



Risoluzione automatica di problemi: gli Algoritmi

Cos'è l'INFORMATICA ??

Il termine "*informatica*" ha un' accezione molto ampia.

Esistono varie definizioni:

- *l'informatica è la scienza che si occupa della conservazione, dell'elaborazione e della rappresentazione dell'informazione.*
- “Scienza dei calcolatori elettronici” (Computer Science)
- “Scienza dell’informazione”
- ...

→ Definizione proposta nell'ambito di questo corso:

“Scienza della rappresentazione e dell’elaborazione automatica dell’informazione.”

Informatica

Informazione:

e` tutto ciò che può essere **rappresentato** all'interno di un computer è informazione:

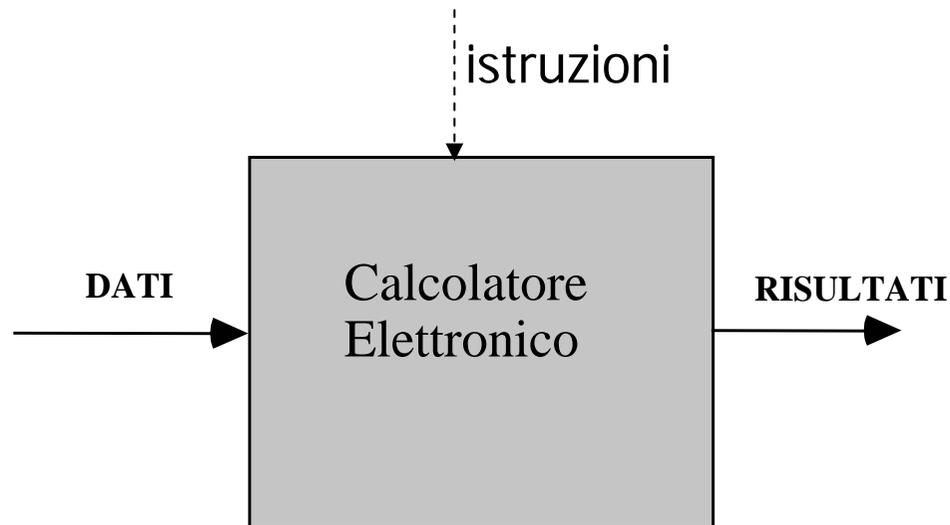
- Numeri
 - Caratteri, parole e testi
 - Immagini
 - Suoni
 - Filmati
 - comandi (istruzioni) e sequenze di comandi (programmi) che il calcolatore deve eseguire
- Le modalita` di **rappresentazione** dipendono dalle caratteristiche dell'elaboratore.

Elaboratore Elettronico (computer):

e` lo strumento per la rappresentazione e l'elaborazione delle informazioni.

Programmazione

- È l'attività con cui si predispongono l'elaboratore ad eseguire un particolare insieme di azioni su particolari informazioni (*dati*), allo scopo di risolvere un certo problema.



I Problemi da risolvere

I problemi che siamo interessati a risolvere con l'elaboratore sono di natura molto varia. Ad esempio:

- *Somma di due numeri interi*
- *Dati a e b , risolvere l'equazione $ax+b=0$*
- *Calcolare il massimo comun divisore fra due numeri dati.*
- *Dato un'insieme di parole, metterle in ordine alfabetico.*
- *Calcolare l'intersezione di due insiemi.*
- *Dato un elenco di nomi e relativi numeri di telefono trovare il numero di telefono di una determinata persona*
- *Dati gli archivi dei dipendenti di un'azienda, calcolare lo stipendio medio del dipendente dell'azienda.*
- *Data l'immagine satellitare di un'area geografica, calcolare le previsioni del tempo per domani.*
- ...

Risoluzione dei Problemi

- La descrizione del problema non fornisce (in genere) un metodo per calcolare il risultato.
- Non tutti i problemi sono risolvibili attraverso l'uso del calcolatore. In particolare esistono classi di problemi per le quali la soluzione automatica non è proponibile.
- **Ad esempio:**
 - se il problema presenta infinite soluzioni
 - per alcuni dei problemi **non è stato trovato** un metodo risolutivo.
 - per alcuni problemi è stato dimostrato che **non esiste** un metodo risolutivo automatizzabile

Noi ci concentreremo sui problemi che, ragionevolmente, ammettono un metodo risolutivo (esprimibile mediante una ***funzione calcolabile***).

