Capitolo 3
L'elaborazione e la strutturazione dell'informazione

Problemi e algoritmi

Definire il problema

- **Eliminare le ambiguità nella formulazione del problema**
- **Individuare il risultato che si vuole ottenere, gli obiettivi da raggiungere**
- **Evidenziare**
  - le regole da rispettare
  - i vincoli interni ed esterni
  - i dati espliciti ed impliciti
- **Eliminare i dettagli inutili ed ambigui**

Soluzioni ed esecutori

- **La descrizione della soluzione di un problema dipende dall'esecutore**
  - esecutore con un livello medio di scolarità ⇒
    "**determina la superficie *s* di un cerchio di cui è noto il raggio *r***";
  - esecutore che non conosce come calcolare l'area del cerchio ⇒
    "**la superficie di un cerchio è $s = \pi r^2$**";
  - esecutore che non conosce π ⇒
    "**la superficie di un cerchio è $s = \pi r^2$ e π indica *pi greco* ed è 3.1415**";
  - … ⇒ "**eleva al quadrato il raggio e quindi moltiplica il risultato per pi greco**";
  - … ⇒ "**moltiplica il raggio per se stesso e poi il risultato per 3.14**";
  - …

- **Descrizione della soluzione di un problema sia accettabile per un esecutore**
  - si scompone il problema originario in sottoproblemi;
  - si scompongono i sottoproblemi in sotto-sottoproblemi;
  - si prosegue nella scomposizione fino a giungere a ***problemi elementari*** (o ***primitivi***), cioè problemi che possono essere risolti direttamente dall'esecutore

Istruzioni e azioni elementari

- **Ogni *istruzione elementare* è associata a un'*azione elementare* (oppure a una *successione di azioni elementari*) che può essere direttamente compiuta dall'esecutore.**
- **Processo che porta alla soluzione di un problema:**
  - per il soggetto **descrittore** richiede un'attività di scomposizione progressiva del problema, fino a giungere a una successione di **istruzioni elementari** (ciascuna associata al corrispondente problema elementare);
  - per il soggetto **esecutore** richiede l'esecuzione delle **azioni elementari** associate alle istruzioni elementari identificate. La condizione che le azioni elementari siano eseguite in successione, cioè secondo un ordine definito, è necessaria perché, in generale, ognuna di esse opera sui dati che riceve dalle azioni eseguite precedentemente.

- **Le azioni elementari vengono interpretate in termini *funzionali*, come delle entità che trasformano i dati che ricevono in ingresso (*input*) in risultati (*output*), con ciò prescindendo dalle modalità con cui tale trasformazione viene effettuata, cioè assumendo un modello "a scatola nera" (*black box*).**

Procedura effettiva

- **Si dice *procedura effettiva per un esecutore* una successione di azioni tale che:**
  - tutte le azioni della successione sono *elementari* per l'esecutore, che è in grado di eseguire ciascuna di esse in un tempo finito e in modo deterministico, cioè ottenendo sempre gli stessi risultati a parità di input;
  - è fissato l'ordine di esecuzione delle azioni;
  - è esplicitamente specificato il modo in cui un'azione utilizza i risultati delle azioni che la precedono.

Soluzione effettiva

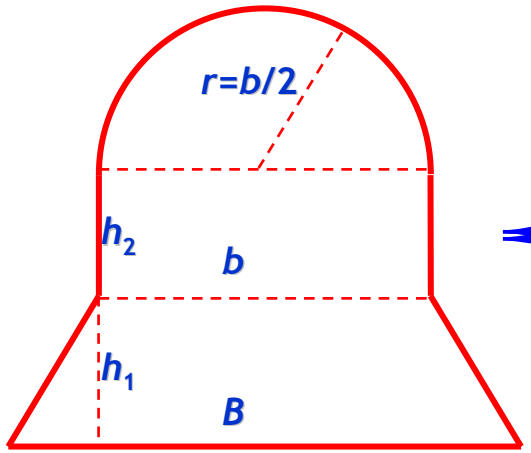
- **Dati un problema *P* e un esecutore *E*, si definisce *soluzione effettiva del problema P per l'esecutore E* una successione di istruzioni elementari tale che:**
  - l'esecutore è in grado di interpretare le istruzioni nella successione e quindi di associare a ciascuna di esse l'azione (o la successione di azioni) che deve compiere per eseguirla;
  - la successione di azioni risultante dall'interpretazione delle istruzioni costituisca una procedura effettiva per l'esecutore stesso.

- **In generale, possono esistere diverse soluzioni effettive dello stesso problema per lo stesso esecutore**

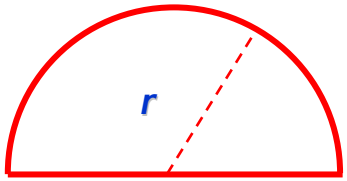
Area di una campana (1)

Scomposizione del problema in tre sottoproblemi

Problema

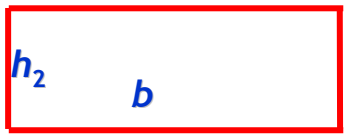


soluzione *elementare*:
$s = ??$



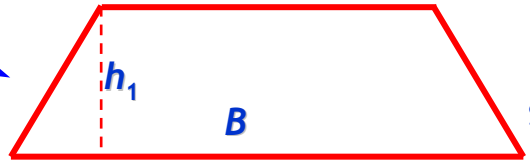
Sottoproblema 1

soluzione *elementare*:
$s = \frac{1}{2} \pi r^2$



Sottoproblema 2

soluzione *elementare*:
$s = b\ h_2$

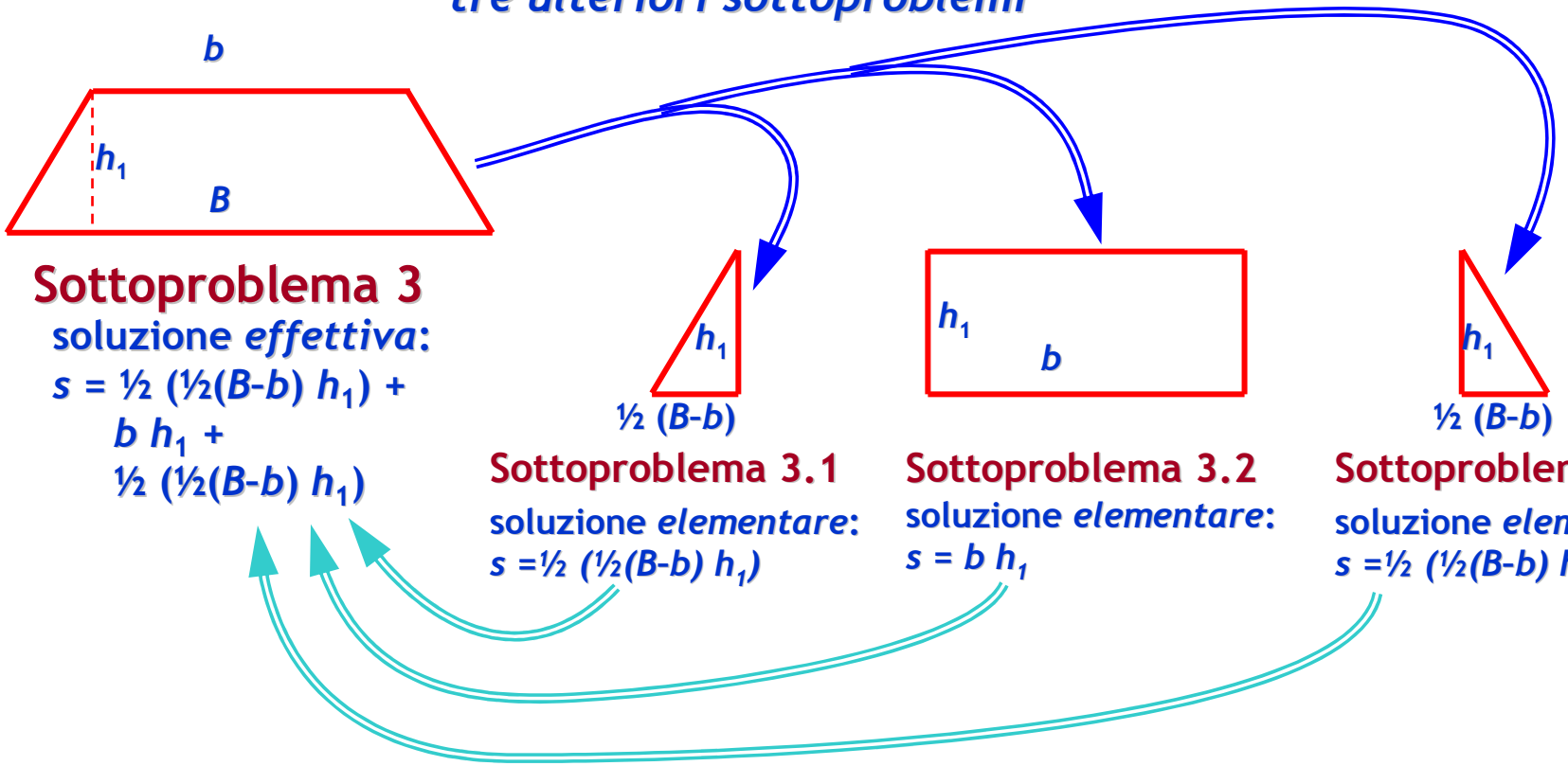


Sottoproblema 3

soluzione *elementare*:
$s = ??$

Area di una campana (2)

