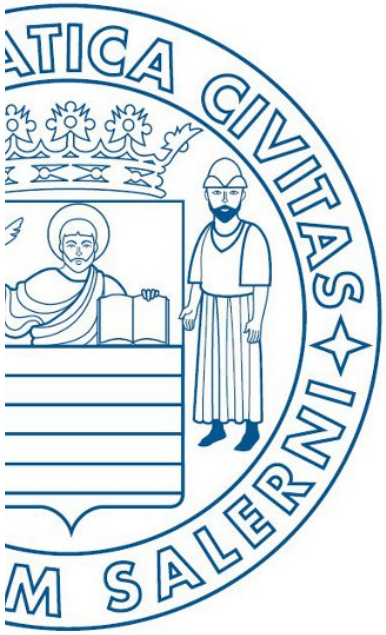


Università di Salerno
Dipartimento di
Ingegneria Industriale

UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

Introduzione ad AlgoBuild

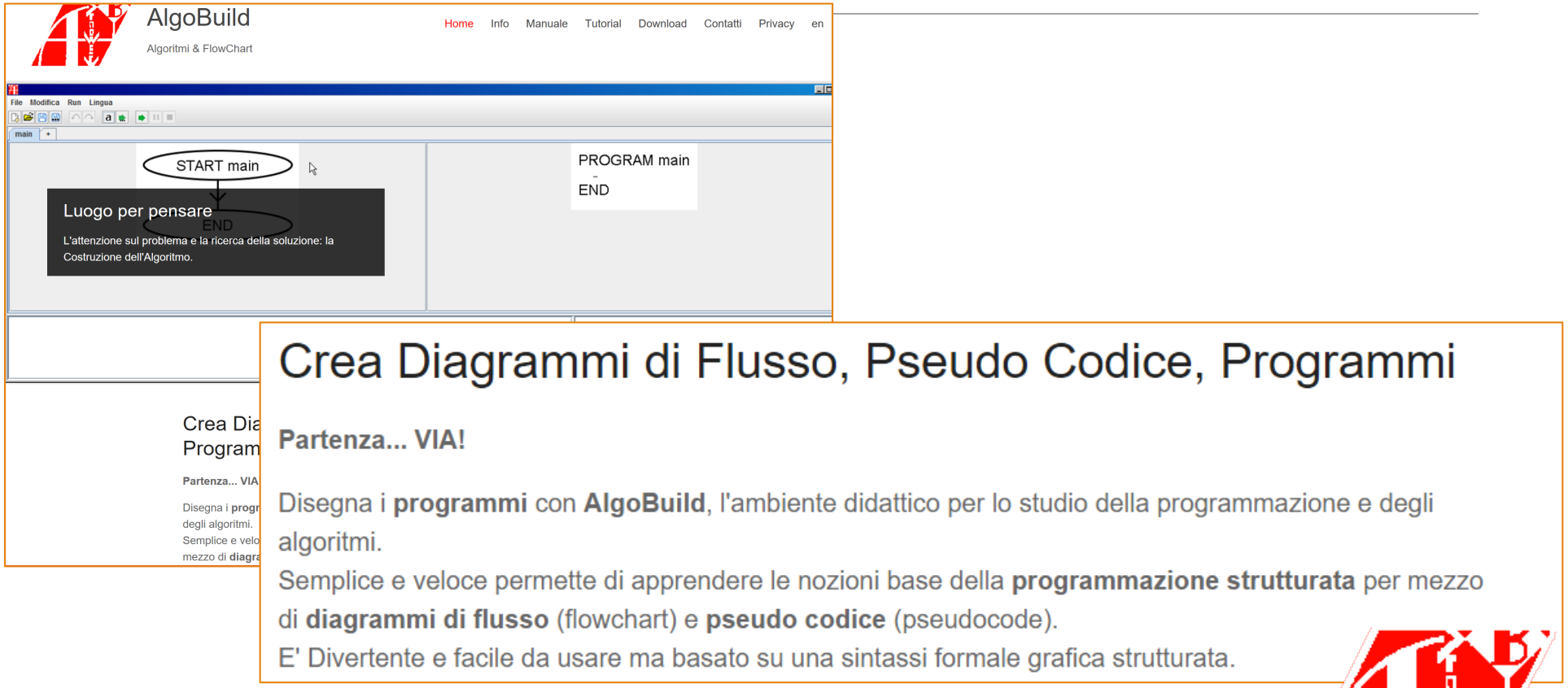
Prof. Christian Esposito
Corso di Laurea in Ingegneria Meccanica e Gestionale (Classe I)
A.A. 2017/18

AlgoBuild

Introduzione ad AlgoBuild: OUTLINE

- Caratteristiche
- Come si presenta
- Utilizzo del blocco di output
  - *Esempio*
- Utilizzo dei blocchi di input, output ed esecuzione
  - *Esempio*

Caratteristiche – 1/3



- Fonte
  - https://algobuild.com/it/index.html

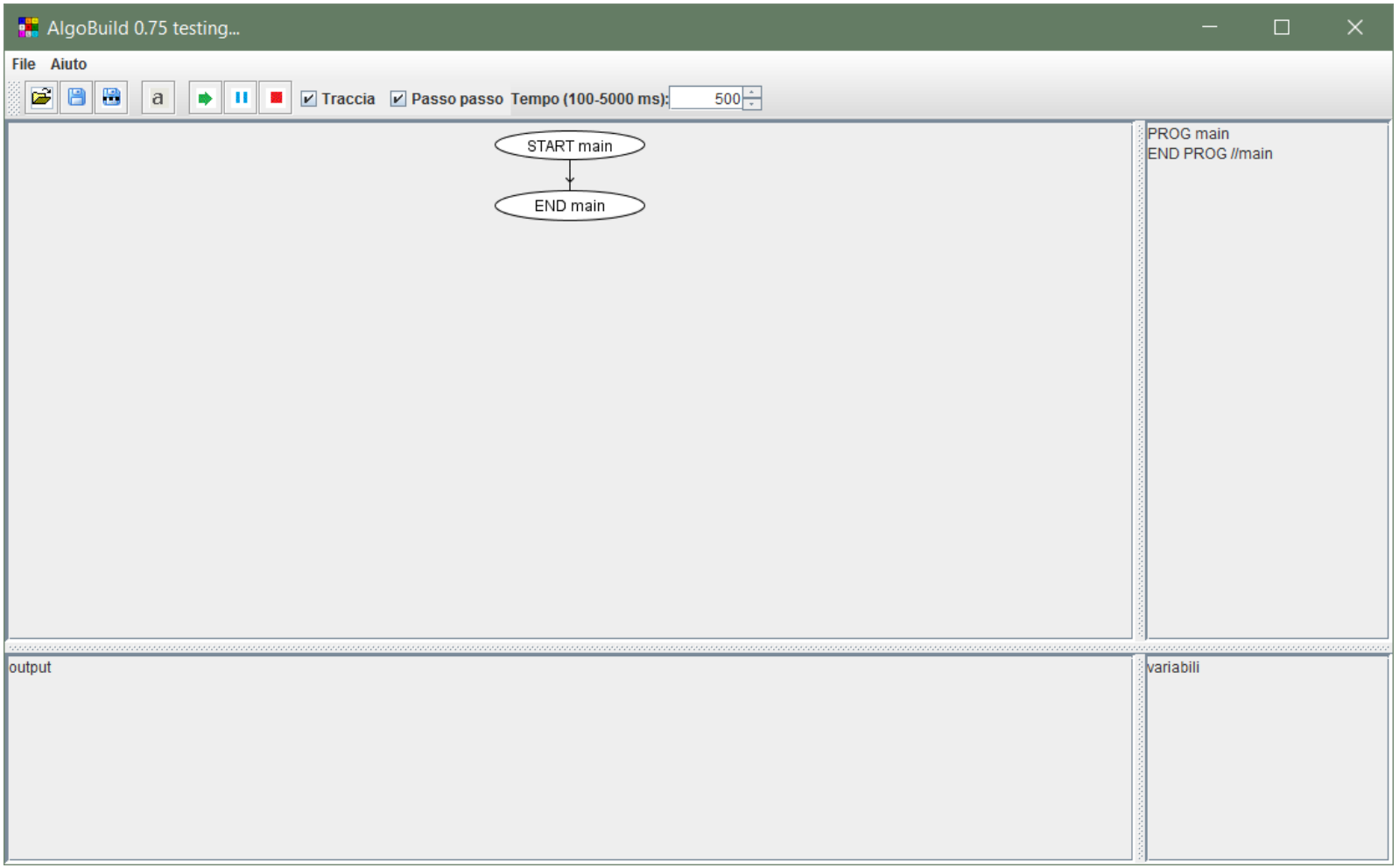
Caratteristiche – 2/3

- Con AlgoBuild è possibile disegnare in maniera semplice ed efficace **diagrammi di flusso**
- AlgoBuild permette anche di tradurre i diagrammi di flusso in **pseudo-codice**
- *Maggiori informazioni*
  - https://algobuild.com/it/index.html

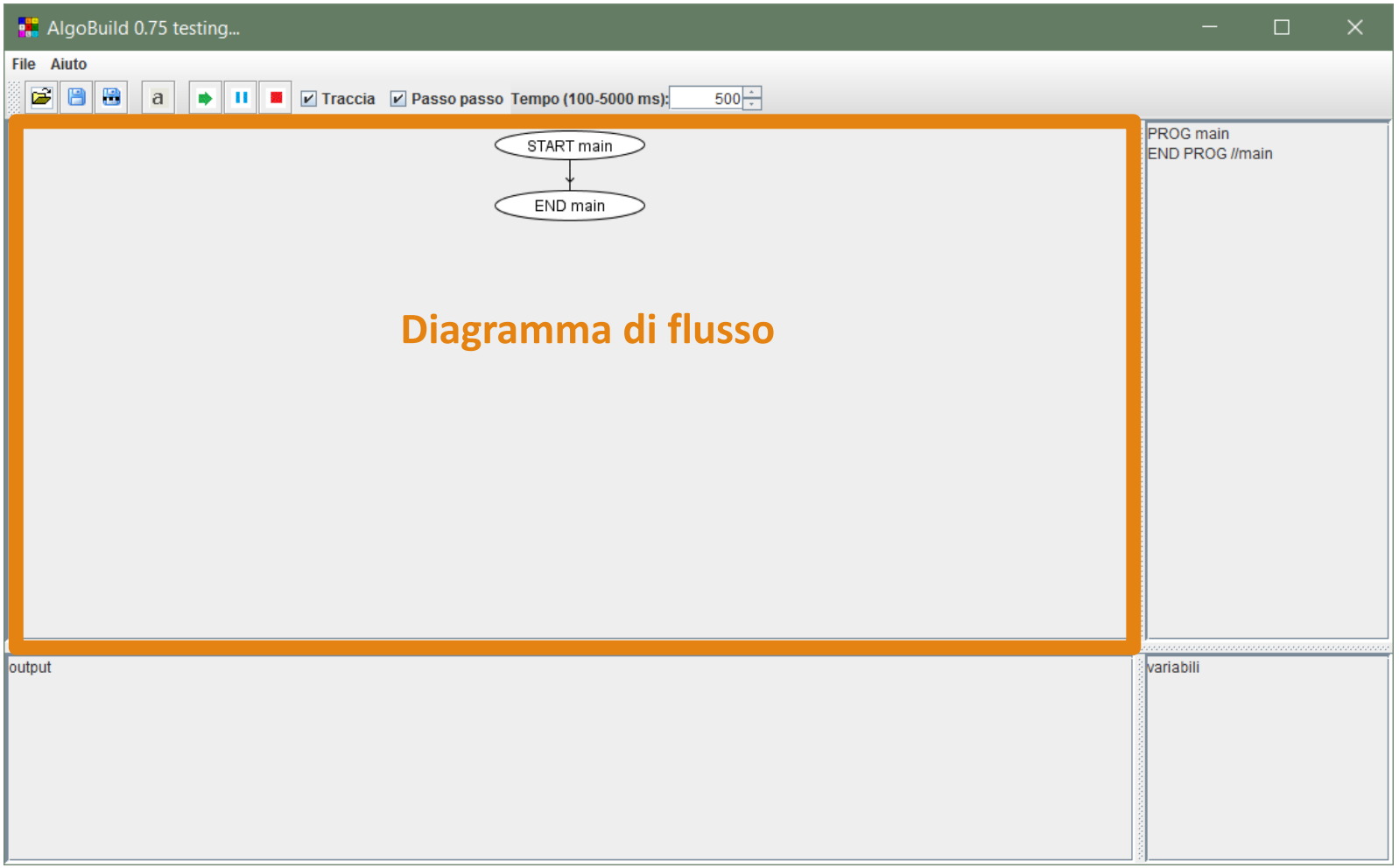
Caratteristiche – 3/3

- Dove reperirlo?
  - AlgoBuild è scaricabile gratuitamente
  - L'indirizzo da cui può essere scaricato è
    - https://algobuild.com/it/download.html
  - La versione *stabile* attualmente è la **0.80**

