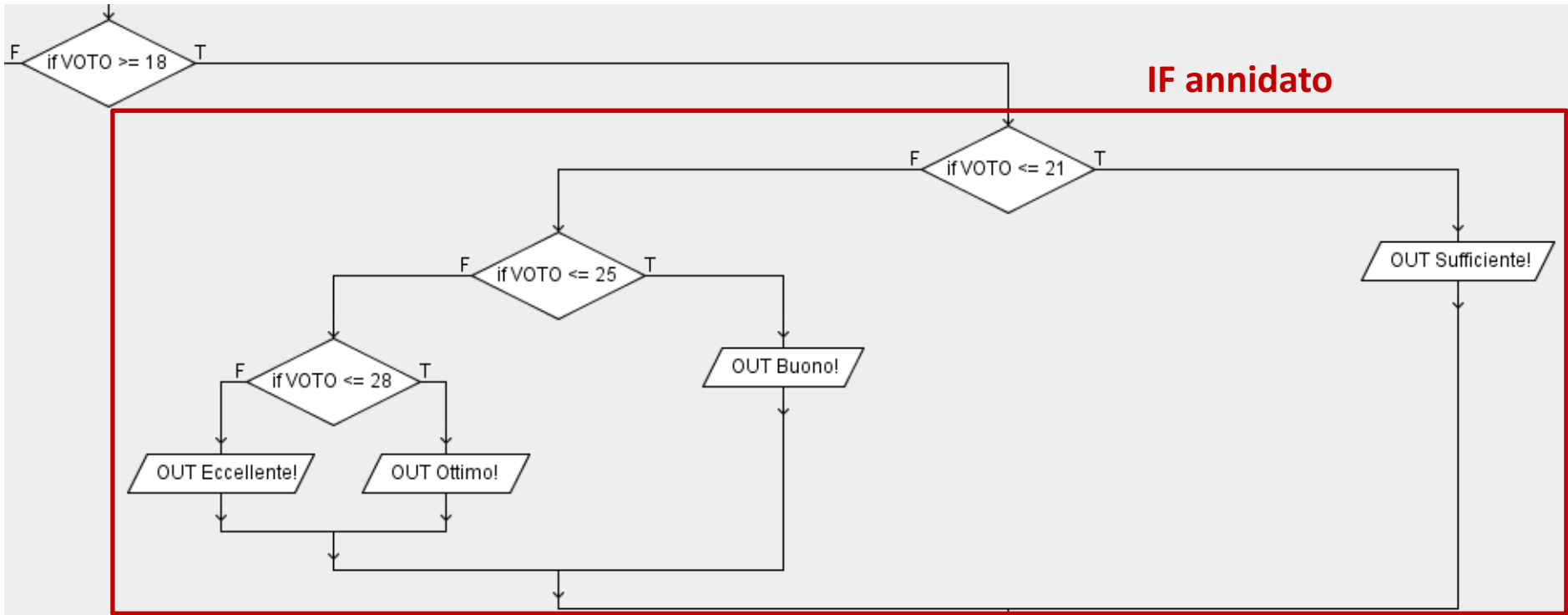


```
PROG main
IN VOTO
IF VOTO >= 0 && VOTO <= 30
  IF VOTO >= 18
    OUT Esame non superato!
  END IF //VOTO >= 18
ELSE //if VOTO >= 0 && VOTO <= 30
  OUT Voto non valido!
END IF //VOTO >= 0 && VOTO <= 30
END PROG //main
```

Struttura Selettiva IF (Annidata): Esempio 1



```
PROG main
IN VOTO
IF VOTO >= 0 && VOTO <= 30
  IF VOTO >= 18
    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!

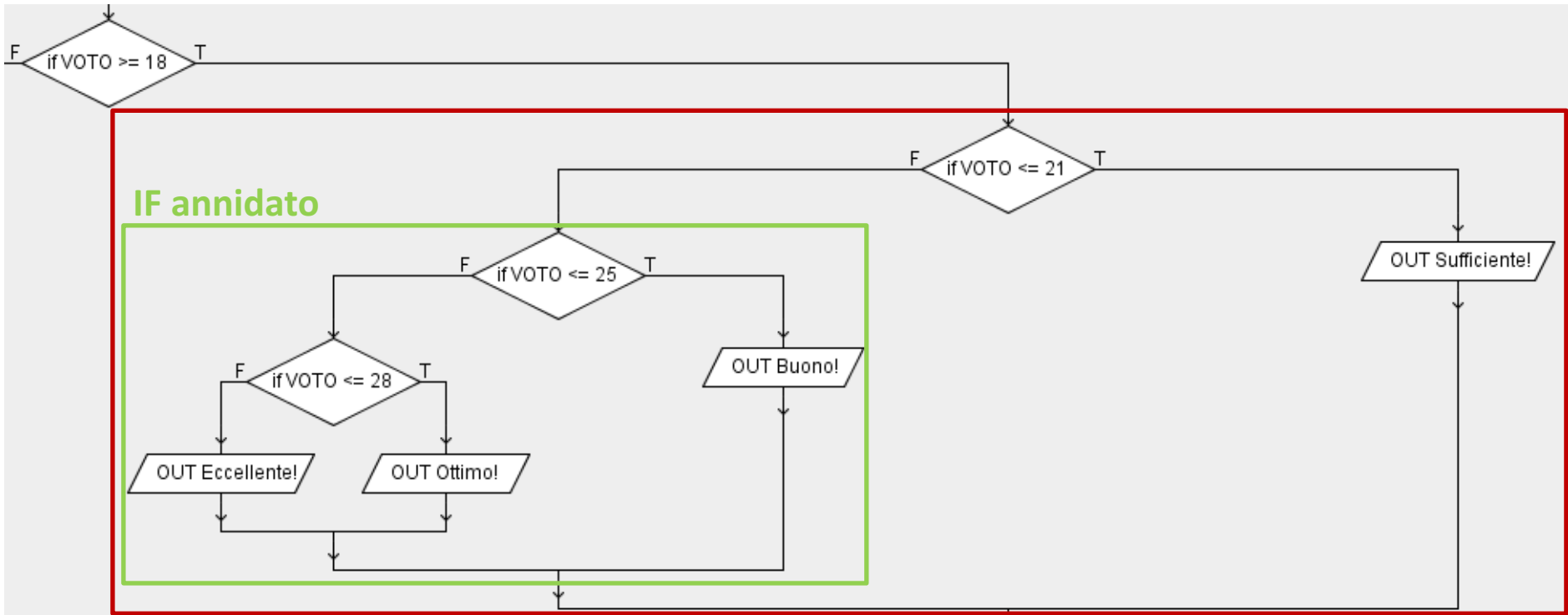
END IF //VOTO >= 18

ELSE //if VOTO >= 0 && VOTO <= 30

OUT Voto non valido!

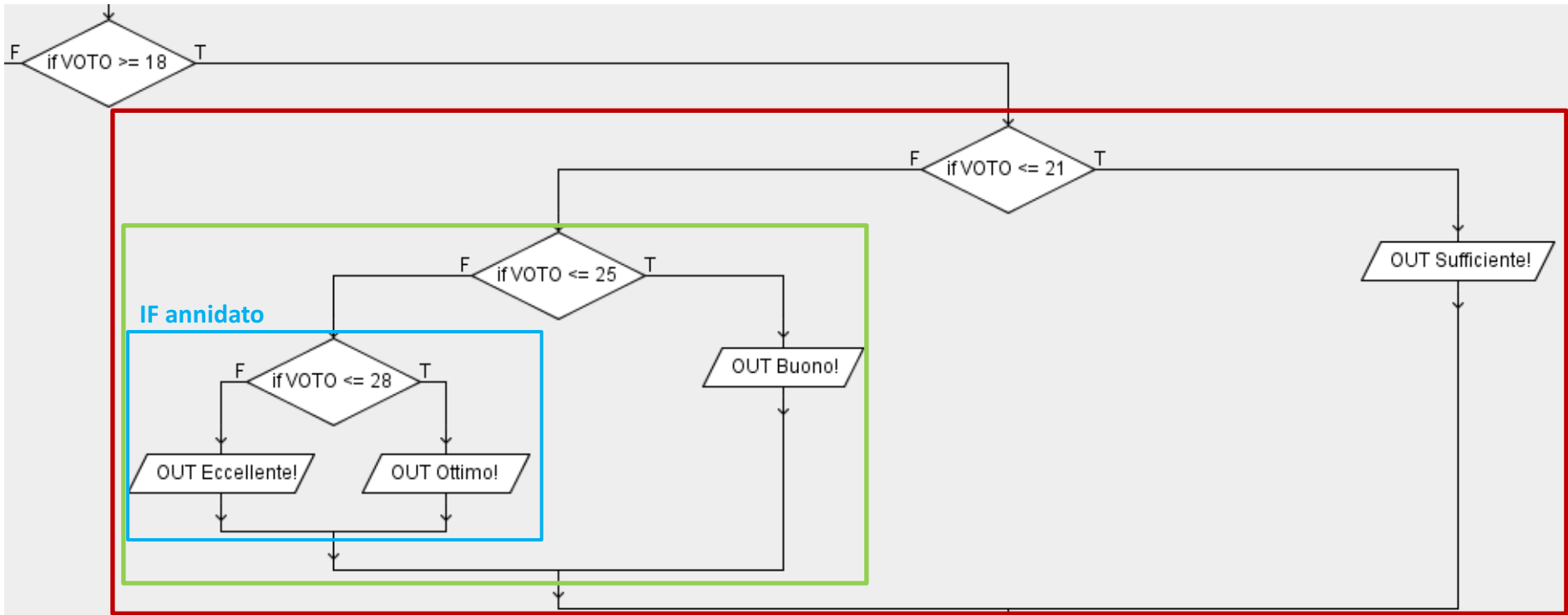
END IF //VOTO >= 0 && VOTO <= 30

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
  