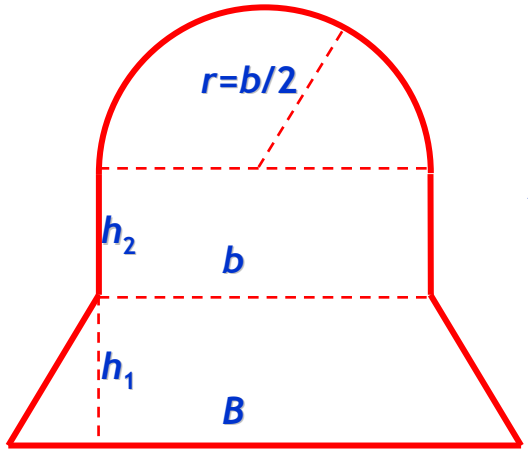
Scomposizione del sottoproblema 3 in tre ulteriori sottoproblemi



Composizione delle soluzioni dei tre sottoproblemi 3.1, 3.2 e 3.3 per risolvere il sottoproblema 3

Area di una campana (3)

Problema

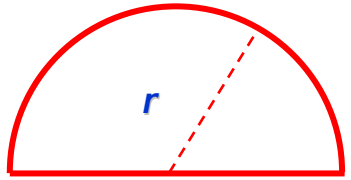


soluzione *effettiva*:

$s = \frac{1}{2} \pi r^2 +$
$b\ h_2 +$
$\frac{1}{2}(\frac{1}{2}(B-b)\ h_1) +$
$b\ h_1 +$
$\frac{1}{2}(\frac{1}{2}(B-b)\ h_1)$

Composizione delle soluzioni dei tre sottoproblemi 1, 2 e 3 per risolvere il problema originario

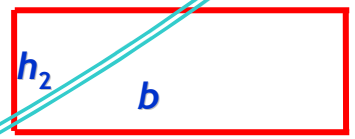
Sottoproblema 1



soluzione *elementare*:

$s = \frac{1}{2} \pi r^2$

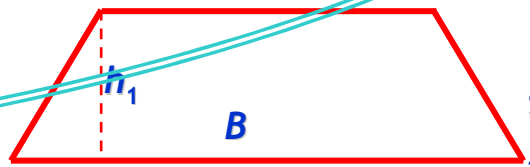
Sottoproblema 2



soluzione *elementare*:

$s = b\ h_2$

Sottoproblema 3



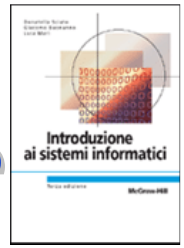
soluzione *effettiva*:

$s = \frac{1}{2}(\frac{1}{2}(B-b)\ h_1) +$
$b\ h_1 +$
$\frac{1}{2}(\frac{1}{2}(B-b)\ h_1)$

Esecutori e linguaggi

- **Un esecutore è definito in base a tre elementi:**
  - l'insieme delle **operazioni** che è capace di compiere;
  - l'insieme delle **istruzioni** che capisce (**sintassi**);
  - quali **operazioni** associa ad ogni **istruzione** che riconosce (**semantica**).
- **Il calcolatore "capisce" le istruzioni che fanno parte del linguaggio macchina**
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore
- **La soluzione si dice *effettiva* se l'esecutore è in grado di:**
  - interpretarla
  - compiere le azioni (in un tempo finito!)

Proprietà di un'azione elementare



- **Finitezza**
  - l'azione deve concludersi in un tempo finito
- **Osservabilità**
  - l'azione deve avere un effetto osservabile, cioè deve produrre qualcosa
- **Riproducibilità**
  - a partire dallo stesso stato iniziale, la stessa azione deve produrre sempre lo stesso risultato

Dal problema alla soluzione automatica

- Specifiche dei requisiti:
  descrizione precisa e corretta dei requisiti (verificabilità) ---> cosa?

- Progetto:
  procedimento con cui si individua la soluzione ---> come?

- Soluzione: algoritmo

Proprietà degli algoritmi

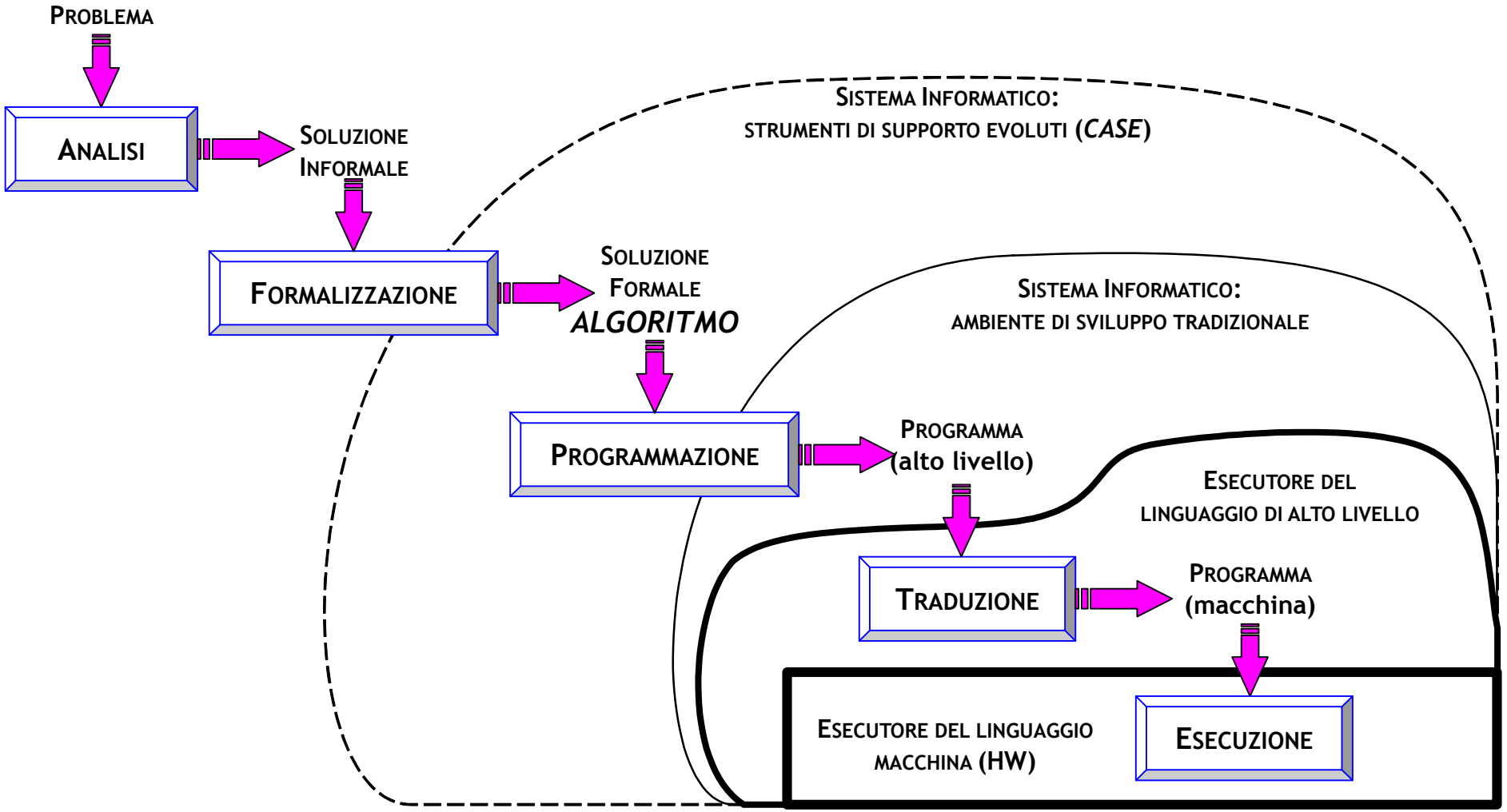
- **Correttezza**
  - L'algoritmo perviene alla soluzione del compito cui è preposto, senza difettare di alcun passo fondamentale
- **Efficienza**
  - L'algoritmo perviene alla soluzione del problema usando la minima quantità di risorse fisiche
    - tempo di esecuzione, memoria, …

