Elaborazione e Strutturazione dell'Informazione

Capitolo 3

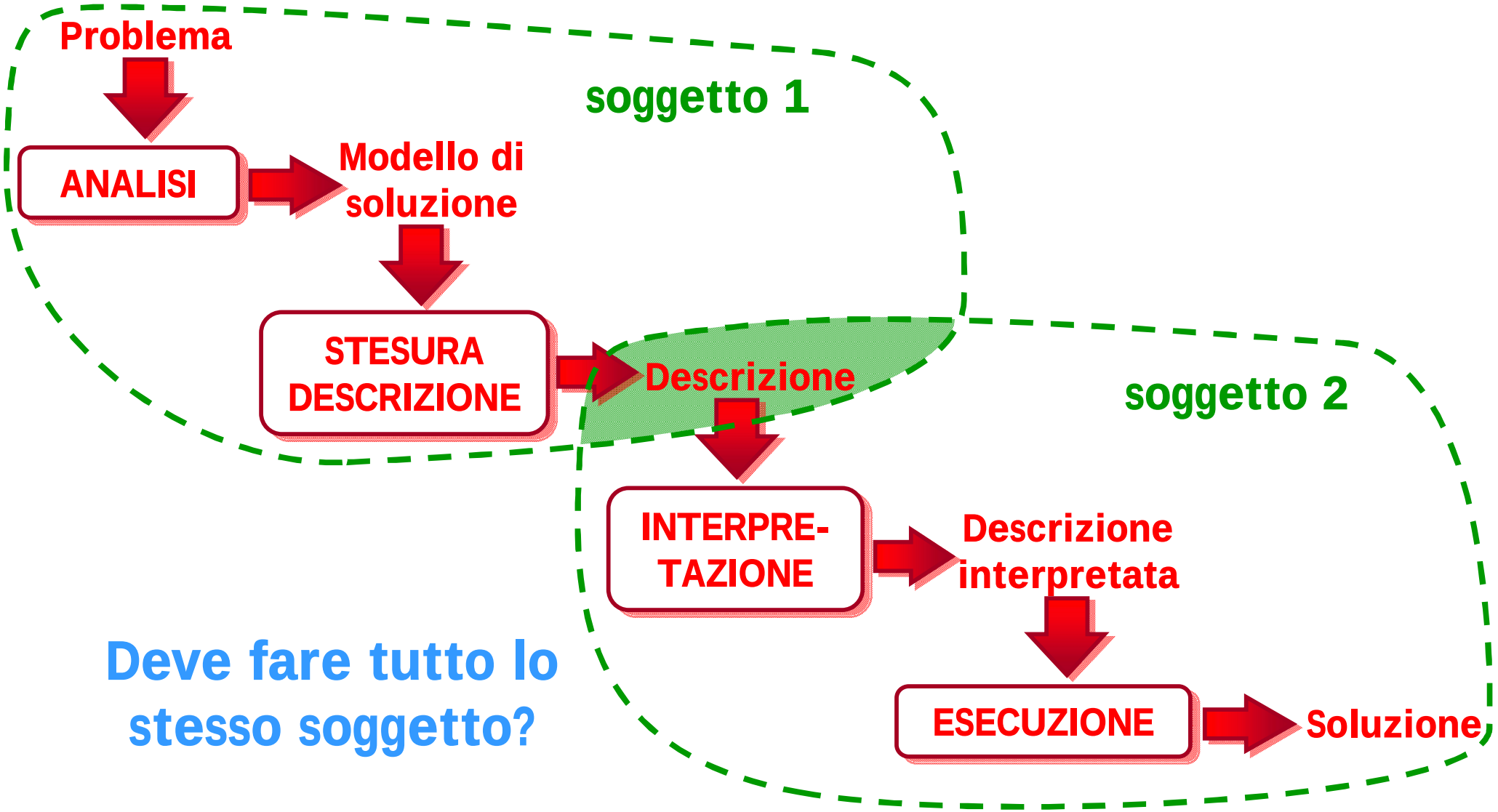
(Introduzione ai Sistemi Informatici – Sciuto et alii)

Il calcolatore come strumento per gestire informazione

tradizionalmente l'enfasi è sui

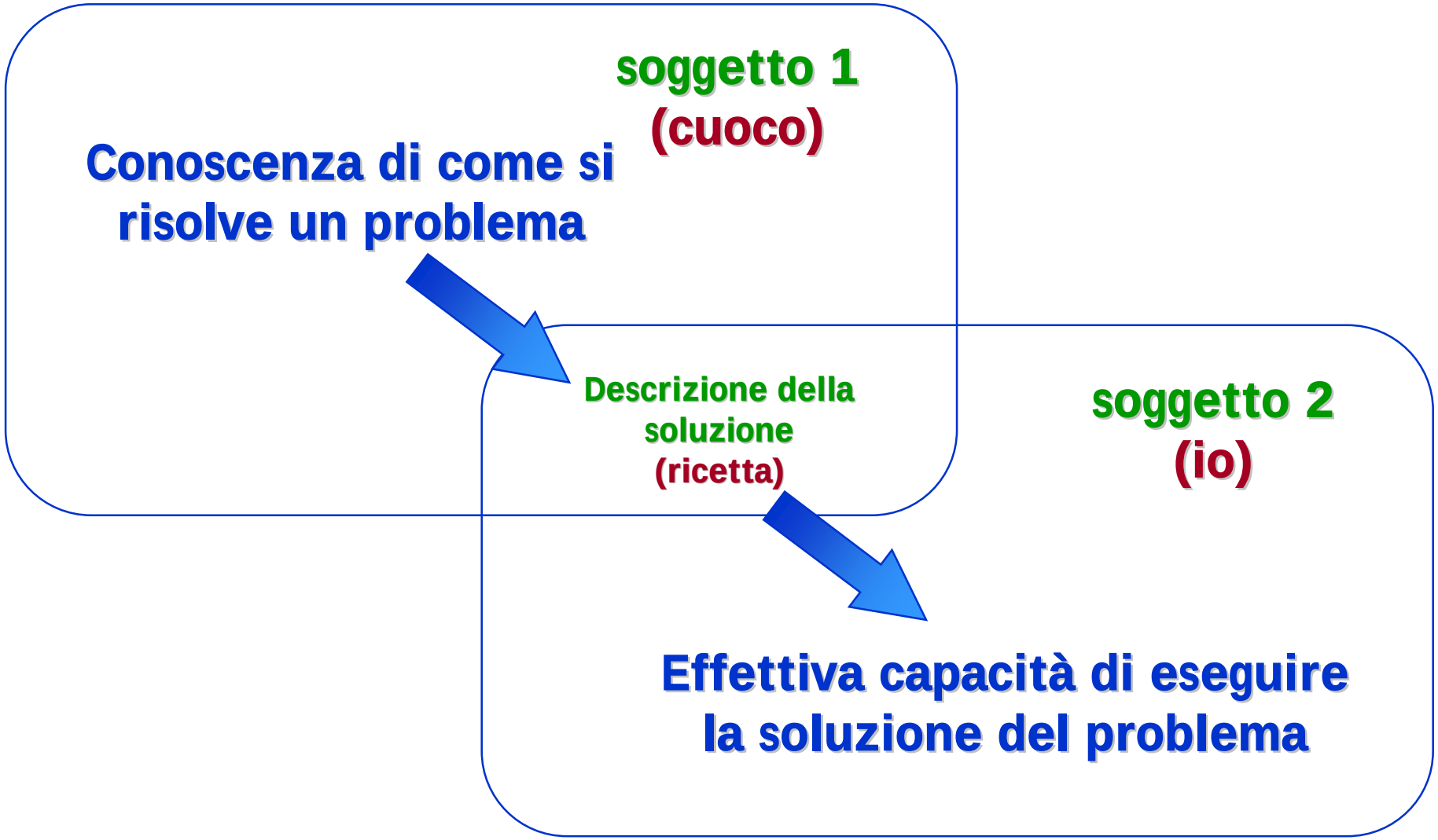
sistemi di elaborazione delle informazioni ...

Soluzione di un problema



Deve fare tutto lo stesso soggetto?

La soluzione di un problema



Problemi ed 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 π**;
  - ... ⇒ "**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***), risolvibili 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 <u>successione di azioni</u> 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.