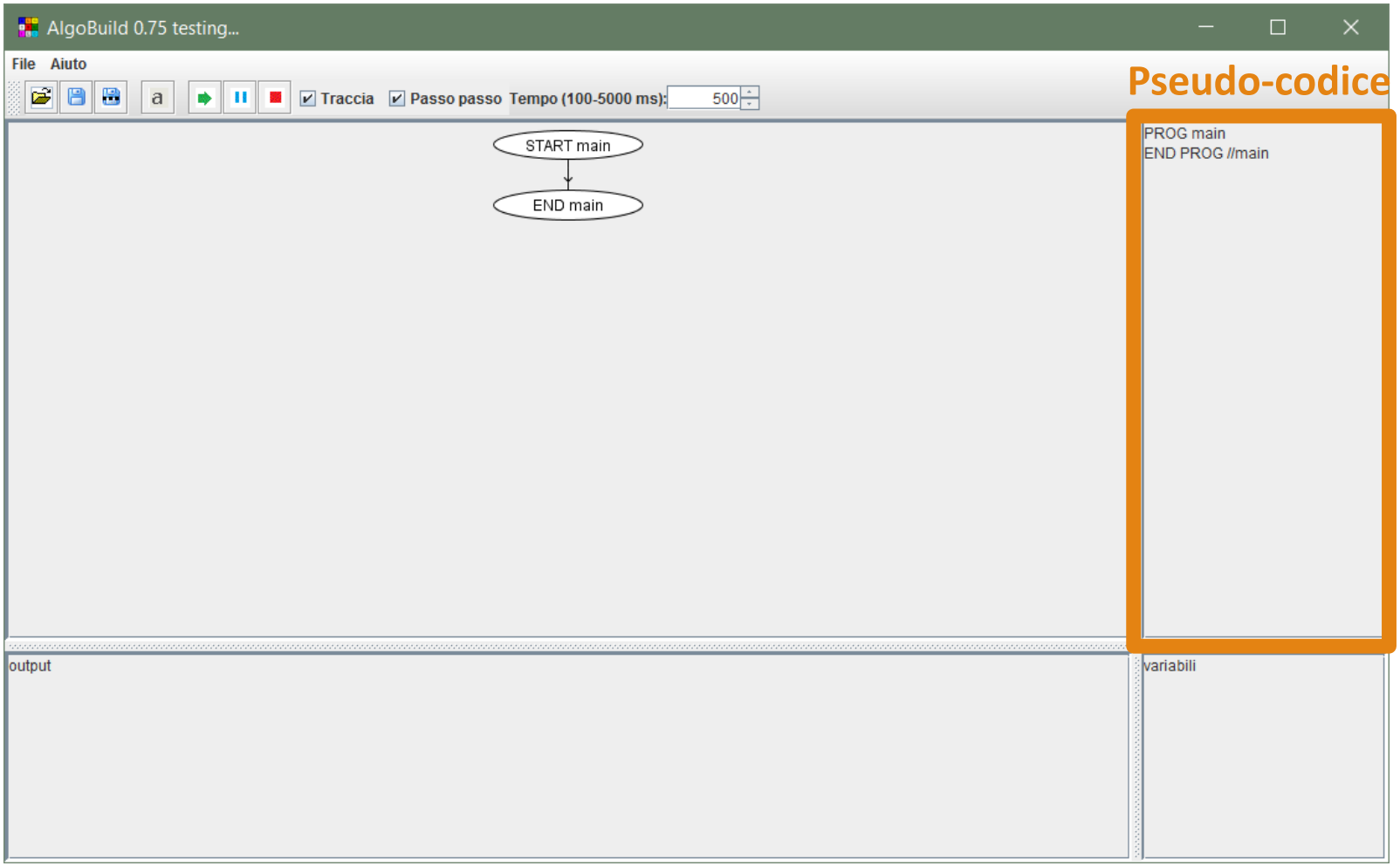
Come si Presenta



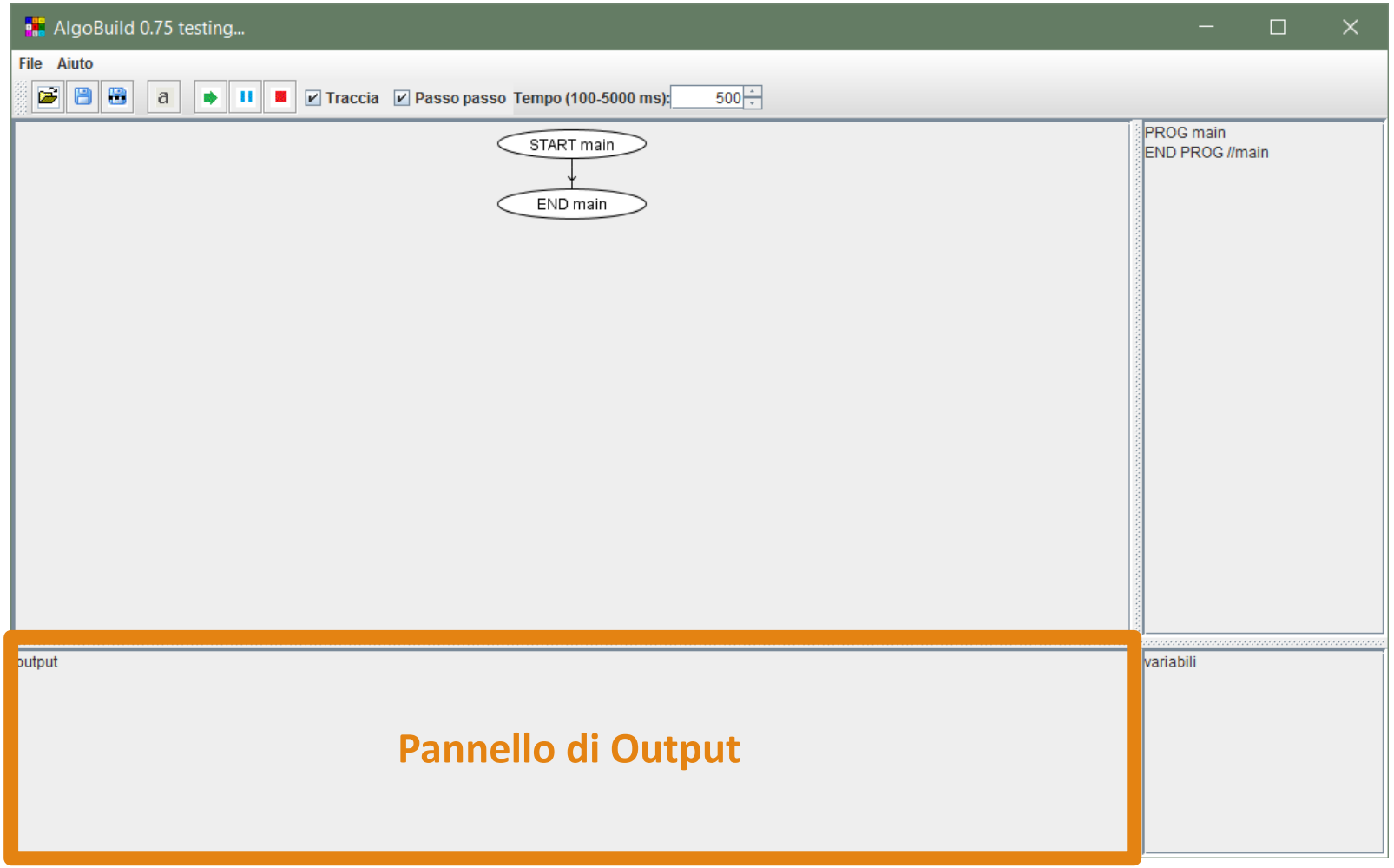
Come si Presenta



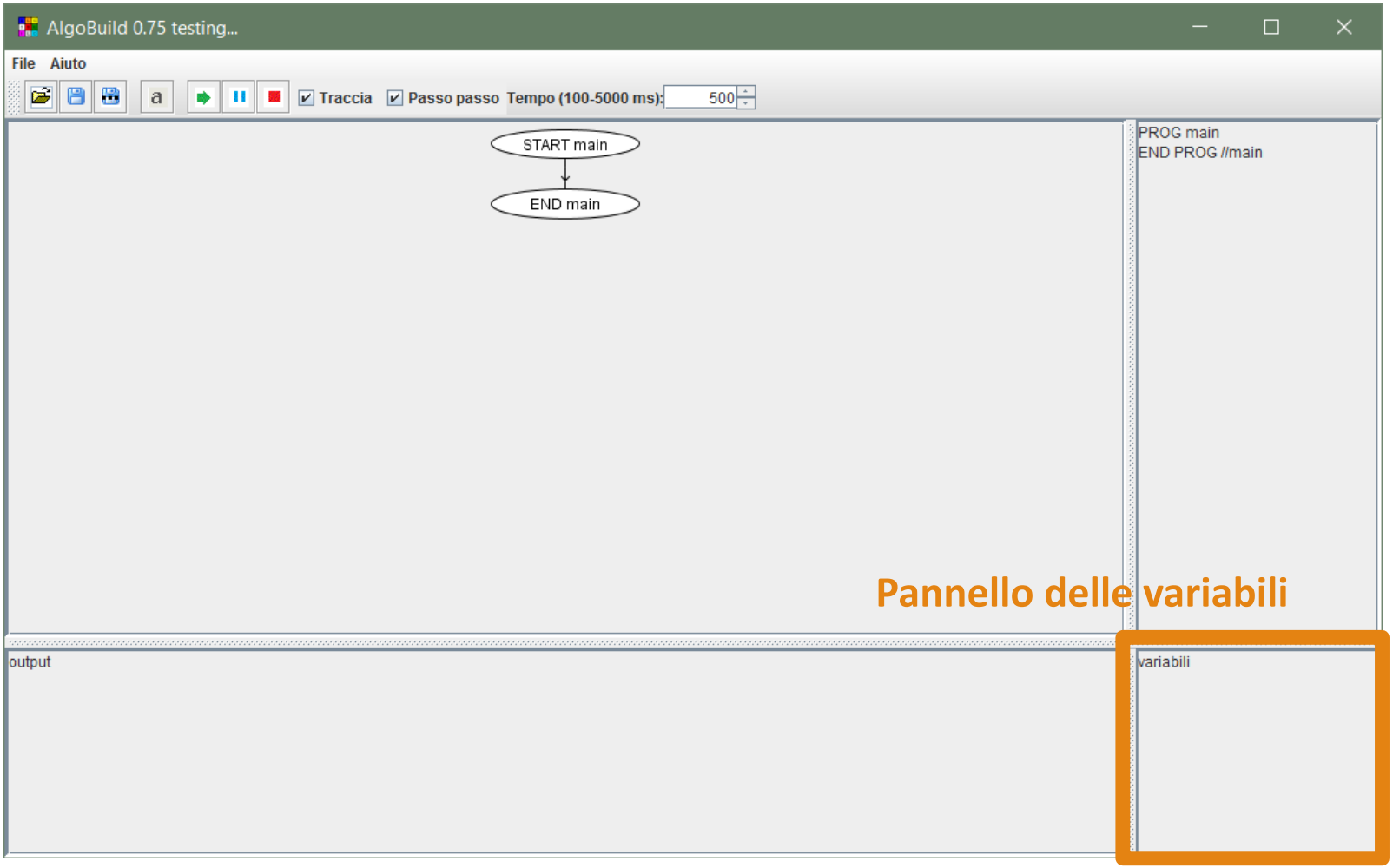
Come si Presenta



Come si Presenta



Come si Presenta



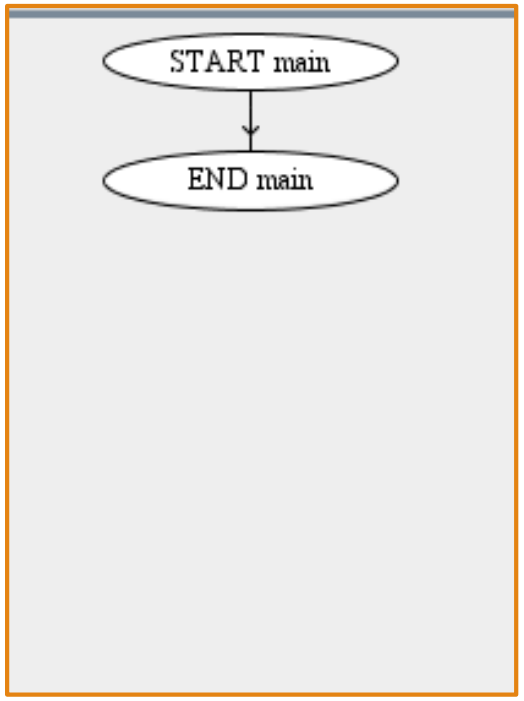
Come si Presenta

Esempio 1: “Hello, World!” – 1/13

- Iniziamo ad utilizzare AlgoBuild con l’esempio *Hello, World!*
- *Hello, World!* mostra semplicemente la stringa
  - **“Ciao, Mondo!”**
- Storicamente, molti manuali di programmazione usano l’esempio “Hello, world!” per mostrare lessico, sintassi e semantica basilare di un dato linguaggio di programmazione

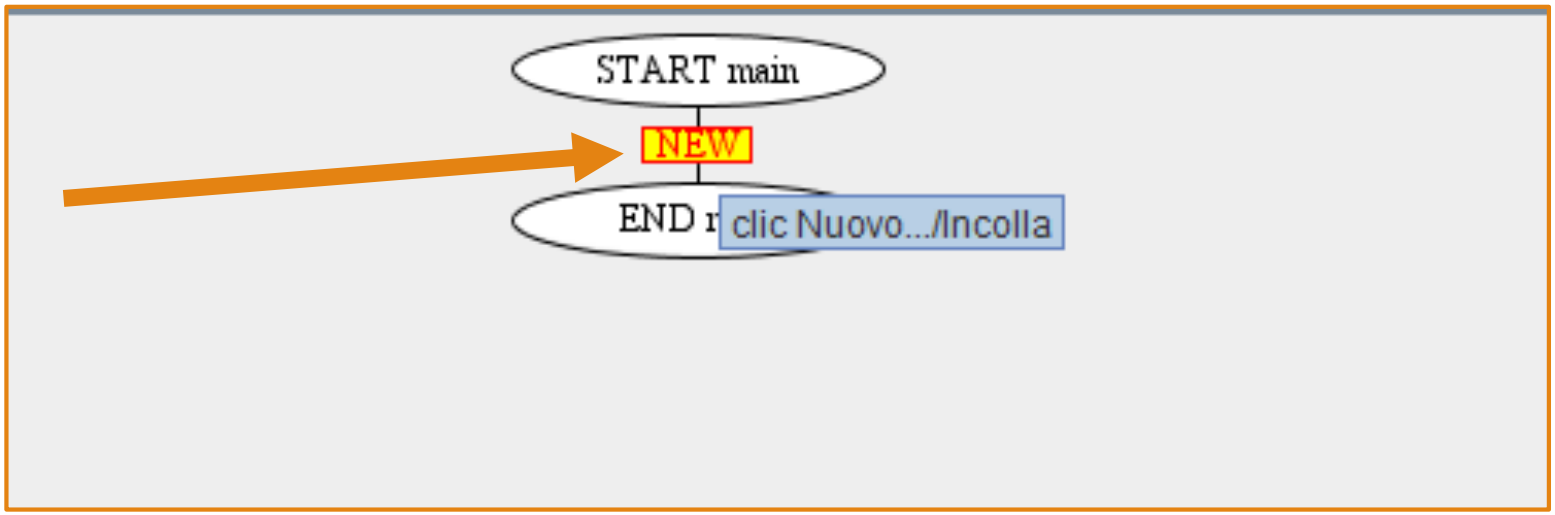
Esempio 1: “Hello, World!” – 2/13

- Nell’area del diagramma di flusso possiamo notare i due blocchi di inizio (*START*) e fine (*END*)
  - Sono inseriti **automaticamente** da AlgoBuild