Alcuni concetti

- ***Algoritmo*** = **descrizione di come si risolve un problema**
- ***Programma*** = **algoritmo scritto in modo che possa essere eseguito da un calcolatore (linguaggio di programmazione)**
- ***Linguaggio macchina*** = **linguaggio effettivamente "compreso" da un calcolatore, caratterizzato da**
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore

- **Due aspetti rilevanti:**
  - **produrre algoritmi** (cioè capire la sequenza di passi che portano alla soluzione di un problema)
  - **codificarli in programmi** (cioè renderli comprensibili al calcolatore)

Sviluppo di un programma



Algoritmi

Formalizzazione

Codifica degli algoritmi

- **Algoritmo formulato per essere comunicato tra esseri umani**
  - sintetico e intuitivo
  - codificato in linguaggi informali o semi-formali (linguaggio naturale, diagrammi di flusso, …)
- **Algoritmo formulato per essere eseguito automaticamente**
  - preciso ed eseguibile
  - codificato in linguaggi comprensibili dagli esecutori automatici (linguaggio macchina o linguaggio di programmazione di alto livello)

Algoritmi e variabili

- **Gli algoritmi sono parametrici:**
  - producono un risultato che dipende da un insieme di dati di partenza;
  - descrivono la soluzione non di un singolo problema, ma di una intera **classe di problemi** strutturalmente equivalenti.
  - Esempi:
    - l'algoritmo per la moltiplicazione di due numeri specifica come effettuare il prodotto di *tutte* le possibili coppie di numeri;
    - l'algoritmo per la ricerca di un libro nello schedario della biblioteca vale per tutti i possibili libri;
    - …
- **Le istruzioni dell'algoritmo fanno riferimento a *variabili*, il cui valore non è fissato a priori ma cambia a seconda della situazione elaborativa in cui l'esecutore si trova.**

Uso delle variabili

- **All'interno di espressioni,**
  - l'esecutore usa il valore contenuto nelle variabili per calcolare il risultato dell'espressione,
  - per esempio **op1 + op2 × op3** oppure **op1 / op2 - op3**, …
- **in istruzioni di assegnamento**
  - introdurre nel contenitore identificato dal nome della variabile il valore specificato a destra dell'assegnamento;
  - per esempio **r ← 35** (assegna 35 alla variabile il cui nome è **r**), **pi ← 3,14**, …
- **in istruzioni di assegnamento combinate con espressioni**
  - assegna a una variabile il risultato ottenuto dalla valutazione di un'espressione, per esempio in "**circ ← 2 × r × pi**" il risultato dell'espressione **2 × r × pi** viene calcolato utilizzando i valori contenuti nelle variabili **r** e **pi** e il risultato viene poi assegnato alla variabile **circ**;
  - la stessa variabile può comparire in entrambi i lati dell'istruzione di assegnamento, per esempio in "**k ← k + 1**" il valore contenuto in **k** viene utilizzato per trovare il valore dell'espressione **k + 1** che viene memorizzato come nuovo valore di **k**.

Dati e istruzioni

- Tipi di dati
  - Numeri naturali o interi o reali (1, -2, 0.34)
  - Caratteri alfanumerici (A, B, ..)
  - Dati logici o booleani (Vero, Falso)
  - Array o vettore di n elementi ({1,2,3})
- Istruzioni
  - Operazioni di Input/Output (es. *leggi*, *scrivi*)
  - Operazioni Aritmetico-logiche (es. *max* = *A* + *B*)
  - Strutture di controllo (es. *SE*, *RIPETI*)

I dati

- **Ogni variabile è caratterizzata dal suo *tipo*.**
  - Tipi predefiniti: numeri, caratteri, booleani, …
  - Altri tipi: stringhe, date, …
- **Variabili strutturate:**
  - Vettori (o array)
  - Record

Operazioni elementari

- **Operazioni aritmetiche e assegnamenti di valori a singole variabili**
  - Es. C ← (A + B)

- **Condizioni sul valore di singole variabili**
  - se (A > B) allora … altrimenti …

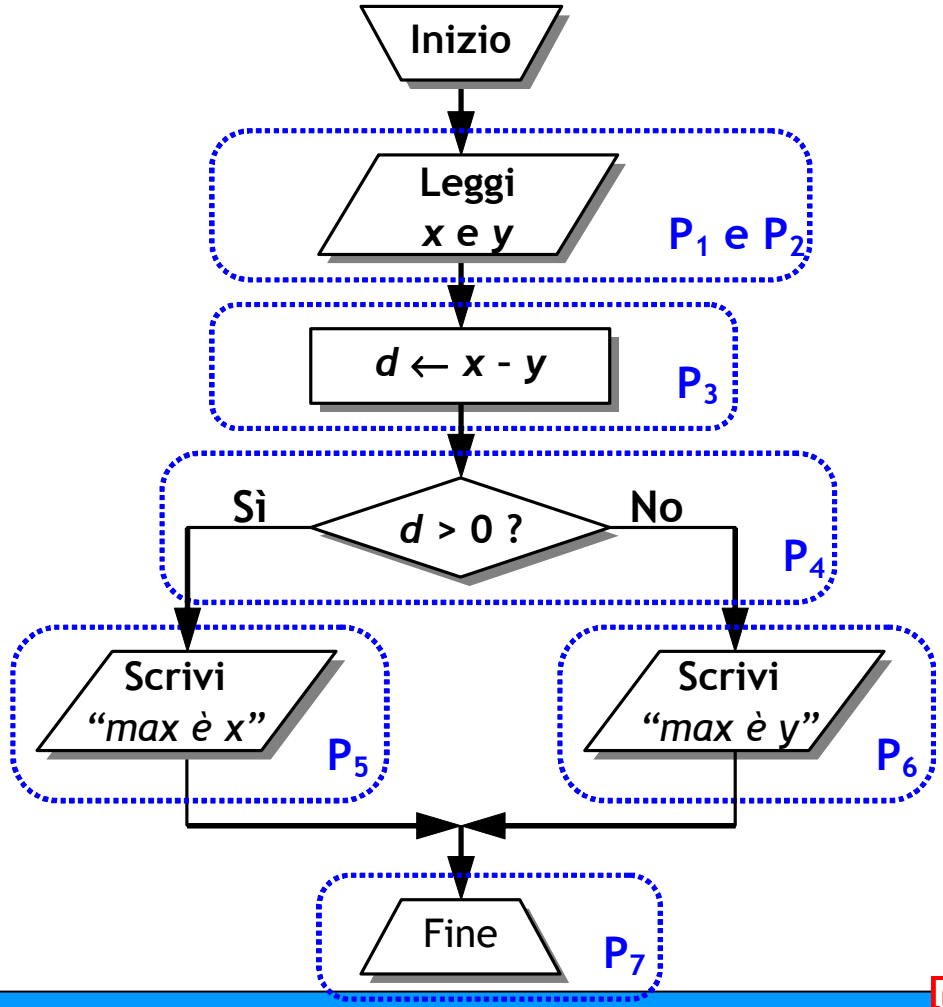
- **Lettura e scrittura di variabili**
  - “Leggi A” oppure “Stampa B”

Rappresentazione degli algoritmi

- Linguaggio naturale
- Diagramma a blocchi
- Pseudo codice
- Linguaggio di programmazione

Rappresentazione degli algoritmi

- **Linguaggio Naturale**
  - Sollevare il ricevitore
  - Attendere il segnale di linea libera
  - Comporre il numero
  - …
- **Diagramma di flusso**