Risoluzione di un problema

Con questo termine si indica il processo che:

- dato un problema, e
- individuato un metodo risolutivo

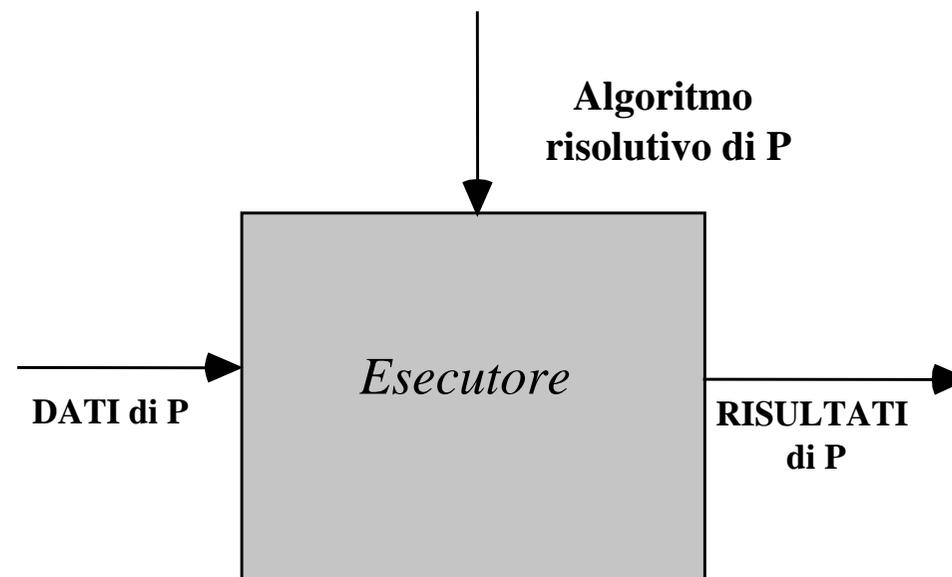
trasforma i dati iniziali nei corrispondenti risultati finali.

- Affinche` la risoluzione di un problema possa essere realizzata attraverso l'uso del calcolatore, tale processo deve poter essere definito come sequenza di **azioni elementari**, esprimibili mediante **istruzioni**.

ALGORITMO

e` l'insieme ordinato delle azioni che risolve un dato problema P.

- l'algoritmo descrive un metodo risolutivo attraverso un insieme ordinato di azioni.
- l'esecuzione dell'algoritmo e` affidata ad un generico "***esecutore***", cioe` una macchina astratta (non necessariamente un calcolatore !) in grado di interpretare ed eseguire ogni azione specificata nell'ordine indicato.



Esecutore e istruzioni primitive

- Ad un generico esecutore e` implicitamente associato un insieme di istruzioni primitive (set di istruzioni):
 - sono le sole istruzioni che e` in grado di interpretare ed eseguire.

Esempio: la preparazione del caffè`

Esecutore: essere umano corredato di caffettiera "moka", cucina a gas e macina-caffè`;

Algoritmo:

1. **svitare** la caffettiera;
2. se si dispone di caffè` macinato:
 - **riempire** il filtro con il caffè` macinato,
 - altrimenti se si dispone di caffè` in chicchi:
 - **macinarlo** e ripetere il punto 2;
 - altrimenti **terminare** (il caffè` non si puo` fare..).
3. **riempire** la parte inferiore della caffettiera con acqua;
4. **inserire** il filtro nella macchina;
5. **avvitare** la caffettiera;
6. **accendere** il fuoco a gas;
7. **collocare** la moka sul fuoco;
8. **attendere** l'uscita del caffè`;
9. **spegnere** il fuoco;
10. **fine** (il caffè` e` pronto).

Esempio: la preparazione del caffè`

Esecutore: essere umano corredato di caffettiera "moka", cucina a gas e macina-caffè`;

Set di istruzioni:

- operazioni fondamentali sulla caffettiera:
 - svitare
 - avvitare
 - riempire il filtro
 - riempire con acqua
- operazioni fondamentali sulla cucina a gas:
 - accendere
 - spegnere
- operazioni fondamentali sul macina-caffè`:
 - macinare
- altre operazioni:
 - verifica di condizioni
 - ripetizione di operazioni
 - attesa
 - ..

Proprietà fondamentali dell'Algoritmo

1. **Eseguibilità**: ogni "istruzione" deve essere eseguibile da parte dell'esecutore dell'algoritmo;
 2. **Non Ambiguità**: ogni istruzione deve essere univocamente interpretabile dall'esecutore
 3. **Finitezza**: il numero totale di azioni da eseguire, per ogni insieme di dati di ingresso, è finito.
- se almeno una delle 3 proprietà non è soddisfatta, la sequenza non è un algoritmo.

Altre proprietà desiderabili:

- **generalità**: corretto funzionamento dell'algoritmo anche variando alcuni aspetti del problema (ad esempio, la dimensione dell'insieme dei dati, il tipo dei dati, ecc.)
- **efficienza**: tanto minore è il numero di azioni eseguite per la risoluzione del problema, tanto maggiore è l'efficienza.
- **determinismo**: possibilità di prevedere esattamente prima dell'esecuzione la sequenza di azioni che verranno eseguite, per ogni insieme di dati.
- ...