```

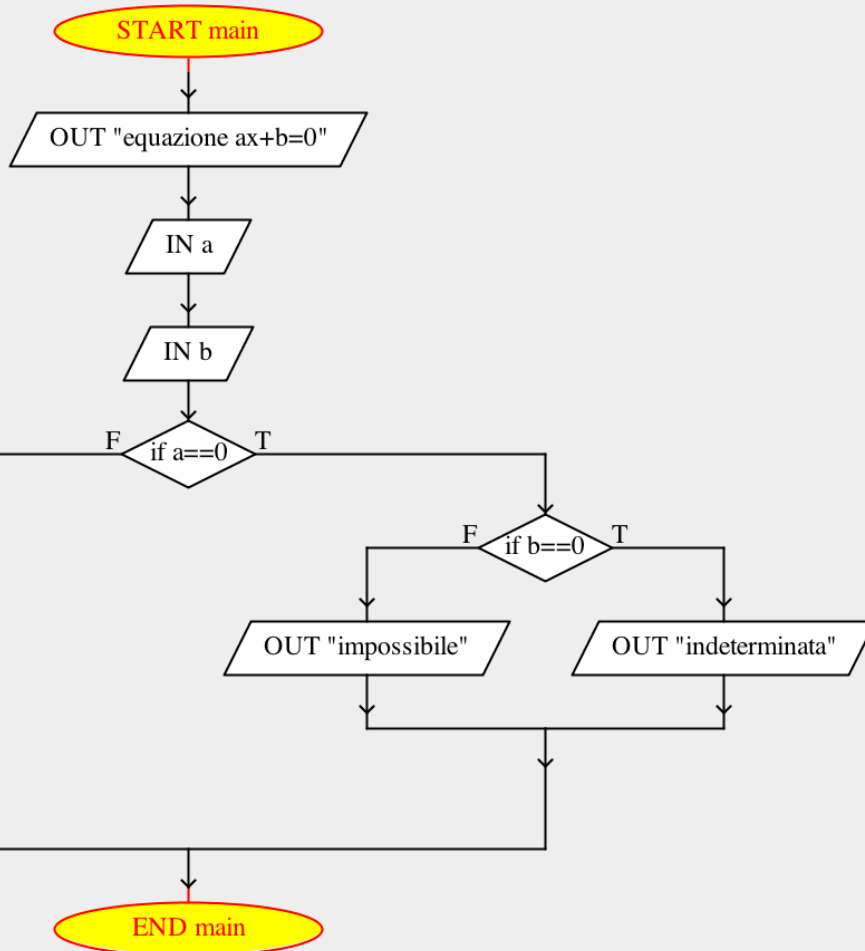
IF annidato

```

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
  
```

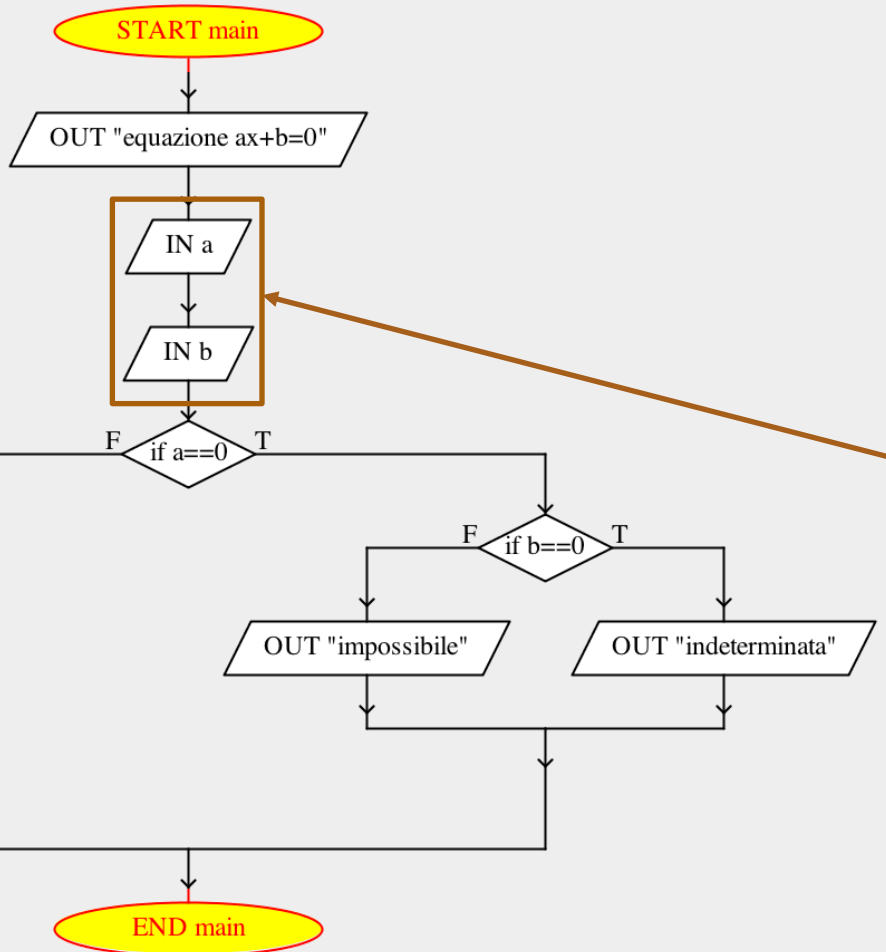
Esempio: Risoluzione di equazioni di primo grado

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



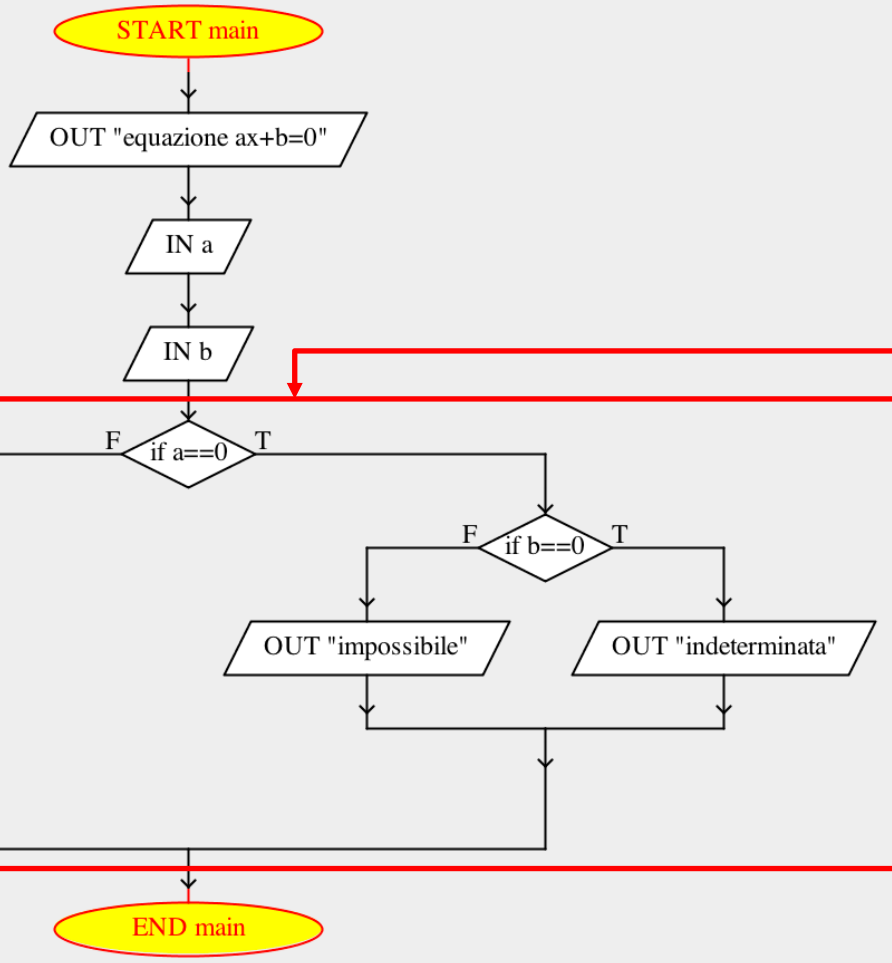
```

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
  
```



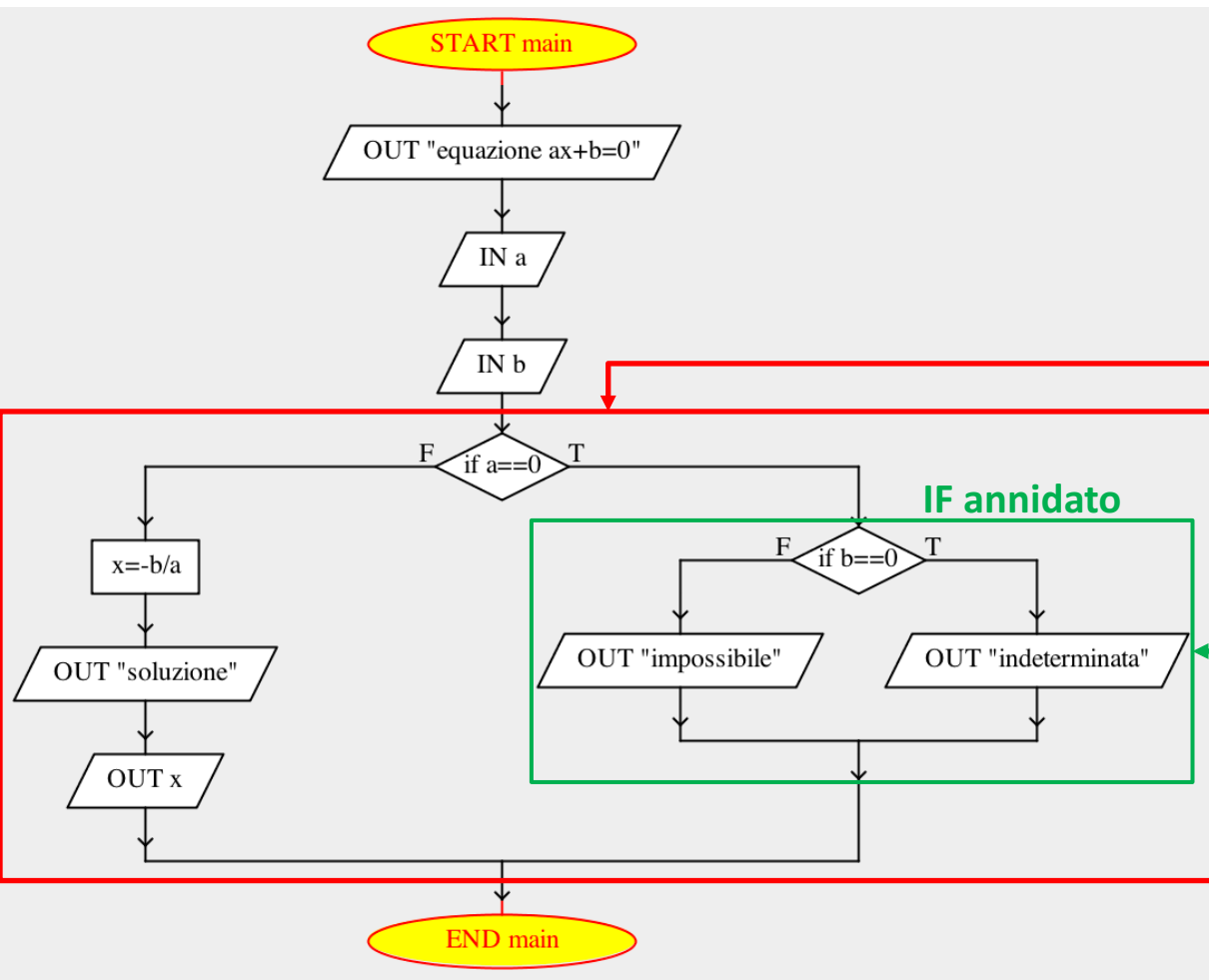
```

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
  