Rappresentazione degli algoritmi

- **Pseudo Codice**

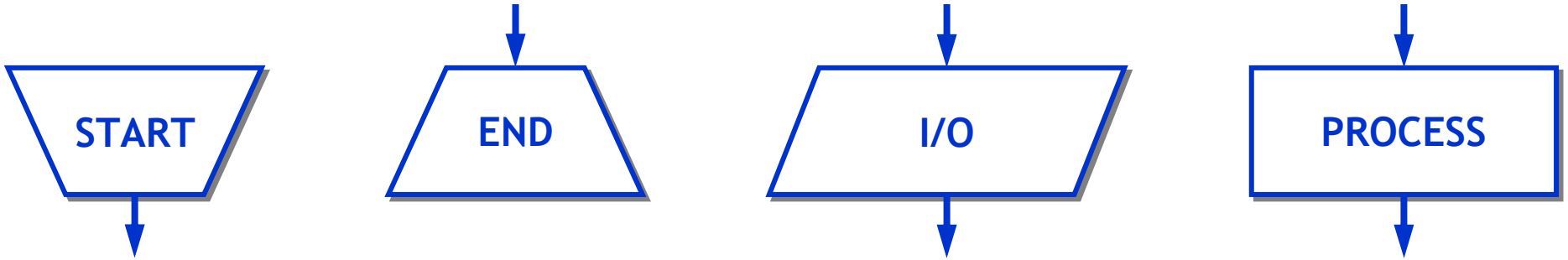
```
leggi alfa, beta
prod ⇦ 0
finché alfa ≠ 0 ripeti
    prod ⇦ prod + beta;
    alfa ⇦ alfa - 1;
stampa prod;
```

- **Linguaggio di programmazione**

```
#include <stdio.h>

int main (void) {
  puts ("ciao mondo!");
  return Exit_success;
}
```

Diagrammi di flusso

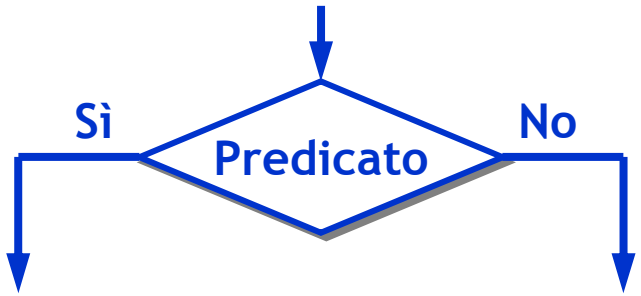


Inizio

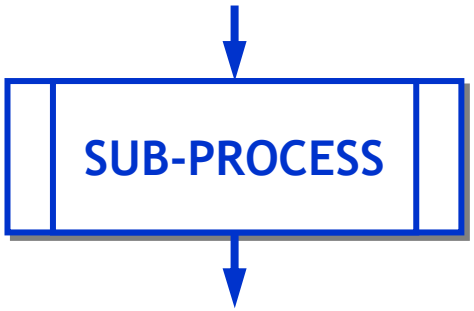
Fine

Operazioni di ingresso/uscita

Elaborazione



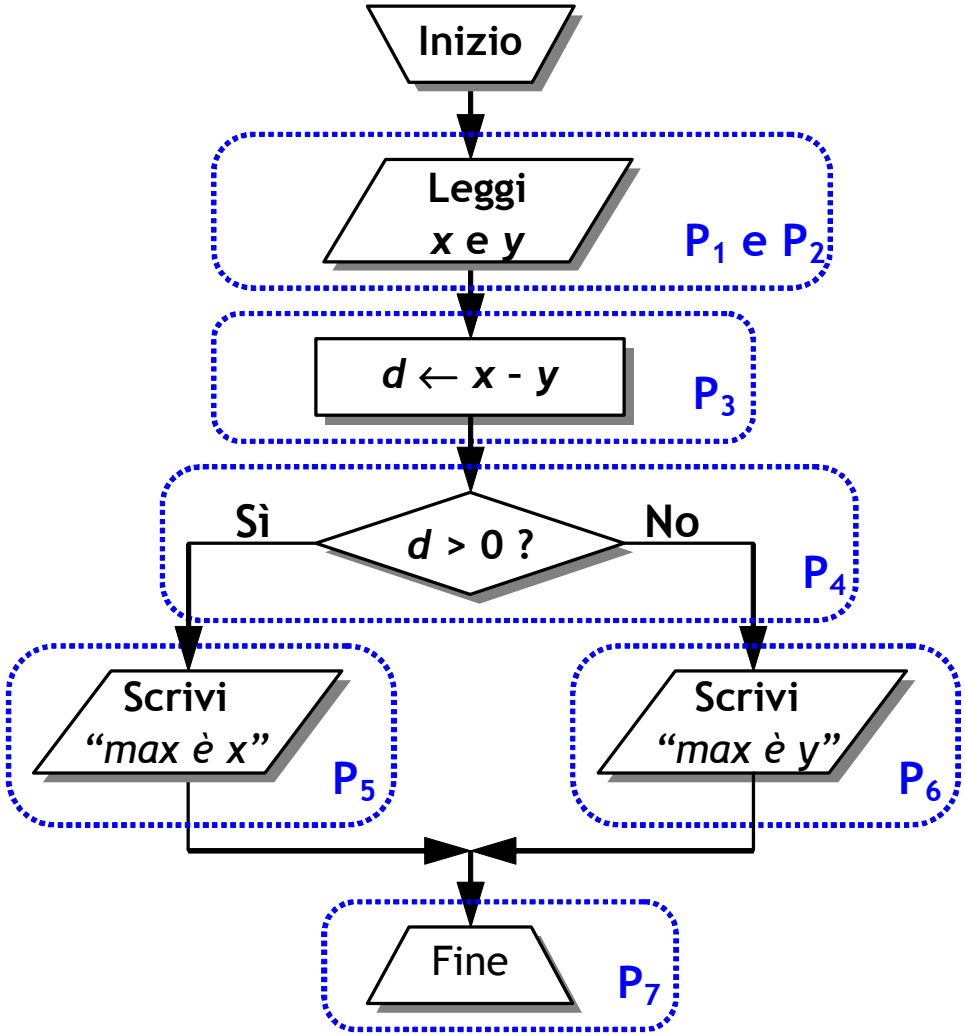
Selezione a due vie



Sottoprogramma

Esempio

Esempio:
dati in ingresso due numeri X e Y, si calcoli e stampi il maggiore.

