Progettare la Logica: L’Architettura degli Algoritmi

Dal problema al codice attraverso gli schemi di flusso.

Corso di Fondamenti di Informatica

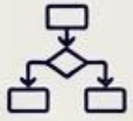
Il Processo di Formalizzazione

1. Analisi

Individuare input, output e relazioni.

2. Sviluppo

Costruire l'algoritmo risolutore.



3. Simulazione

Testare con dati fittizi.

4. Codifica

Tradurre in linguaggio macchina.

5. Programma

Eseguire le istruzioni.

L'elaboratore non pensa. Il programmatore “trasmette” il ragionamento sotto forma di algoritmo.

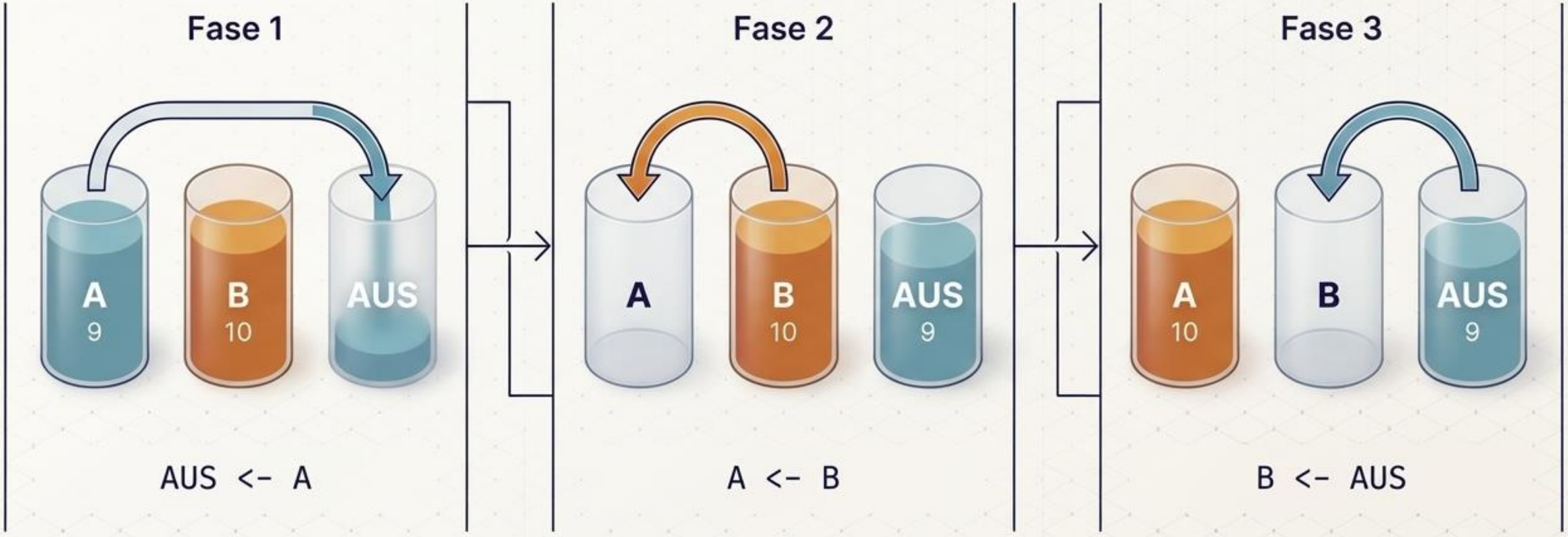
La Memoria: Il Concetto di Variabile

Fase 1: Vuota	Fase 2: Assegnazione	Fase 3: Sovrascrittura
A inizio esecuzione, il contenitore è vuoto.	L'operazione di assegnazione (`A <- 9`) inserisce un valore.	`A <- A + 1`. Il vecchio valore va perso, sostituito da quello nuovo.

Una variabile ha un **identificatore** (nome univoco) e un **valore** che cambia durante l'esecuzione.