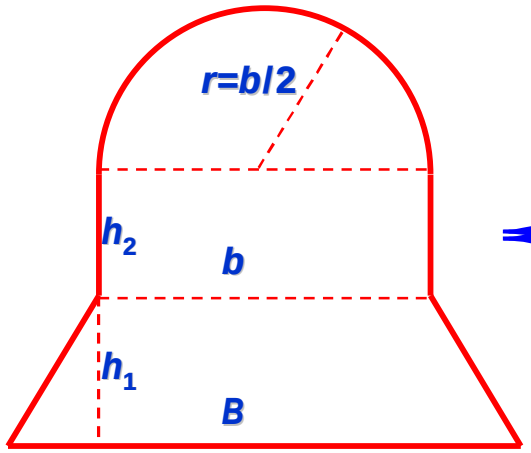
Soluzione effettiva

- **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**

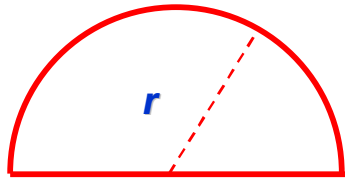
Area di una campana (1)

Problema



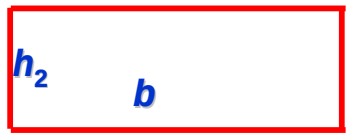
soluzione *elementare*:
$s = ??$

Scomposizione del problema in tre sottoproblemi



Sottoproblema 1

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



Sottoproblema 2

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

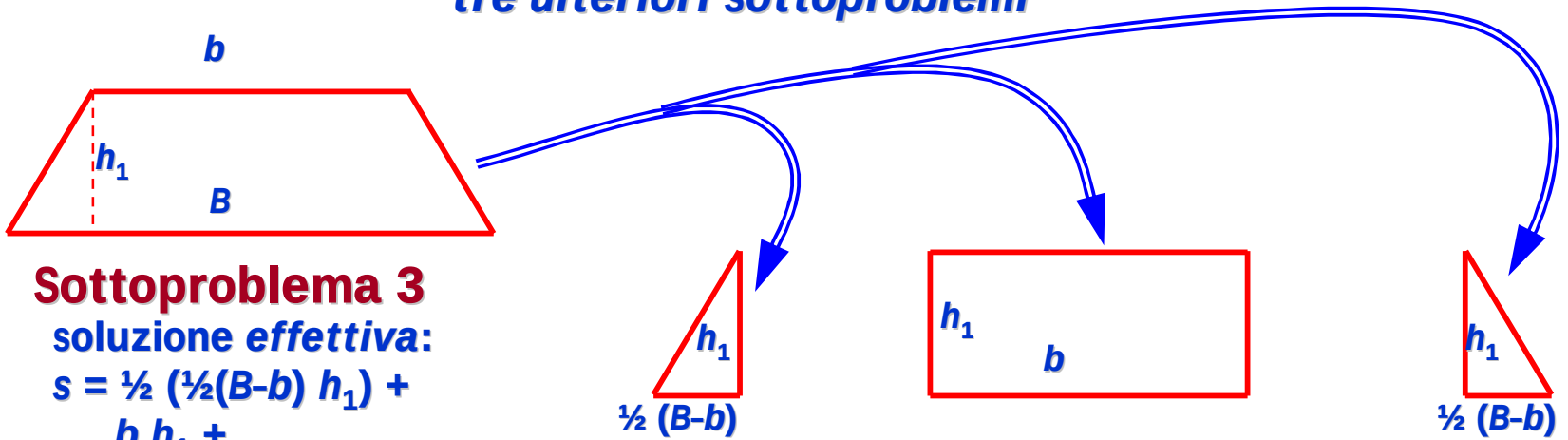
Sottoproblema 3

soluzione *elementare*:

$s = ??$

Area di una campana (2)

Scomposizione del sottoproblema 3 in tre ulteriori sottoproblemi



Sottoproblema 3
soluzione *effettiva*:
$s = ½ (½(B-b)\ h_1) + b\ h_1 + ½ (½(B-b)\ h_1)$

Sottoproblema 3.1
soluzione *elementare*:
$s = ½ (½(B-b)\ h_1)$

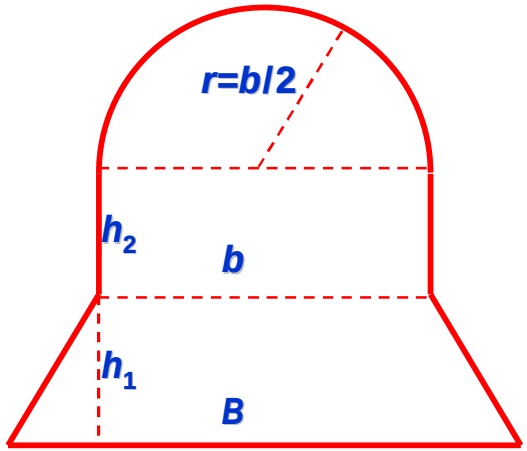
Sottoproblema 3.2
soluzione *elementare*:
$s = b\ h_1$

Sottoproblema 3.3
soluzione *elementare*:
$s = ½ (½(B-b)\ h_1)$

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

Area di una campana (3)

Problema

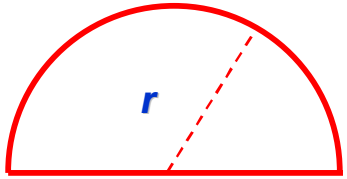


soluzione *effettiva*:

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

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

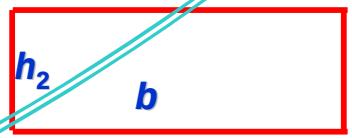
Sottoproblema 1



soluzione *elementare*:

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

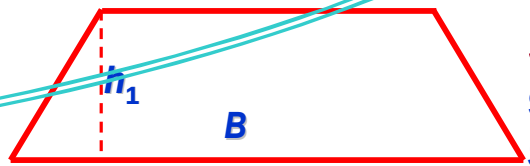
Sottoproblema 2



soluzione *elementare*:

$s = b\ h_2$

Sottoproblema 3



soluzione *effettiva*:

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

Algoritmo (definizione informale)

- **Sequenza finita di istruzioni,**
- **comprensibili da un esecutore (automatico),**
- **che descrive come realizzare un compito (come risolvere un "problema").**

- **Alcuni esempi**
  - Istruzioni di montaggio di un elettrodomestico
  - Uso di un terminale Bancomat
  - Calcolo del massimo comune divisore di numeri naturali

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 (associa alle istruzioni le corrispondenti operazioni)
  - eseguire 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

  ---> **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

- **Completezza**

  descrivono la soluzione non di un singolo problema, ma di una intera classe di problemi strutturalmente equivalenti.

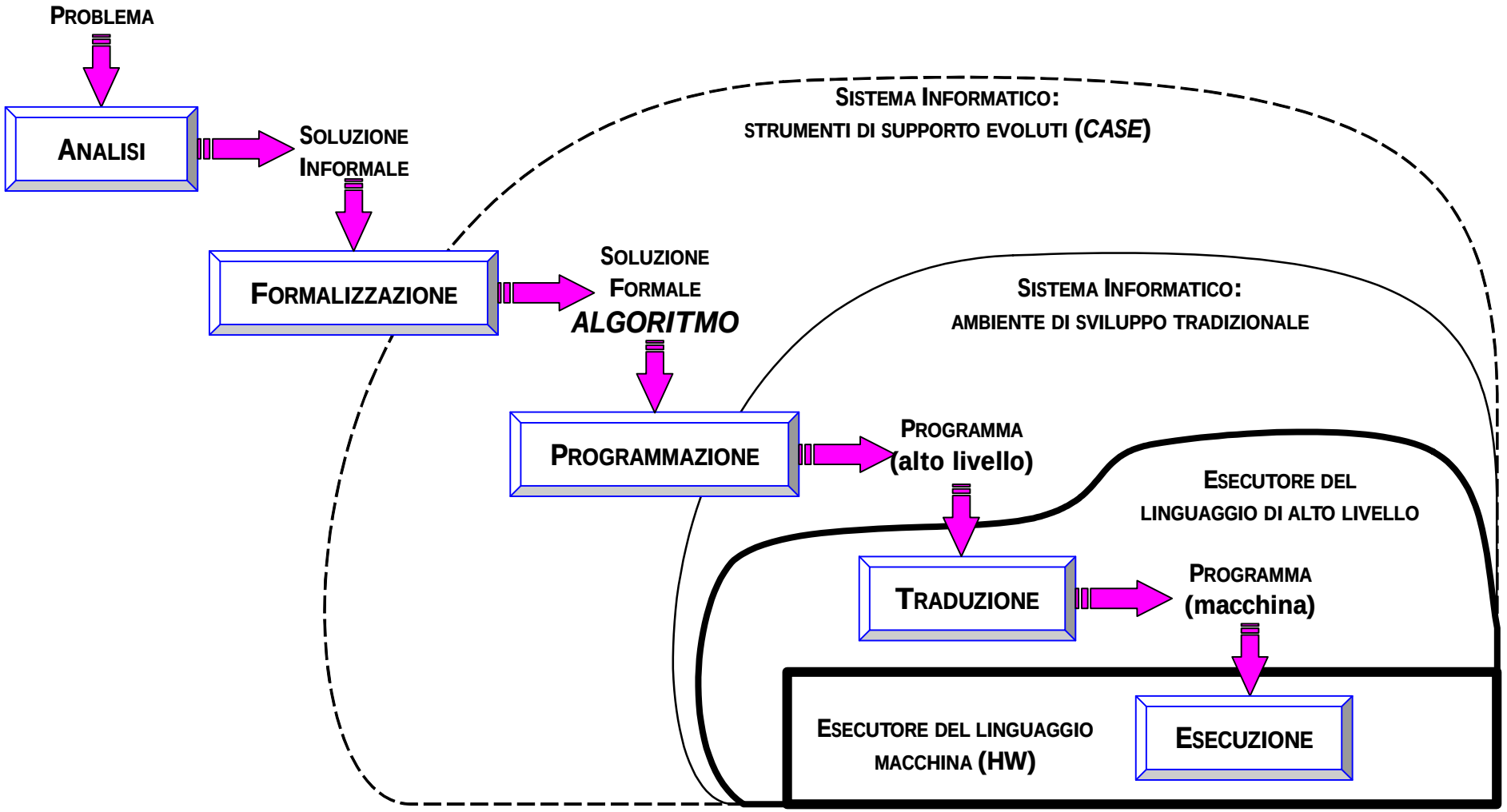
- **Efficienza**
  - L'algoritmo perviene alla soluzione del problema usando la minima quantità di risorse fisiche
    - tempo di esecuzione, memoria, …

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** (capire la sequenza di passi che portano alla soluzione di un problema)
  - **codificarli in programmi** (renderli comprensibili al calcolatore)

Sviluppo di un programma



Algoritmi

Formalizzazione

Codifica degli algoritmi

- **Algoritmo formulato per essere comunicato tra esseri umani (*il nostro punto di vista*)**
  - 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 (parametri) 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 (o parametri)*, il cui valore non è fissato a priori ma cambia a seconda della situazione elaborativa in cui l'esecutore si trova.**

Uso delle variabili (1/3)

- **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),

    sono variabili che contengono valori

Uso delle variabili (2/3)

- **in istruzioni di assegnamento**
  - introdurre nel contenitore identificato dal nome della variabile il valore specificato a destra dell'assegnamento (←);
  - ***Esempio:***

r ← 35

(assegna 35 alla variabile il cui nome è **r**)

pi ← 3,14

...