Algoritmi e Programmi

- Se l'esecutore e` un **elaboratore elettronico**:
 1. e` necessario conoscere l'insieme di istruzioni che e` in grado di interpretare
 2. e` necessario conoscere quali tipi di informazioni (dati) e` in grado di rappresentare

Gli aspetti 1. e 2. sono peculiari del **formalismo** scelto per esprimere l'algoritmo all'interno del sistema di elaborazione, cioe` del

Linguaggio di Programmazione

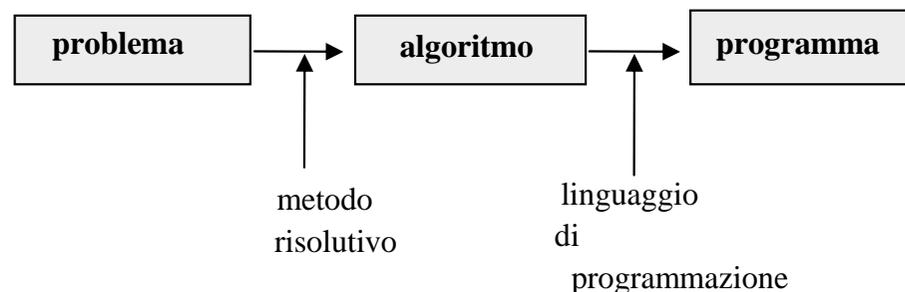
Algoritmi e Programmi

Quindi:

Dato un problema P, la sua soluzione puo` essere ottenuta mediante l'uso del calcolatore, compiendo i seguenti passi:

1. individuazione di un **metodo risolutivo**
2. scomposizione del procedimento in insieme ordinato di azioni: **algoritmo**
3. rappresentazione dei dati e dell'algoritmo attraverso un formalismo comprensibile per l'elaboratore (il linguaggio di programmazione).

Si ottiene così il **PROGRAMMA**, che potra` essere eseguito dall'elaboratore per risolvere automaticamente ogni istanza del problema P.



Algoritmi equivalenti

Due algoritmi si dicono equivalenti quando:

- hanno lo stesso dominio dei dati (dominio di ingresso);
- hanno lo stesso dominio dei risultati (dominio di uscita);
- in corrispondenza degli stessi valori nel dominio di ingresso producono gli stessi valori nel dominio di uscita

Due algoritmi equivalenti:

- forniscono lo stesso risultato
- possono essere profondamente diversi
- possono avere differente efficienza

Algoritmi Equivalenti: Calcolo del massimo comun divisore

Dati due interi m ed n , calcolare il massimo comune divisore di essi.

Algoritmo a:

1. Calcola l'insieme I dei divisori di m
2. Calcola l'insieme J dei divisori di n
3. Calcola l'insieme K dei divisori comuni:

$$K = I \cap J$$

4. Calcola il massimo in K : questo e' il risultato

Algoritmi Equivalenti:

Calcolo del massimo comun divisore

Algoritmo b: si basa sul *metodo di Euclide*: detta *mcd* la funzione che calcola la soluzione del problema, la sua definizione è data come segue:

- $\text{mcd}(m,n) = m$ (oppure n) se $m=n$
- $\text{mcd}(m,n) = \text{mcd}(m-n, n)$ se $m>n$
- $\text{mcd}(m,n) = \text{mcd}(m, n-m)$ se $m<n$

Quindi l'algoritmo b si può esprimere così:

1. Finché m è diverso da n ripeti le seguenti azioni:
 - se $m>n$ sostituisci a m il valore $(m-n)$
 - altrimenti sostituisci a n il valore $(n-m)$
2. Il massimo comun divisore è n

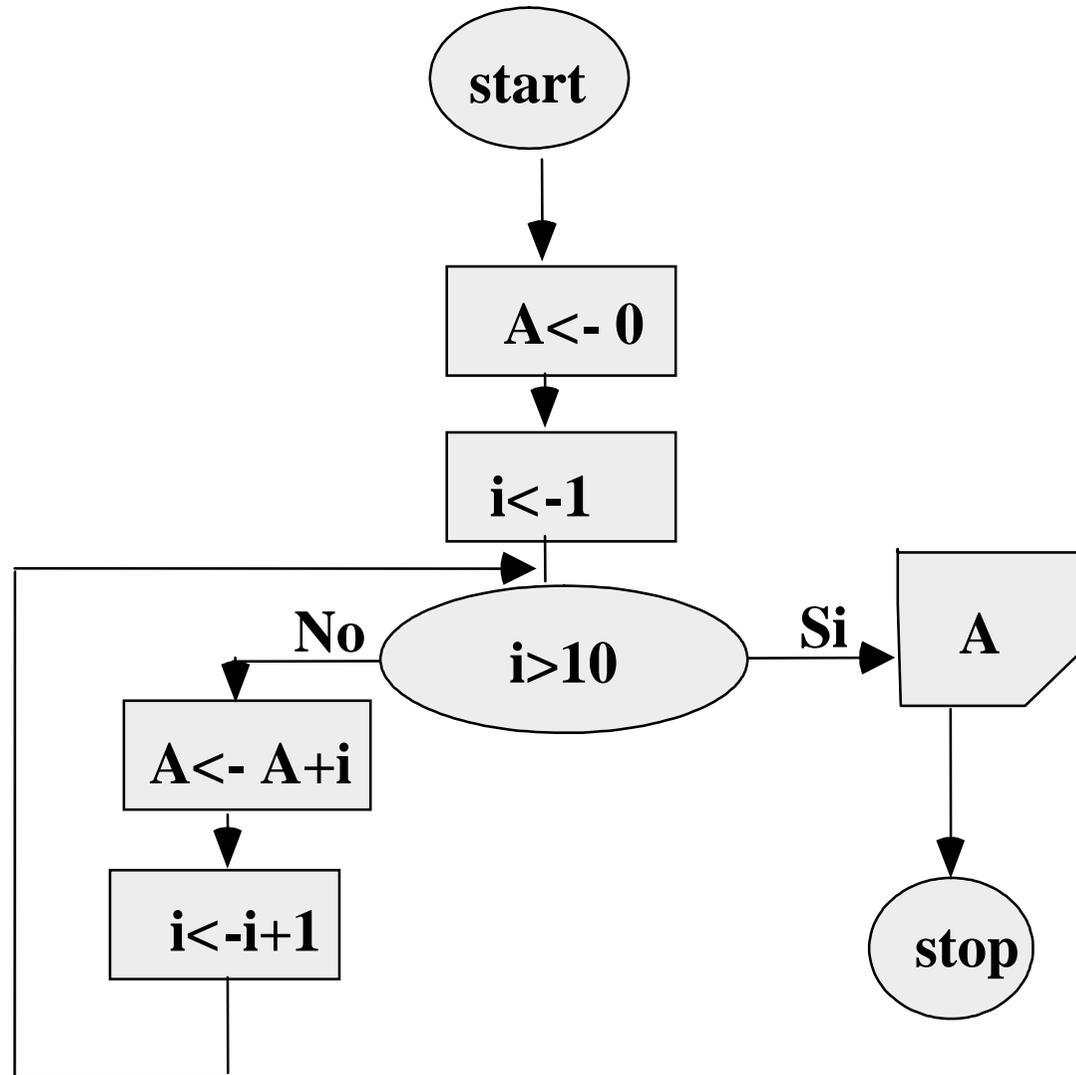
➤ **Gli algoritmi a e b sono equivalenti.**