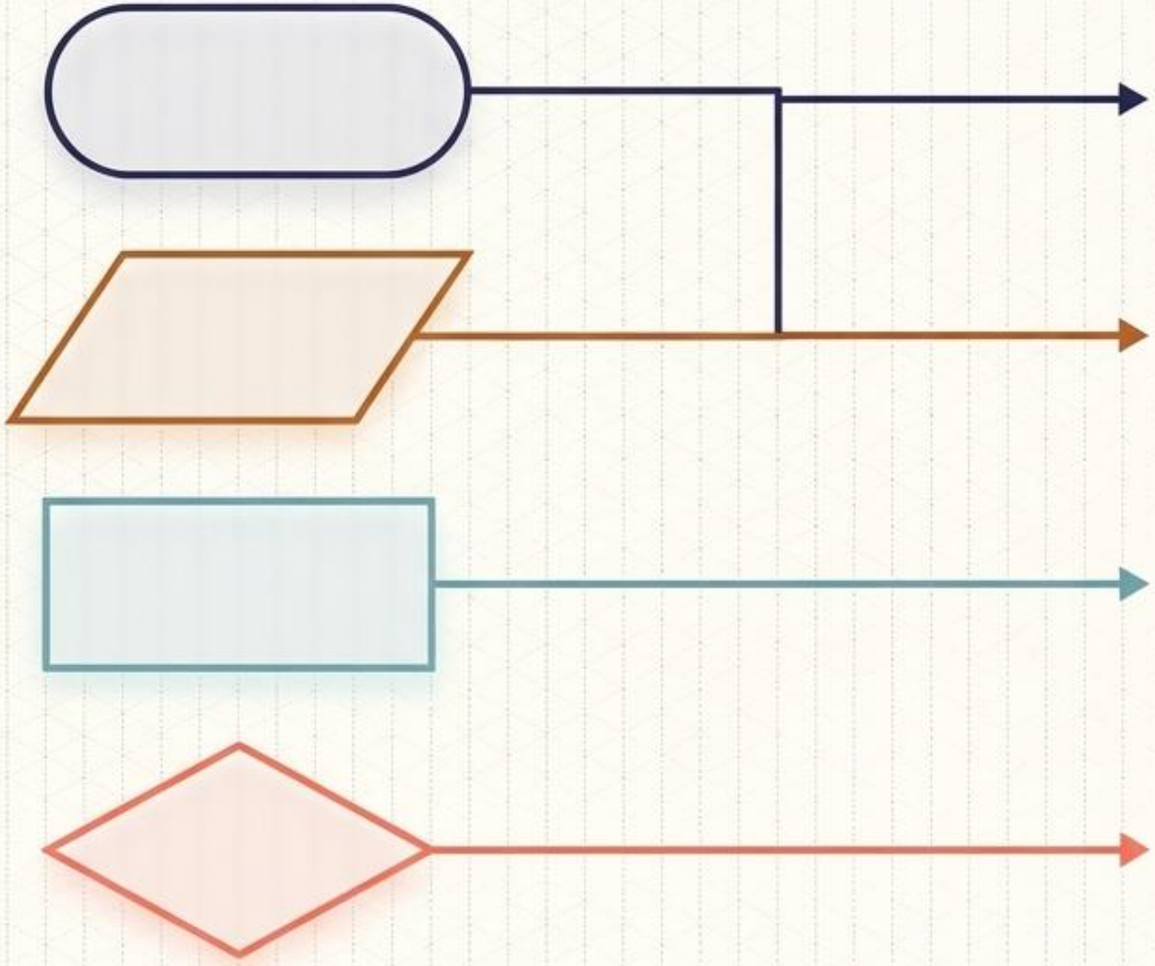
Architettura dei Dati: Lo Scambio di Valori

Come scambiare il contenuto di due variabili (A e B) senza perdere i dati?



L'introduzione di una variabile ausiliaria (AUS) previene la sovrascrittura accidentale dei dati di partenza.

Gli Schemi di Flusso: Il Linguaggio Universale



Ovale (Inizio/Fine):
Delimita l'algoritmo.

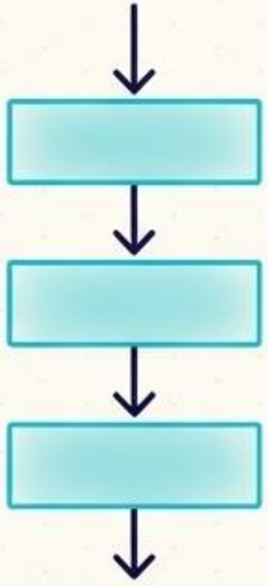
Parallelogramma (I/O):
Comunicazione con l'esterno.
Chiede o fornisce valori.

Rettangolo (Azione):
Blocco di elaborazione (es. calcoli matematici, assegnazioni).

Rombo (Controllo):
Valuta una condizione. Il flusso si divide in due rami (Vero/Falso).