```



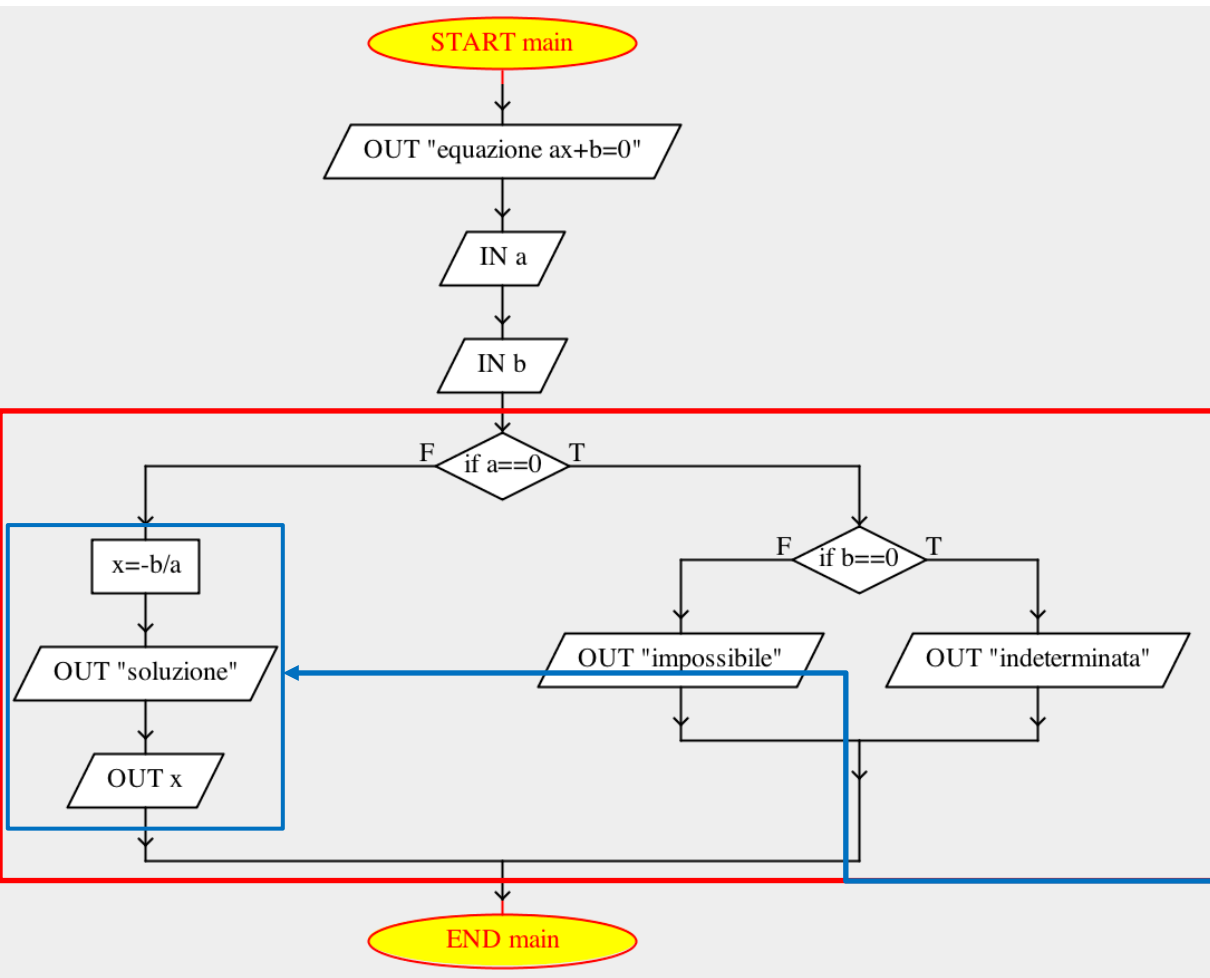
```

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
  
```



```

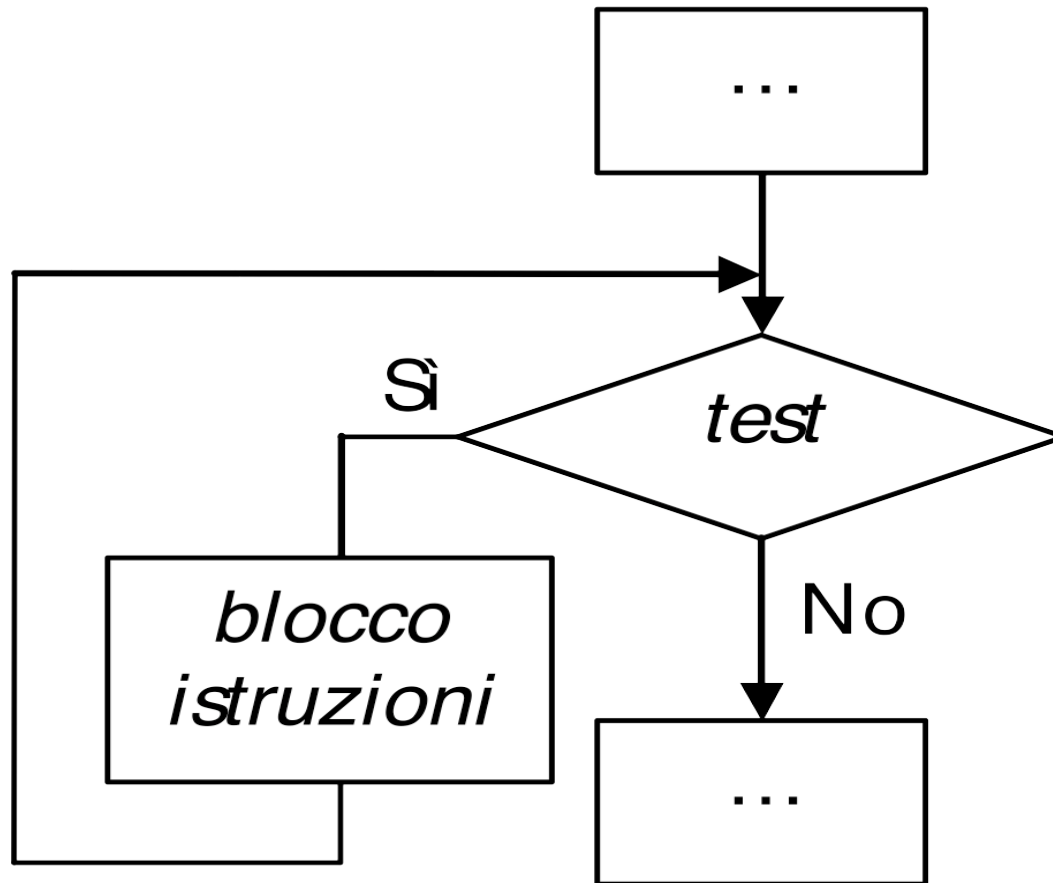
PROG main
  OUT "equazione ax+b=0"
  IN a
  IN b
  IF a==0
    IF annidato
      IF b==0
        OUT "indeterminata"
      ELSE //if b==0
        OUT "impossibile"
      END IF //b==0
    ELSE //if a==0
      ASSIGN x=-b/a
      OUT "soluzione"
      OUT x
    END IF //a==0
  END IF //main
END PROG //main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



```

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
 - **R = X^Y**
 - **N.B.** Il calcolo di **X^Y** deve essere eseguito con il metodo moltiplicativo
 - $\underbrace{X \times X \times \dots \times X}_{Y \text{ volte}}$

Struttura Iterativa FOR – Esempio 1

The screenshot displays the AlgoBuild 0.75 testing environment. The window title is "AlgoBuild 0.75 testing...". The interface includes a menu bar with "File" and "Aiuto", a toolbar with icons for file operations and execution, and a control panel with checkboxes for "Traccia" and "Passo passo", and a "Tempo (100-5000 ms):" field set to 500.

The main workspace contains a flowchart with the following steps:

```
graph TD; Start([START main]) --> InX[/IN X/]; InX --> InY[/IN Y/]; InY --> End([END main]);
```

The code editor on the right shows the following code:

```
PROG main  
IN X  
IN Y  
END PROG //main
```

The bottom section of the interface is divided into two panels: "output" on the left and "variabili" on the right.

Struttura Iterativa FOR – Esempio 1

AlgoBuild 0.75 testing...

File Aiuto

Traccia Passo passo Tempo (100-5000 ms): 500

START main

IN X

IN Y

END

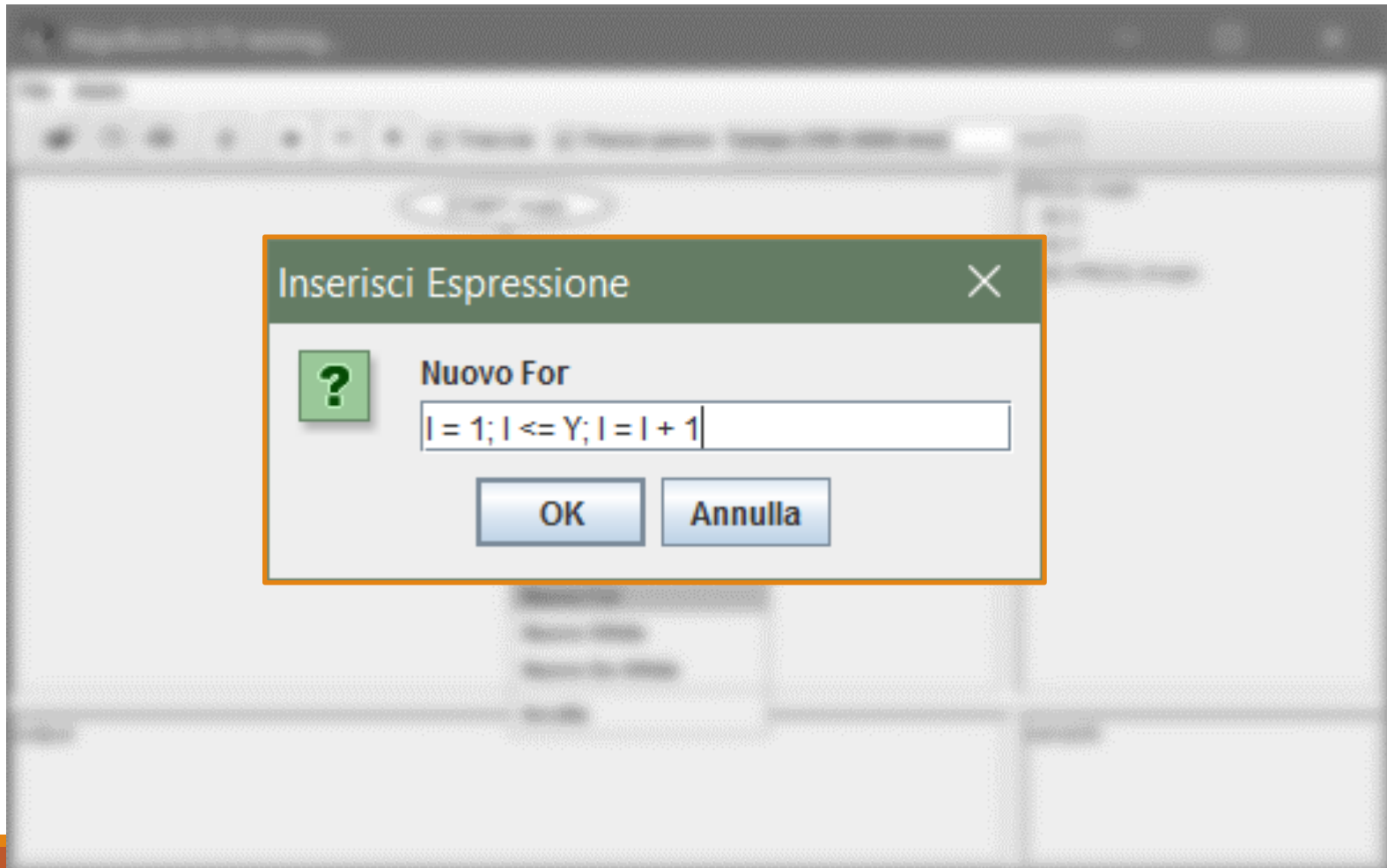
- Nuovo Assegnamento
- Nuovo Input
- Nuovo Output
- Nuovo If
- Nuovo For**
- Nuovo While
- Nuovo Do-While
- Incolla