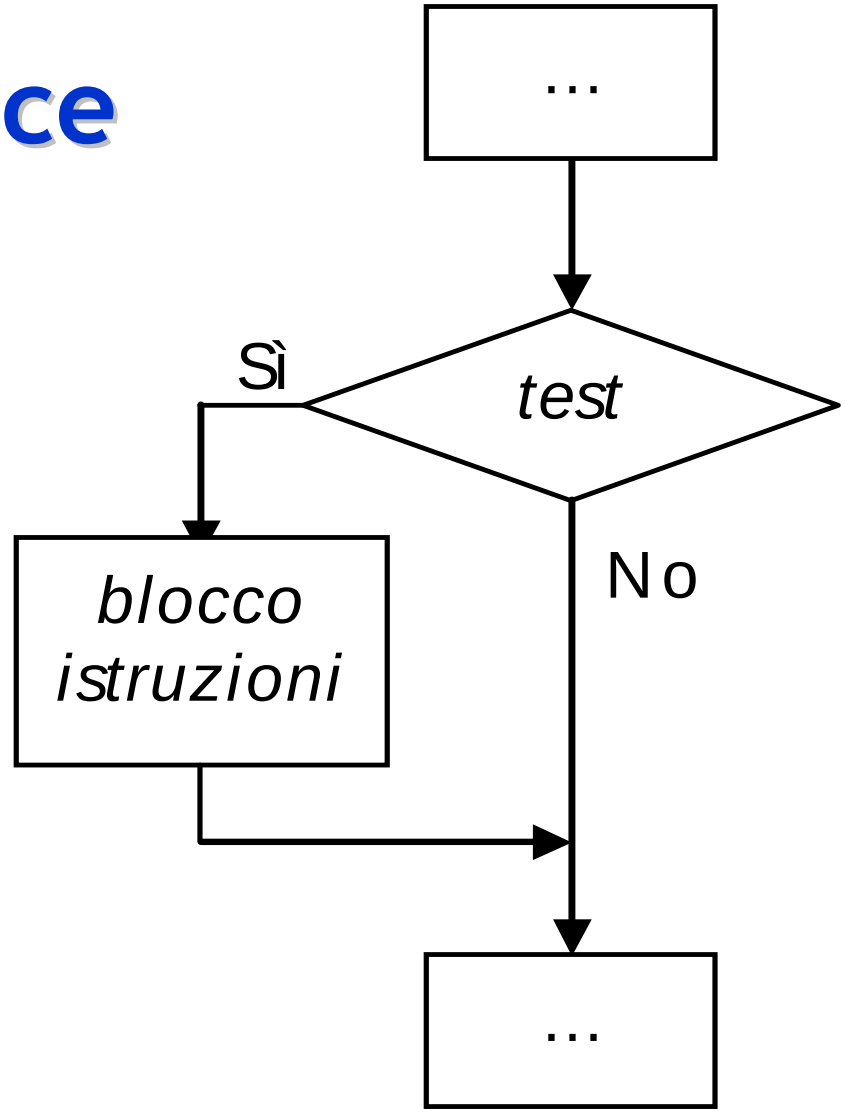


Le strutture di controllo

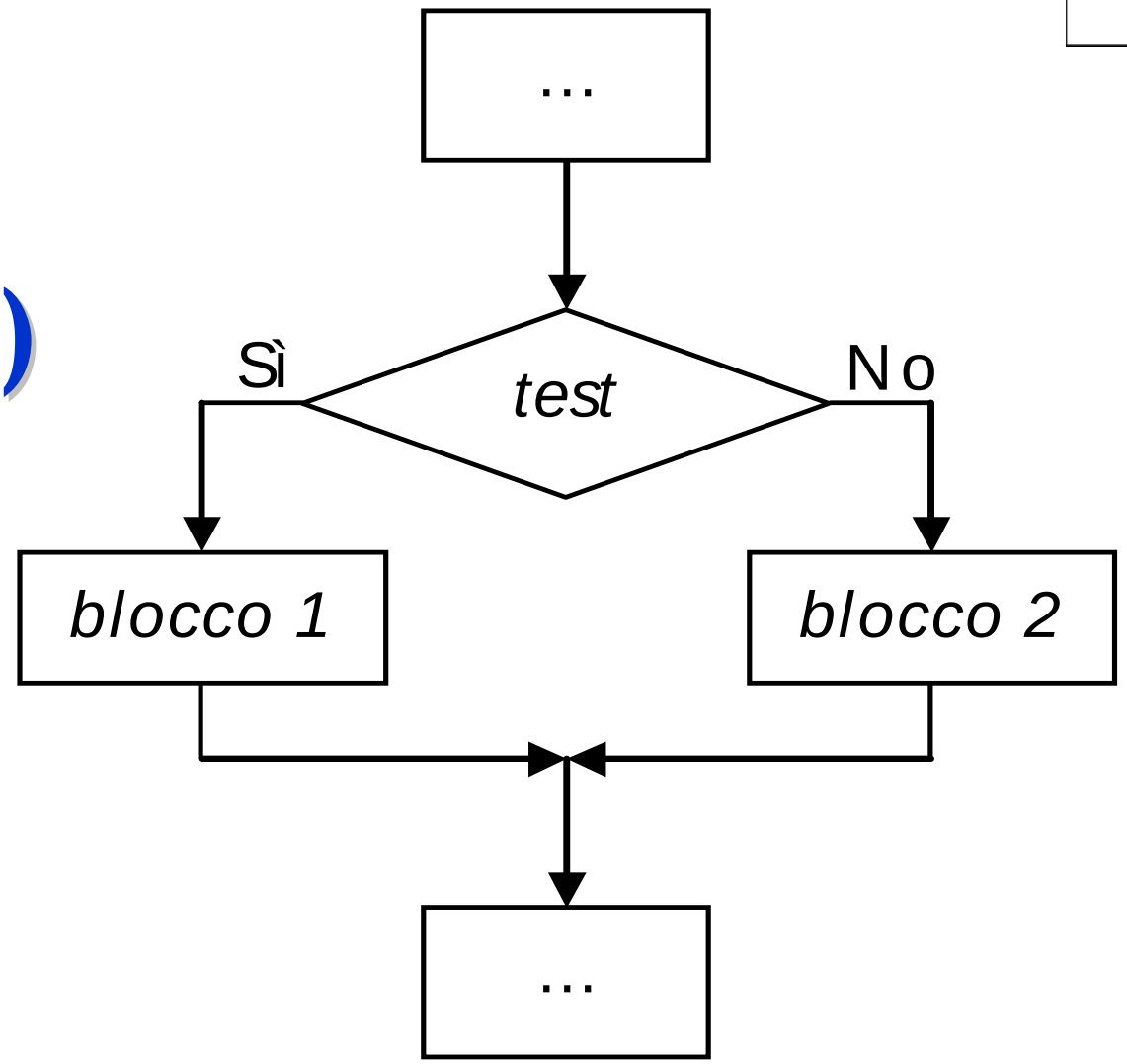
Esempi strutture di controllo

- **Istruzioni condizionate**
  - Se (condizione)
    blocco istruzioni 1
    altrimenti
    blocco istruzioni 2
- **Ripetizione ciclica di istruzioni**
  - Finché (condizione)
    blocco istruzioni

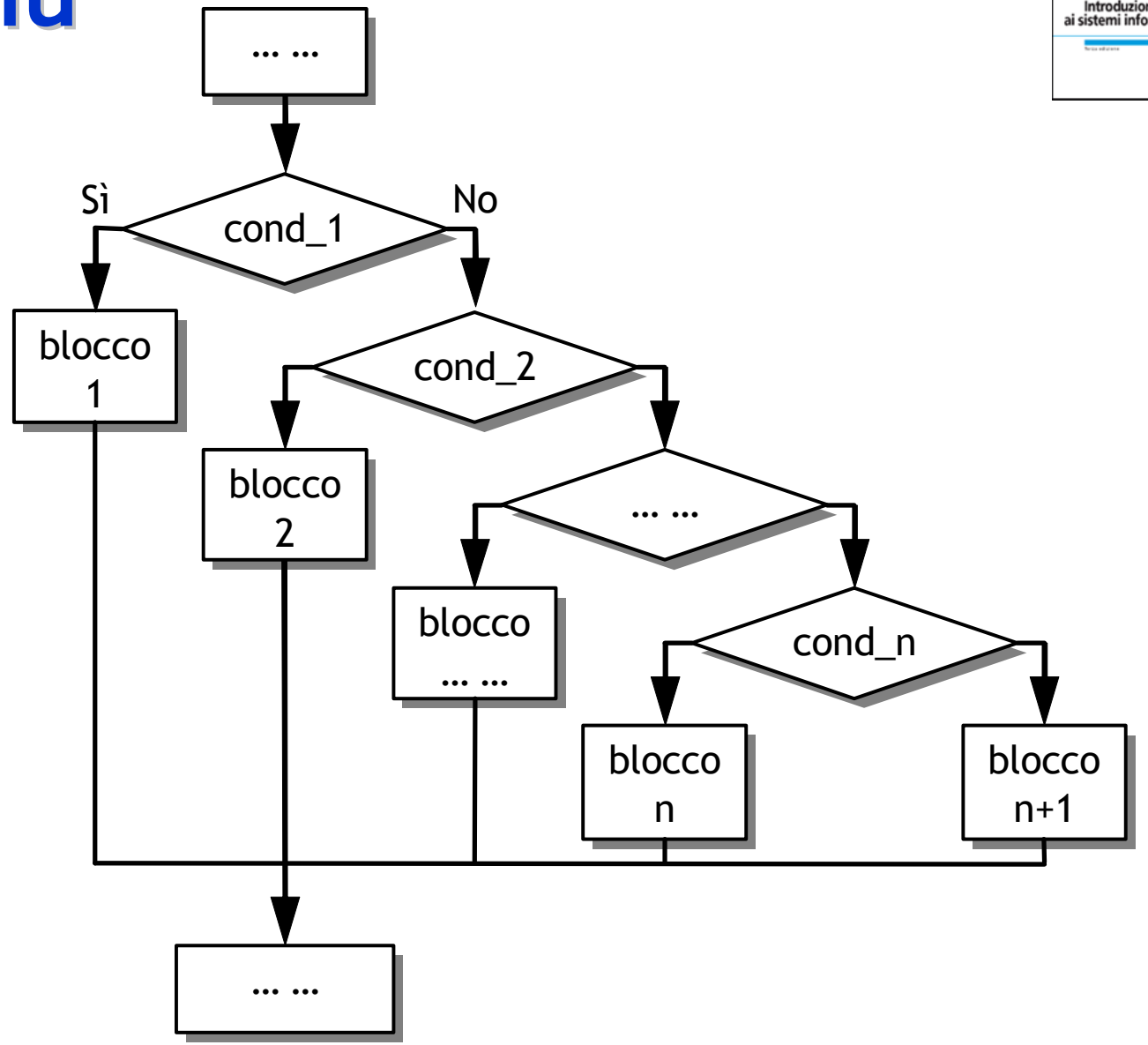
Selezione semplice (if-then)