Uso delle variabili (3/3)

- **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**.

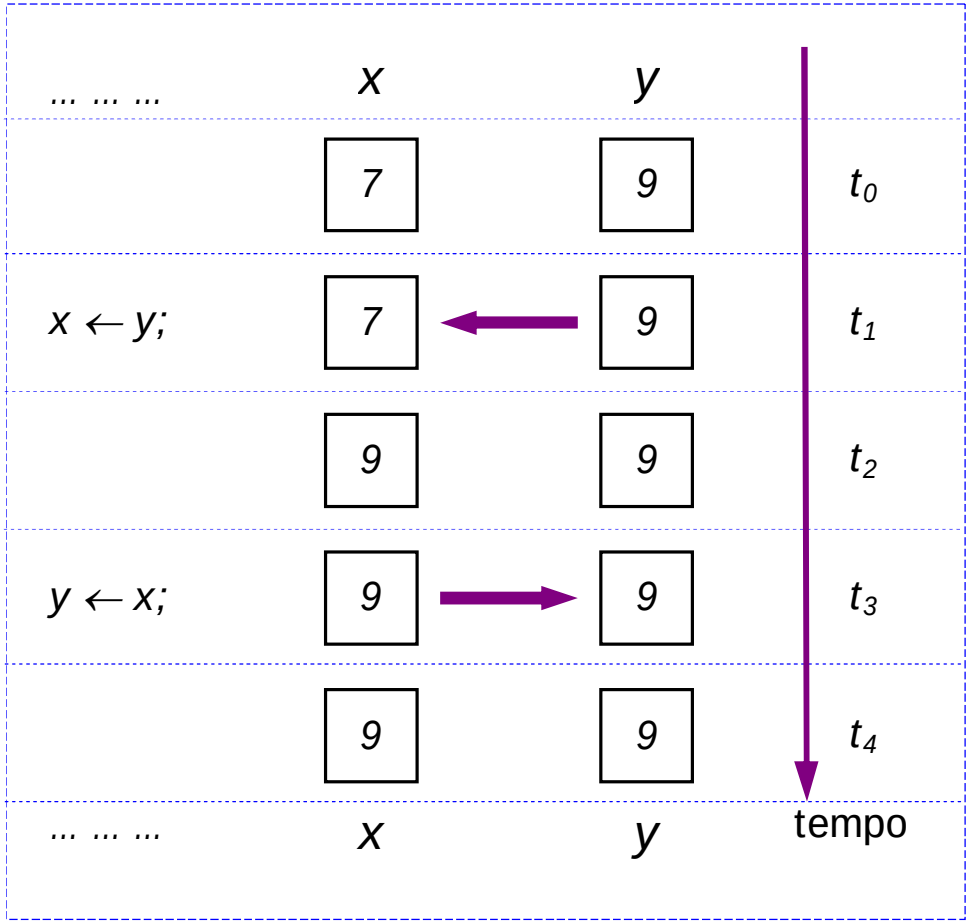
Assegnamento di valori a variabili

- **Esempio:** si ipotizzi di voler scambiare i valori contenuti in due variabili *x* e *y*.
- **Pseudo-soluzione proposta:** doppio assegnamento del tipo

  $x \leftarrow y$

  $y \leftarrow x$

  per indicare che il valore di *y* deve essere copiato in *x* e che, nello stesso tempo, il valore di *x* sia trasferito in *y*.
- **Le istruzioni però vengono eseguite in sequenza!** Quindi l'assegnamento $x \leftarrow y$ viene completato prima di iniziare $y \leftarrow x$.

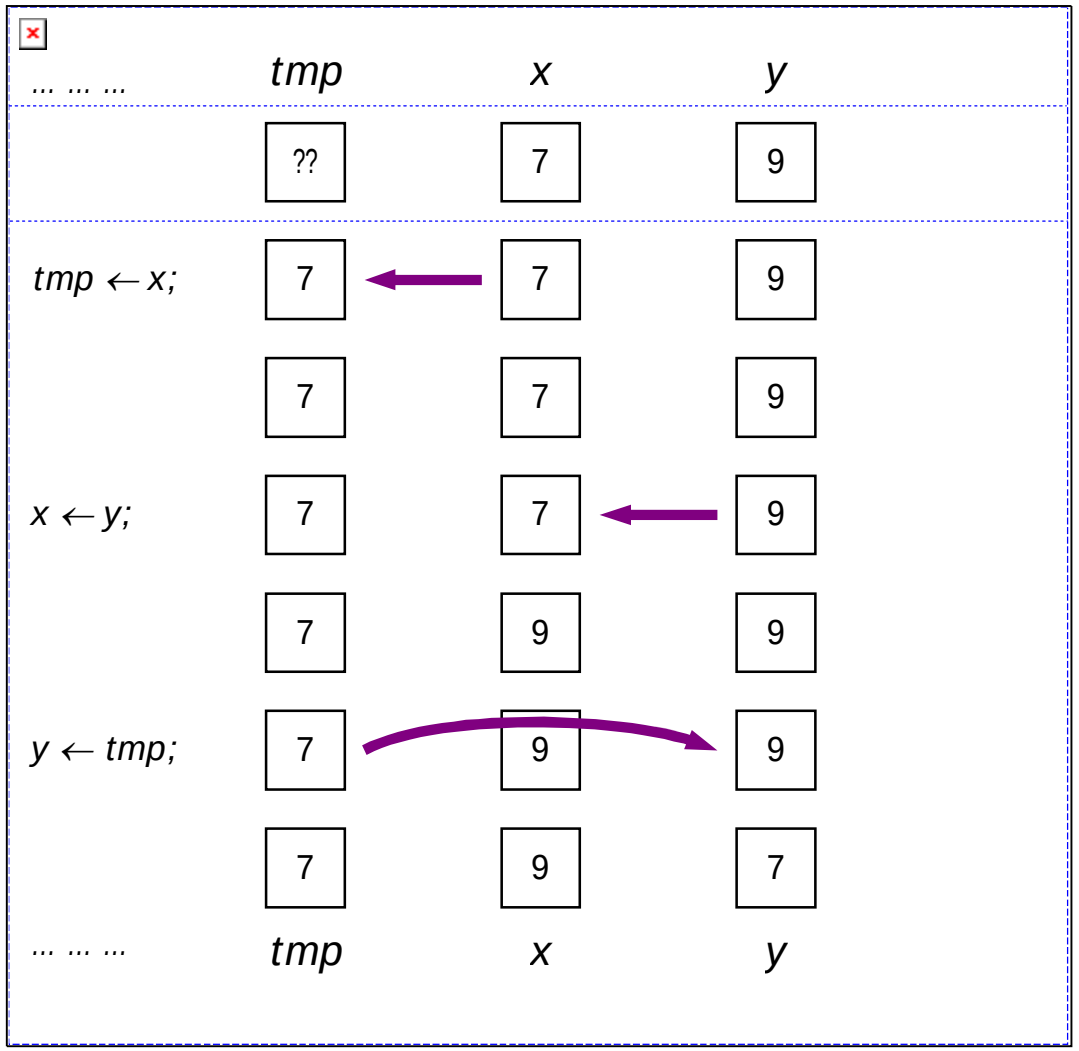


Assegnamento di valori a variabili

Soluzione corretta:
uso di una variabile aggiuntiva (*tmp*), come strumento di memorizzazione temporanea ("buffer") del valore originariamente contenuto in *x*

$tmp \leftarrow x$
$x \leftarrow y$
$y \leftarrow tmp$

In questo modo lo scambio avviene senza perdere i valori originari



Dati e istruzioni

- Tipi di dati
  - Numeri naturali, interi, 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…ALLORA*, *RIPETI*)

Criteri di classificazione dei dati

- **Visibilità da parte dell'utente**
  - visibile (di ingresso o uscita)
  - trasparente (dati temporanei di supporto)
- **Variabilità nel tempo**
  - costanti
  - variabili (acquisizione dall'esterno o assegnazione)
- **Struttura**
  - elementari (interi, alfanumerici, booleani, ...)
  - strutturati (array, matrici, ...)