Rappresentazione di Algoritmi: Diagrammi di flusso

E' un formalismo che consente di rappresentare graficamente gli algoritmi.

- un **diagramma di flusso** descrive le azioni da eseguire ed il loro ordine di esecuzione.
- ad ogni tipo di **azione** corrisponde un simbolo grafico (**blocco**) diverso.
- ogni blocco ha un ramo in ingresso ed uno o piu` rami in uscita; collegando tra loro i vari blocchi attraverso i rami, si ottiene un diagramma di flusso
- un diagramma di flusso appare, quindi, come un insieme di blocchi, collegati fra loro da linee orientate che specificano la sequenza in cui i blocchi devono essere eseguiti (flusso del controllo di esecuzione).

Esempio



Diagrammi di Flusso

- **Dati:**
 - **Variabile:** Rappresenta un dato ed è individuata da un nome simbolico cui è assegnato un valore che può cambiare durante l'esecuzione dell'algoritmo.
 - **Costante:** è una grandezza nota a priori, il cui valore non cambia durante l'esecuzione.
- **Blocco(o istruzione):** rappresenta una operazione mediante un simbolo grafico
 - **Blocco semplice:** esecuzione di una singola operazione elementare sui dati
 - **Blocco condizione:** in base al verificarsi di una condizione, permette di differenziare il comportamento dell'algoritmo, mediante la scelta tra due alternative.

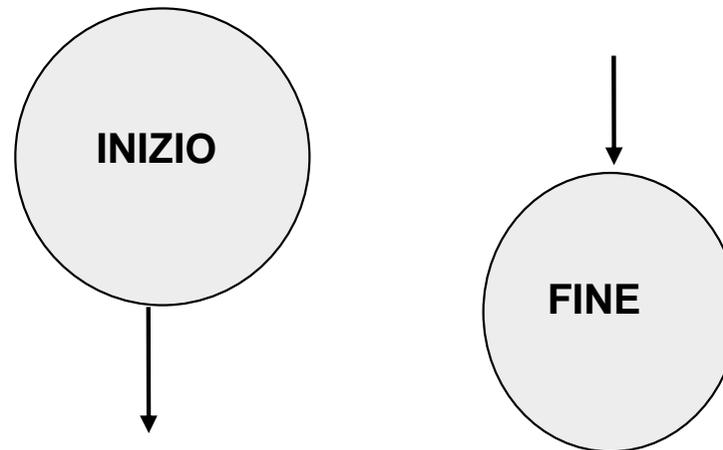
Diagrammi di Flusso

- **Espressioni:** sequenze di variabili e costanti combinate fra loro mediante **operatori**
 - operatori aritmetici: ad esempio $\{+, -, *, /\}$:
 $s + 5 \rightarrow$ producono un risultato *aritmetico*
 - operatori logici e relazionali: ad esempio $\{\text{and, or, not}\}$ e $\{<, >, =, \leq, \geq, \neq\}$
 $\text{not } (C > B) \rightarrow$ producono un risultato *logico* $\{\text{vero, falso}\}$

Blocchi semplici

Inizio e fine esecuzione (start e stop): marcano inizio e fine di un algoritmo

- Inizio e` il blocco da cui deve iniziare l'esecuzione (uno solo epr ogni algoritmo).
- Il blocco fine fa terminare l'esecuzione dell'algoritmo (almeno uno).