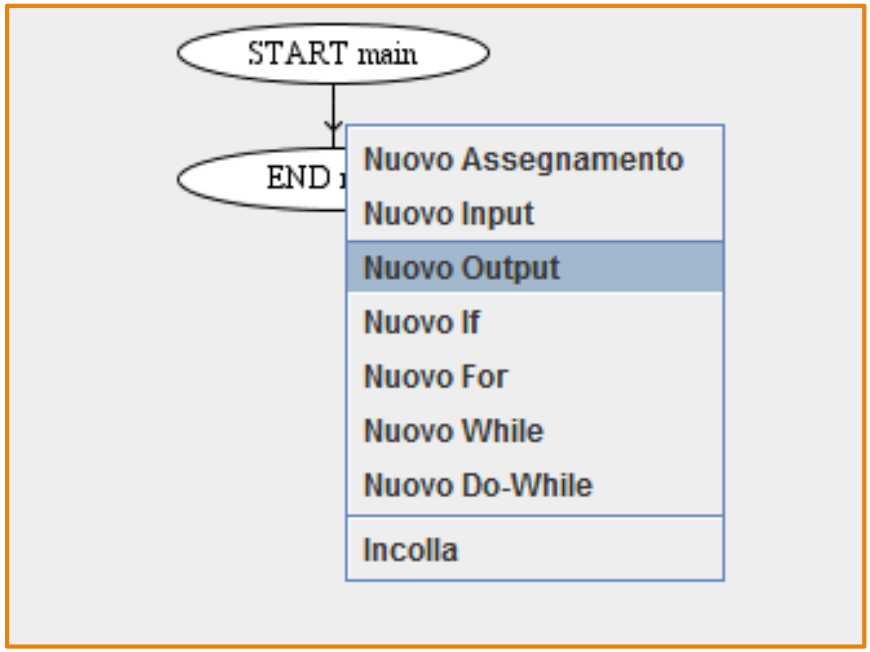
Esempio 1: “Hello, World!” – 3/13

- Posizioniamoci con il mouse **sulla freccia** che collega lo *START* e l’*END* del nostro diagramma di flusso



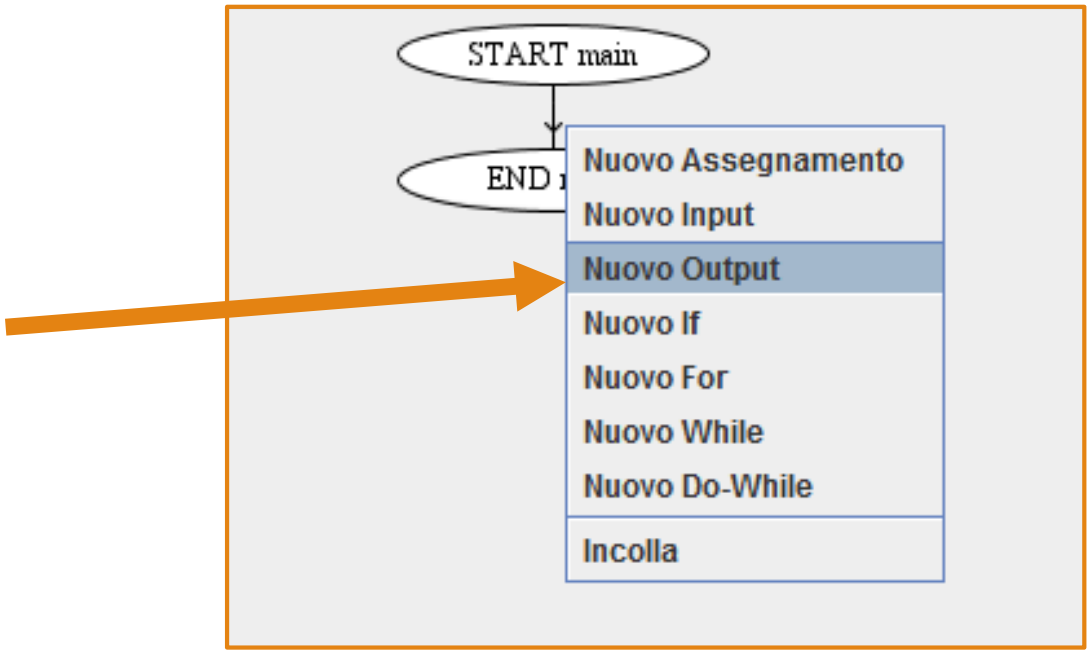
Esempio 1: “Hello, World!” – 4/13

- Cliccando su **«NEW»**, ci verranno proposte diverse alternative per l’inserimento di un nuovo blocco



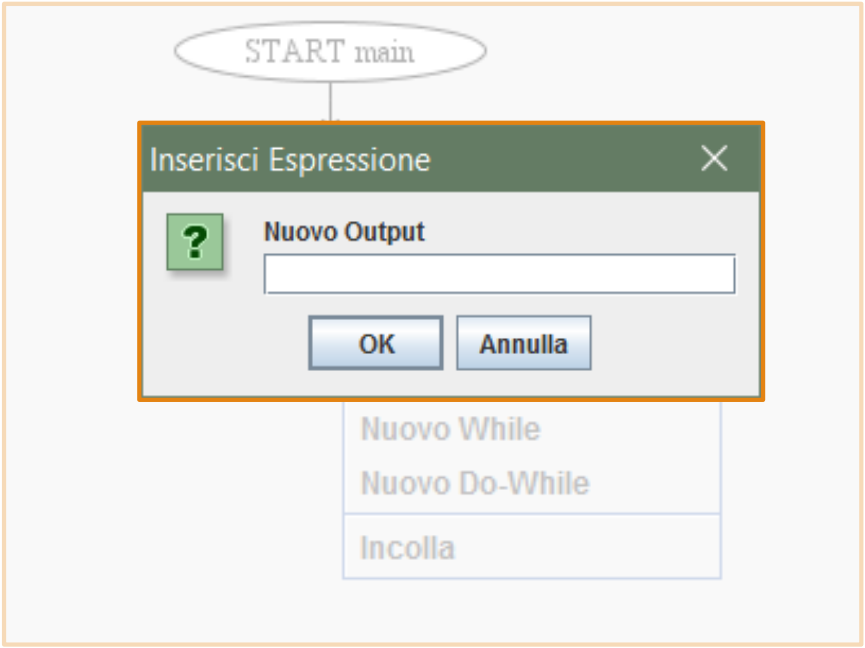
Esempio 1: “Hello, World!” – 5/13

- Selezioniamo **Nuovo Output**



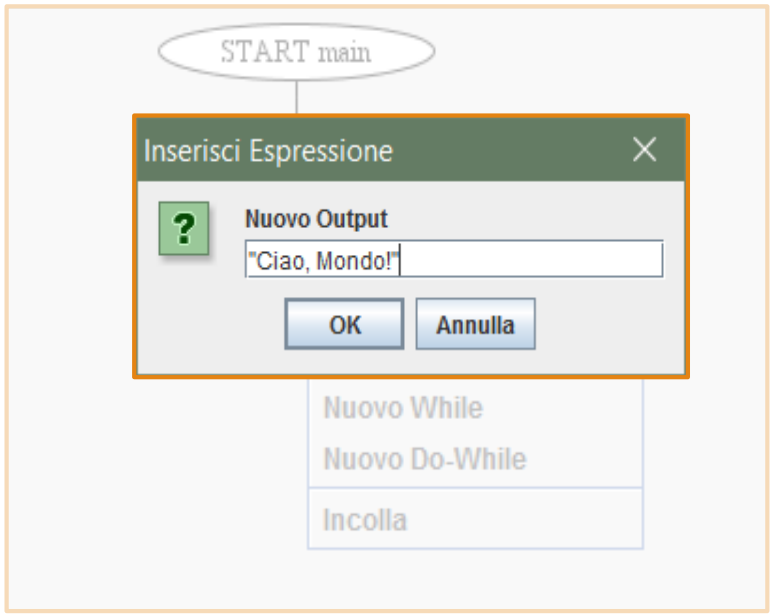
Esempio 1: “Hello, World!” – 6/13

- Selezioniamo **Nuovo Output**
  - Ci verrà richiesto qual è l’output che vogliamo mostrare



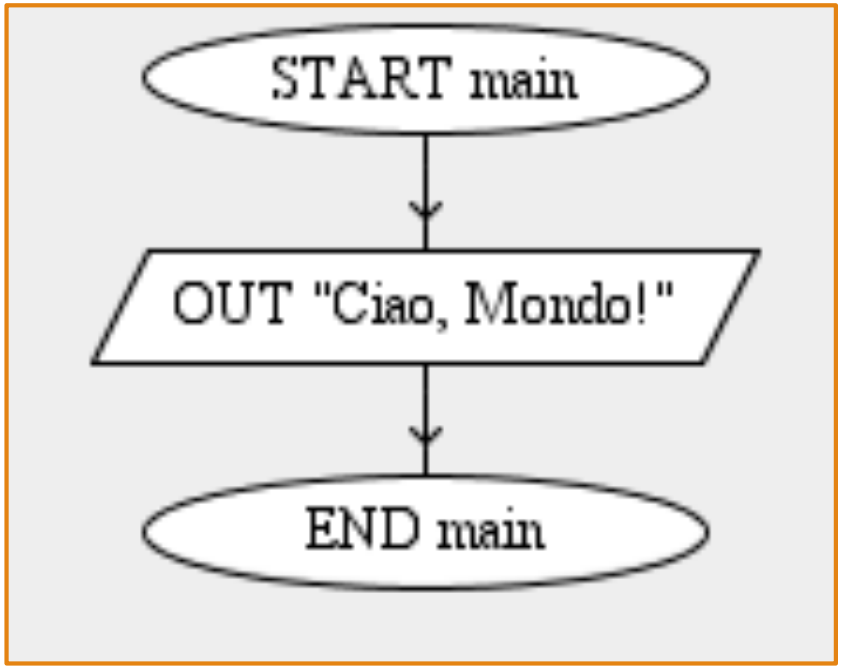
Esempio 1: “Hello, World!” – 7/13

- Scriviamo la stringa “Ciao, Mondo!”, poi
  - Clicchiamo su **OK**
  - Oppure premiamo il tasto **Invio** della tastiera



Esempio 1: “Hello, World!” – 8/13

- Ecco il nostro **diagramma di flusso**



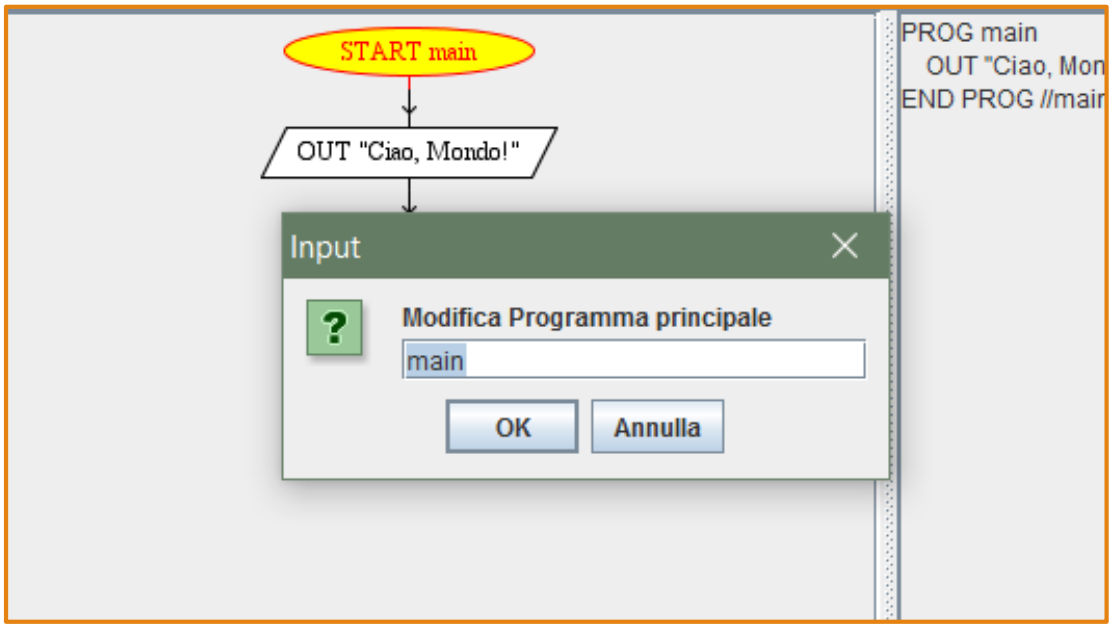
Esempio 1: “Hello, World!” – 9/13

- AlgoBuild ha contestualmente generato anche lo pseudo-codice
  - Ecco cosa ci presenterà l’area preposta

```
PROG main
   OUT "Ciao, Mondo!"
END PROG //main
```