Schemi di Composizione Fondamentali

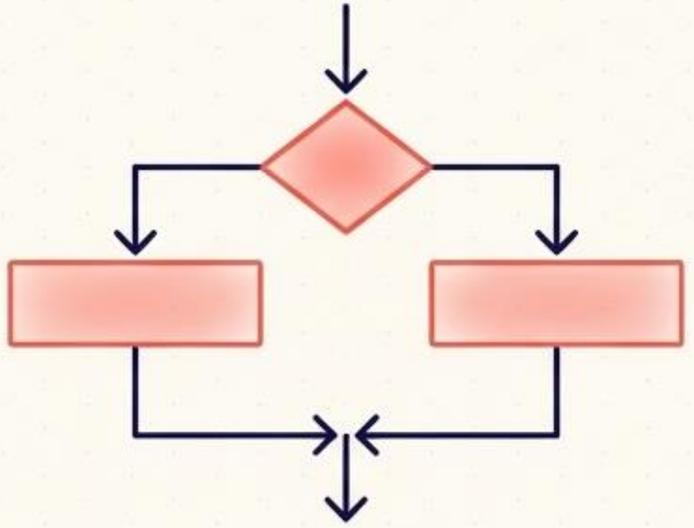
1. Sequenza



Istruzioni eseguite in successione, una dopo l'altra.

Analogia: *Seguire una ricetta passo dopo passo.*

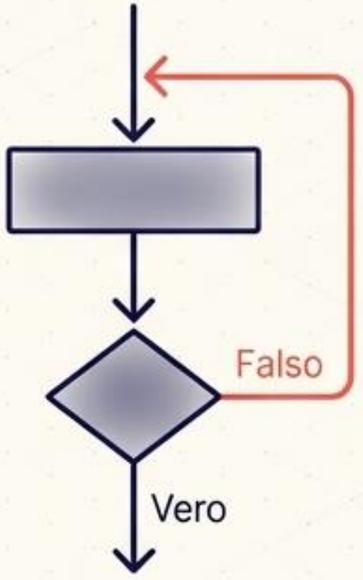
2. Selezione



Scelta del percorso basata su una condizione (Vero/Falso).

Analogia: *Prendere l'ombrello solo se piove.*

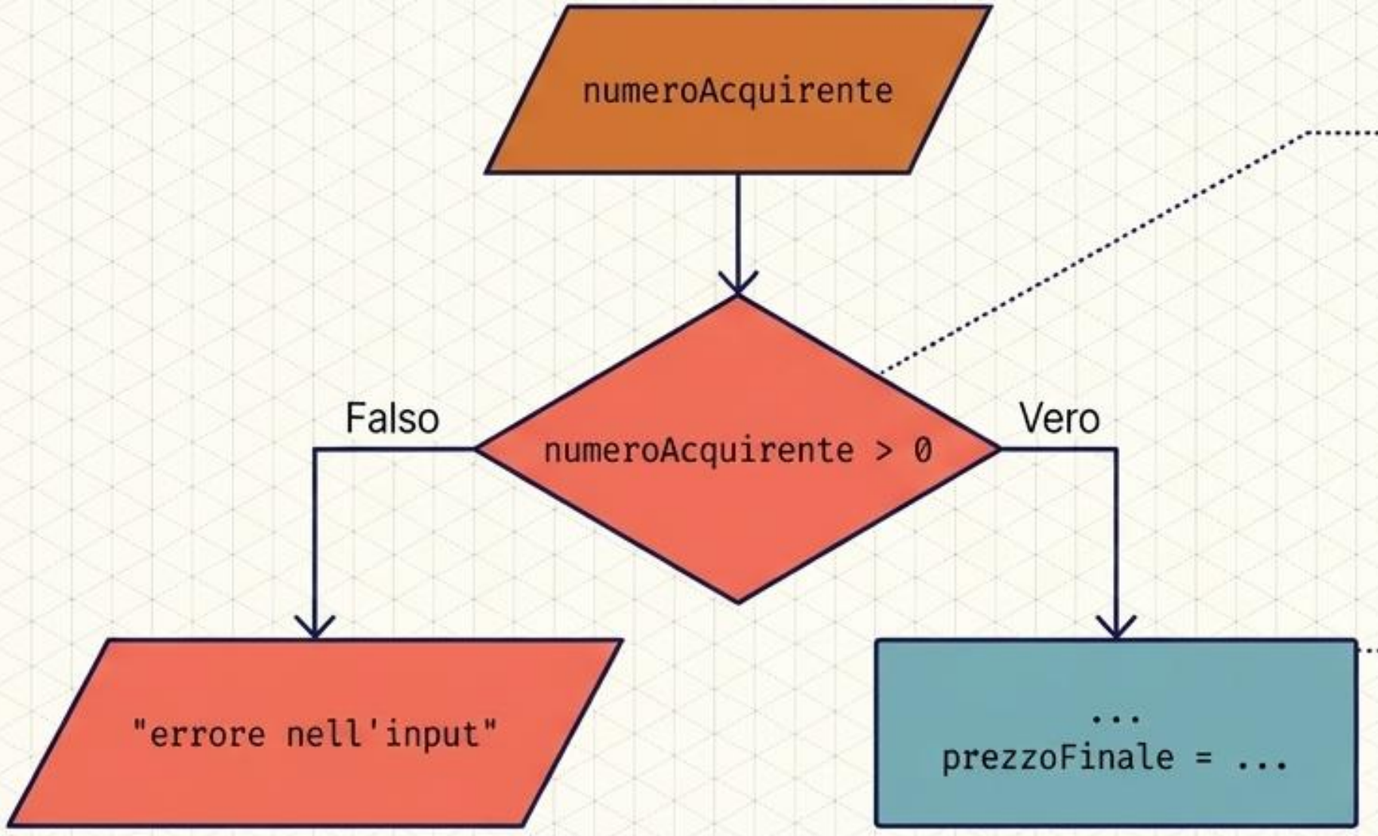
3. Ripetizione (Ciclo)