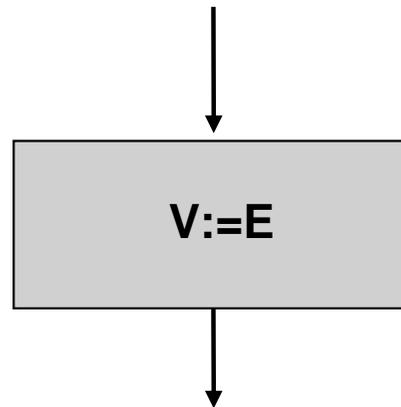


Blocchi semplici

Assegnamento :

calcola il valore dell'espressione a destra del simbolo " := " (o "<-") e lo si attribuisce (lo si *assegna*) alla variabile indicata a sinistra del simbolo (con eventuale perdita del valore precedente di V)

Esempio:



dove V e` il nome di una variabile, E e` una espressione.

Significato: " *Calcola il valore dell'espressione E e assegnalo alla variabile V.*"

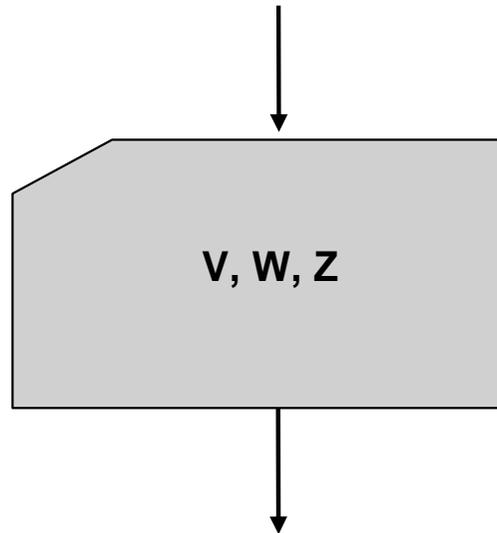
N.B. Il valore di V viene, in generale, **modificato**.

Blocchi semplici

Ingresso (lettura, read, input):

Si ricevono dal dispositivo di ingresso (per esempio, la tastiera) tanti valori quante sono le variabili specificate all'interno del blocco (separate da virgole), e si attribuiscono (*si assegnano*) nello stesso ordine alle variabili.

Ad esempio:



Significato: *"leggi i tre valori dati in ingresso, ed assegnali rispettivamente alle variabili V, W, e Z."*

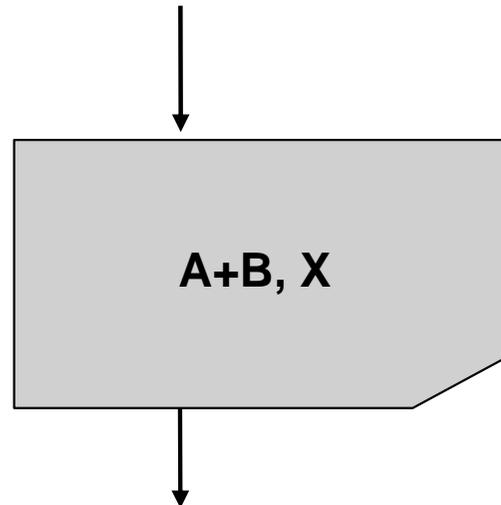
Quindi: durante l'esecuzione, se vengono digitati dalla tastiera i valori: 5, 7, 9, allora la variabile V assumerà il valore 5, W il valore 7 e Z il valore 9.

Blocchi semplici

Uscita (stampa, print, output):

i valori delle espressioni specificati all'interno del blocco vengono calcolati e successivamente trasmessi al dispositivo di uscita (per esempio, il video).

Ad esempio:

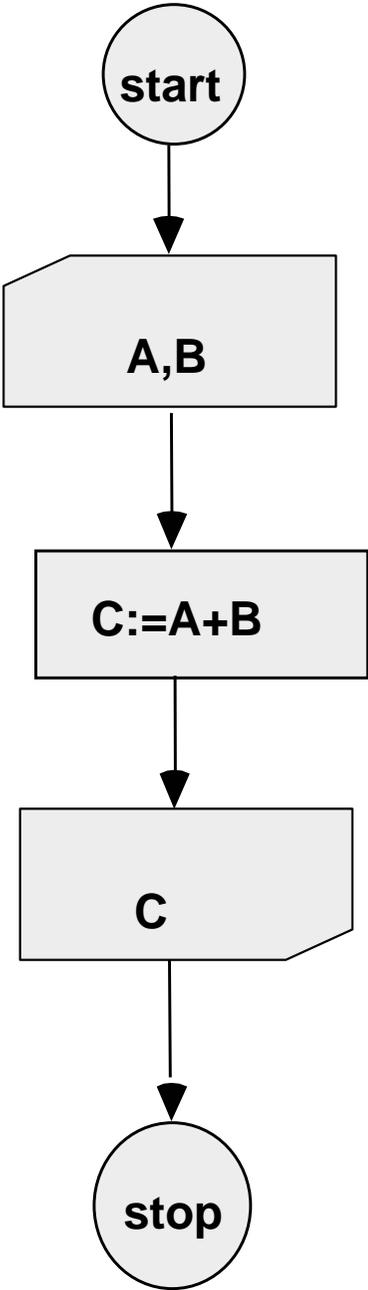


Significato: *"calcola il valore dell'espressione $A+B$ e di X e trasmettili in uscita."*

Quindi: se A vale 10, B vale 7 e X vale -25, l'esecuzione del blocco provocherà la stampa dei 2 valori: 17 e -25.