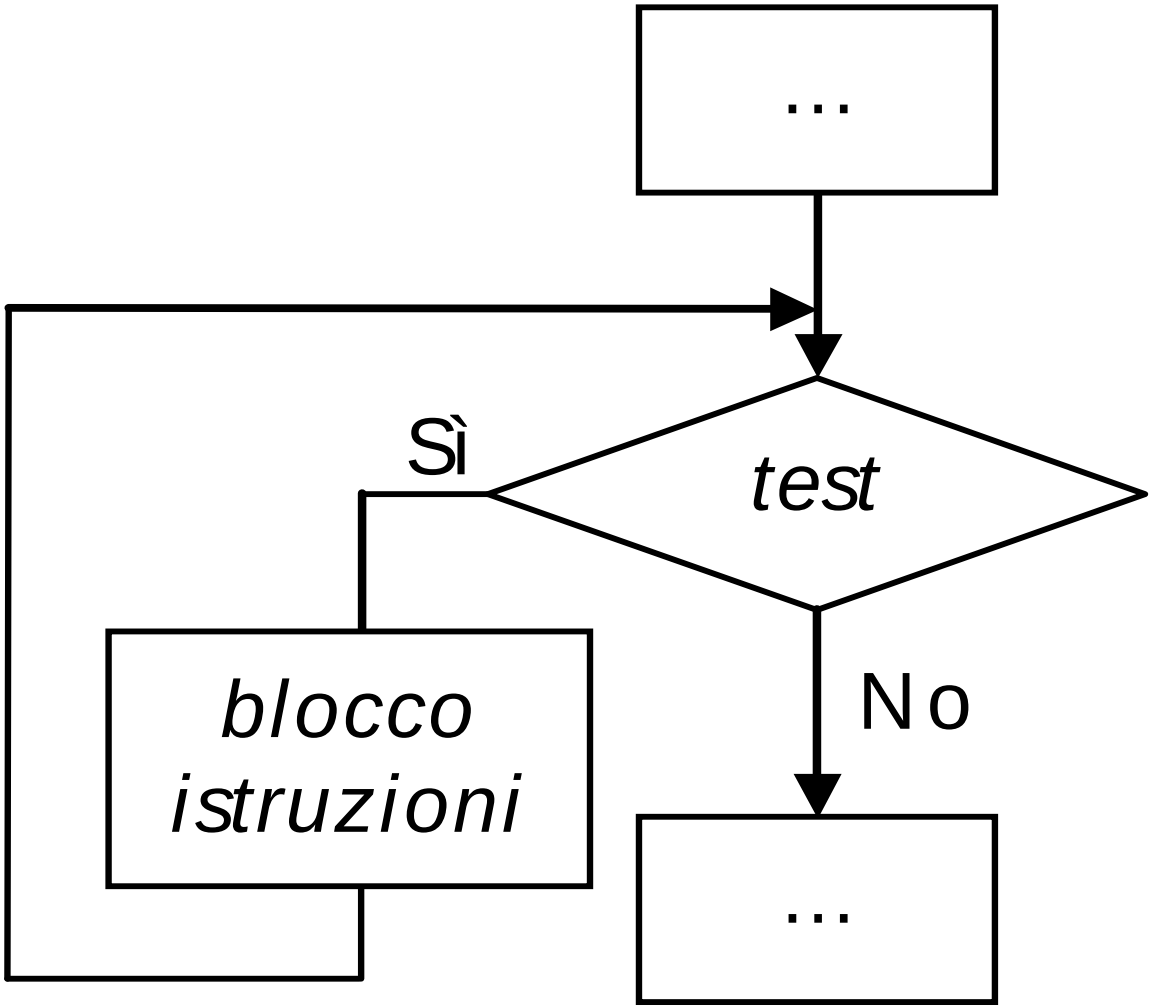
Selezione a due vie (if-then-else)



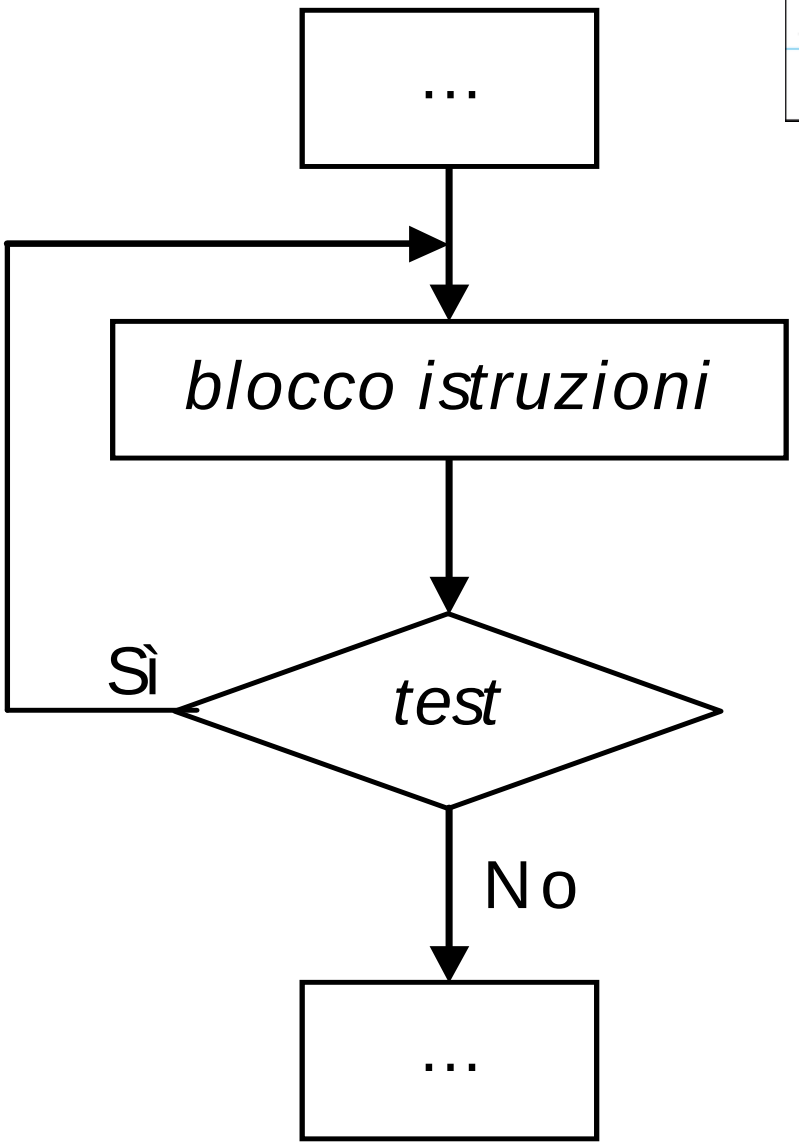
Selezione a più vie (if-then elseif-then elseif-then …)