Operazioni elementari

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

- **Condizioni sul valore di singole variabili**
  - se A allora C, altrimenti D

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

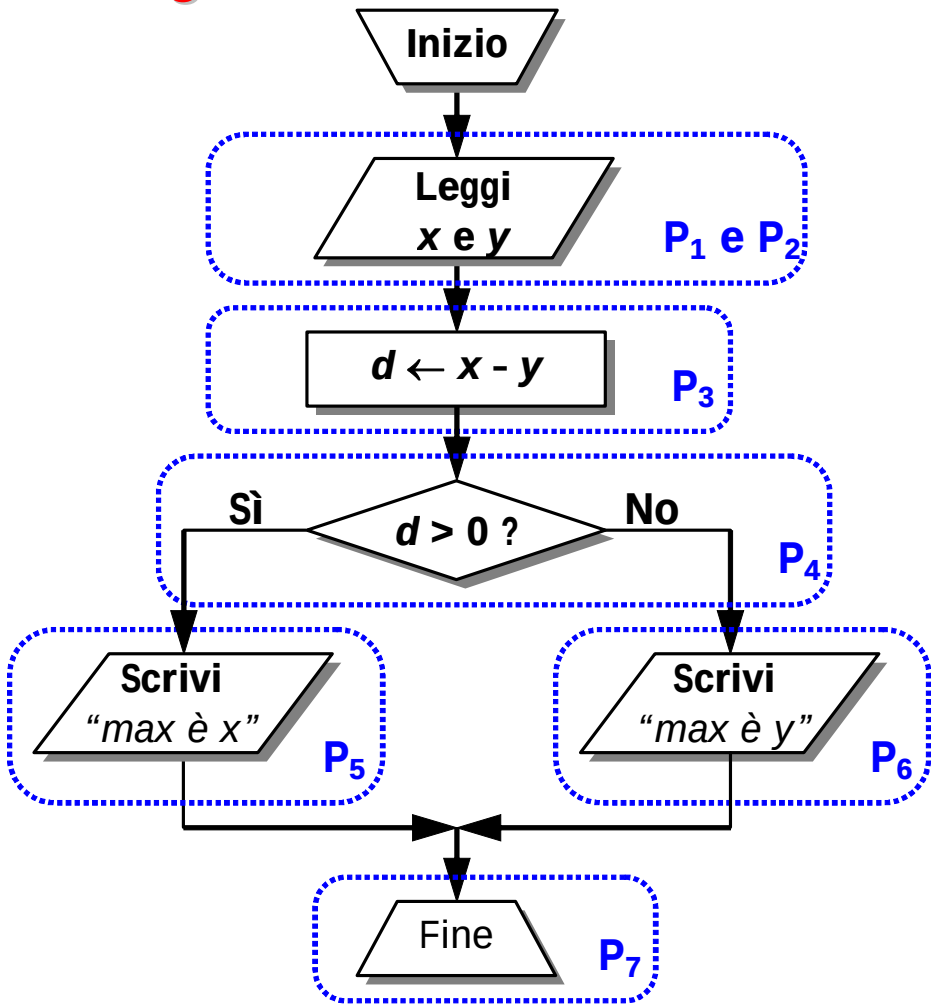
Rappresentazione degli algoritmi

- Linguaggio naturale
- Diagramma a blocchi
- Macchine di Turing
- 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**