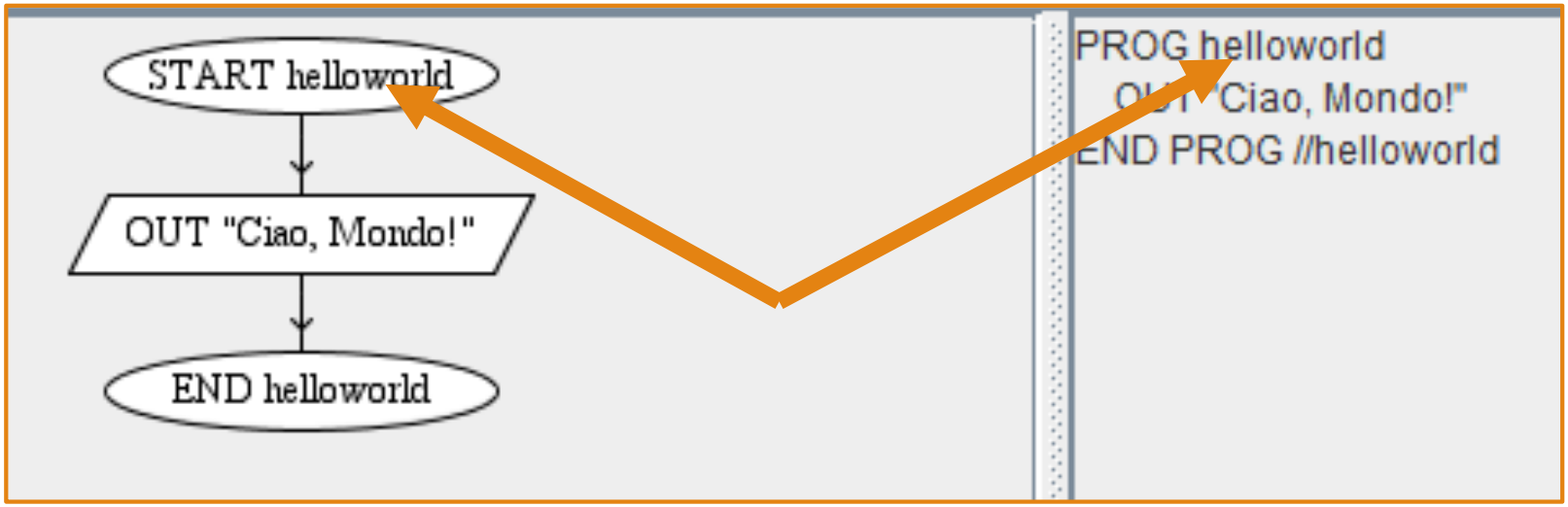
Esempio 1: “Hello, World!” – 10/13

- Possiamo anche modificare il nome del diagramma di flusso
  - Cliccando sul blocco *START* oppure *END*
  - Scrivendo il nome che vogliamo assegnare al diagramma



Esempio 1: “Hello, World!” – 11/13

- Possiamo modificare anche il nome del diagramma di flusso
  - Cliccando sul blocco *START* oppure *END*
  - Scrivendo il nome che vogliamo assegnare al diagramma
    - Ad esempio, lo chiamiamo **helloworld**

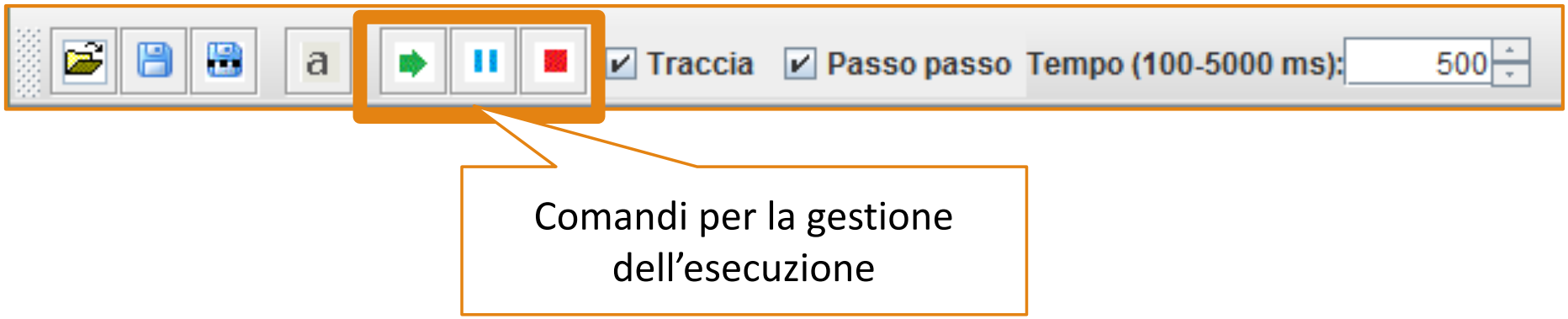


Esempio 1: “Hello, World!” – 12/13

- **NOTA IMPORTANTE**
  - Per inserire un **nuovo blocco** dobbiamo sempre **cliccare sulla freccia** che collega i **due blocchi tra i quali vogliamo inserire un nuovo blocco**

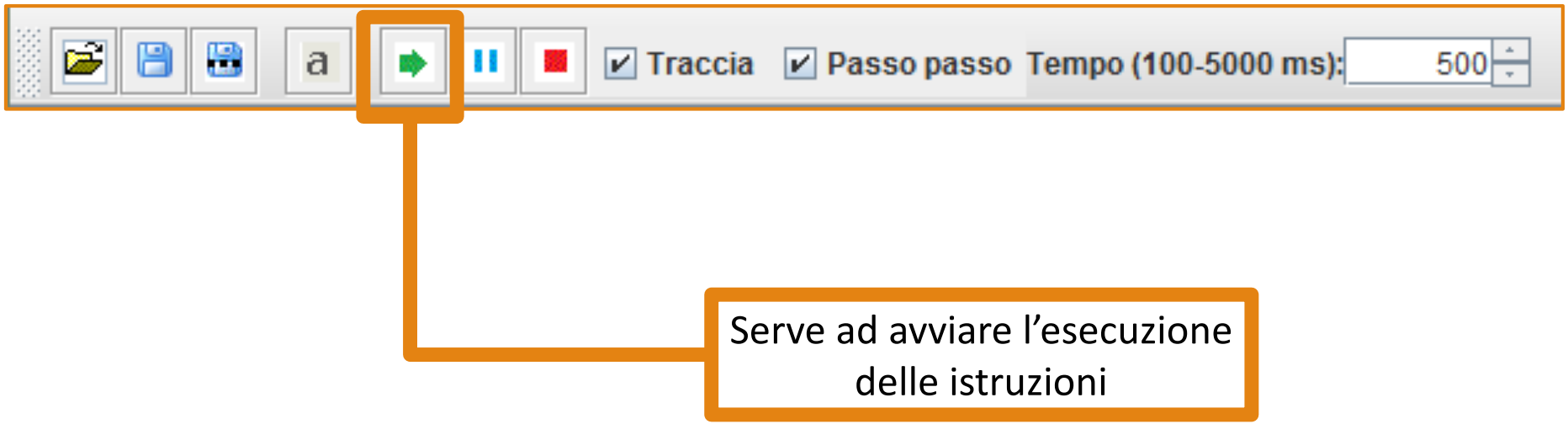
Esempio 1: “Hello, World!” – 13/13

- L’esecuzione…
  - Ora che il nostro diagramma è stato generato, possiamo simulare la sua esecuzione tramite AlgoBuild



Esempio 1: “Hello, World!” – 13/13

- L’esecuzione…
  - Ora che il nostro diagramma è stato generato, possiamo simulare la sua esecuzione tramite AlgoBuild

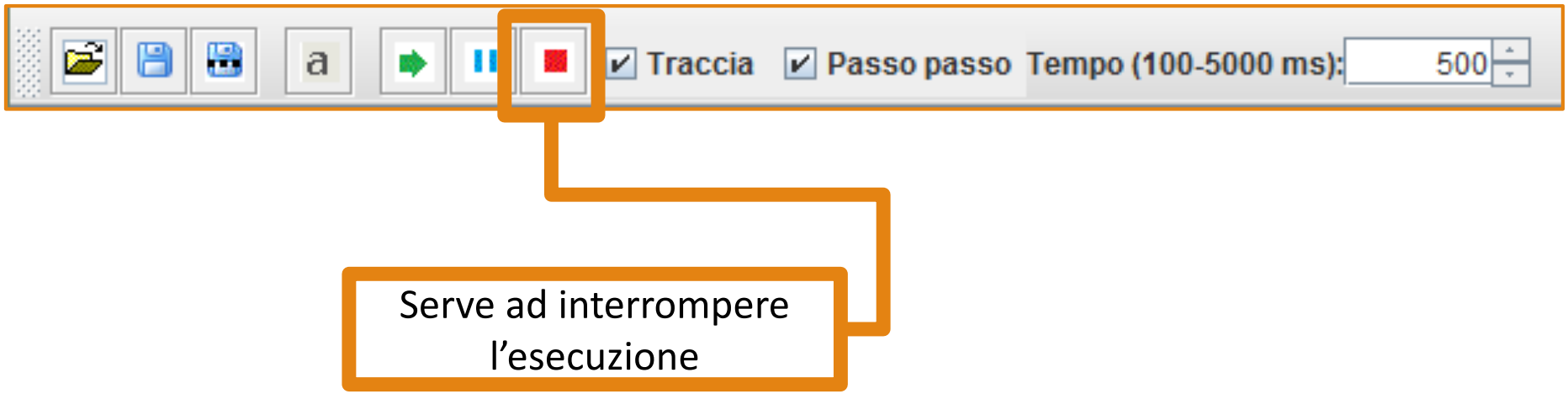


Esempio 1: “Hello, World!” – 13/13

- L’esecuzione…
  - Ora che il nostro diagramma è stato generato, possiamo simulare la sua esecuzione tramite AlgoBuild

Esempio 1: “Hello, World!” – 13/13

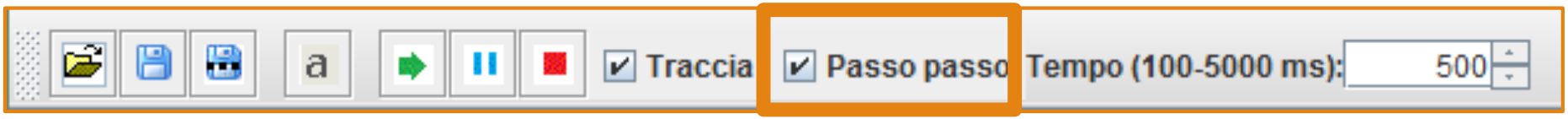
- L’esecuzione…
  - Ora che il nostro diagramma è stato generato, possiamo simulare la sua esecuzione tramite AlgoBuild



Esempio 1: “Hello, World!” – 13/13

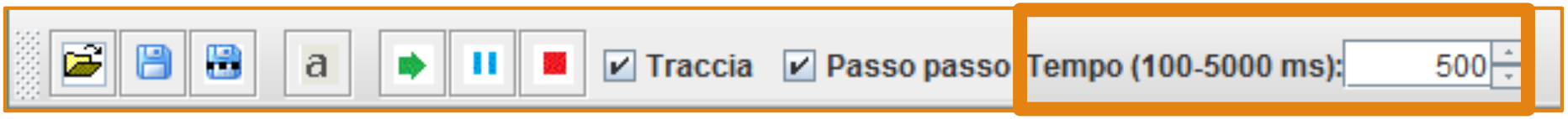
- L’esecuzione…
  - Ora che il nostro diagramma è stato generato, possiamo simulare la sua esecuzione tramite AlgoBuild

- AlgoBuild permette di simulare l’esecuzione anche *passo passo*
  - In questo caso sarà necessario cliccare ogni volta su  per eseguire l’istruzione successiva

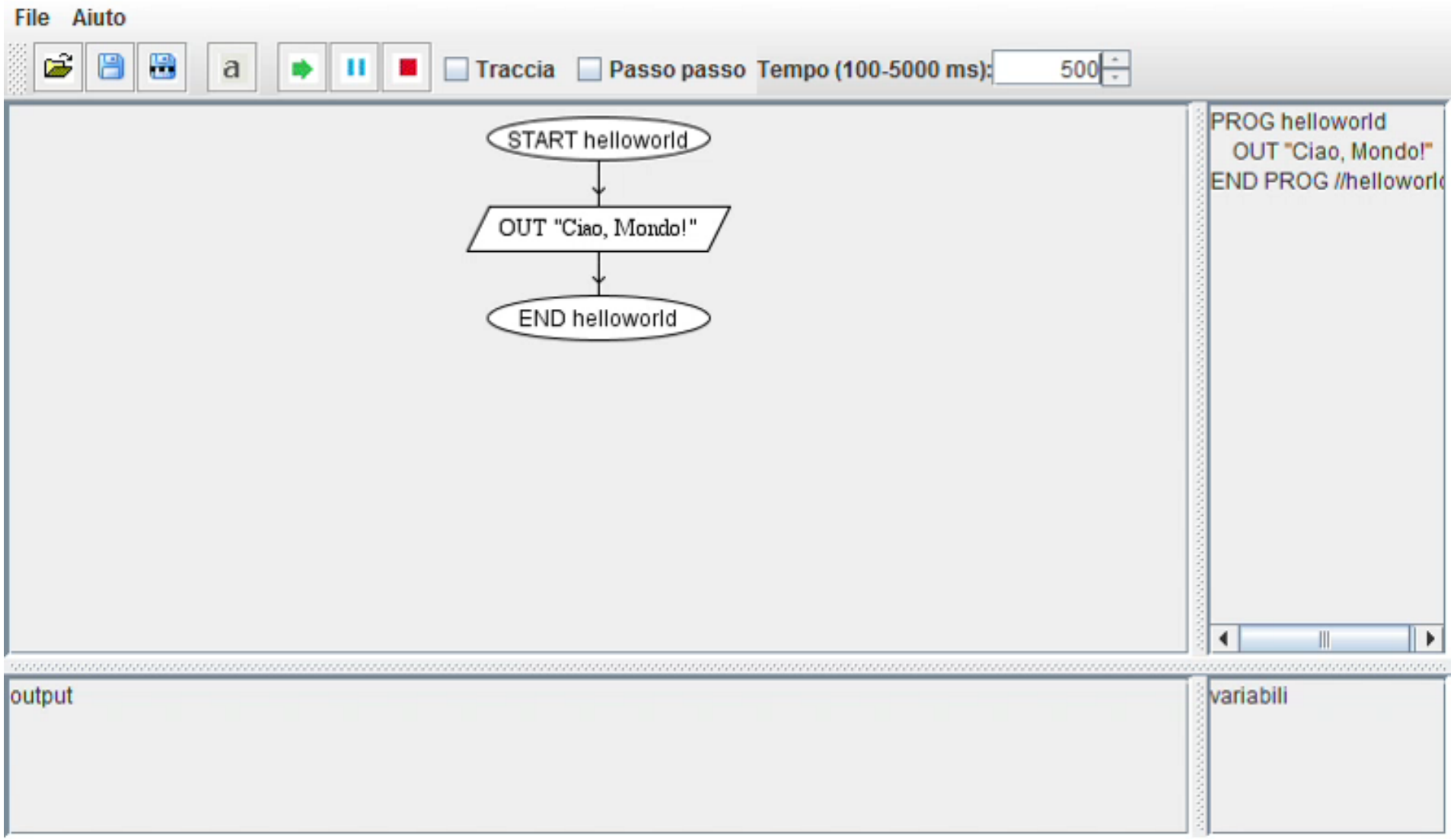


Esempio 1: “Hello, World!” – 13/13

- L’esecuzione…
  - Ora che il nostro diagramma è stato generato, possiamo simulare la sua esecuzione tramite AlgoBuild
- AlgoBuild permette di simulazione l’esecuzione, anche *passo passo*
- Possiamo anche decidere il tempo (in millisecondi) che intercorre tra ogni istruzione eseguita