Ripetizione di azioni fino al verificarsi di una condizione.

Analogia: *Mescolare il caffè finché lo zucchero non è sciolto.*

La Selezione e i Vincoli di Integrità



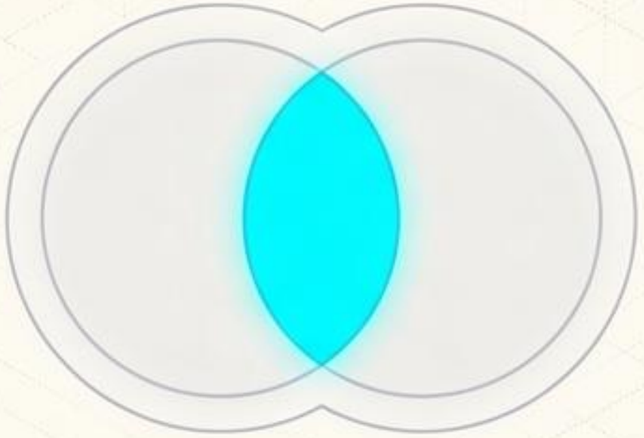
Vincoli di integrità

Condizioni che i dati devono rispettare per essere accettati dal programma (es. un numero di studenti non può essere negativo).

Controlli Annidati

Inserire un blocco di selezione dentro l'altro permette di gestire scenari complessi (es. fasce di sconto in cascata a seconda dell'input).

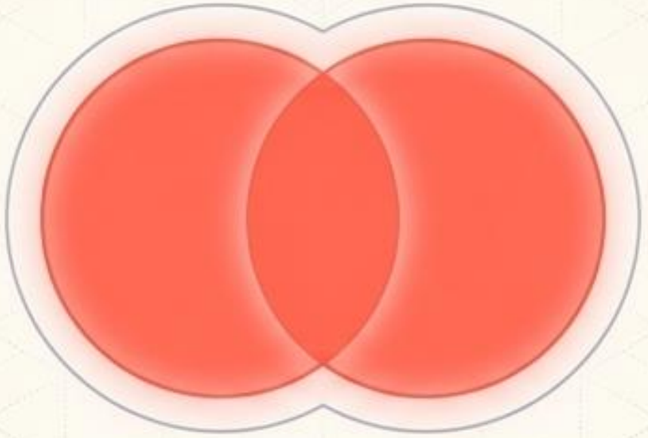
Operatori Booleani: Fondere le Condizioni



AND

Risultato Vero solo se entrambe le condizioni di partenza sono Vere.

```
dividendo >= 0 AND divisore > 0
```



OR

Risultato Vero se almeno una delle condizioni è Vera.

NOT

Inverte il risultato. Un Falso diventa Vero e viceversa.