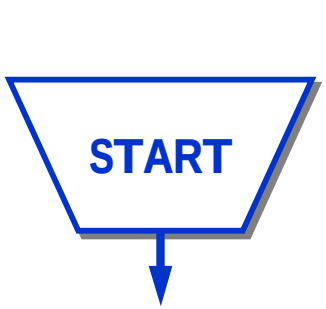
```
#include <stdio.h>

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

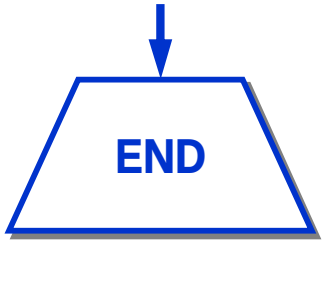
Diagrammi di flusso
(o algortimi a blocchi):

algoritmi visti/scritti come grafi

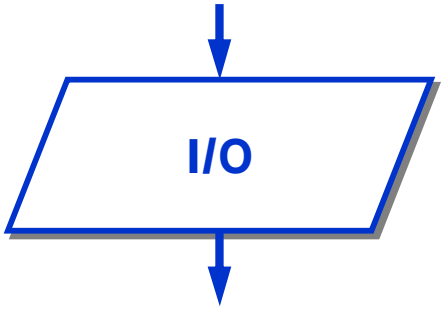
Diagrammi di Flusso: sintassi



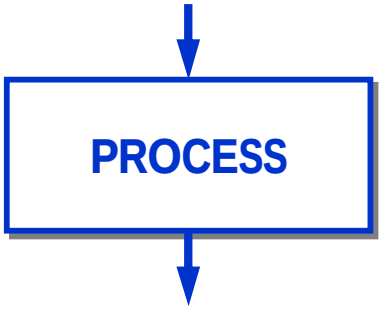
Inizio



Fine

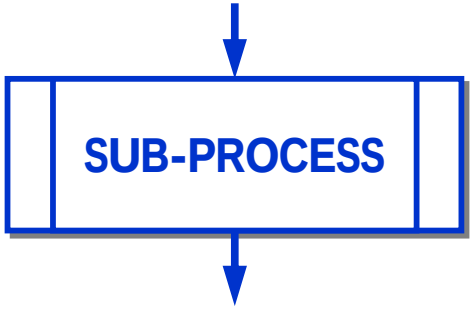


Operazioni di ingresso/uscita leggi/scrivi



Elaborazione

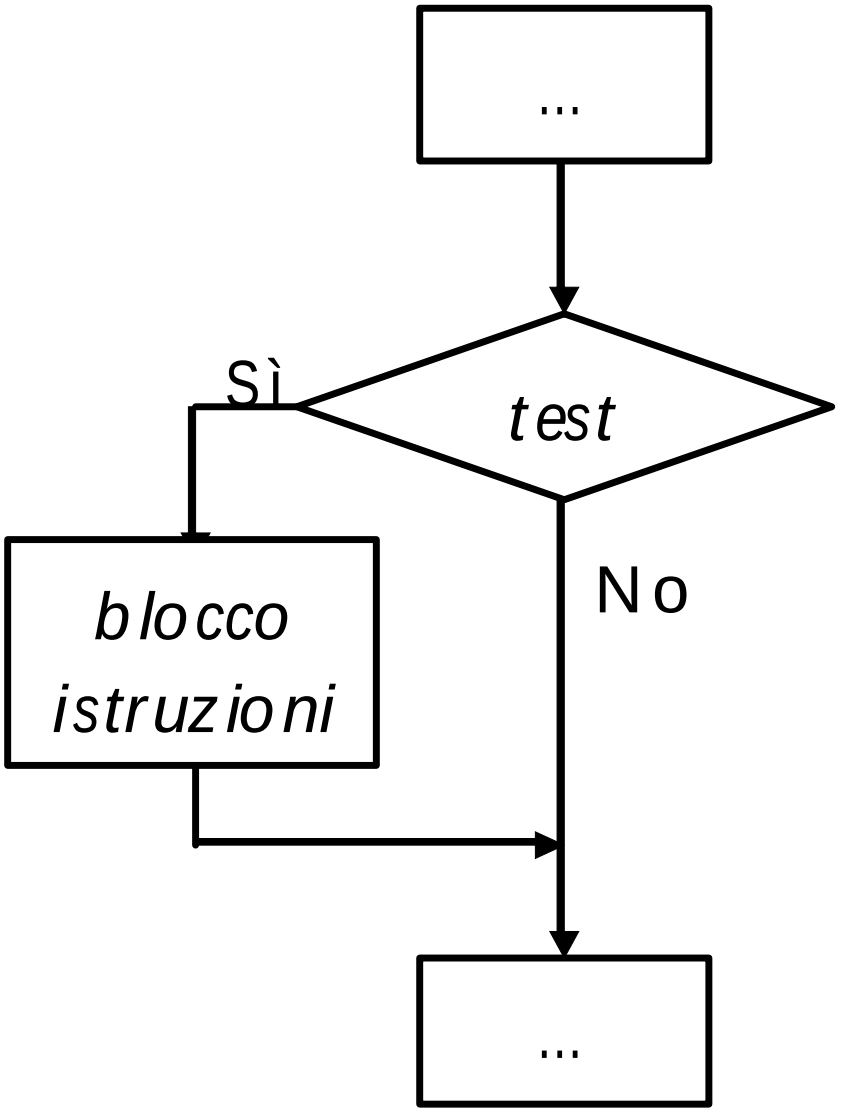
Selezione a due vie