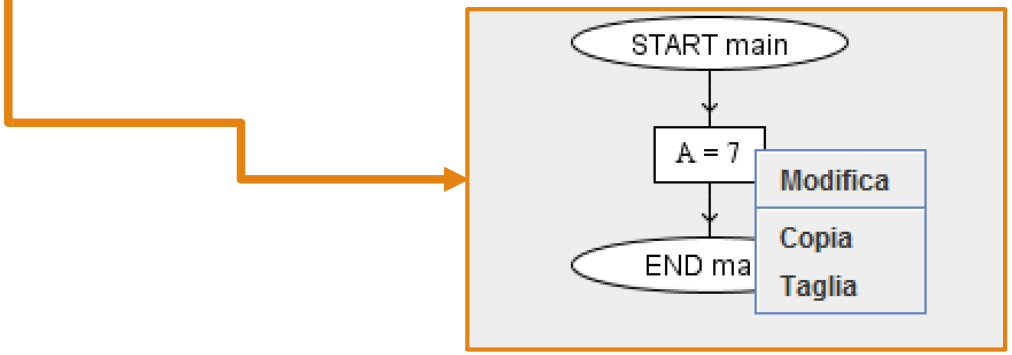


DEMO Esecuzione «Hello, World!» (Tempo passo: 5000ms, ovvero 5 secondi)



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*)



AlgoBuild:
Operatori Artimetici, Relazionali e Logici

- **Operatori Aritmetici**

Operatore	Descrizione
+	Addizione
-	Sottrazione
*	Moltiplicazione
/	Divisione
%	Resto della divisione intera

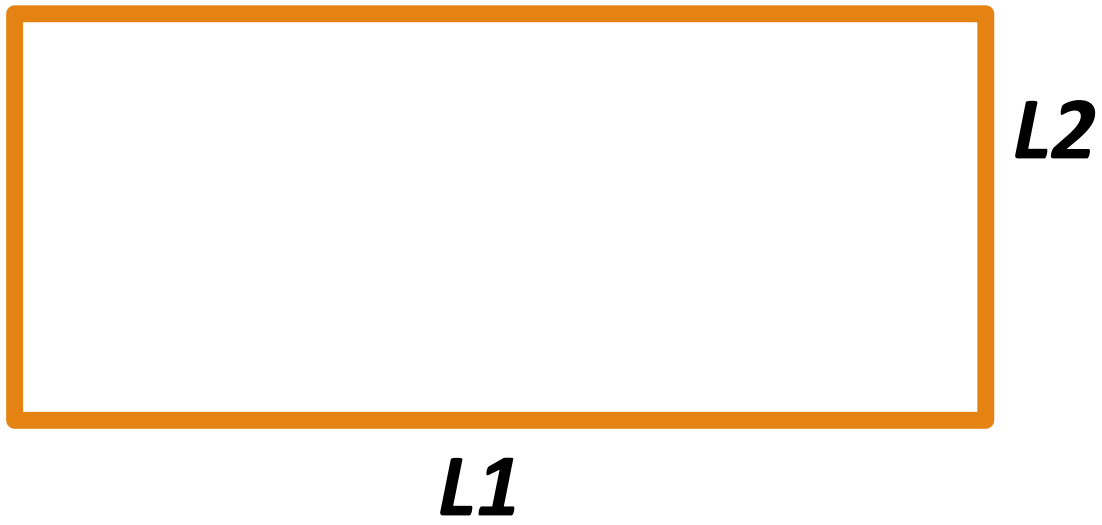
- **Operatori Logici**

Operatore	Descrizione
&&	AND
||	OR
!	NOT

- **Operatori Relazionali**

Operatore	Descrizione
<	Minore
<=	Minore o uguale
>	Maggiore
>=	Maggiore o uguale
==	Uguale
!=	Diverso