Combinare le condizioni ottimizza l'algoritmo, riducendo drasticamente i blocchi di controllo annidati.

Strumenti del Ciclo: Contatori vs. Accumulatori

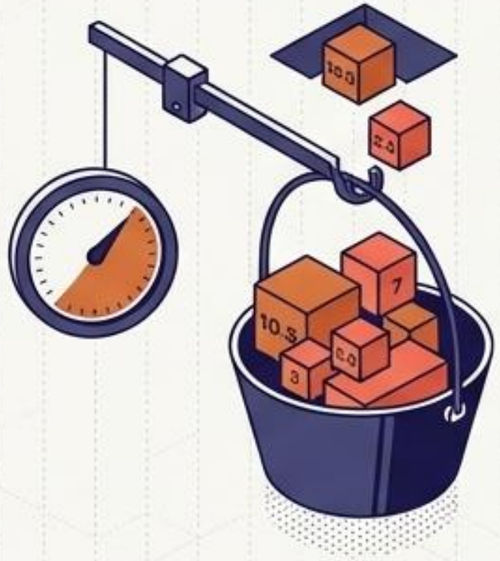
Il Contatore

```
contatore ← contatore + 1
```

(or “+ costante")

Conta il numero di volte in cui si svolge un evento o un’iterazione. Cresce a step fissi e costanti.

L’Accumulatore (Somme Successive)

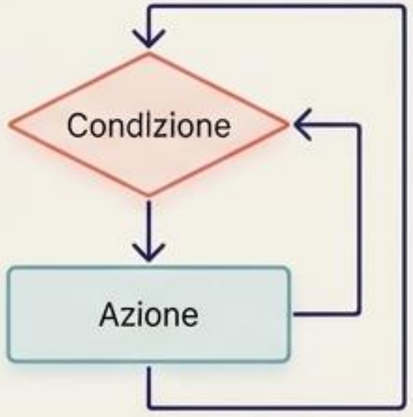


```
somma ← somma + x
```

(x variable in ingresso)

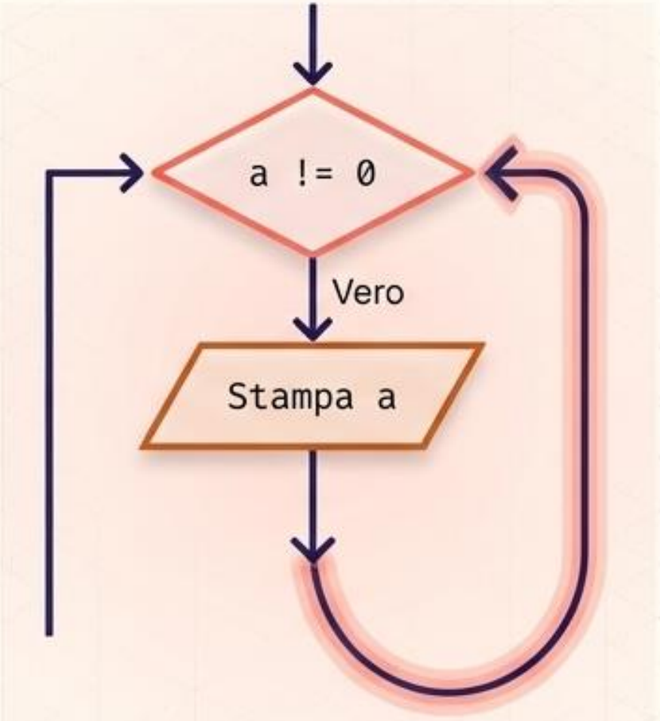
Somma valori progressivi tra loro. Appoggia il nuovo valore variabile al totale parziale calcolato in precedenza (es. scontrino della spesa).

Diagnostica dei Cicli: Pre vs. Post-Condizionale

Ciclo Precondizionale (While)		Ciclo Postcondizionale (Do-While)	
			
Controllo:	A inizio ciclo.	**Controllo:**	A fine ciclo.
Esecuzioni Minime:	0. Il nucleo potrebbe non essere mai eseguito se la condizione iniziale è subito falsa.	**Esecuzioni Minime:**	1. Il nucleo viene eseguito sempre almeno una volta prima della verifica.
Caso d'uso:	Divisione per sottrazioni successive (se il dividendo è già minore del divisore, il ciclo non parte affatto).	**Caso d'uso:**	Indovinare un numero segreto o forzare l'utente a inserire un input corretto (il primo tentativo avviene sempre).