PROG main
IN X
IN Y
END PROG //main

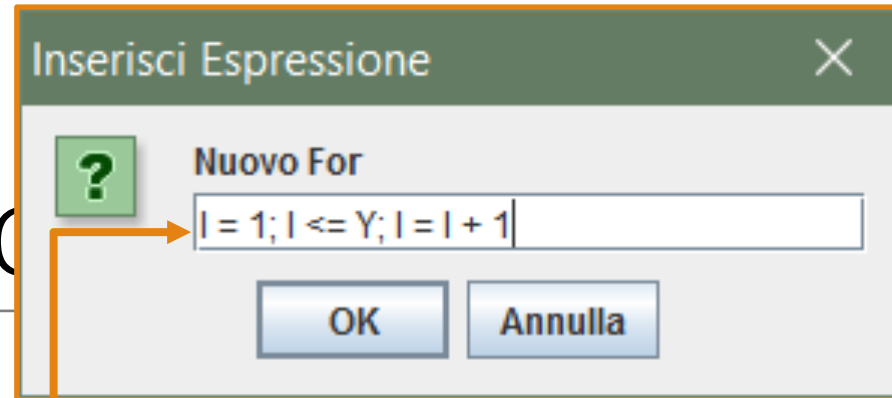
output

variabili

Struttura Iterativa FOR – Esempio 1

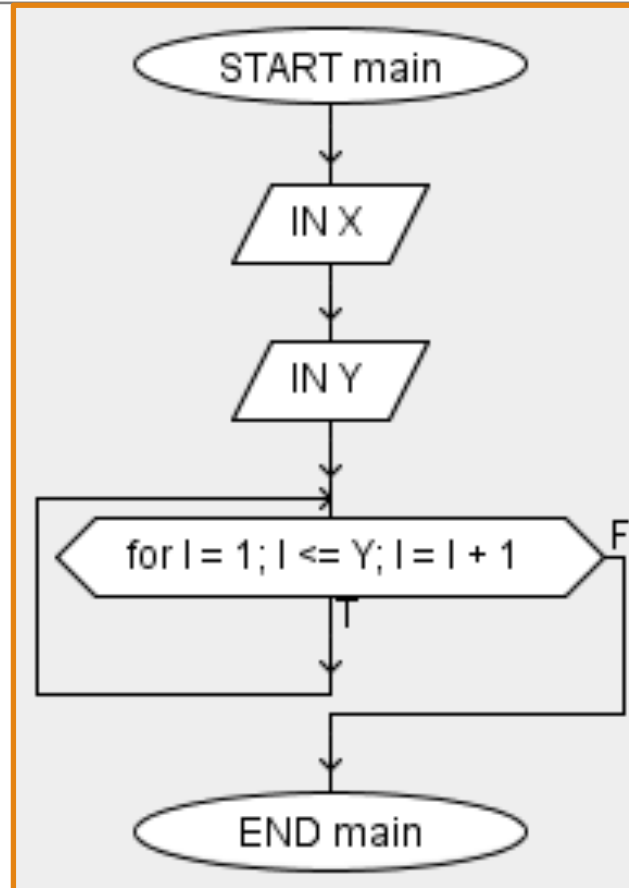


Struttura Iterativa For

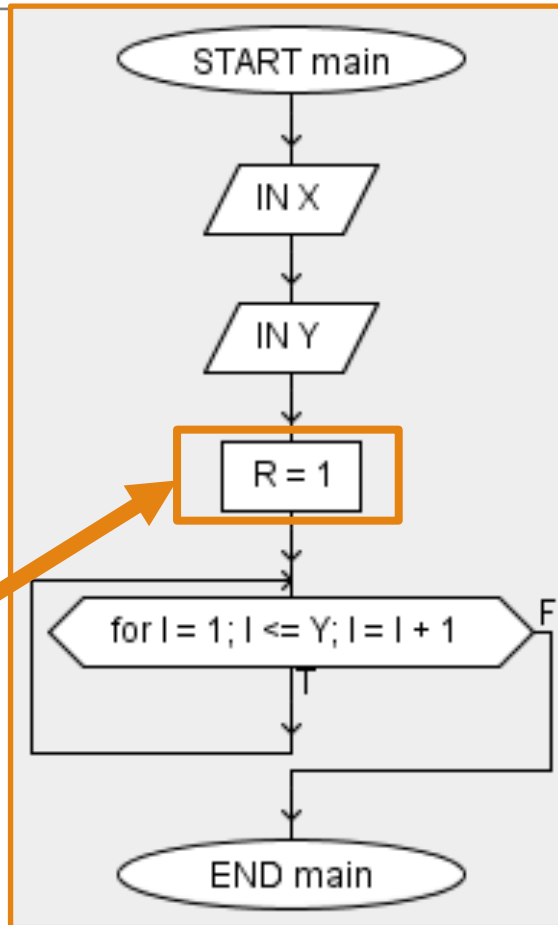


- I = 1; I <= Y; I = I + 1
 - I = 1
 - *Inizializzazione* del ciclo FOR (alla prima iterazione I = 1)
 - I <= Y
 - *Condizione di uscita* dal ciclo FOR (appena I sarà maggiore di Y, il ciclo FOR si concluderà)
 - I = I + 1
 - *Incremento*: ad ogni iterazione la variabile I verrà incrementata automaticamente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

Struttura Iterativa FOR – Esempio 1

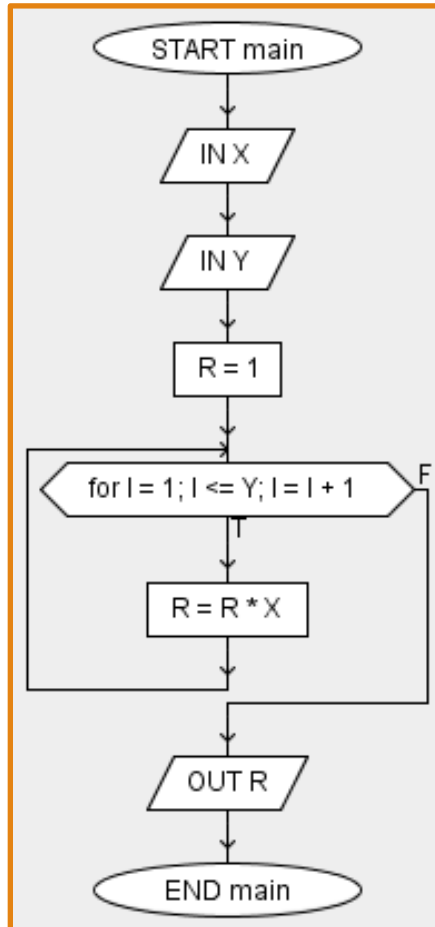


Struttura Iterativa FOR – Esempio 1



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

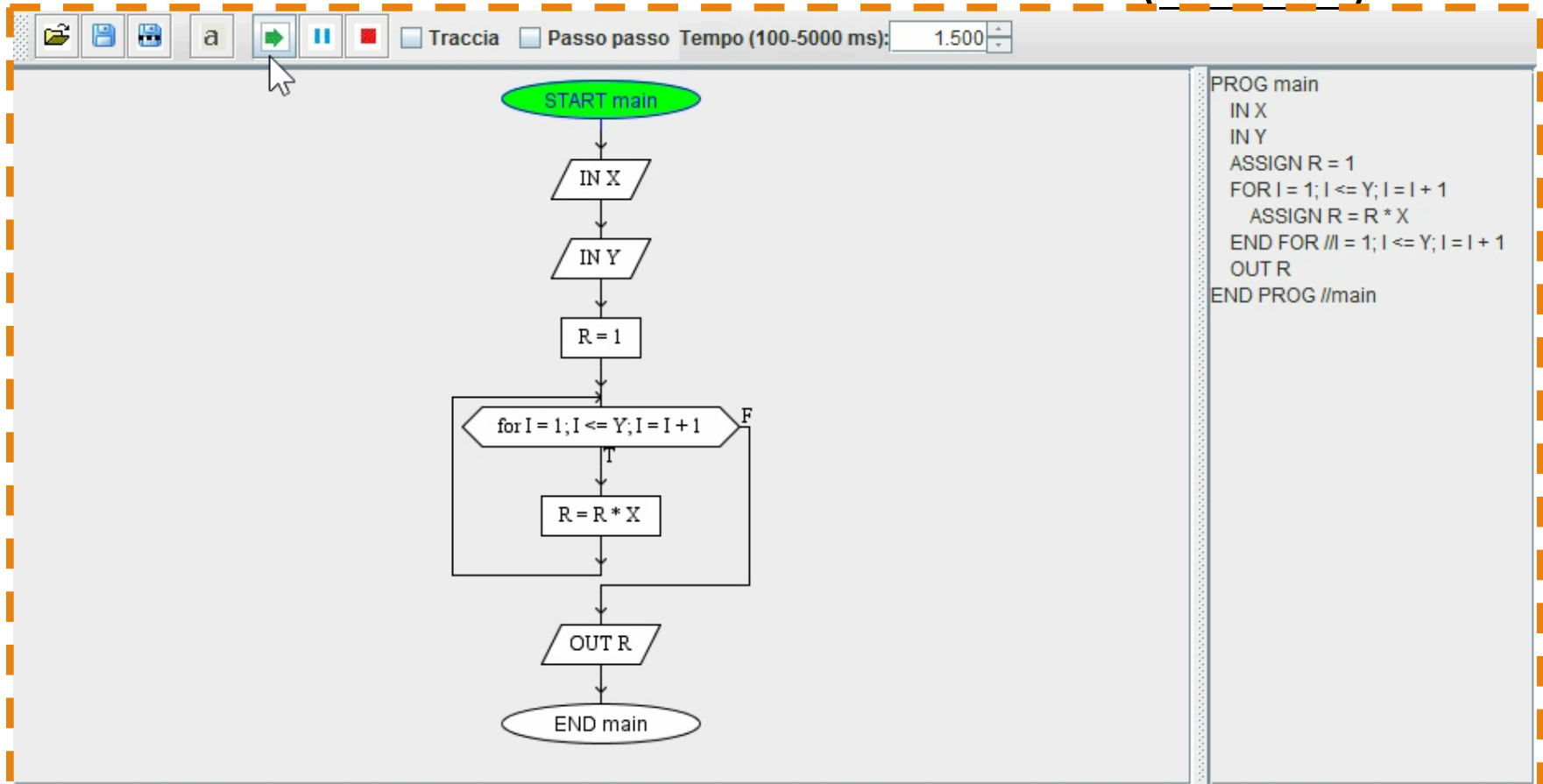
Struttura Iterativa FOR – Esempio 1



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)