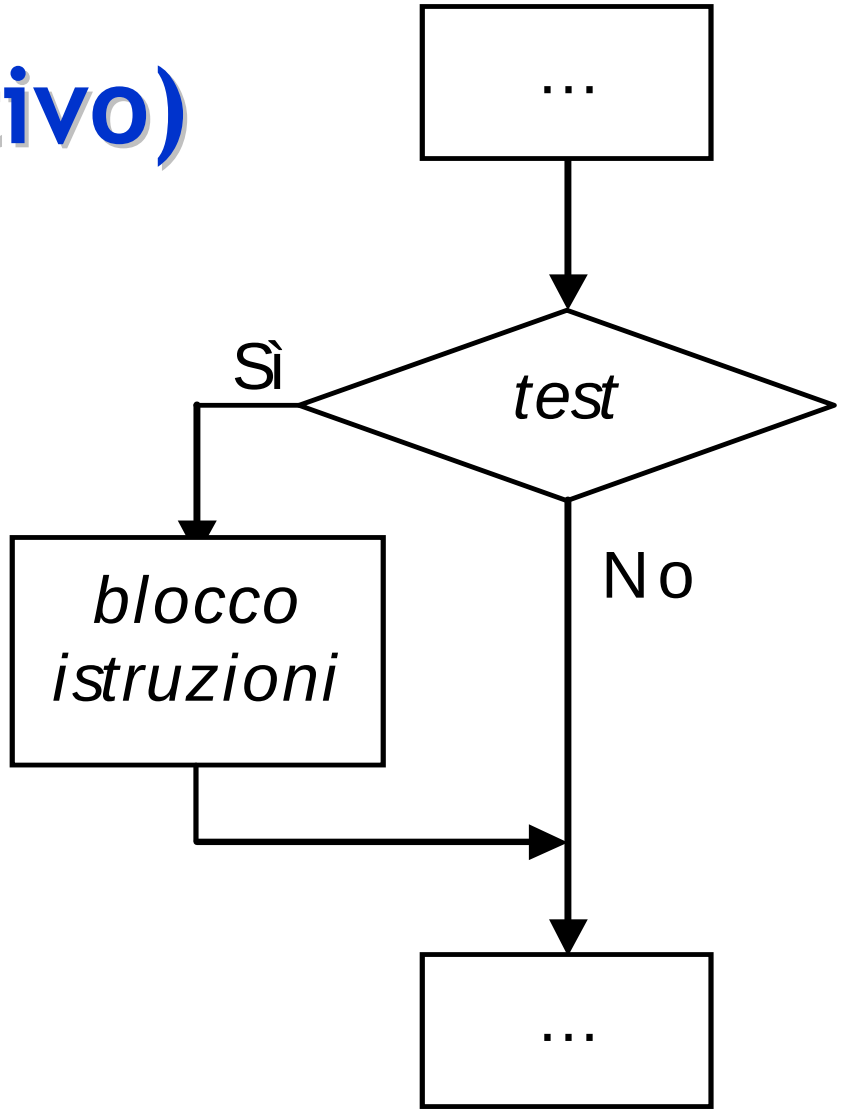
Ciclo a condizione iniziale (while-do)



Ciclo a condizione finale (do-while)



Ciclo FOR (Iterativo) (for loop)



La Programmazione

Linguaggio di Programmazione

- La notazione con cui è possibile descrivere gli algoritmi.
- Programma: è la rappresentazione di un algoritmo in un particolare linguaggio di programmazione.
- Ogni linguaggio di programmazione dispone di un insieme di “parole chiave” (keywords)
- Ogni linguaggio è caratterizzato da una sintassi e da una semantica

Alcuni concetti

- *Algoritmo* = descrizione di come si risolve un problema
- *Programma* = algoritmo scritto in modo che possa essere eseguito da un calcolatore (linguaggio di programmazione)
- *Linguaggio macchina* = linguaggio effettivamente "compreso" da un calcolatore, caratterizzato da
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore

- Due aspetti rilevanti:
  - produrre algoritmi (cioè capire la sequenza di passi che portano alla soluzione di un problema)
  - codificarli in programmi (cioè renderli comprensibili al calcolatore)

Il linguaggio macchina

- **Il linguaggio macchina è direttamente eseguibile dall'elaboratore, senza nessuna traduzione.**
  - Istruzioni ed operandi relativi al programma in esecuzione sono caricati in memoria e quindi sono memorizzati in forma *binaria*.
  - Vincolo: conoscenza dei metodi di rappresentazione delle informazioni utilizzati.

- **Istruzione: carica nell'accumulatore**

10010000 11001100

Il linguaggio Assembler

- Le istruzioni corrispondono univocamente a quelle macchina, ma vengono espresse tramite nomi simbolici (parole chiave).
- Il programma prima di essere eseguito deve essere tradotto in linguaggio macchina (assemblatore).
- Vincolo: necessità di conoscere in dettaglio le caratteristiche della macchina (registri, dimensione dei dati, set di istruzioni)
- Anche semplici algoritmi implicano la specifica di molte istruzioni

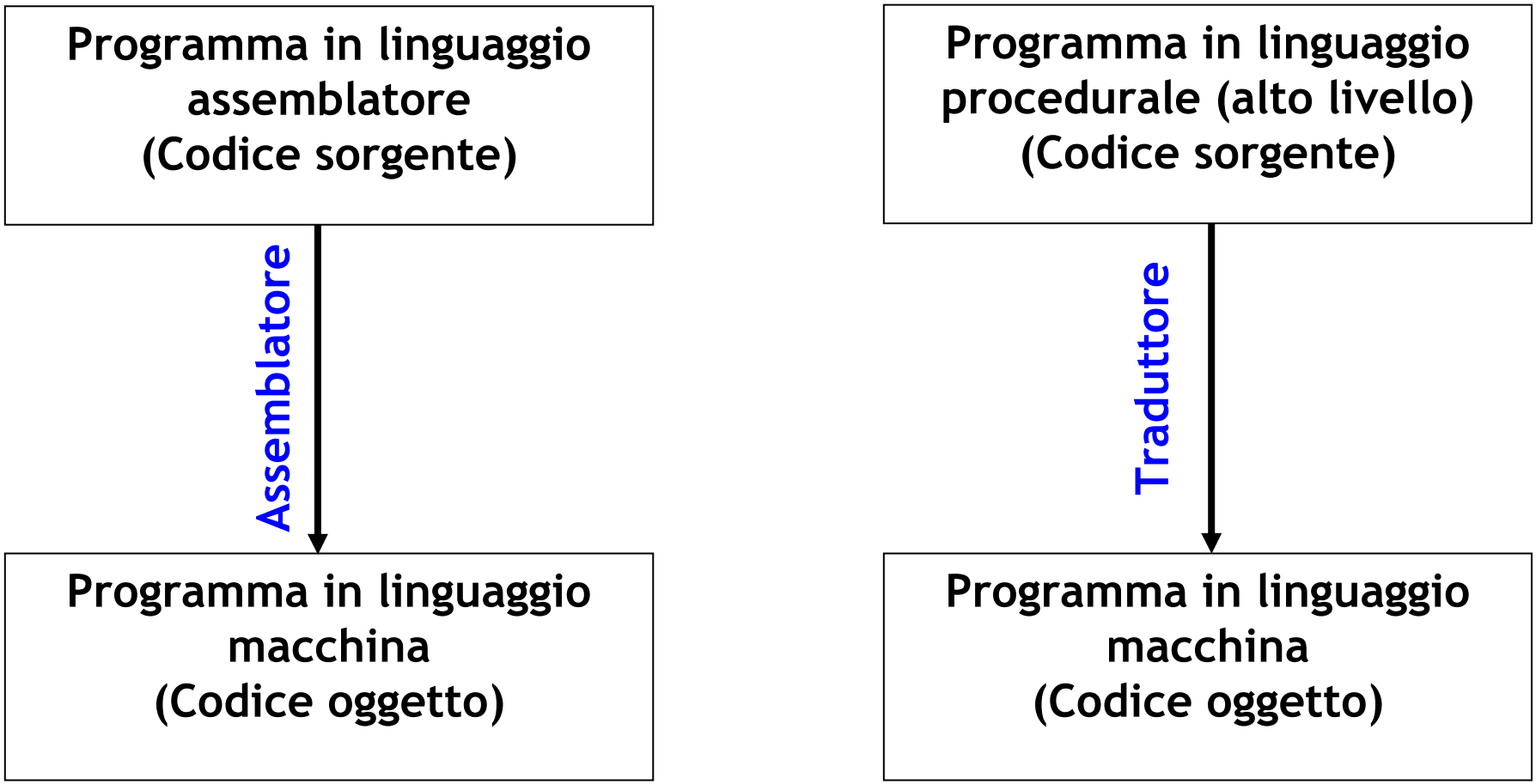
I linguaggi di alto livello

- Sono i linguaggi di terza generazione. Le istruzioni esprimono una serie di azioni. Il programma prima di essere eseguito deve essere tradotto in linguaggio macchina (traduttore)

- Il programmatore può astrarre dai dettagli legati all'architettura ed esprimere i propri algoritmi in modo simbolico

- Sono indipendenti dalla macchina (astrazione)

Linguaggi di II/III generazione



Sviluppo di un Programma

- I traduttori sono programmi particolari che provvedono a convertire il codice di programmi scritti in un dato linguaggio di programmazione (sorgenti), nella corrispondente rappresentazione in linguaggio macchina (eseguibili)

Due tipi di traduttori

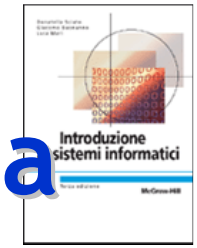
- **Compilatori**
  - Accettano in ingresso l'intero programma e producono in uscita la rappresentazione dell'intero programma in linguaggio macchina.