Esempio 2: Perimetro Rettangolo – 1/5



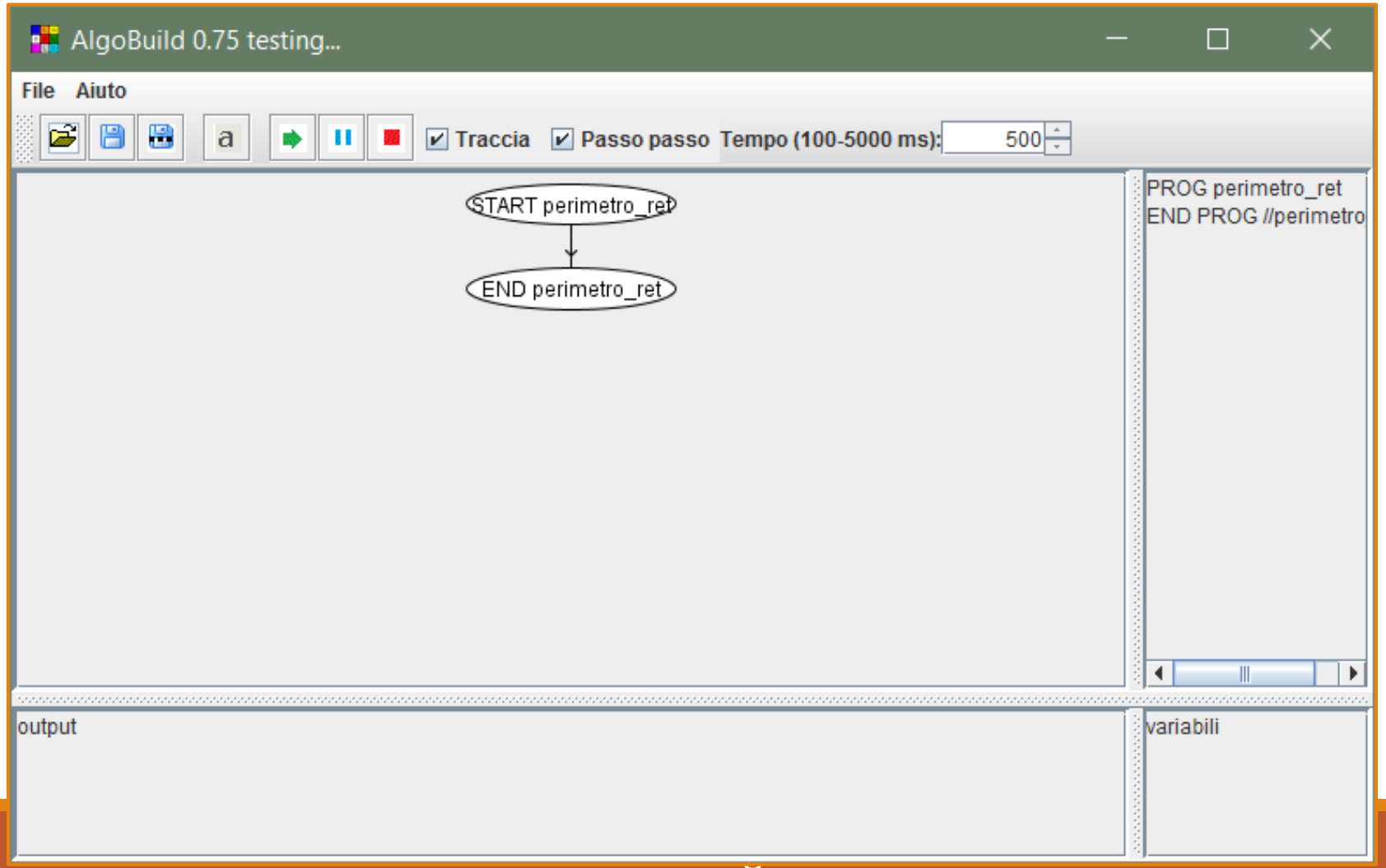
$P = 2 * (L1 + L2)$

Esempio 2: Perimetro Rettangolo – 1/5

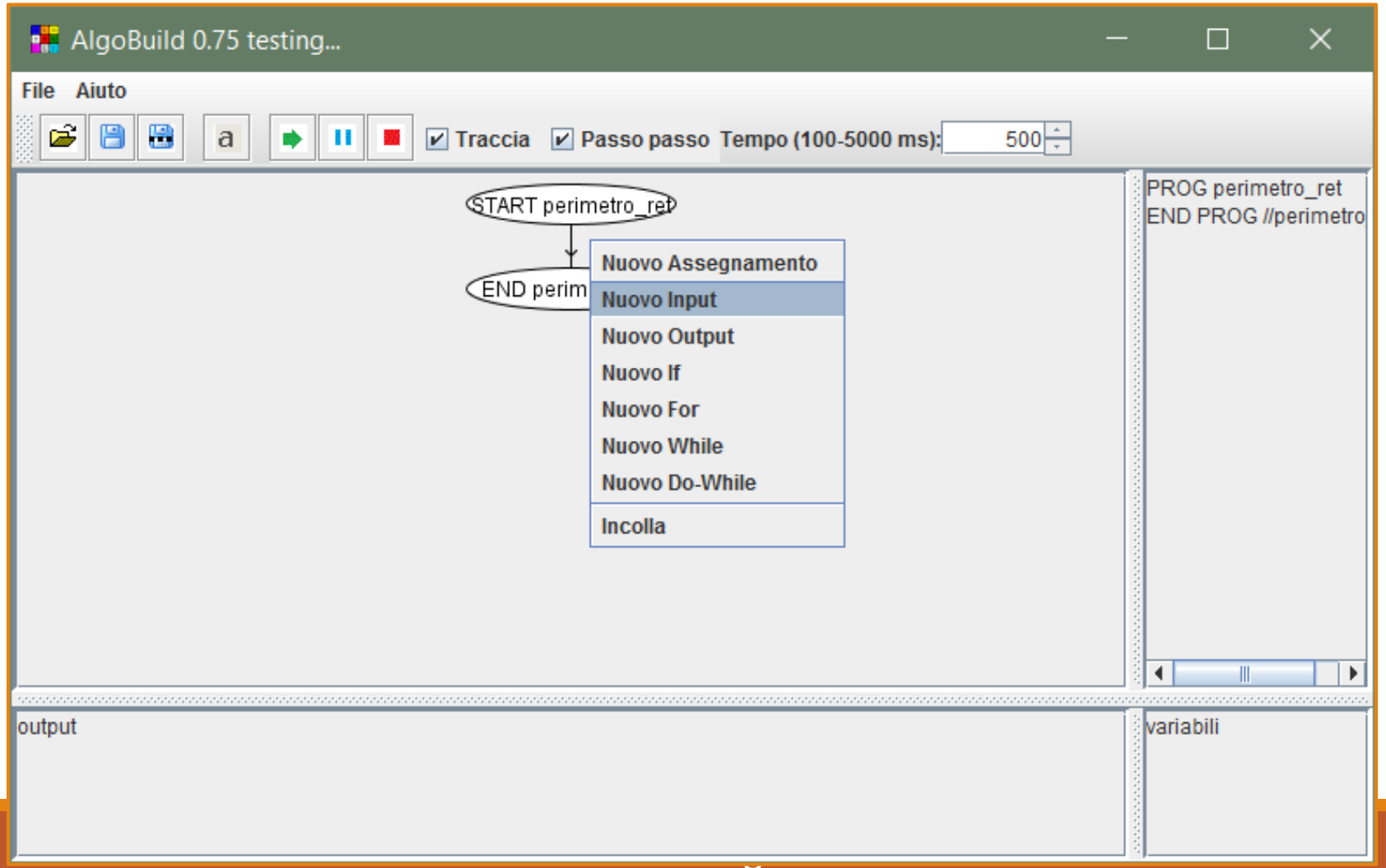
*P = 2 * (L1 + L2)*

L1 → input
L2 → input
P → output

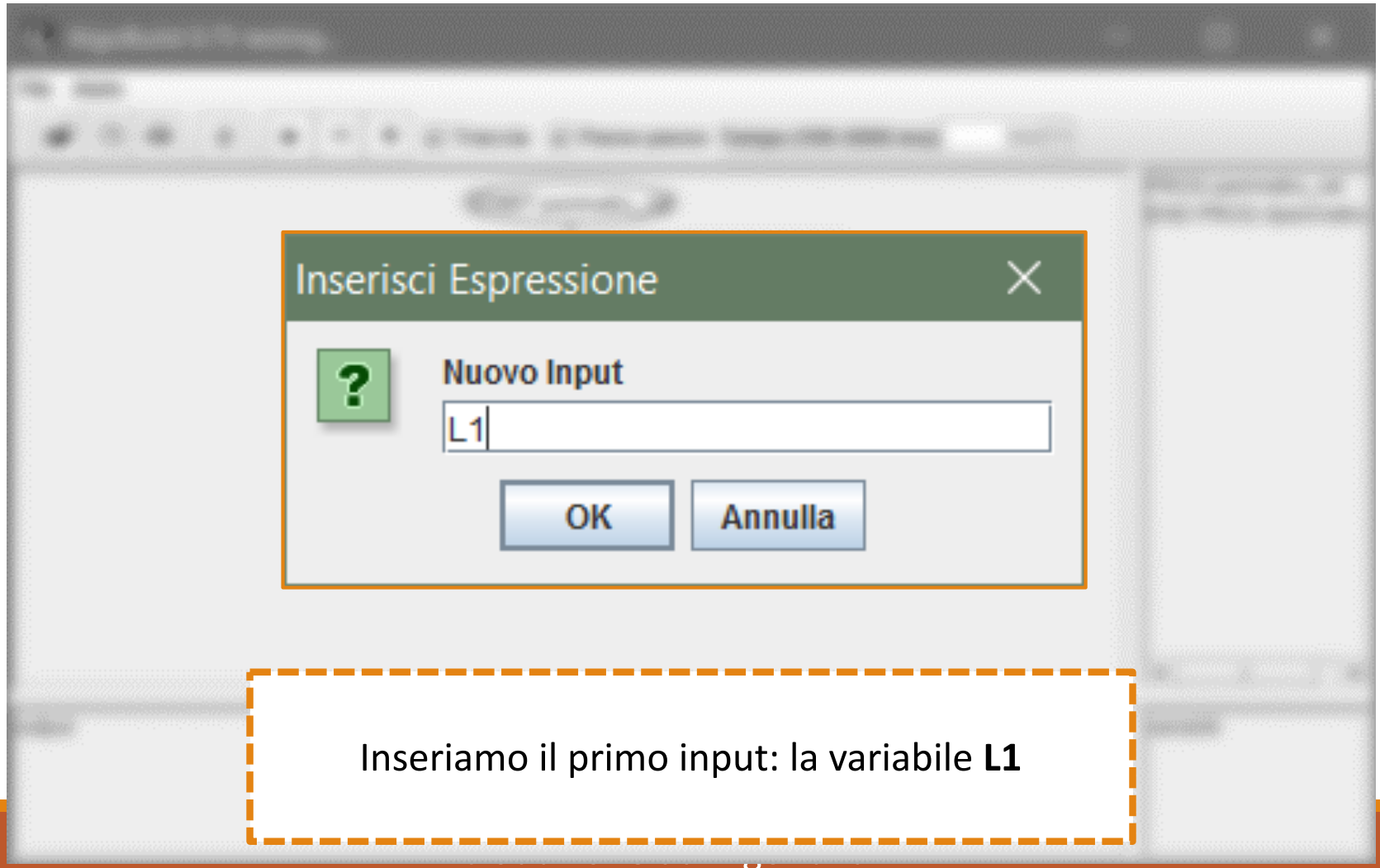
Esempio 2: Perimetro Rettangolo – 2/5



Esempio 2: Perimetro Rettangolo – 2/5

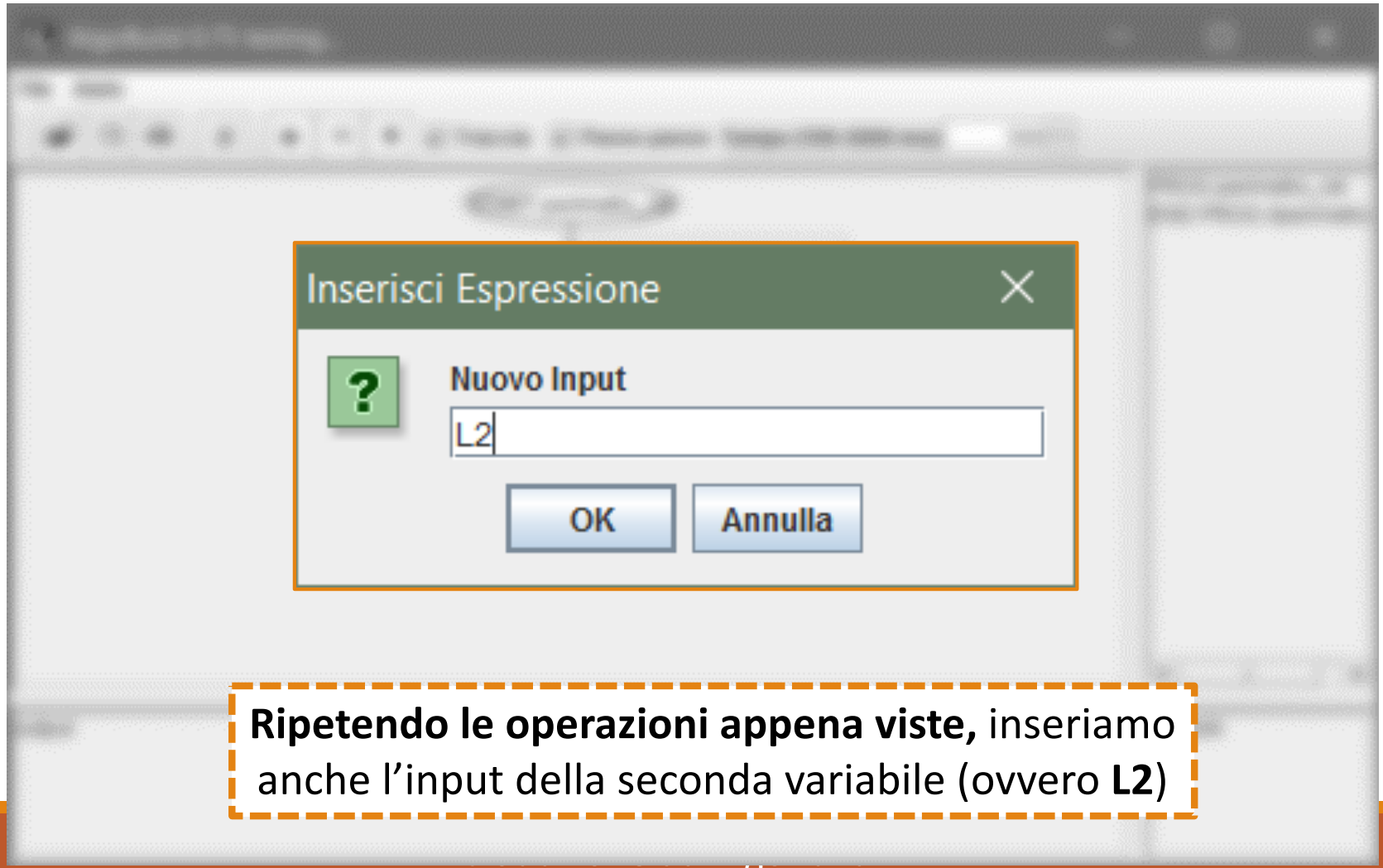


Esempio 2: Perimetro Rettangolo – 2/5



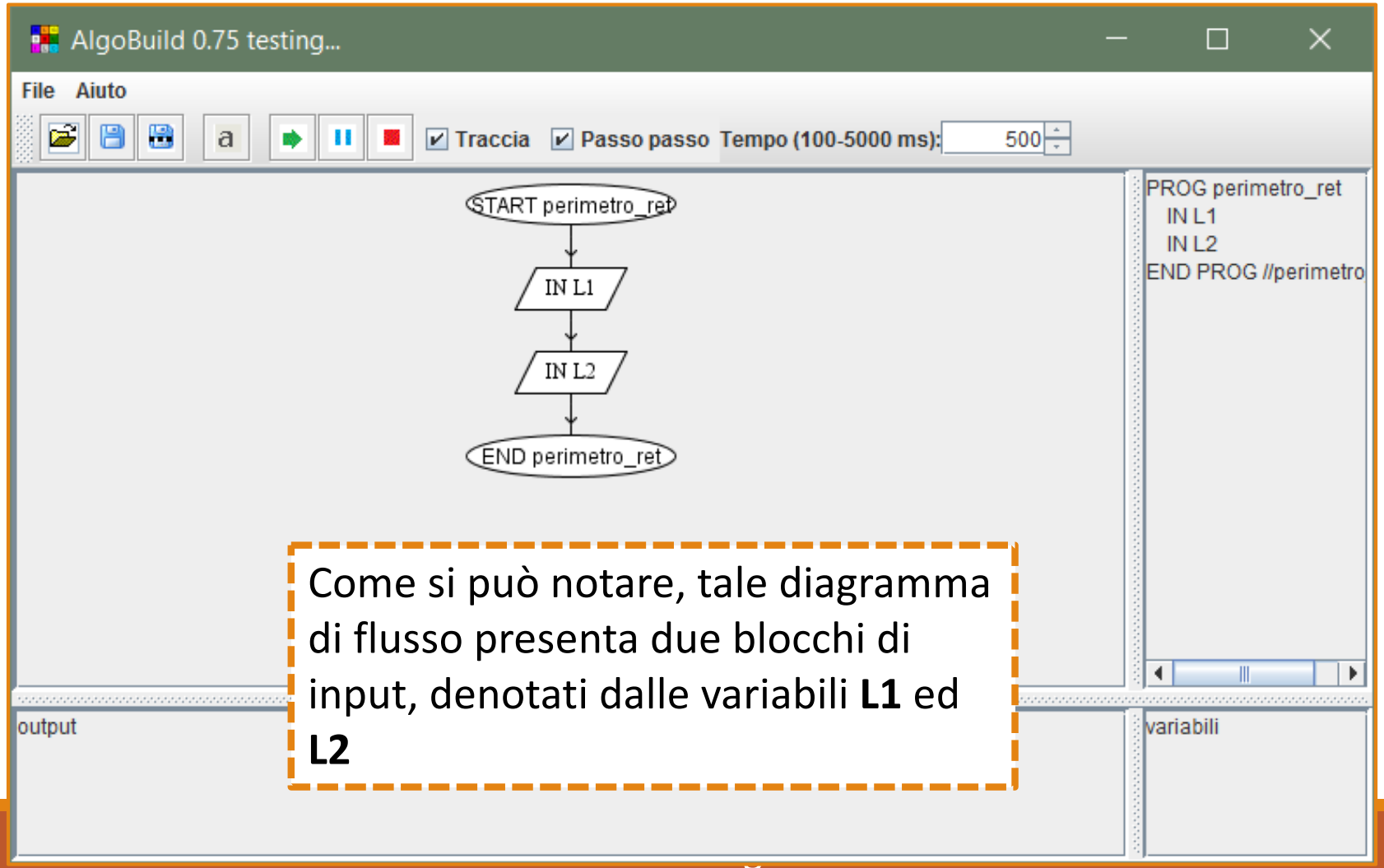
Inseriamo il primo input: la variabile **L1**

Esempio 2: Perimetro Rettangolo – 2/5



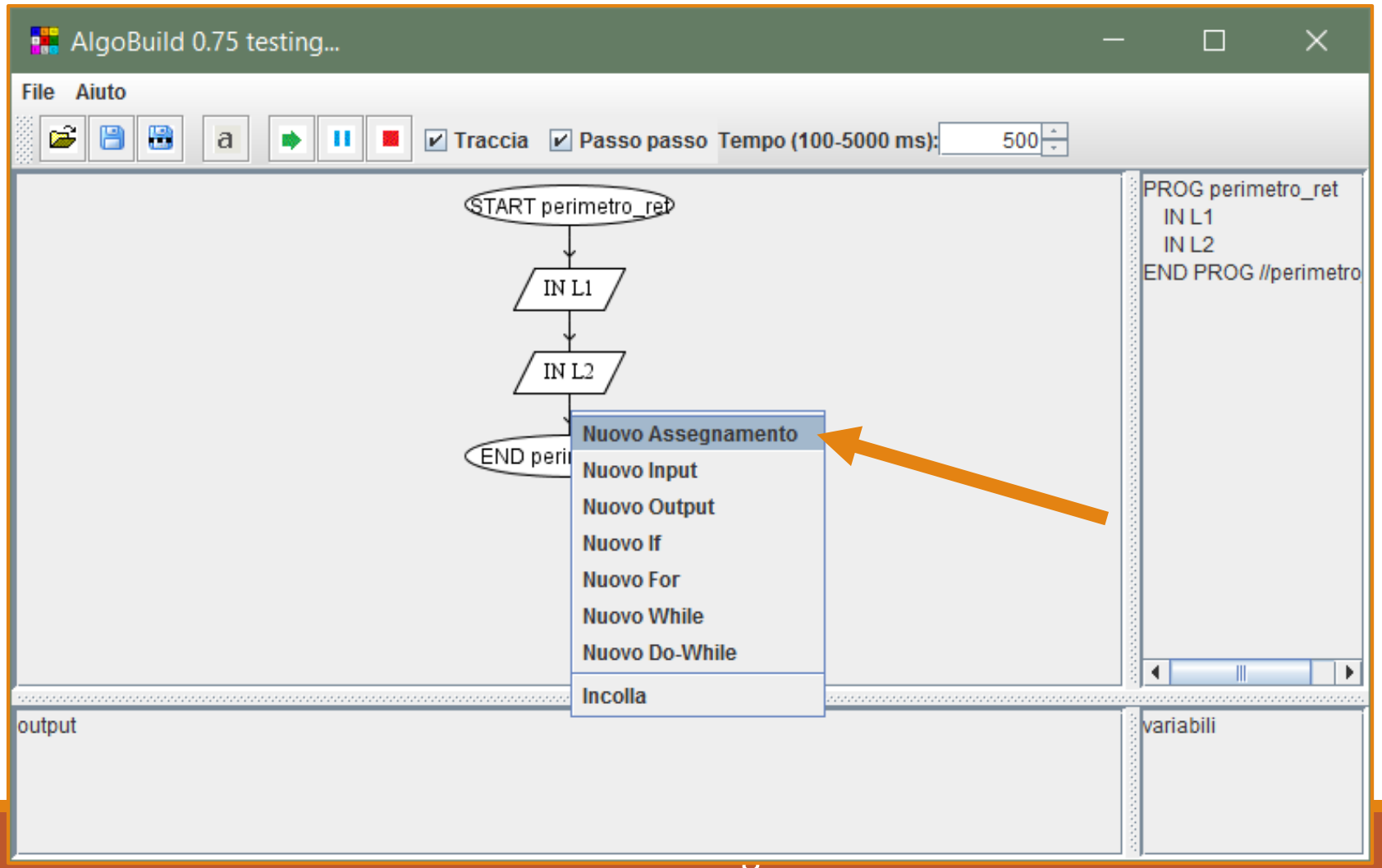
Ripetendo le operazioni appena viste, inseriamo anche l'input della seconda variabile (ovvero **L2**)