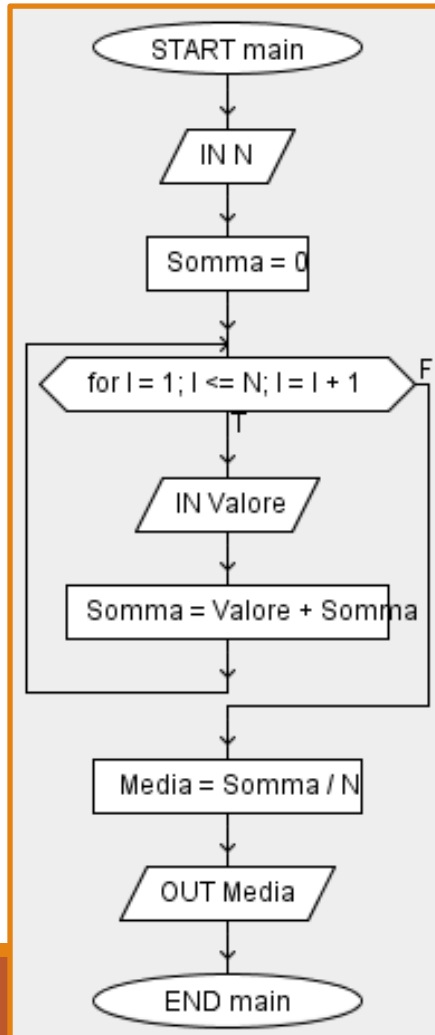
output
*** PROGRAMMA main inizia.

variabili

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

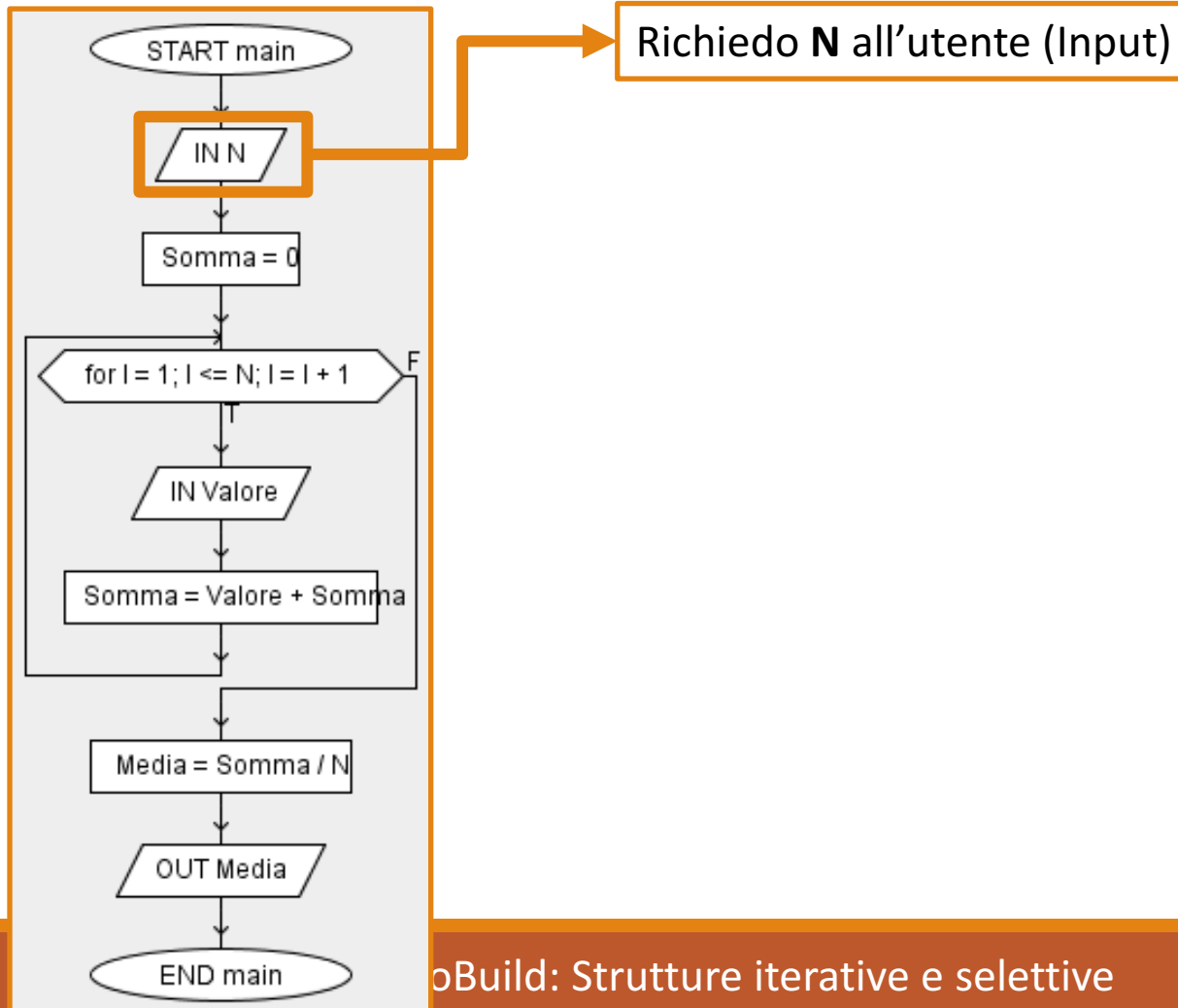
END FOR //I = 1; I <= N; I = I + 1

ASSIGN Media = Somma / N

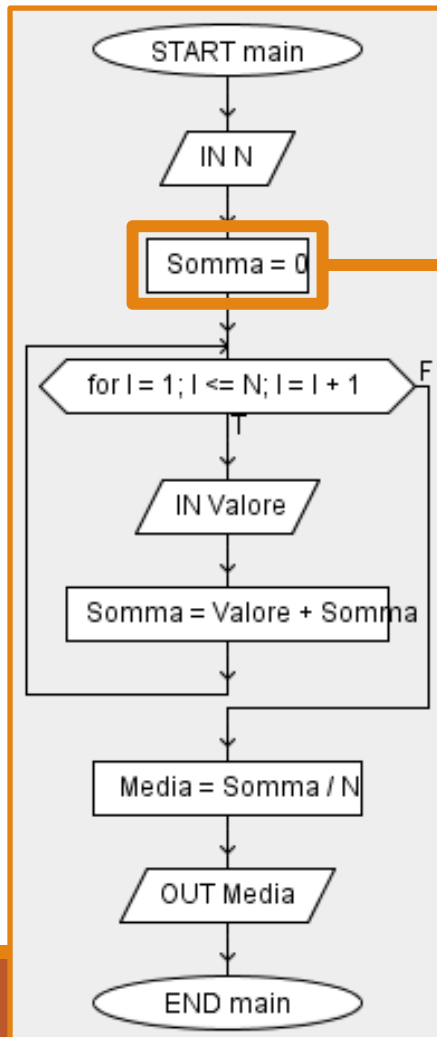
OUT Media

END PROG //main

Struttura Iterativa FOR – Esempio 2

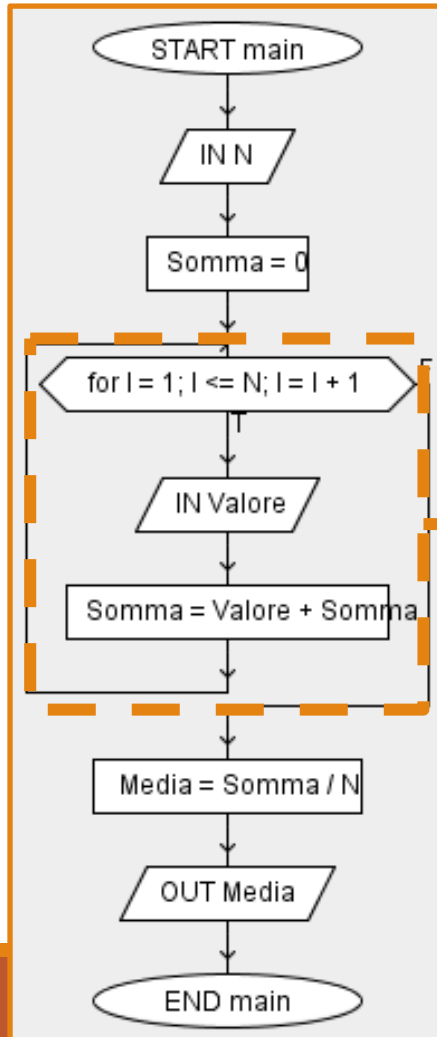


Struttura Iterativa FOR – Esempio 2



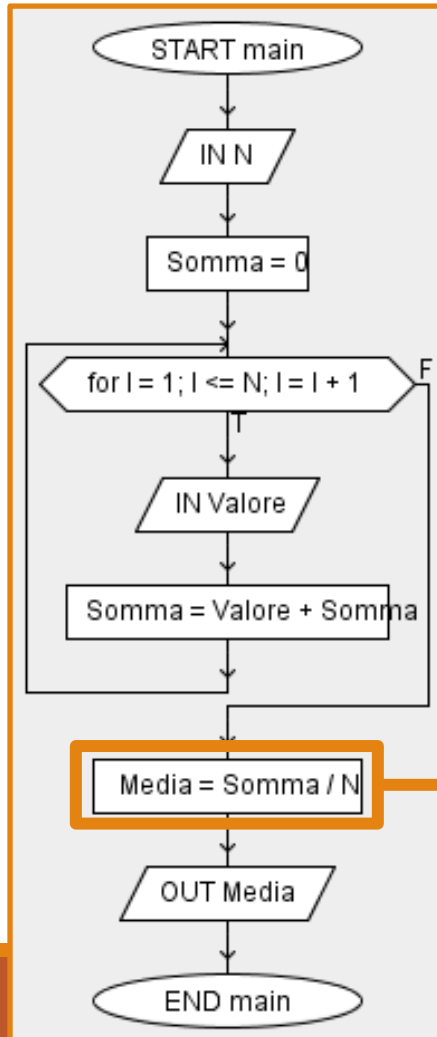
- Imposto **Somma** a zero
 - Ciò mi servirà per tener traccia della somma degli N numeri