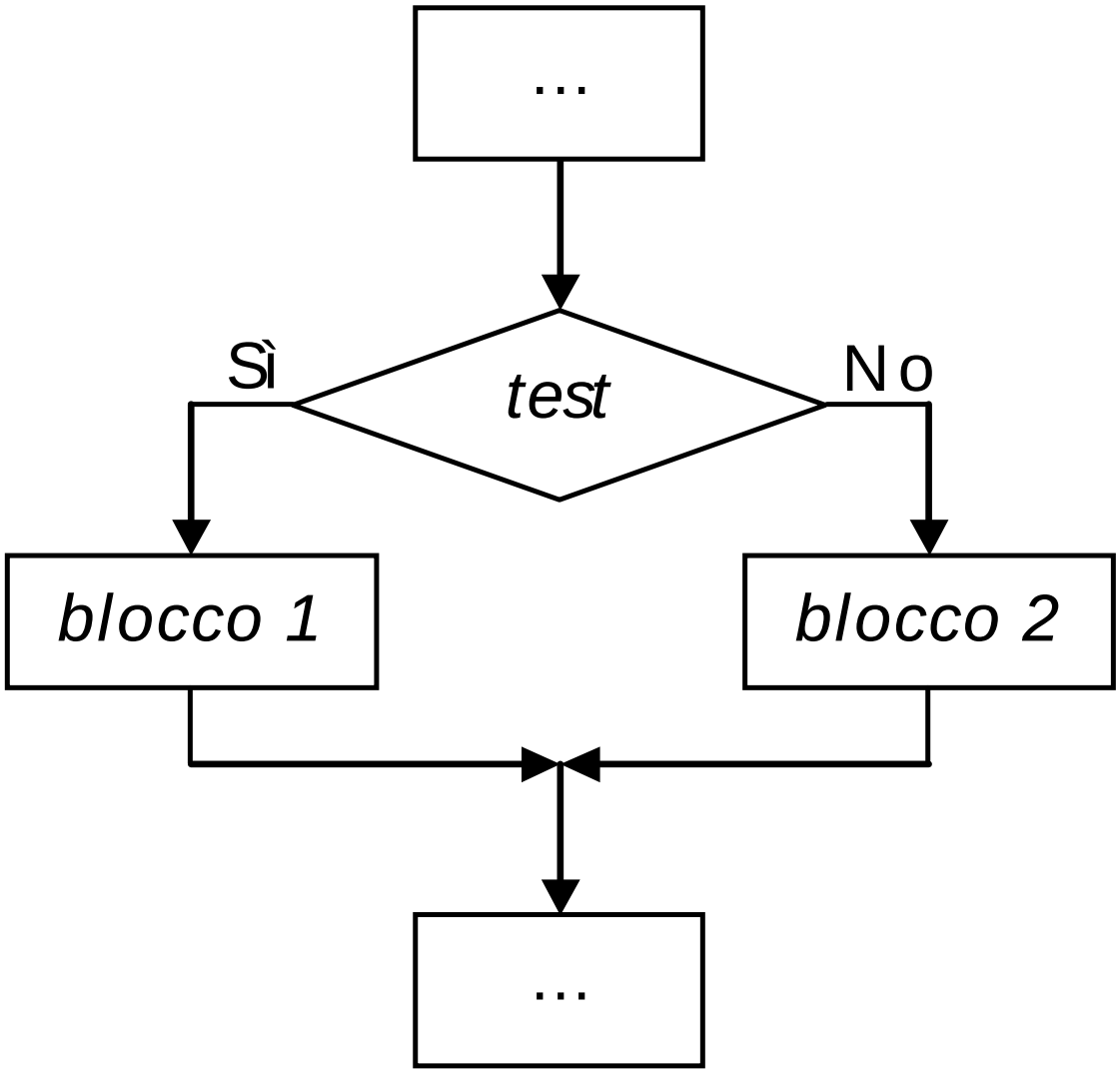
Sottoprogramma

Le strutture di controllo

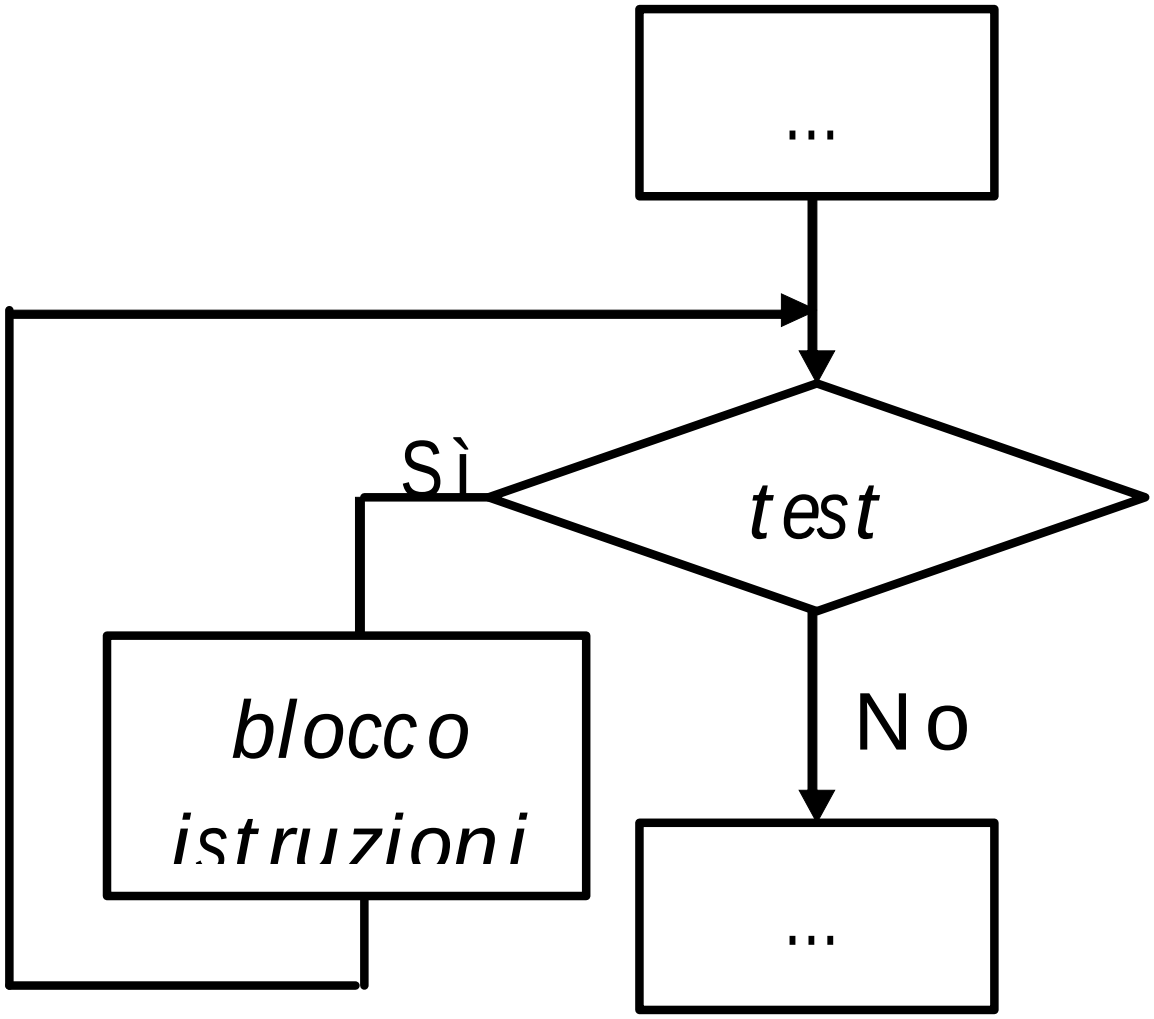
Selezione semplice



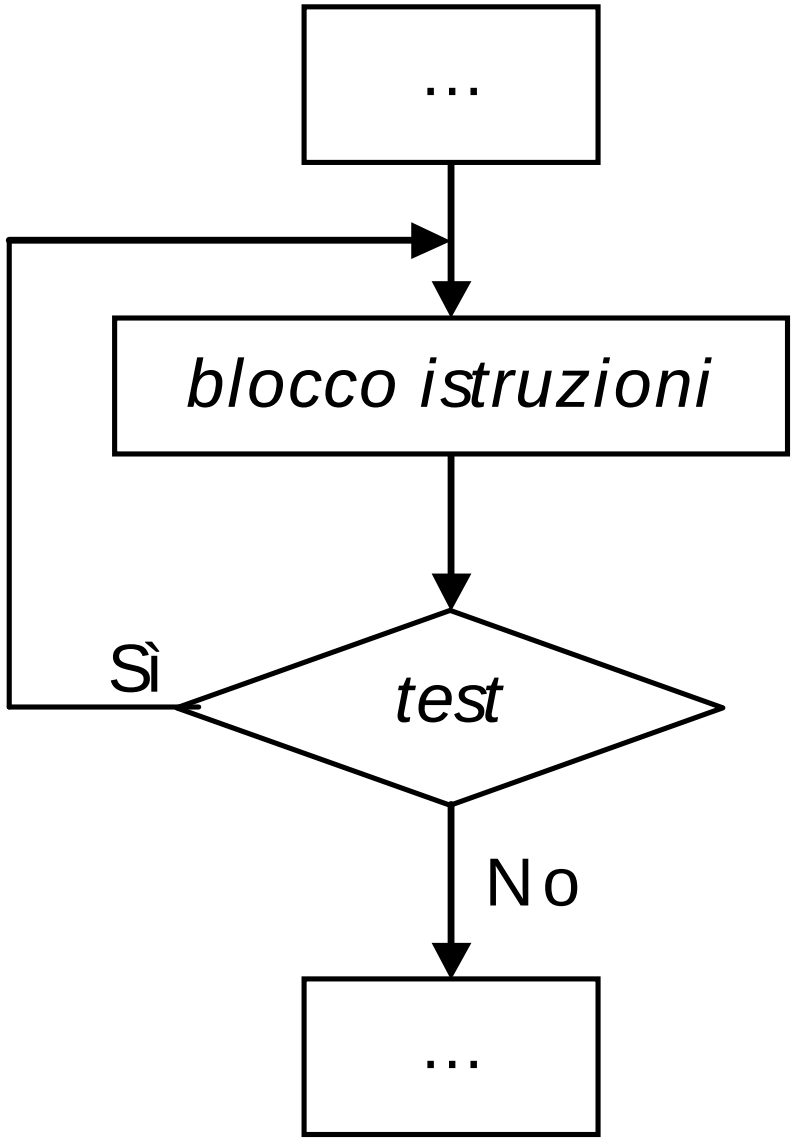
Selezione a due vie



Ciclo a condizione iniziale

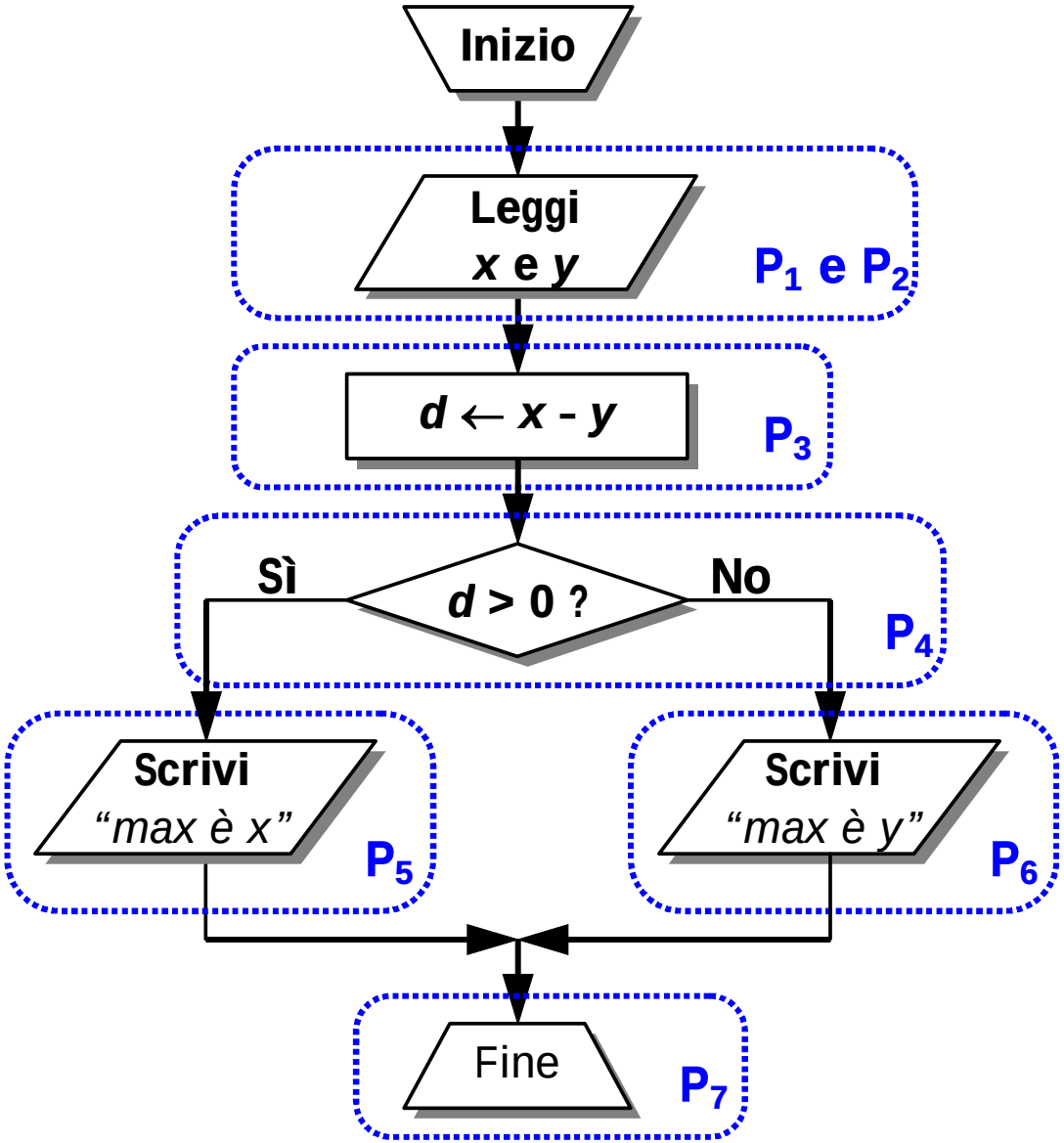


Ciclo a condizione finale



Esempio 1

calcolare il maggiore (max) di due diversi interi X e Y

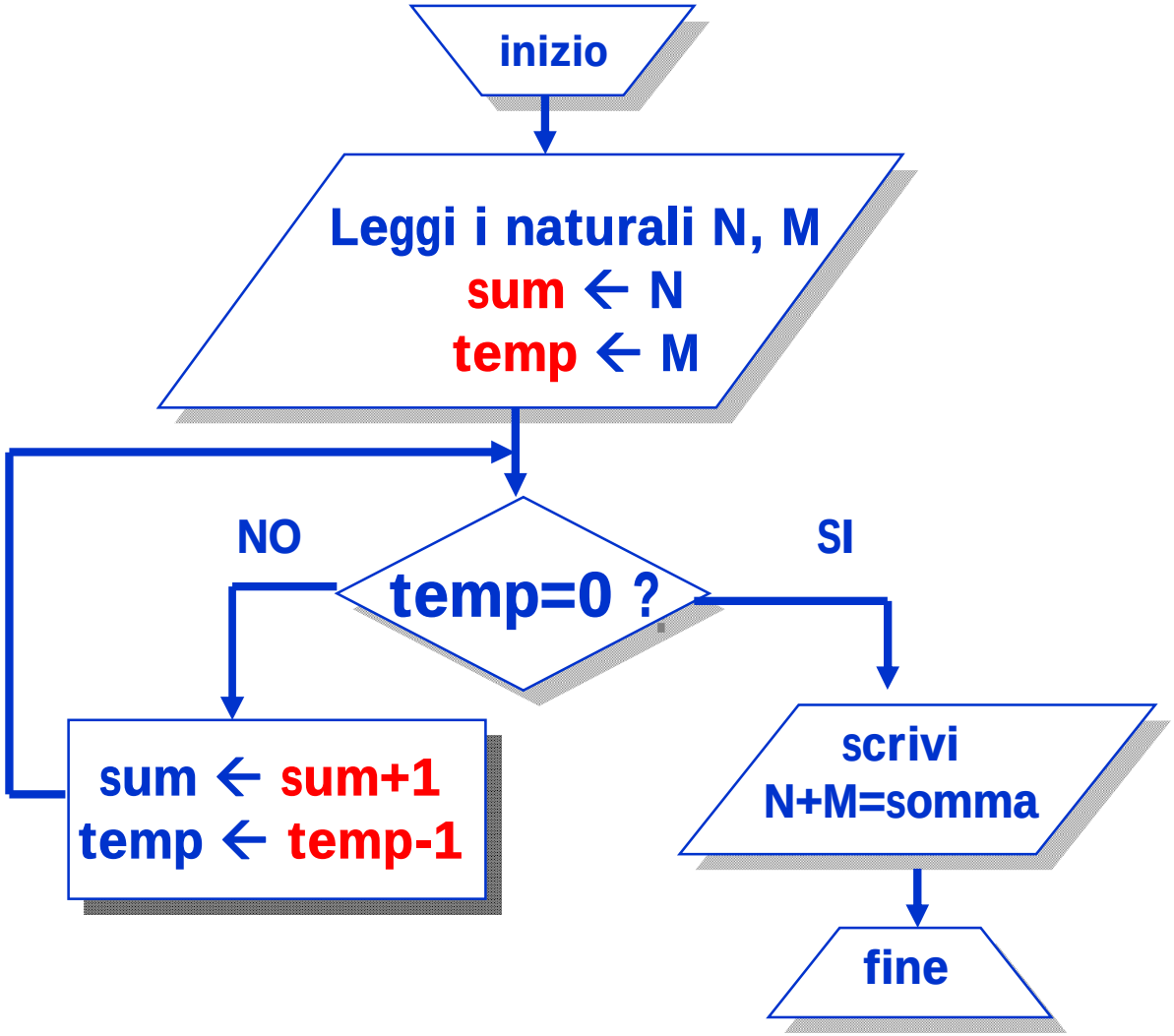


Esempio 2

Data la funzione successore e predecessore di un numero, calcolare la somma di due naturali N ed M

(Idea: N+M equivale a sommare M unità ad N)