- **Interpreti**
  - Traducono ed eseguono direttamente ciascuna istruzione del programma sorgente, istruzione per istruzione.

- **L'esecuzione di un programma compilato è più veloce dell'esecuzione di un programma interpretato.**

I programmi

Parti fondamentali di un programma



- *Identificazione* del programma
- *Dichiarazione* delle variabili utilizzate, di cui sono indicati tipo e nome
- Specificazione della parte *esecutiva* del programma, detta anche *corpo del programma*

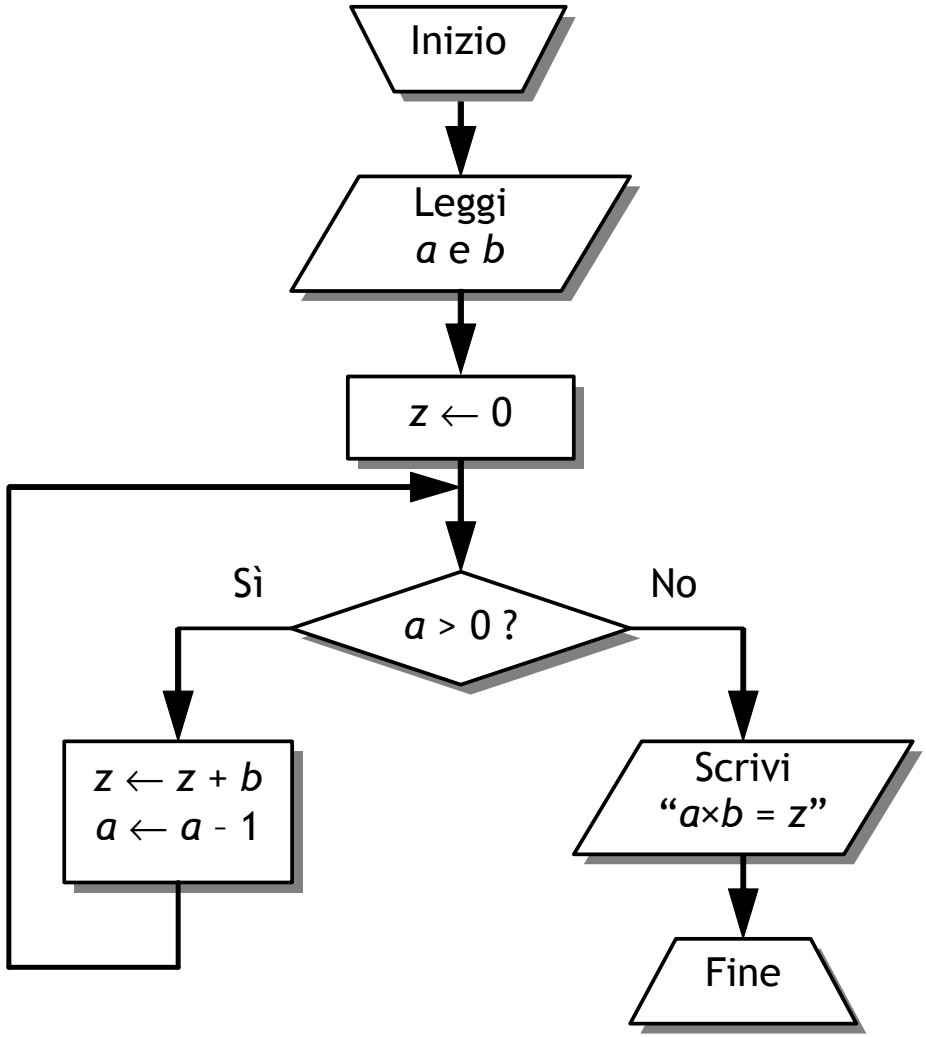
Esempi di algoritmo

Prodotto di due interi positivi

1. Leggi *a*
2. Leggi *b*
3. Somma *b* a se stesso *a* volte
4. Scrivi il risultato

Prodotto di due interi positivi

1. Leggi *a*
2. Leggi *b*
3. *z* ← 0
4. Se (a ≤ 0) salta a 8
5. *z* ← *z* + *b*
6. *a* ← *a* - 1
7. Torna a 4
8. Scrivi "*a* × *b* = *z*"

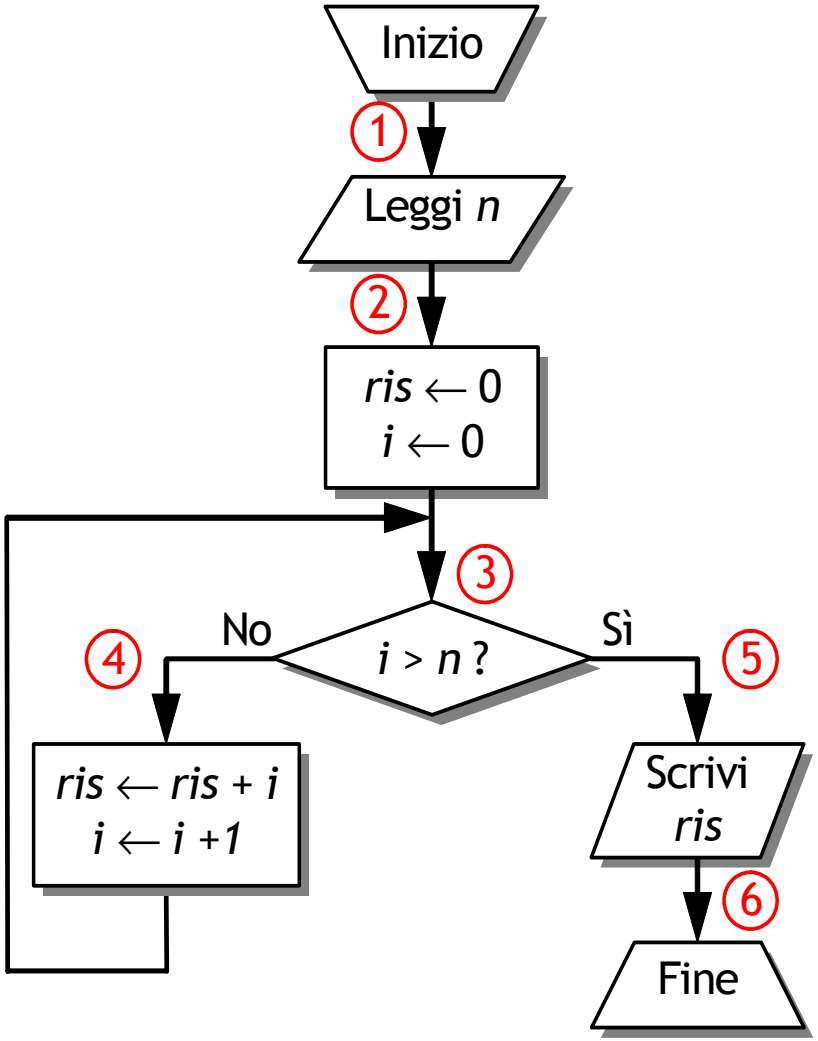


Prodotto di due interi positivi

1. Leggi *a*
2. Leggi *b*
3. *z* ← 0
4. Se (a ≤ 0) salta a 8
5. *z* ← *z* + *b*
6. *a* ← *a* - 1
7. Torna a 4
8. Scrivi "*a* × *b* = *z*"

- Procedimento sequenziale
- Non ambiguo
- Formulazione generale
- Prevede tutti i casi ?? (che succede se *a*<0?)

Somma dei primi N numeri naturali



T	posiz.	*n*	*i*	*ris*	note
t_{01}	①	??	??	??	Variabili non ancora definite
t_{02}	②	4	??	??	Letto il valore 4 e inserito in *n*
t_{03}	③	4	0	0	$i < n \Rightarrow$ posiz. 4
t_{04}	④	4	0	0	
t_{05}	③	4	1	0	$i < n \Rightarrow$ posiz. 4
t_{06}	④	4	1	0	
t_{07}	③	4	2	1	$i < n \Rightarrow$ posiz. 4
t_{08}	④	4	2	1	
t_{09}	③	4	3	3	$i < n \Rightarrow$ posiz. 4
t_{10}	④	4	3	3	
t_{11}	③	4	4	6	$i = n \Rightarrow$ posiz. 4
t_{12}	④	4	4	6	
t_{13}	③	4	5	10	$i > n \Rightarrow$ posiz. 5
t_{14}	⑤	4	5	10	
t_{15}	⑥	4	5	10	Stampato risultato (10)

Somma dei primi N numeri naturali