Struttura Iterativa FOR – Esempio 2



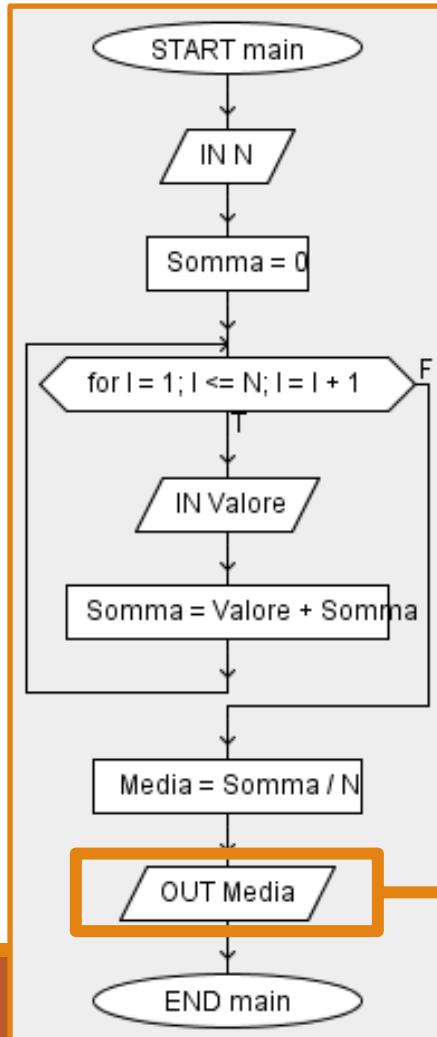
- Ciclo **for** che va da 1 a **N**
 - Ad ogni iterazione
 - Viene richiesto in input un numero (**IN Valore**)
 - Viene aggiornata la variabile **Somma** che verrà utilizzata successivamente per il calcolo della media aritmetica
 - **Somma = Somma + Valore**

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

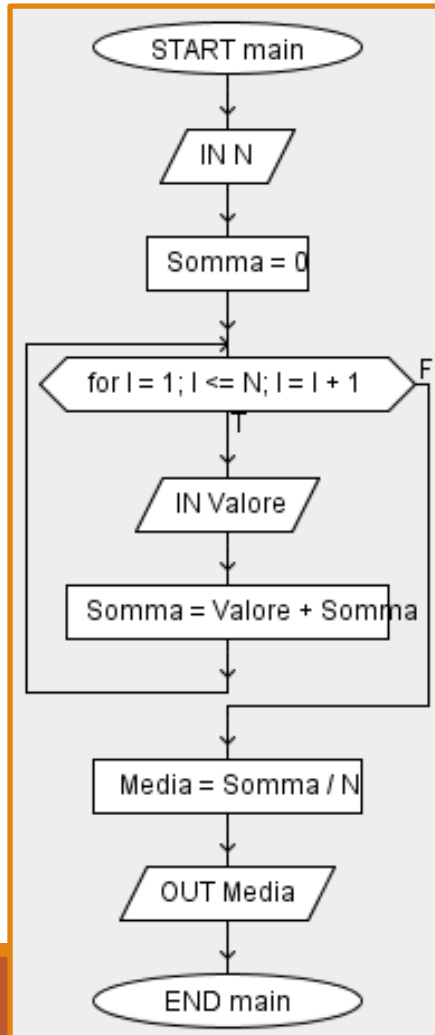
Struttura Iterativa FOR – Demo (Video)

The screenshot displays a flowchart editor interface. At the top, there is a toolbar with icons for file operations and execution control. A status bar shows 'Traccia' and 'Passo passo' checked, and a 'Tempo (100-5000 ms):' field set to '1500'. The main workspace contains a flowchart for a program named 'main'. The flowchart starts with an oval 'START main', followed by a parallelogram 'IN N', a rectangle 'Somma = 0', and a diamond-shaped loop header 'for I = 1; I <= N; I = I + 1'. The loop body consists of a parallelogram 'IN Valore' and a rectangle 'Somma = Valore + Somma'. The loop exits to a rectangle 'Media = Somma / N', a parallelogram 'OUT Media', and finally an oval 'END main'. On the right side, a code editor shows the corresponding pseudocode: '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', and 'END PROG //main'. At the bottom, there are two empty panels labeled 'output' and 'variabili'.

```
graph TD; Start([START main]) --> InN[/IN N/]; InN --> Somma0[Somma = 0]; Somma0 --> LoopHeader{for I = 1; I <= N; I = I + 1}; LoopHeader -- T --> InValore[/IN Valore/]; InValore --> SommaAdd[Somma = Valore + Somma]; SommaAdd --> LoopHeader; LoopHeader -- F --> MediaCalc[Media = Somma / N]; MediaCalc --> OutMedia[/OUT Media/]; OutMedia --> EndMain([END main]);
```

```
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



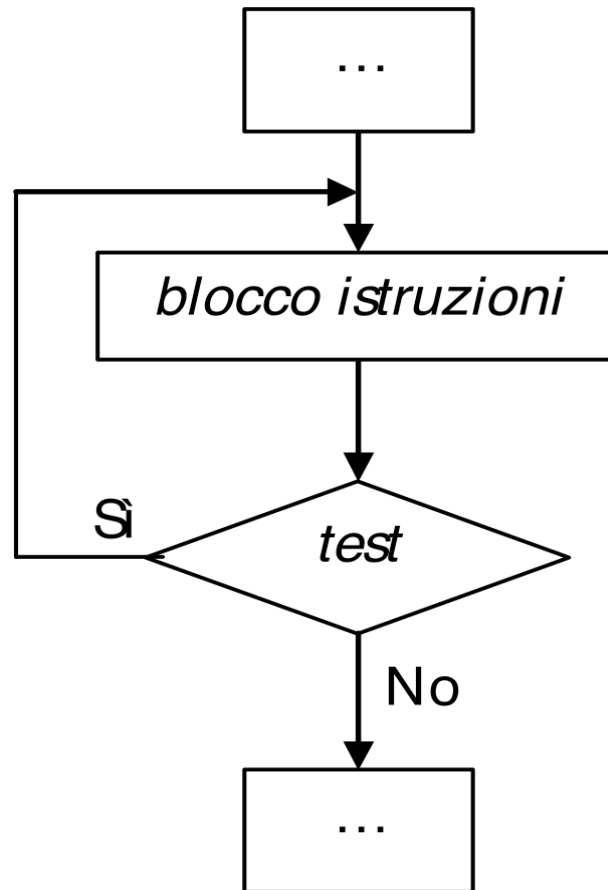
OSSERVAZIONE

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

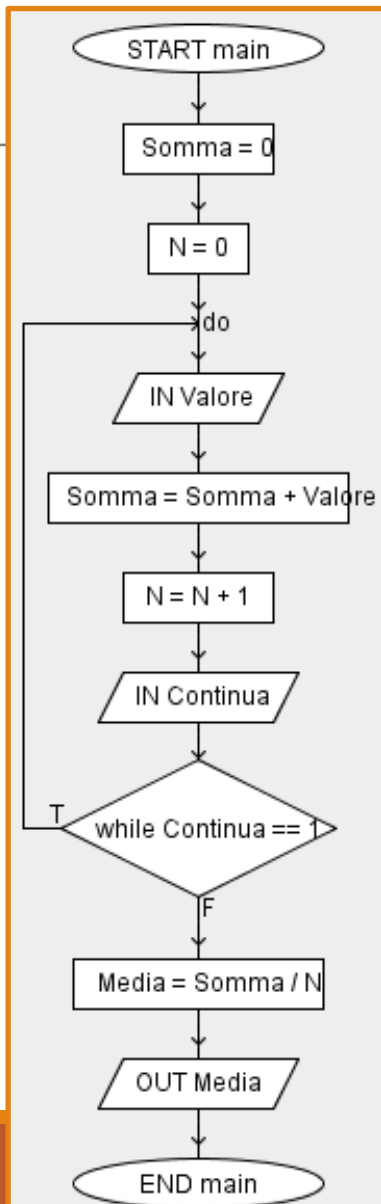


Struttura Iterativa DO/WHILE – Esempio

- 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

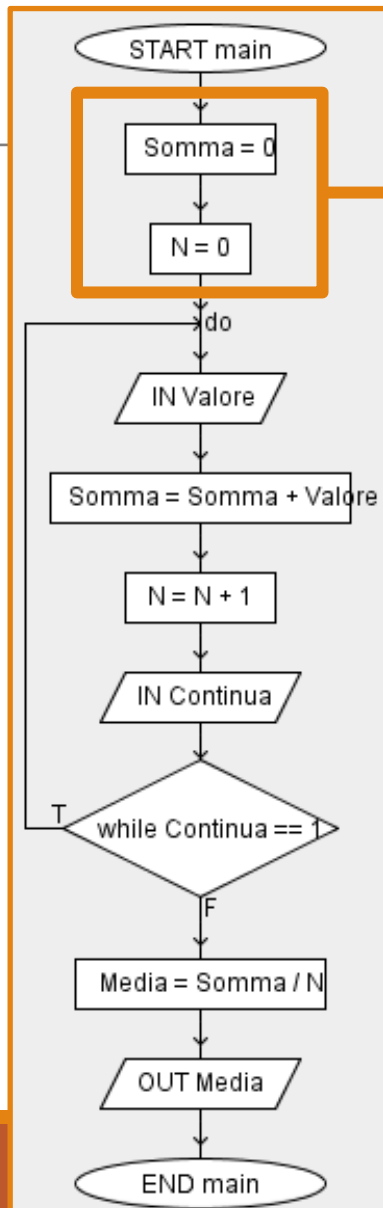
END DO_WHILE Continua == 1

ASSIGN Media = Somma / N

OUT Media

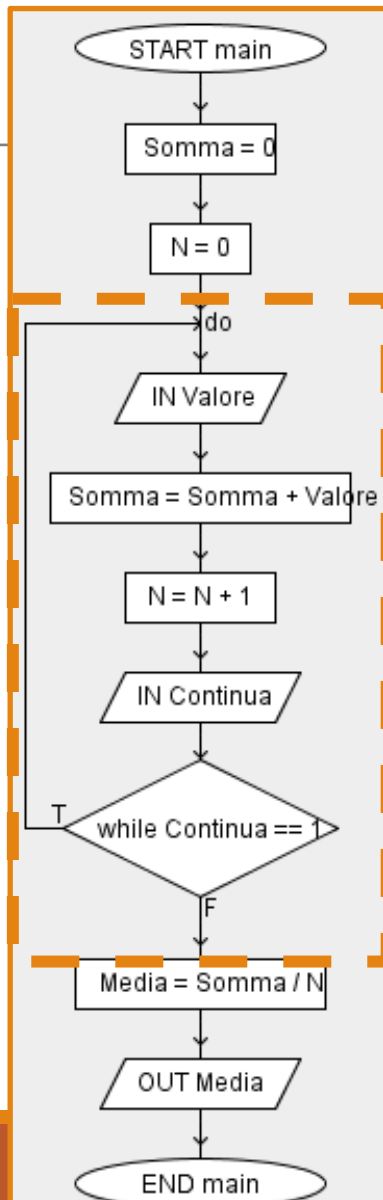
END PROG //main

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)

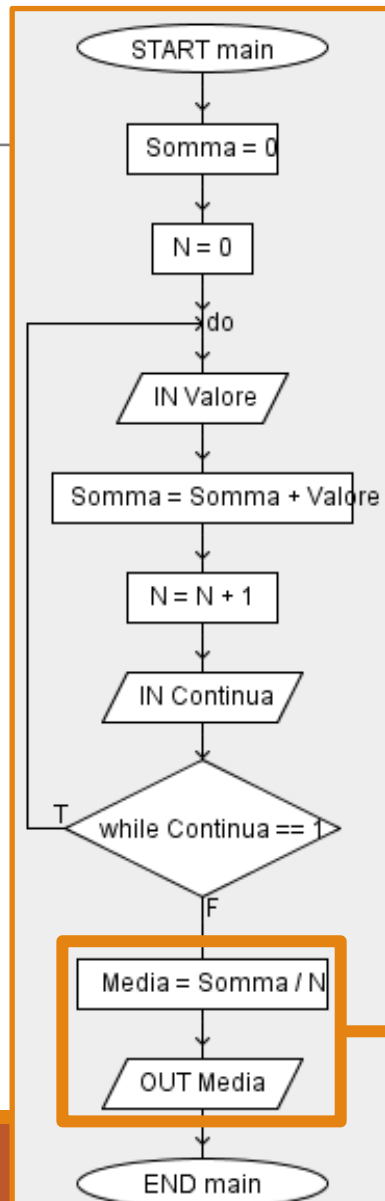
Struttura Iterativa DO/WHILE – Esempio



- 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)

The screenshot displays a flowchart editor interface. At the top, there is a toolbar with icons for file operations and execution control. A green play button is highlighted, and a mouse cursor is positioned over it. To the right of the toolbar, there are checkboxes for 'Traccia' and 'Passo passo', and a 'Tempo (100-5000 ms):' field set to '1500'.

The main area contains a flowchart for a program named 'main'. The flow starts with an oval 'START main', followed by a process box 'Somma = 0', then another process box 'N = 0'. A 'do' connector leads to an input box 'IN Valore', followed by a process box 'Somma = Somma + Valore', then another process box 'N = N + 1', and an input box 'IN Continua'. A decision diamond 'while Continua == 1' follows. The 'T' (True) path loops back to the 'do' connector. The 'F' (False) path leads to a process box 'Media = Somma / N', followed by an output box 'OUT Media'.

On the right side, a code editor shows the corresponding code for the flowchart:

```
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
```

At the bottom of the interface, there are two empty panels labeled 'output' and 'variabili'.

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**

	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

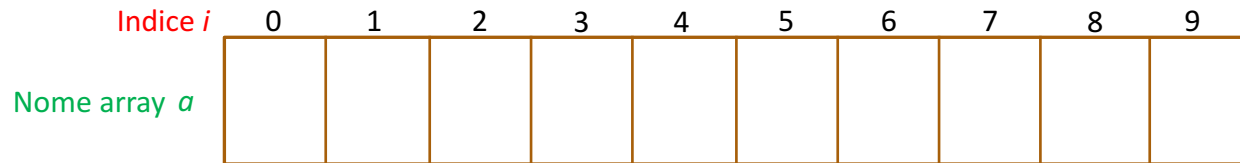
Array (o vettore)

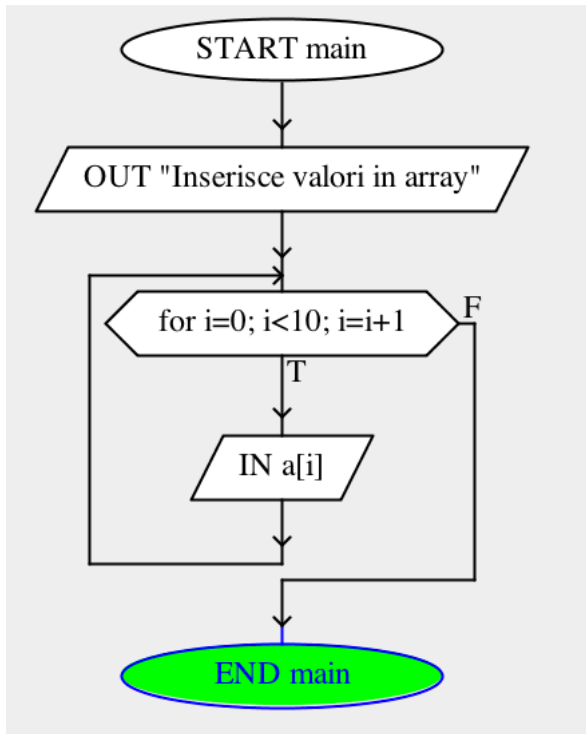
- In AlgoBuild, per accedere ad un singolo elemento di un array, si deve specificare il **nome dell'array** seguito dall'**indice dell'elemento** posto tra parentesi quadre: $a[i]$
- Es: $a[1]$ restituisce l'elemento 15, $a[3]$ restituisce l'elemento 5

	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

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

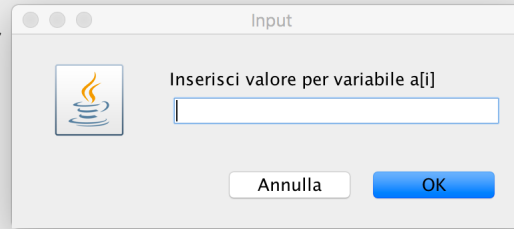
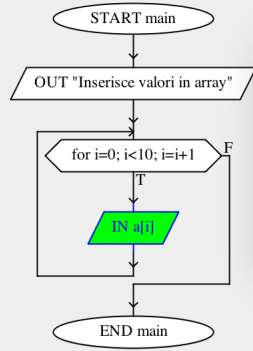




```

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
  
```

Indice <i>i</i>	0	1	2	3	4	5	6	7	8	9
Nome array <i>a</i>										



```

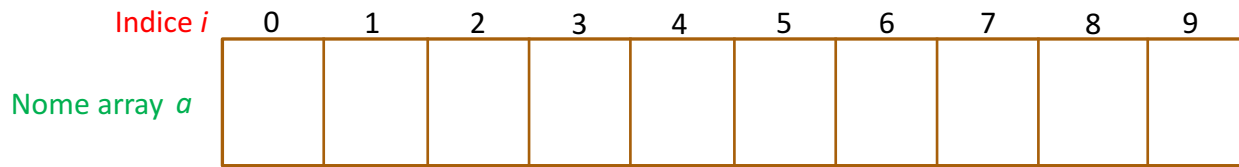
PROG main
  OUT "Inserisce valori in array"
  FOR i=0; i<10; i=i+1
    IN a[i]
  END FOR //i=0; i<10; i=i+1
END PROG //main
  
```

```

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
  
```



START main

OUT "Inserisce valori in array"

Input

Inserisci valore per variabile a[i]

32

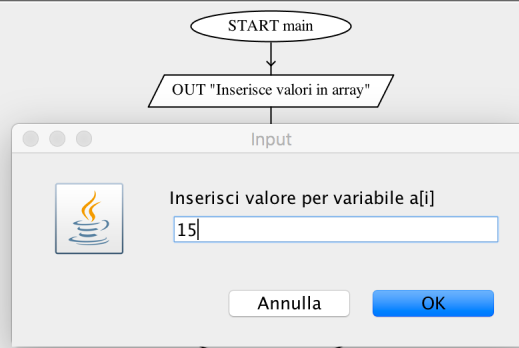
Annulla OK

```
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

Indice <i>i</i>	0	1	2	3	4	5	6	7	8	9
Nome array <i>a</i>										



```

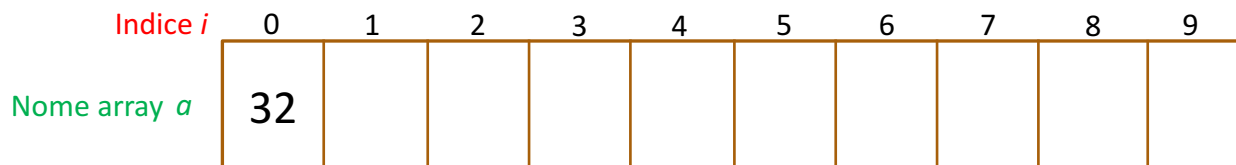
PROG main
  OUT "Inserisce valori in array"
  FOR i=0; i<10; i=i+1
    IN a[i]
  END FOR //i=0; i<10; i=i+1
END PROG //main
  
```

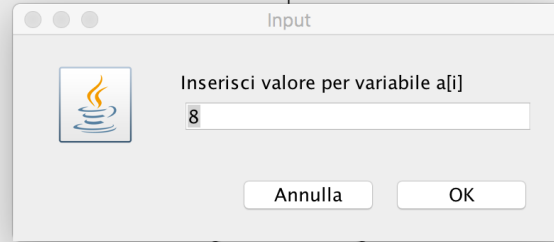
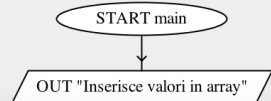
```

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
  
```





```

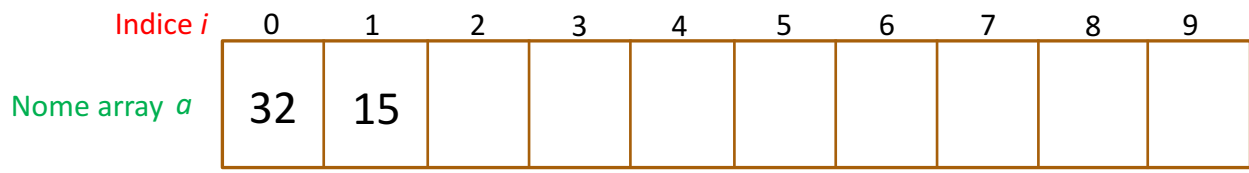
PROG main
  OUT "Inserisce valori in array"
  FOR i=0; i<10; i=i+1
    IN a[i]
  END FOR //i=0; i<10; i=i+1
END PROG //main
  
```

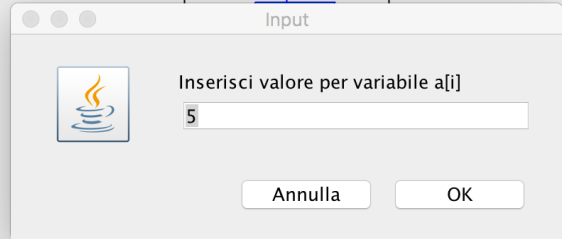
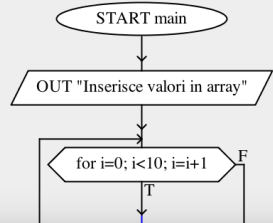
```

PROGRAMMA main.mzda.
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
  