NB: I valori di A , B e X non vengono alterati dall'esecuzione del blocco.

Esempio

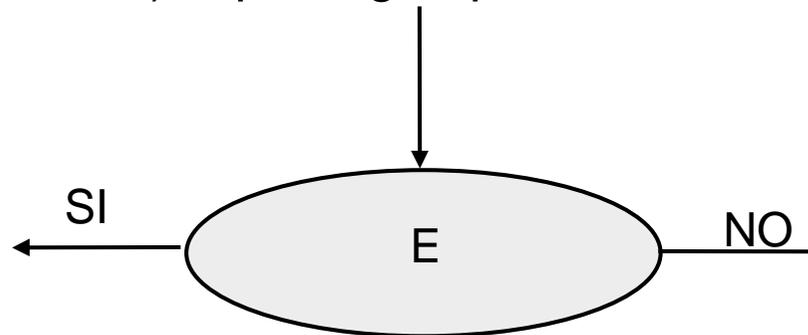


Blocco Condizione

Condizione:

Si valuta la condizione specificata all'interno del blocco: se è verificata, si prosegue con la linea di flusso contrassegnata da "SI" (o *ok, vero, true..*), altrimenti (se non è verificata) si prosegue per il ramo etichettato con "NO" (*false, false,..*).

Esempio:



dove E è un'espressione relazionale (o logica): ritorna valore vero, oppure falso.

Significato: " Calcola il valore dell'espressione E: se è vero, prosegui per il ramo SI, altrimenti prosegui per il ramo NO".

NB. Il blocco condizione è l'elemento di base per realizzare *alternative e ripetizioni*.

Strutture di controllo

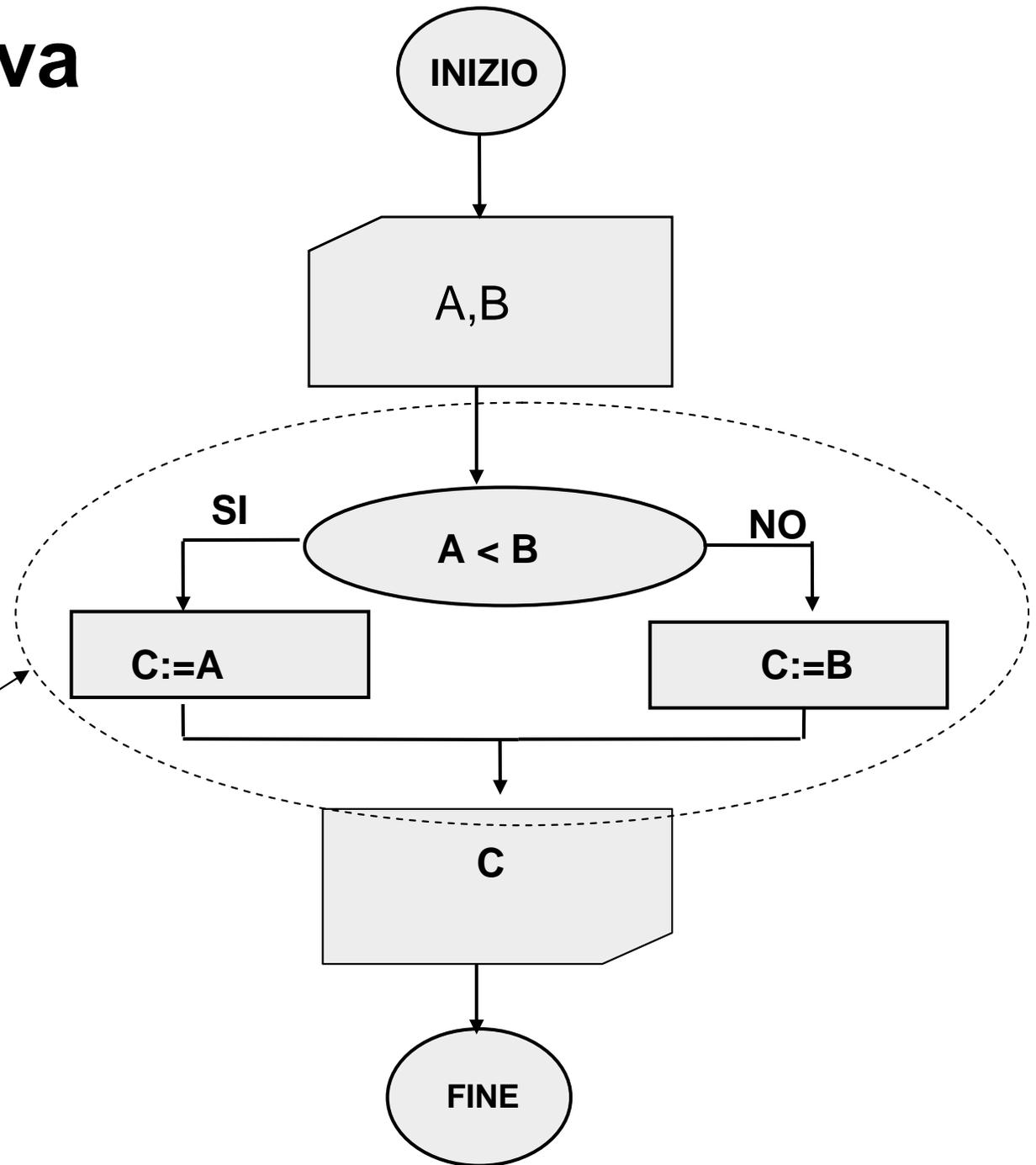
Mediante i blocchi fondamentali finora visti, è possibile costruire delle strutture da utilizzare per il controllo del flusso di esecuzione dell'algoritmo:

- **Alternativa:** esprime la **scelta tra due possibili azioni** (o sequenze di azioni) mutuamente esclusive.
- **Ripetizione:** esprime la **ripetizione di una sequenza di istruzioni.**

Strutture: Alternativa

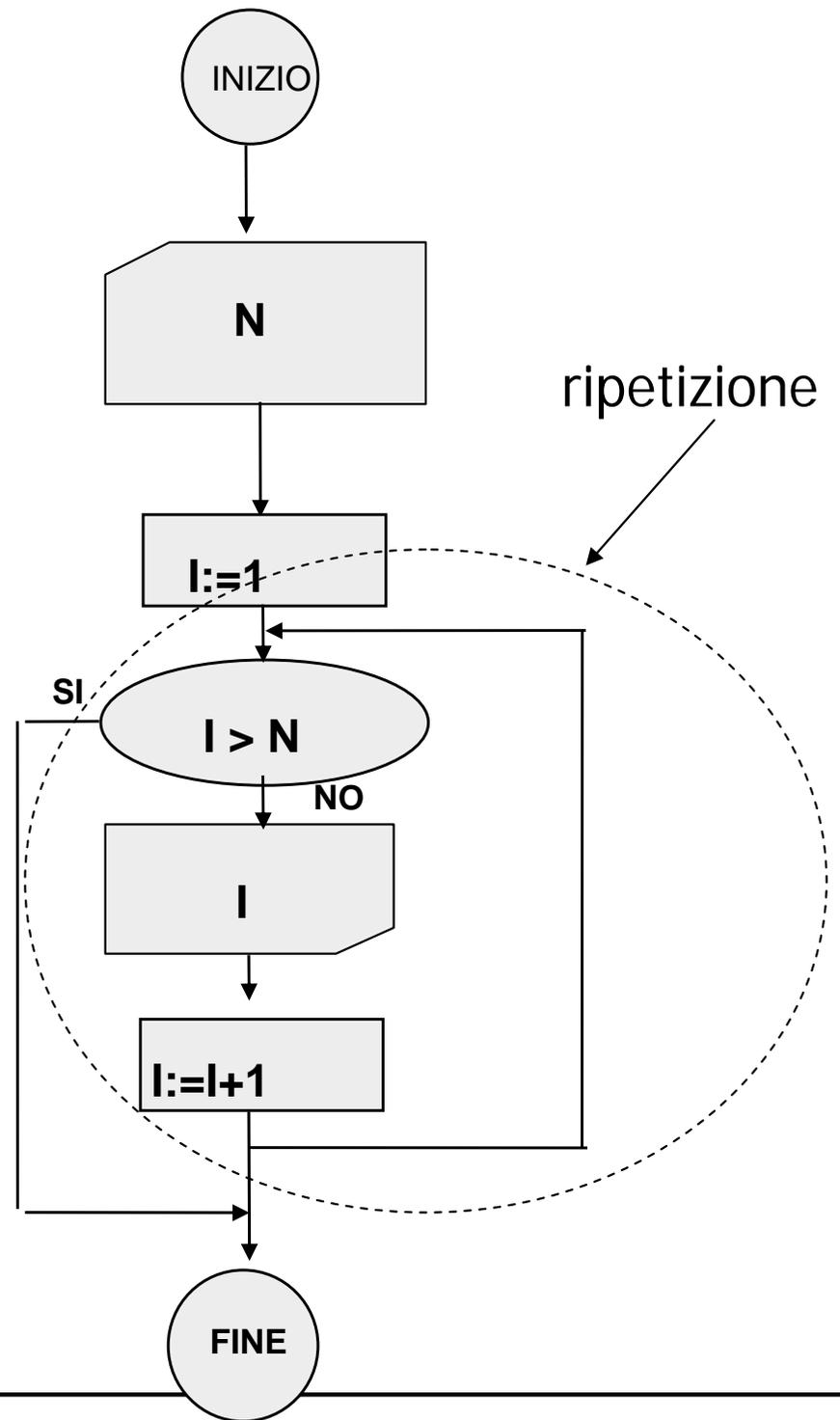
algoritmo che, dati due valori interi A e B, stampa il minore dei due.

alternativa



Strutture: ripetizione

algoritmo che, dato un valore intero positivo N , stampa tutti gli interi >0 e $\leq N$.