```
Esempio di Computazione:
calcola N+M con N=3 e M=2
0) sum=3      temp=2
1) sum=3+1    temp=2-1
3) sum=4+1    temp=1-1
4) sum=5      temp=0
```

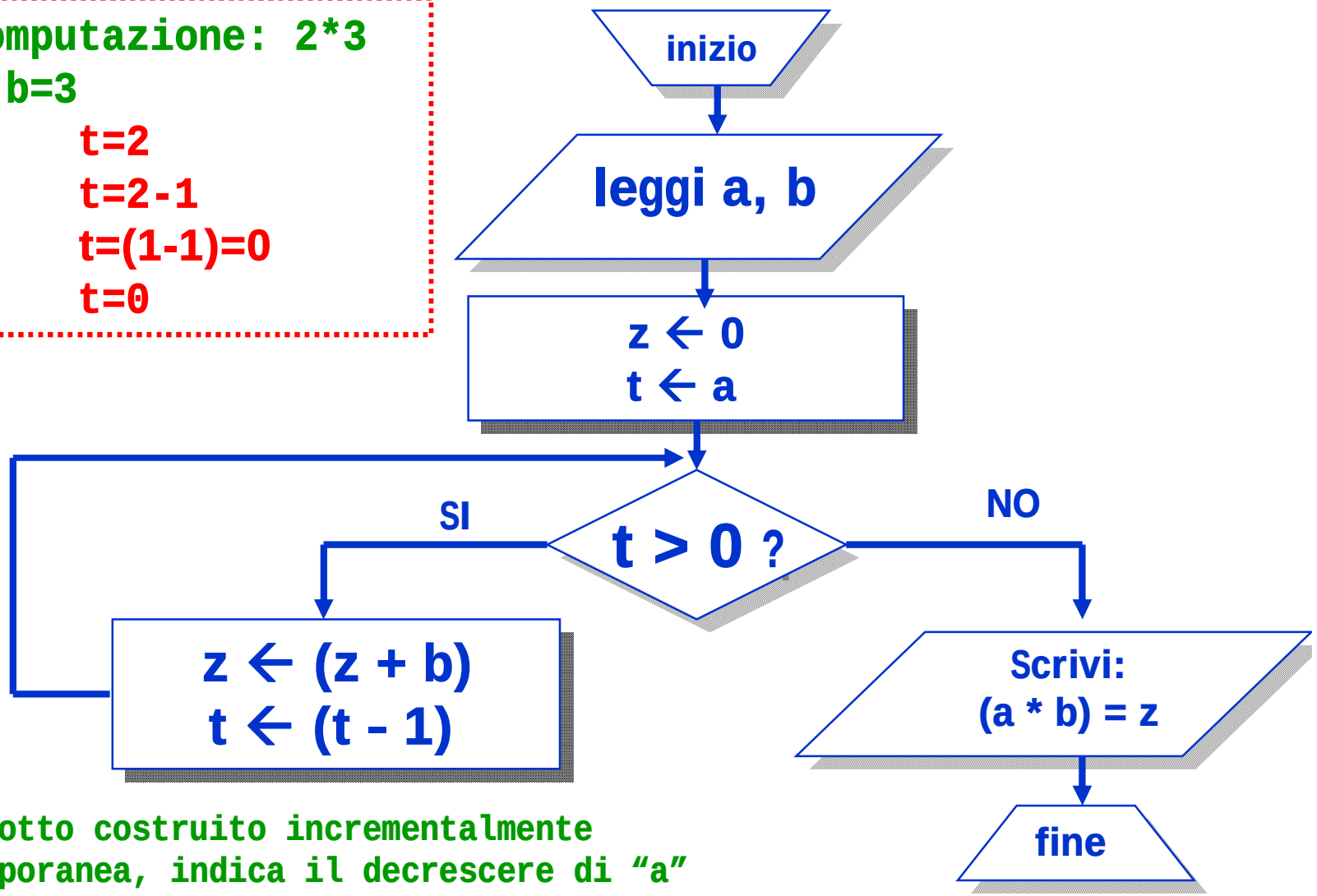


Esempio 3

Data l'operazione di somma aritmetica, calcolare il prodotto di due interi a, b

(Idea: a*b equivale a sommare "b" un numero di volte pari ad "a")

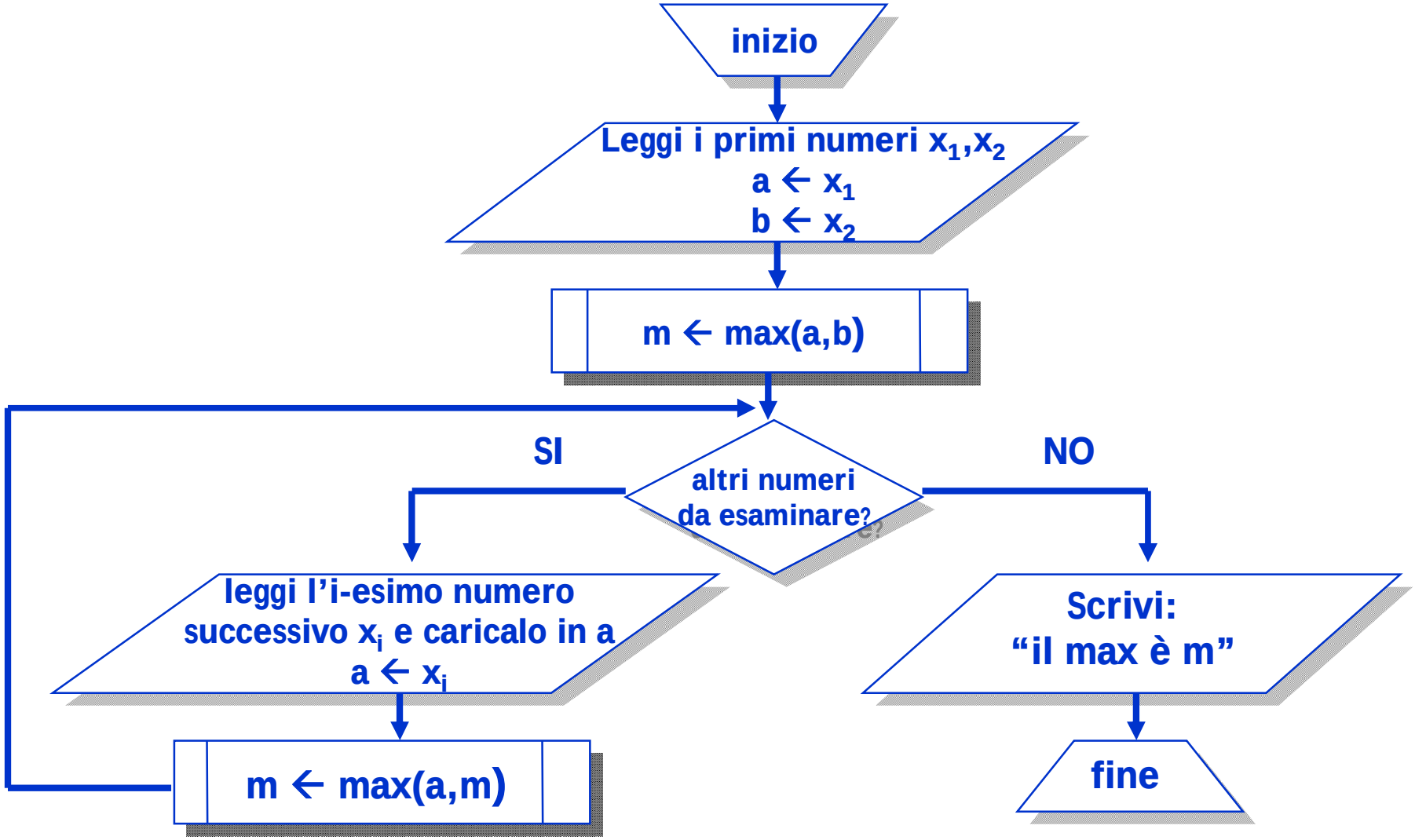
```
Esempio di Computazione: 2*3
          a=2 b=3
0) z=0            t=2
1) z=(0+3)=3      t=2-1
2) z=(3+3)=6      t=(1-1)=0
3) z=2*3=6        t=0
```



NOTA:
"z" indica il prodotto costruito incrementalmente
"t", variabile temporanea, indica il decrescere di "a"

Esempio 4

calcolare il max di una sequenza di interi $x_1, x_2, x_3, \ldots, x_{n-1}, x_n$



Esempio 5

dato N, calcolare la somma dei numeri naturali da 0 ad N

Computazione (astratta):

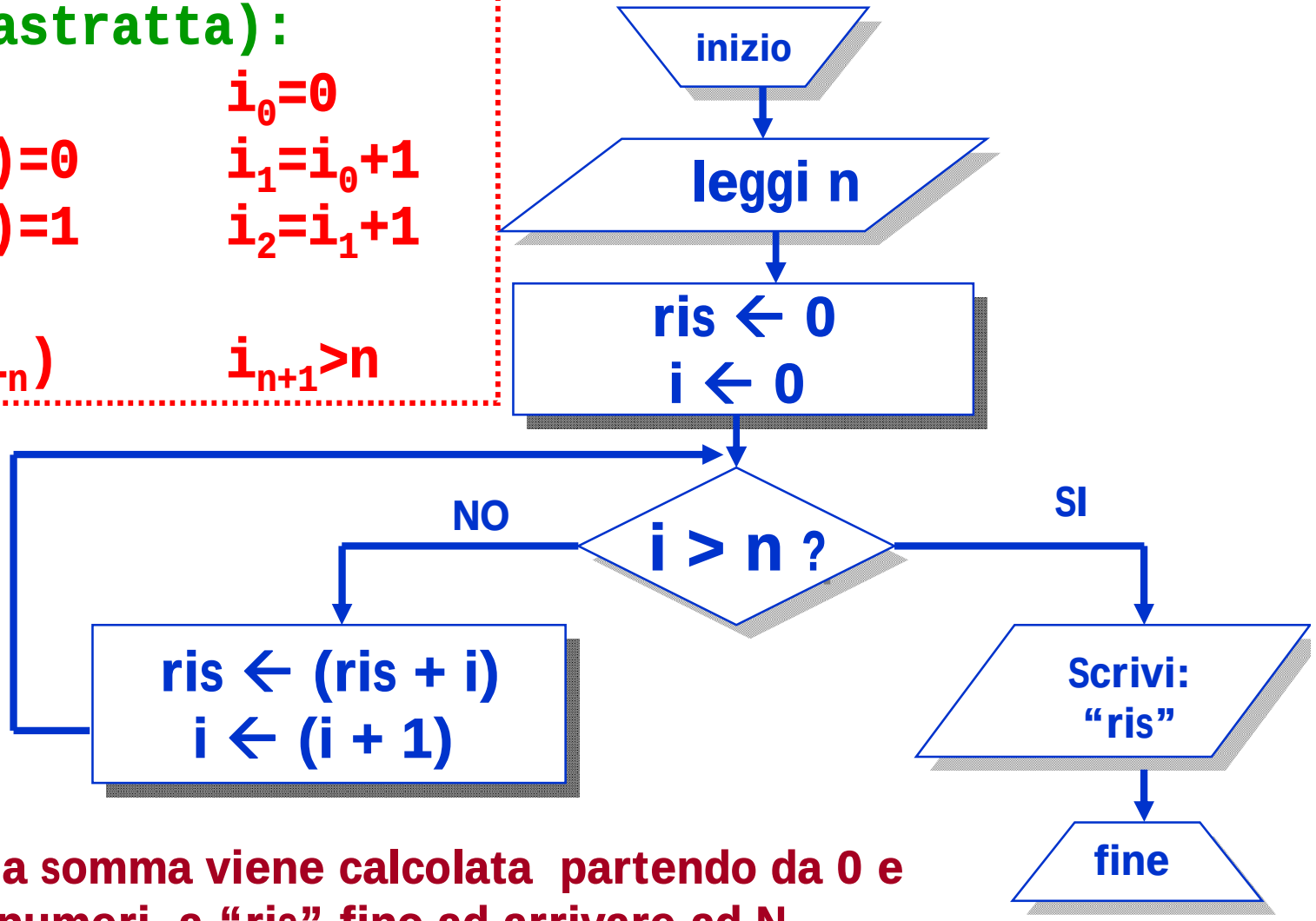
$ris_0 = 0$ $\quad i_0=0$