```

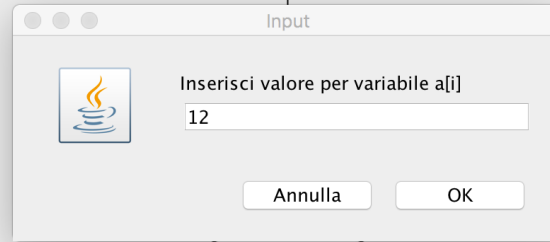
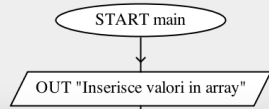
```

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={32.0,15.0,8.0,3.0} | i=3.0 |
FOR i=0; i<10; i=i+1 iterazione: 4
  
```

```

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

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



```

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
  
```

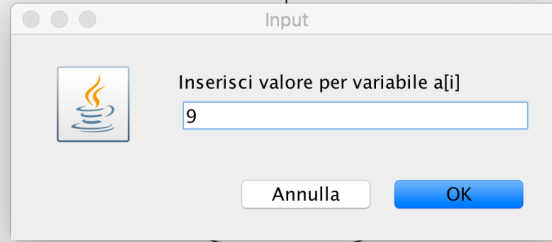
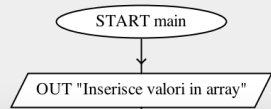
```

INPUT: a[i]=12
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>i</i>	0	1	2	3	4	5	6	7	8	9
Nome array <i>a</i>	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
  
```

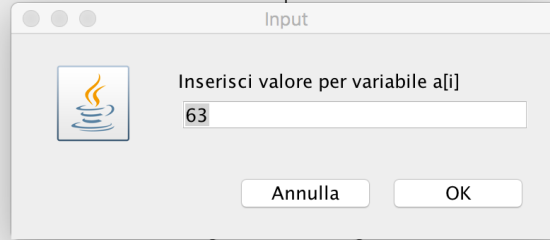
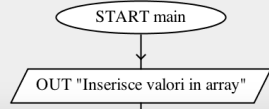
```

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>i</i>	0	1	2	3	4	5	6	7	8	9
Nome array <i>a</i>	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
  
```

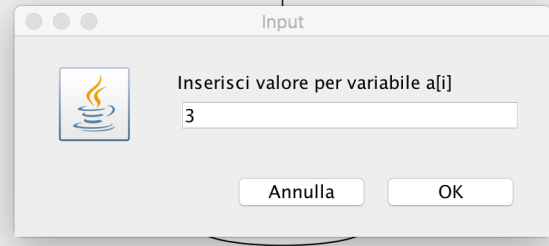
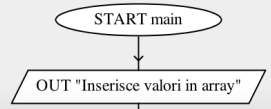
```

//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>i</i>	0	1	2	3	4	5	6	7	8	9
Nome array <i>a</i>	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
  
```

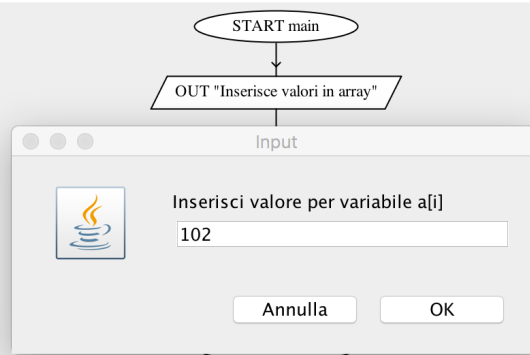
```

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
  
```

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



```

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
  
```

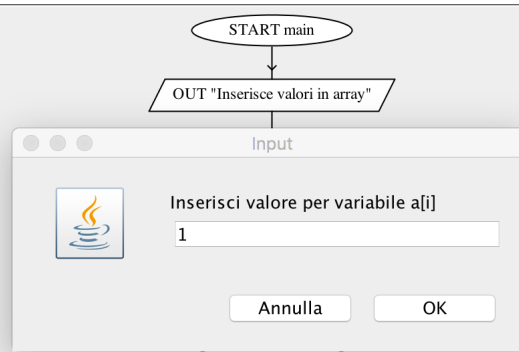
```

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>i</i>	0	1	2	3	4	5	6	7	8	9
Nome array <i>a</i>	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
  
```

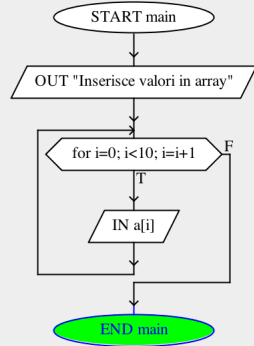
```

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={32.0,15.0,8.0,5.0,12.0,9.0,63.0,3.0,102.0}
i=9.0
  
```

Indice <i>i</i>	0	1	2	3	4	5	6	7	8	9
Nome array <i>a</i>	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
  
```

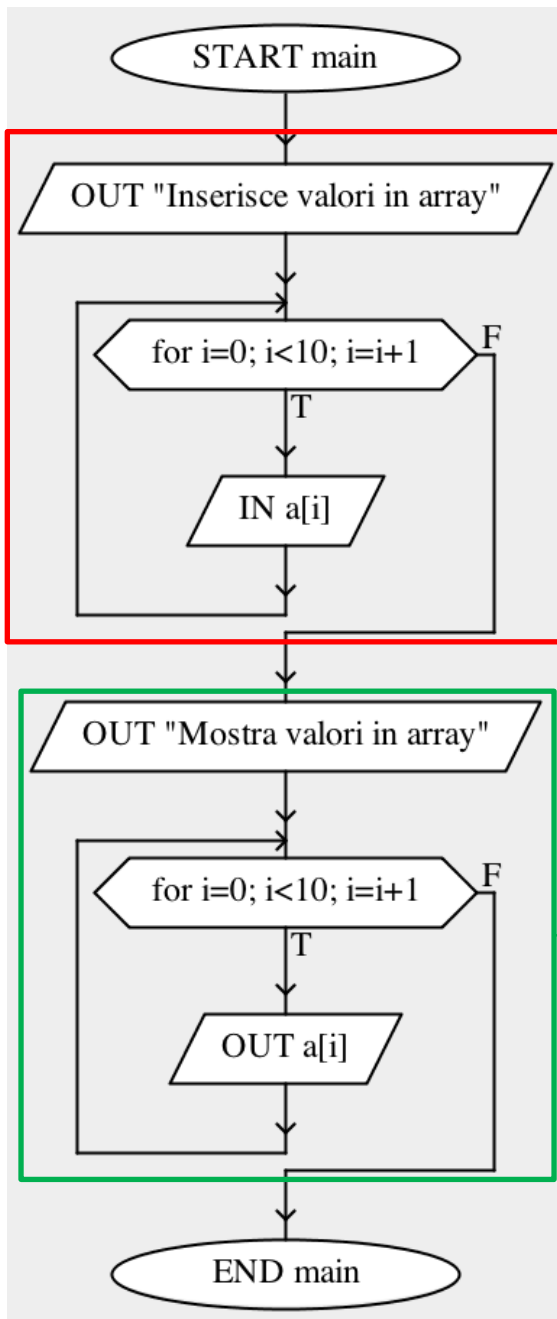
```

    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>i</i>	0	1	2	3	4	5	6	7	8	9
Nome array <i>a</i>	32	15	8	5	12	9	63	3	102	1



Prende valori in input

```

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
  
```

Restituisce valori in output

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 $a[i] > a[\text{posizione}]$

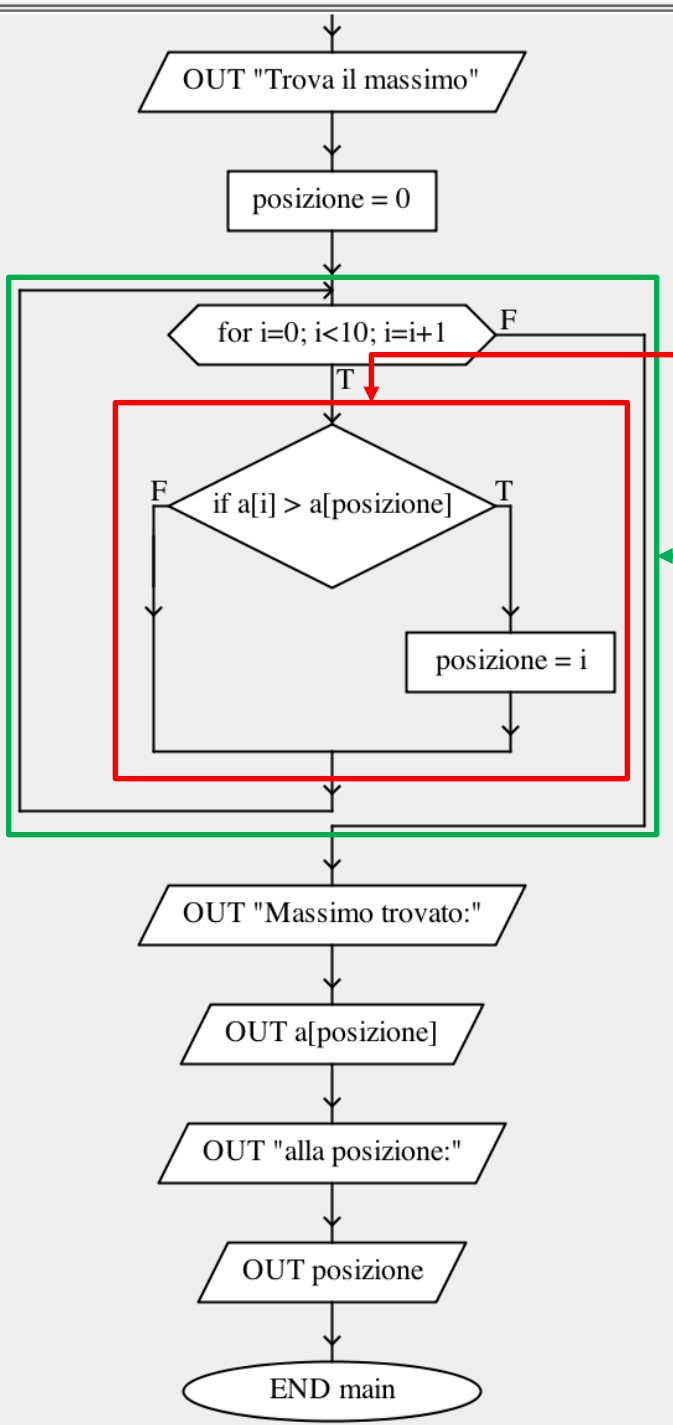
posizione = i //istruzione eseguita se $A(i) > \max$

Incrementa i

restituisce $a[\text{posizione}]$, posizione

FINE ALGORITMO trovaMax

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



```

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*)