Esempio 2: Perimetro Rettangolo – 2/5

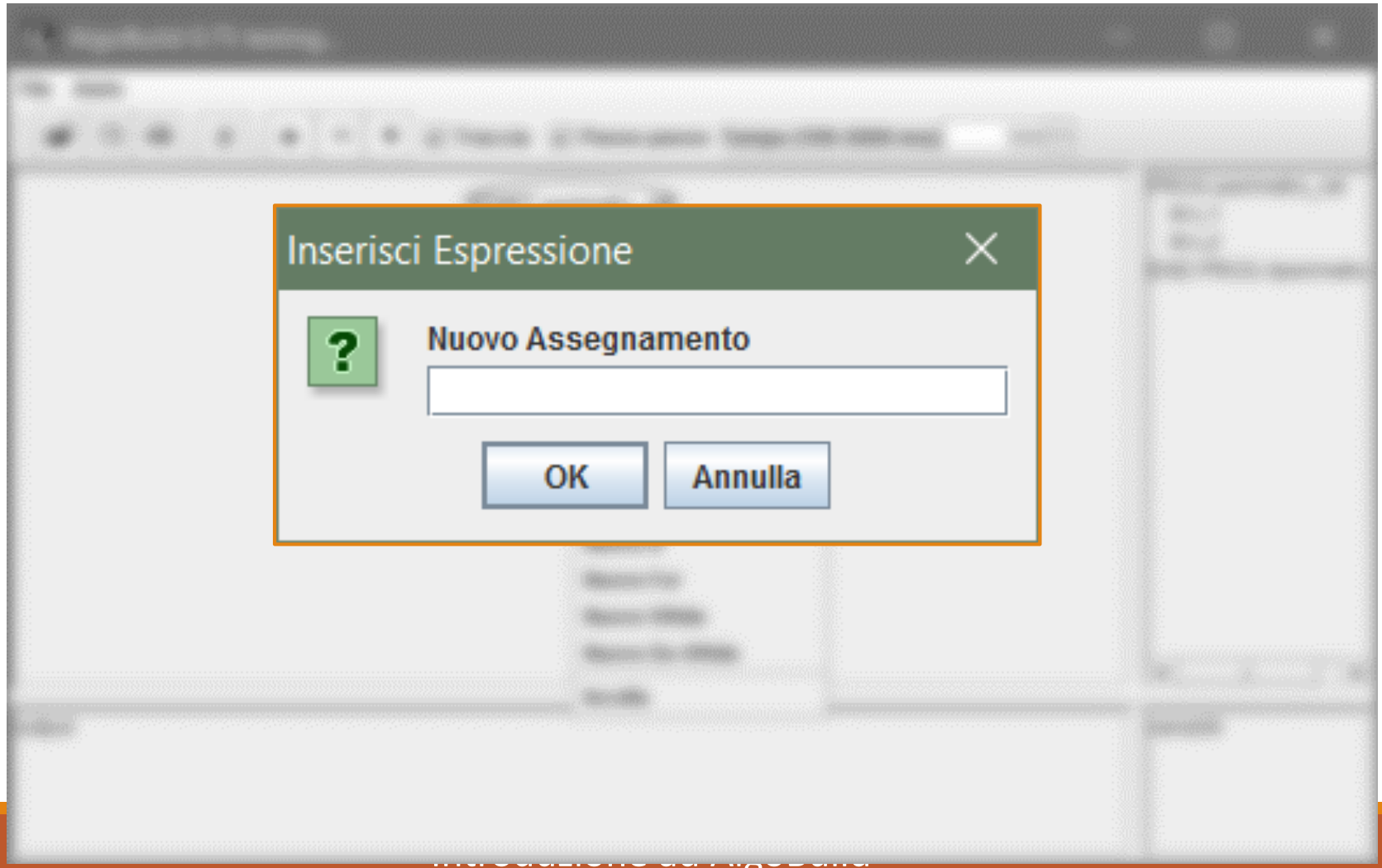


Come si può notare, tale diagramma di flusso presenta due blocchi di input, denotati dalle variabili **L1** ed **L2**

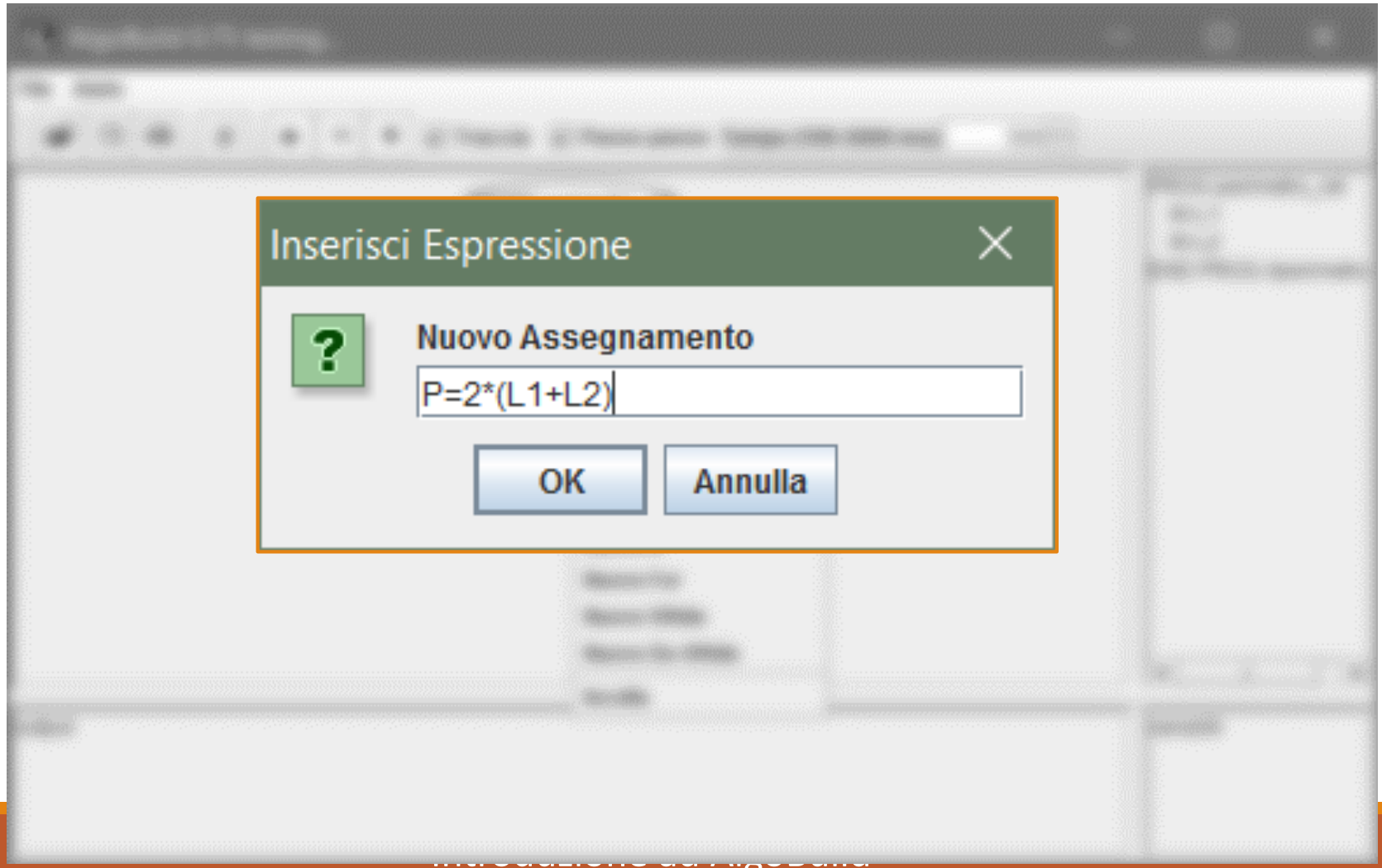
Esempio 2: Perimetro Rettangolo – 2/5



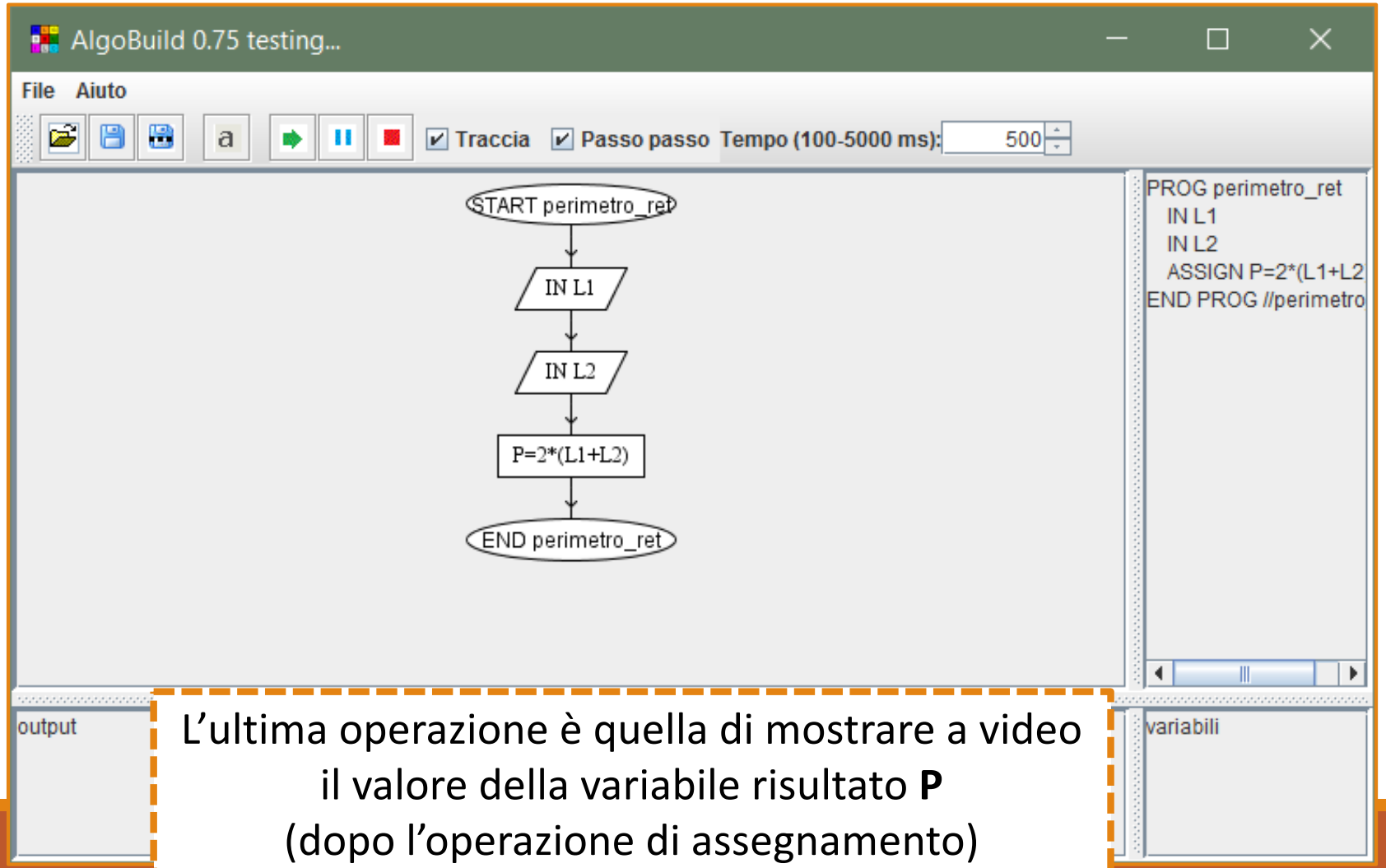
Esempio 2: Perimetro Rettangolo – 2/5



Esempio 2: Perimetro Rettangolo – 2/5

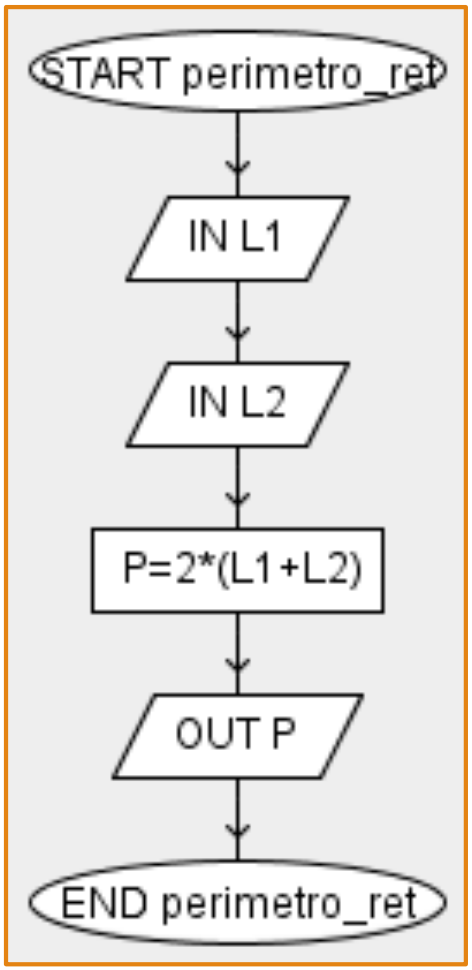


Esempio 2: Perimetro Rettangolo – 2/5



L'ultima operazione è quella di mostrare a video il valore della variabile risultato **P** (dopo l'operazione di assegnamento)