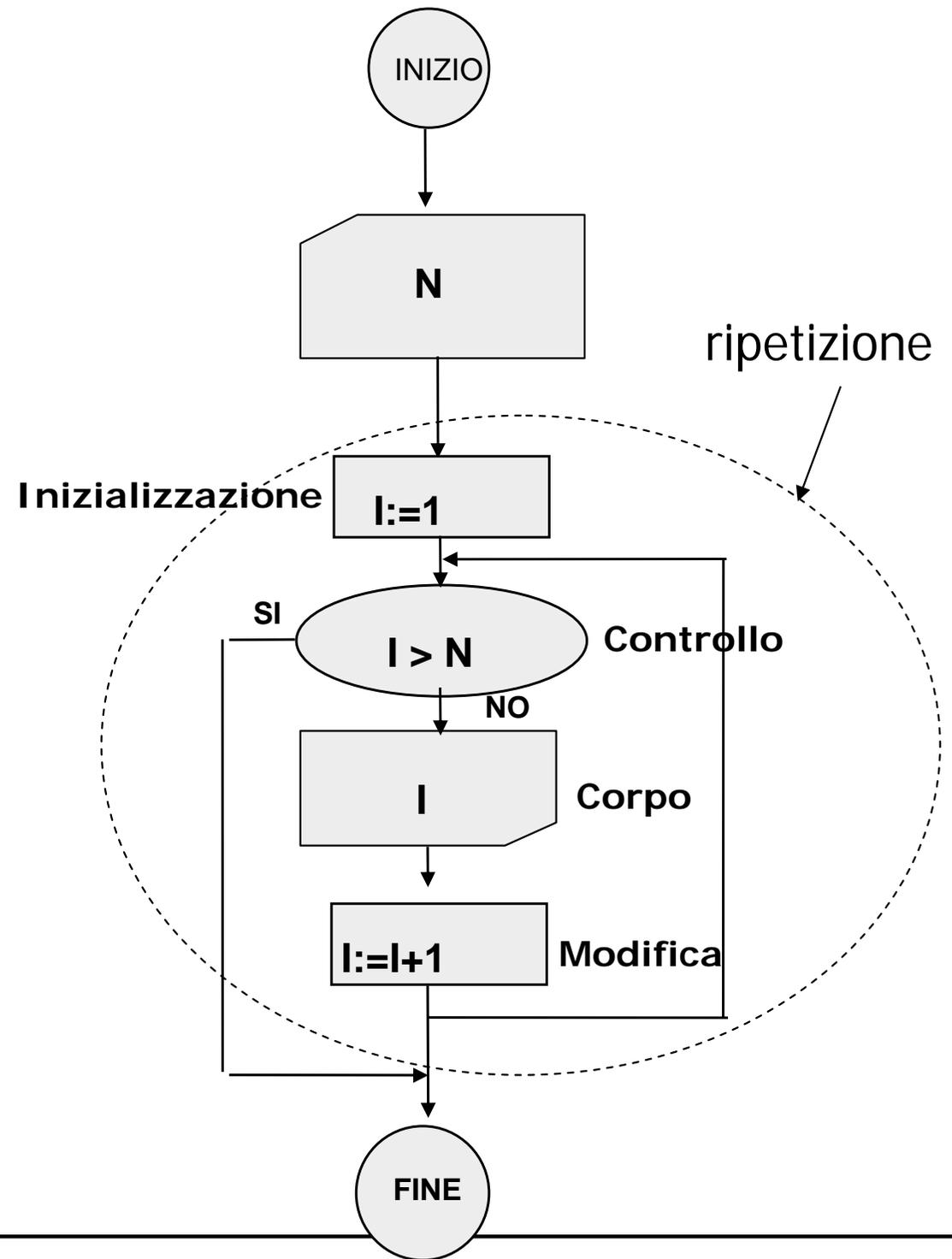


Strutture: Ripetizione (o *iterazione*)

Nel caso piu` generale, e` costituita da 4 elementi:

- **Inizializzazione:** assegnazione dei valori iniziali alle variabili caratteristiche del ciclo (viene eseguita una sola volta);
- **Corpo:** esecuzione delle istruzioni fondamentali del ciclo che devono essere eseguite in modo ripetitivo;
- **Modifica:** modifica dei valori delle variabili che controllano l'esecuzione del ciclo (eseguito ad ogni iterazione);
- **Controllo:** determina, in base al valore delle variabili che controllano l'esecuzione del ciclo se il ciclo deve essere ripetuto o meno.

Ripetizione



Esempio:

Algoritmo che calcola il prodotto come sequenza di somme (si suppone $X \geq 0$).