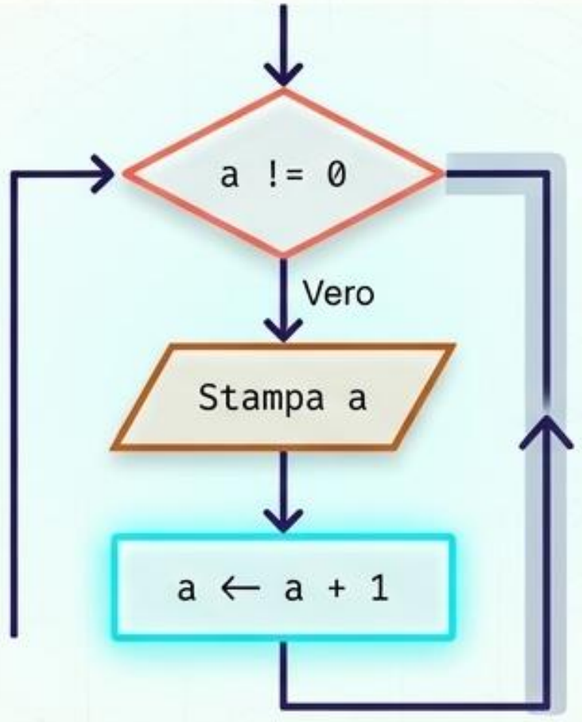
La Proprietà di Finitezza (La Trappola del Loop Infinito)

Errore: Loop Infinito



L'algoritmo non termina mai. Manca l'istruzione interna che altera la condizione di controllo. Il programma è bloccato.

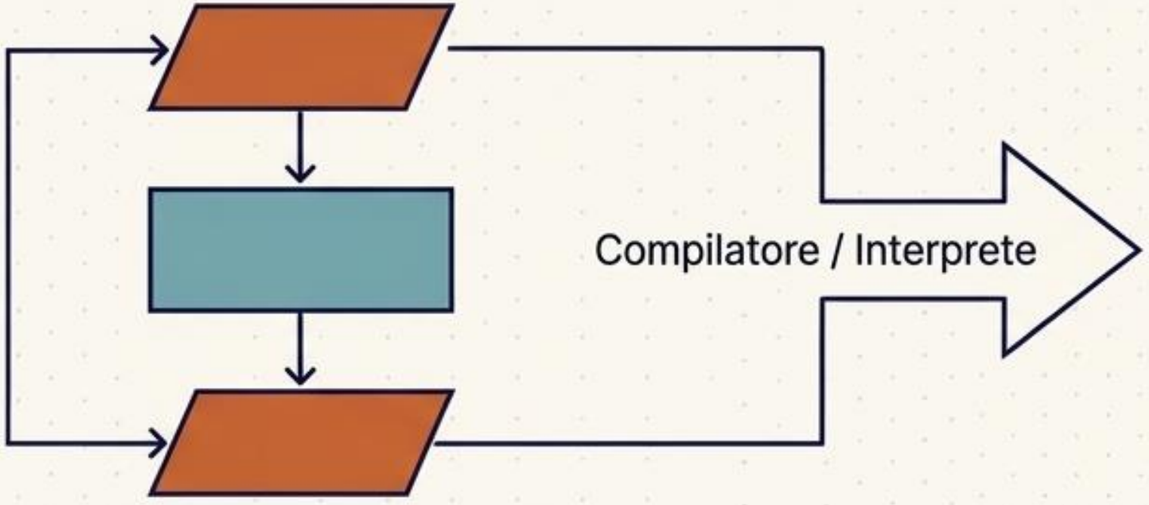
Corretto: La Via d'Uscita



Un algoritmo gode della proprietà di finitezza solo quando presenta un inizio, una fine, e un meccanismo certo che garantisce l'uscita da ogni ciclo.

La Codifica: Dal Disegno al Linguaggio Macchina

Il Progetto



La Traduzione

Linguaggio Python

```python
media = (valore1 + valore2 + valore3) / 3
print(media)
```

Linguaggio C++

```cpp
media = (valore1 + valore2 + valore3) / 3;
cout << media;
```

L'ultimo passo è l'Editor. Gli schemi di flusso vengono tradotti nel codice sorgente. Un programma traduttore converte poi questa sintassi in istruzioni binarie per il microprocessore.

L'architettura logica rimane invariata; cambia solo il linguaggio. Il progetto è ora un programma.