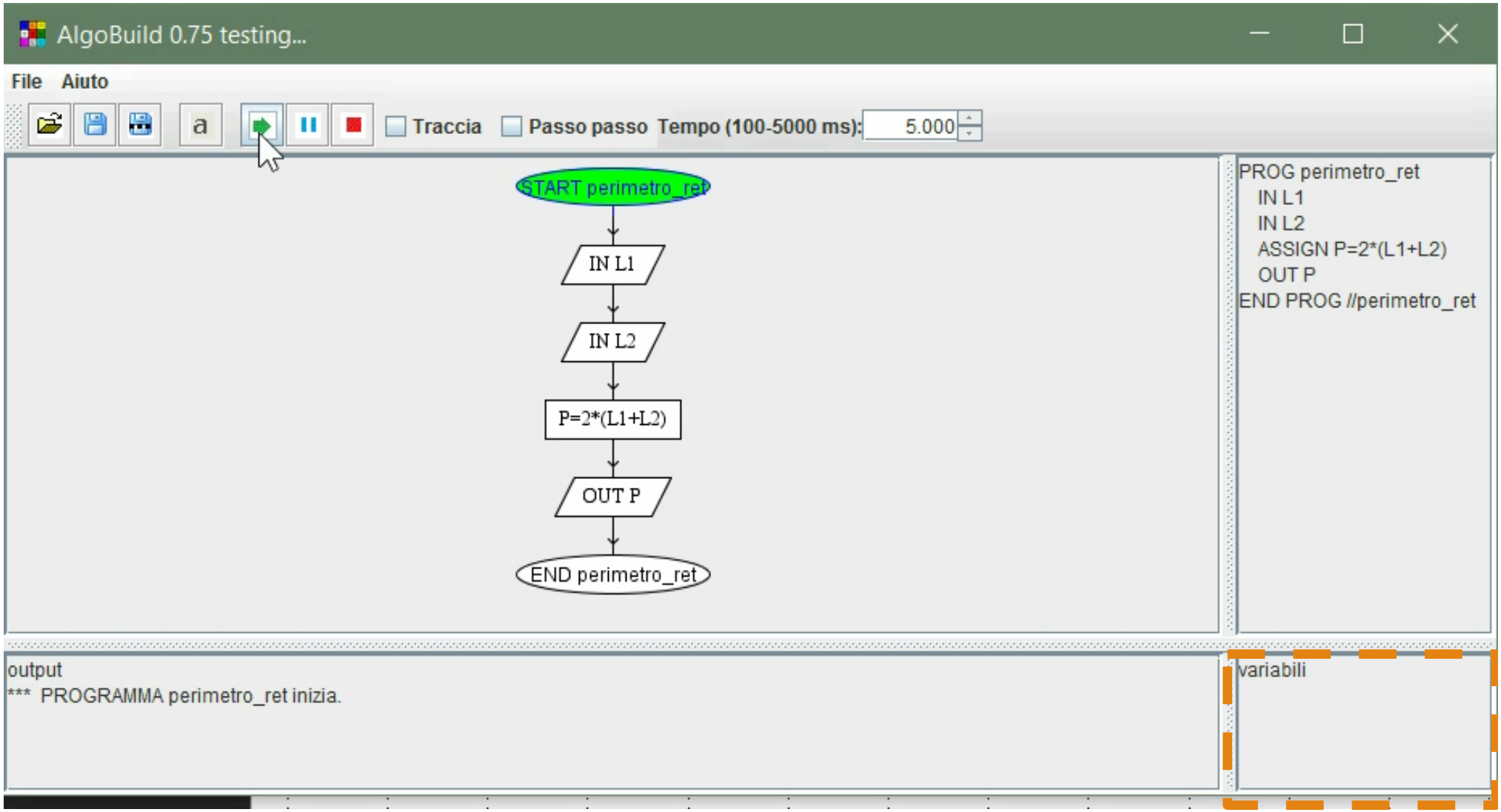
Esempio 2: Perimetro Rettangolo – 3/5



PSEUDO-CODICE

```
PROG perimetro_ret
   IN L1
   IN L2
   ASSIGN P=2*(L1+L2)
   OUT P
END PROG //perimetro_ret
```

DEMO Esecuzione *Perimetro Rettangolo* (Tempo passo: 5000ms, ovvero 5 secondi)



Esempio 2: Perimetro Rettangolo – 4/5

- Selezionando l'opzione **«Traccia»** verranno fornite (nel Pannello di Output) ulteriori informazioni riguardanti il flusso di esecuzione
  - Oltre ad *eventuali errori che possono intercorrere*

Esempio 2: Perimetro Rettangolo – 5/5

Selezionando *l'opzione* ***«Traccia»***

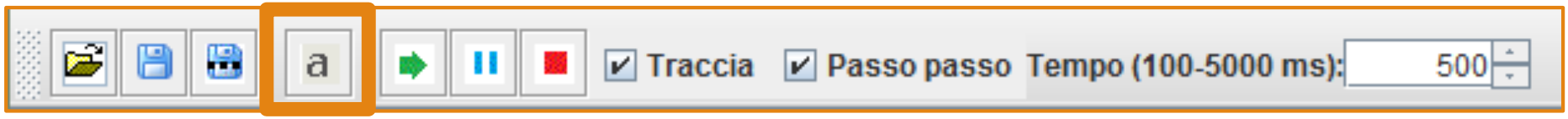
```
output
***  PROGRAMMA L2 inizia.
10
8
36.0
***  PROGRAMMA L2 termina.
```

Non selezionando *l'opzione* ***«Traccia»***

```
output
***  PROGRAMMA L2 inizia.
INPUT: L1=10
   var:  | L1=10.0 |
INPUT: L2=8
   var:  | L1=10.0 | L2=8.0 |
   ASSEGNA: P <- 36.0
   var:  | L1=10.0 | L2=8.0 | P=36.0 |
OUTPUT P: 36.0
***  PROGRAMMA L2 termina.
```

Altre Opzioni AlgoBuild – 1/3

- AlgoBuild permette di modificare le opzioni di visualizzazione del diagramma di flusso, permettendo di cambiare
  - Tipo di carattere
  - Dimensioni del carattere
  - Dimensioni del carattere nel pannello di output
  - Spessore delle linee relative al contorno dei blocchi ed agli archi orientati che collegano i blocchi (frecce)



- Cliccando sul tasto 

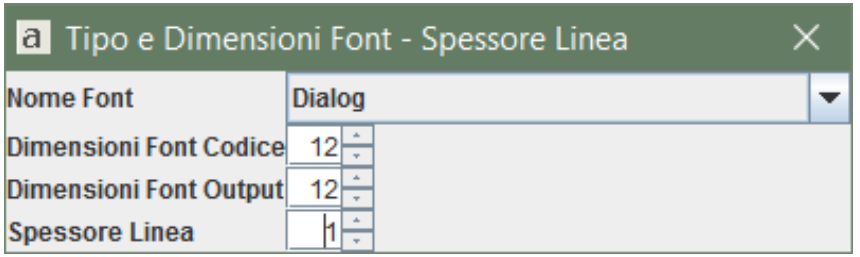
apparirà una finestra di dialogo che ci permetterà di vedere e modificare le suddette caratteristiche di visualizzazione

Altre Opzioni AlgoBuild – 1/3

- AlgoBuild permette di modificare le opzioni di visualizzazione del diagramma di flusso, permettendo di cambiare
  - Tipo di carattere
  - Dimensioni del carattere
  - Dimensioni del carattere nel pannello di output
  - Spessore delle linee relative al contorno dei blocchi ed agli archi orientati che collegano i blocchi (frecce)

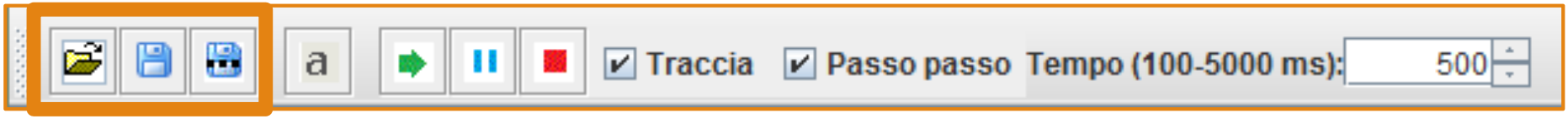
- Cliccando sul tasto  appari
  permetterà di vedere e modi
  di visualizzazione

Finestra di Dialogo
Modifica opzioni di visualizzazione

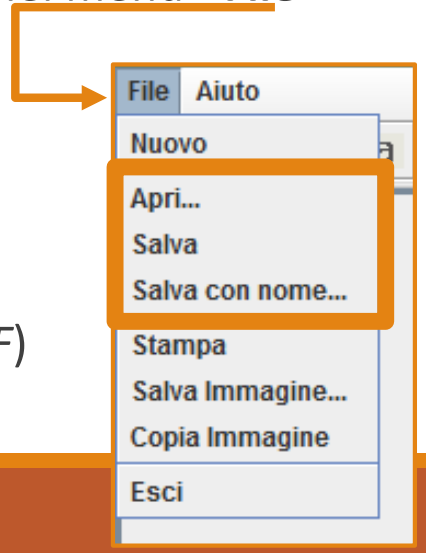


Altre Opzioni AlgoBuild – 2/3

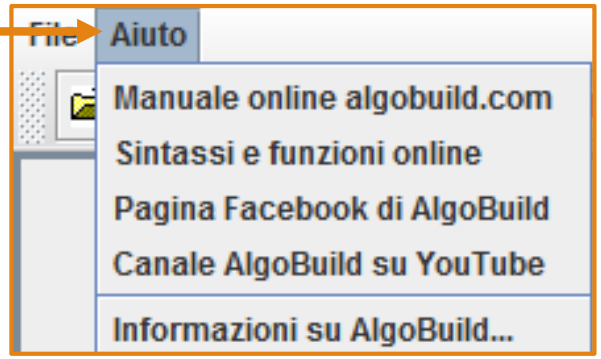
- È anche possibile ***salvare*** un diagramma di flusso, per poi ***aprirlo*** successivamente (ad esempio, per fare modifiche, per simulare altre esecuzioni, etc)
  - Le opzioni di salvataggio ed apertura sono presenti nella barra strumenti



  - Le opzioni di salvataggio/apertura sono presenti anche nel menu **«File»**
    - In particolare, sono presenti anche opzioni per
      - Stampare il diagramma di flusso
      - Copiare il diagramma di flusso negli appunti
        - Il diagramma verrà copiato come immagine
      - Salvare il diagramma come immagine (*JPG*, *PNG*, *GIF*)



Altre Opzioni AlgoBuild – 3/3



- Il menu **«Aiuto»** permette di accedere rapidamente a
  - Manuale AlgoBuild (online)
  - Sintassi e funzioni (online)
  - Pagina *Facebook ufficiale* e canale *Youtube* (con suggerimenti e video)
  - Informazioni sulla versione di AlgoBuild

Novità della nuova versione (1/5)

- Tipi di dati: ogni espressione e ogni variabile ha associato un tipo. I tipi sono int (numeri interi), double (numeri decimali a doppia precisione), string (sequenze di caratteri), boolean (logico true/false).
- Il modo di scrivere una espressione determina anche il suo tipo.
  - int: sono numeri scritti come sequenza di cifre senza il punto oppure espressioni algebriche contenenti SOLO int e variabili di tipo int.
  - double: sono numeri scritti come sequenza di cifre con il punto oppure espressioni algebriche contenenti ALMENO un double o una variabile di tipo double.
  - string: sono sequenze di caratteri racchiuse tra doppi apici. Esempi: "ciao", "tanti quanti", "torno\nsubito" (il simbolo \n inserisce un "a capo"). nome = "Mondo" messaggio = "Ciao " + nome + "!"
  - boolean: possono avere solo due valori true e false. condizione = true test = x>y

Novità della nuova versione (2/5)

- Operazioni sulle stringhe
  - "ciao" + "mondo" (concatenazione)
  - "ciao" == "mondo" (uguaglianza)
  - "ciao" != "mondo" (diverso da)
  - "ciao" > "mondo" (maggiore)
  - "ciao" > "mondo" (maggiore o uguale)
  - "ciao" < "mondo" (minore)
  - "ciao" <= "mondo" (minore o uguale)

Novità della nuova versione (3/5)

- Nuove funzioni di manipolazione stringhe / conversione tra tipi diversi
  - strlen(s) lunghezza della stringa s ritorna il numero (intero) di caratteri di s esempio: lung = strlen("ciao") restituisce 4
  - strchar(s,p) carattere dalla stringa s ritorna una stringa formata dal carattere di s alla posizione p la prima posizione è 0 esempio: ch = strchar("ciao",2) restituisce "a"
  - substr(s,i,j) sottostringa di s ritorna una sottostringa dalla posizione i alla j-1 la prima posizione è 0 esempio: nome = strchar("ciao Mondo!",5,10) restituisce "Mondo"
  - strpos(s,t) posizione della sottostringa ritorna il numero (intero) con la posizione di t in s la prima posizione è 0 esempio: pos = strpos("ciao Mondo!","Mondo") restituisce 5
  - strupr(s) converte in maiuscolo ritorna la stringa convertita in maiuscolo esempio: maiu = strupr("Ciao") restituisce "CIAO"
  - strlwr(s) converte in maiuscolo ritorna la stringa convertita in maiuscolo esempio: maiu = strlwr("Ciao") restituisce "ciao"
  - trim(s) elimina spazi iniziali e finali ritorna la stringa senza spazi a inizi / fine esempio: saluto = trim(" Ciao ") restituisce "Ciao"

Novità della nuova versione (4/5)

- ctoi(s) codice intero del carattere ritorna il numero (intero) codice ASCII del primo carattere di s esempio: cod = ctoi("Ciao") restituisce 67
- itoc(i) carattere dal codice intero ritorna il carattere corrispondente al codice ASCII (intero) passato esempio: char = itoc(65) restituisce "A"
- stoi(s) converte stringa in intero ritorna il numero (intero) convertito dalla stringa data esempio: val = stoi("12") restituisce il numero intero 12
- itos(i) rappresentazione in stringa del numero ritorna la stringa corrispondente al numero (intero) passato esempio: s = itos(65) restituisce "65"
- iformats(i,n) formattazione del numero in stringa ritorna la stringa lunga l corrispondente al numero (intero) passato con eventuali spazi iniziali esempio: s = itos(65,5) restituisce " 65"
- sbasetoi(s,b) converte stringa in intero espresso in una base numerica (da 2 a 36) ritorna il numero (intero) convertito dalla stringa nella base data esempio: val = stoi("12", 16) restituisce il numero intero 18
- ibasetos(i,b) rappresentazione in stringa del numero in una base numerica (da 2 a 36) ritorna la stringa corrispondente al numero (intero) passato nella base specificata esempio: s = itos(10,16) restituisce "A"
- stod(s) converte stringa in double ritorna il numero (double) convertito dalla stringa data esempio: val = stod("12.45") restituisce il numero double 12.45
- dtos(d) rappresentazione in stringa del numero ritorna la stringa corrispondente al numero (double) passato esempio: s = dtos(65.33) restituisce "65.33"

Novità della nuova versione (5/5)

- dformats(d,n,m) formattazione del numero double in stringa ritorna la stringa lunga n con m decimali corrispondente al numero (double) passato con eventuali spazi iniziali esempio: s = itos(65.1234,8,2) restituisce " 65.12"
- dtoi(d) conversione da double a intero ritorna il numero intero eliminando i decimali dal numero (double) passato esempio: i = dtoi(65.33) restituisce il numero intero 65
- type(v) informazioni sul tipo di dato ritorna una stringa con il nome del dato corrispondente esempi type("ciao") restituisce "(string)" type(123.456) restituisce "(double)" type(123) restituisce "(int)”
- Nuova sintassi di input:
  - x (legge un double)
  - double x (legge un double)
  - int x (legge un intero)
  - string x (legge una stringa)
  - v[i] (legge un elemento di un array double)
  - double v[i] (legge un elemento di un array double)
  - int v[i] (legge un elemento di un array intero)
  - string v[i] (legge un elemento di un array stringa)

Riepilogo

- Primo approccio ad AlgoBuild
- Utilizzo dei blocchi di
  - *Input*
  - *Output*
  - *Esecuzione/Assegnamento*
- Simulazione di esecuzione, mediante AlgoBuild