$ris_1 = (ris_0+i_0)=0$ $\quad i_1=i_0+1$

$ris_2 = (ris_1+i_1)=1$ $\quad i_2=i_1+1$

…

$ris_{n+1} = (ris_n+i_n)$ $\quad i_{n+1}>n$



In questo algoritmo la somma viene calcolata partendo da 0 e aggiungendo vi via i numeri a "ris" fino ad arrivare ad N

Esempio 5bis

dato N, calcolare la somma dei numeri naturali da 0 ad N

Computazione

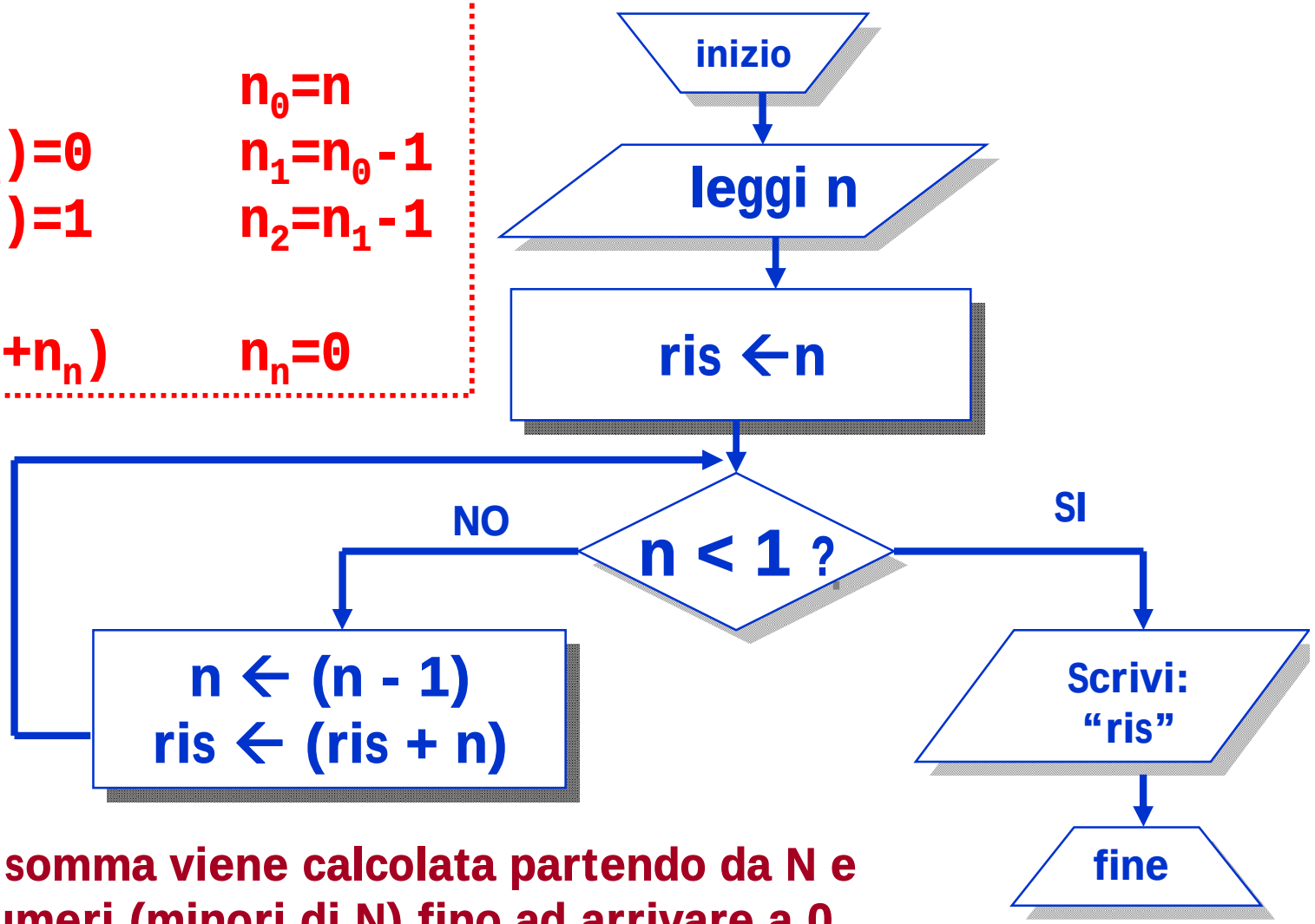
$ris_0 = n$ $\quad n_0=n$

$ris_1 = (ris_0+n_1)=0$ $\quad n_1=n_0-1$

$ris_2 = (ris_1+n_2)=1$ $\quad n_2=n_1-1$

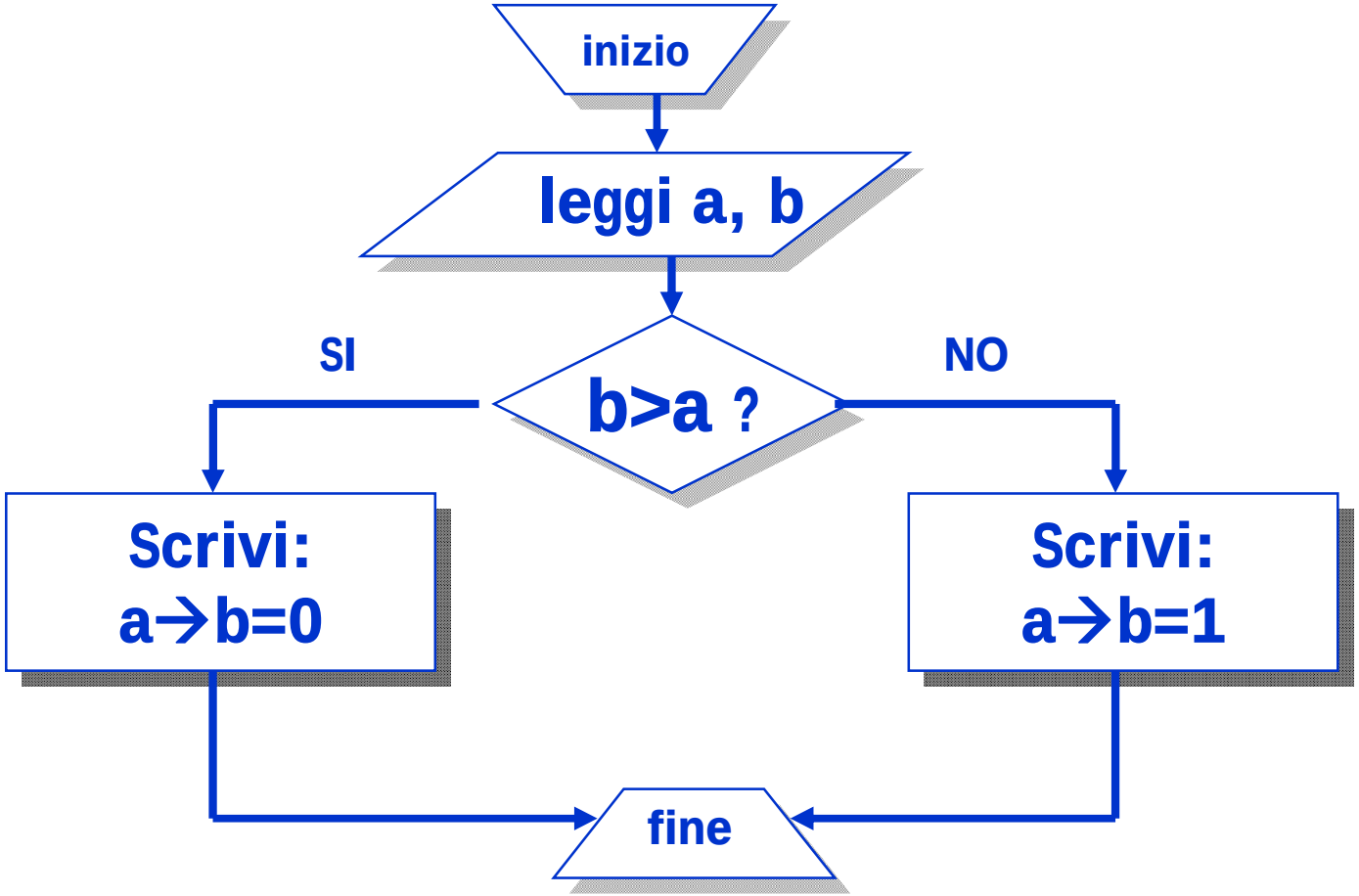
…

$ris_n = (ris_{(n-1)}+n_n)$ $\quad n_n=0$



In questo algoritmo la somma viene calcolata partendo da N e aggiungendo vi via i numeri (minori di N) fino ad arrivare a 0. Non serve il paramentro “i” come nell’algoritmo precedente!

Esempio 6: scrivere l'algoritmo a blocchi che calcola l'*implicazione logica (se A allora B)*



"a" sta per l'antecedente e "b" per il conseguente dell'implicazione logica

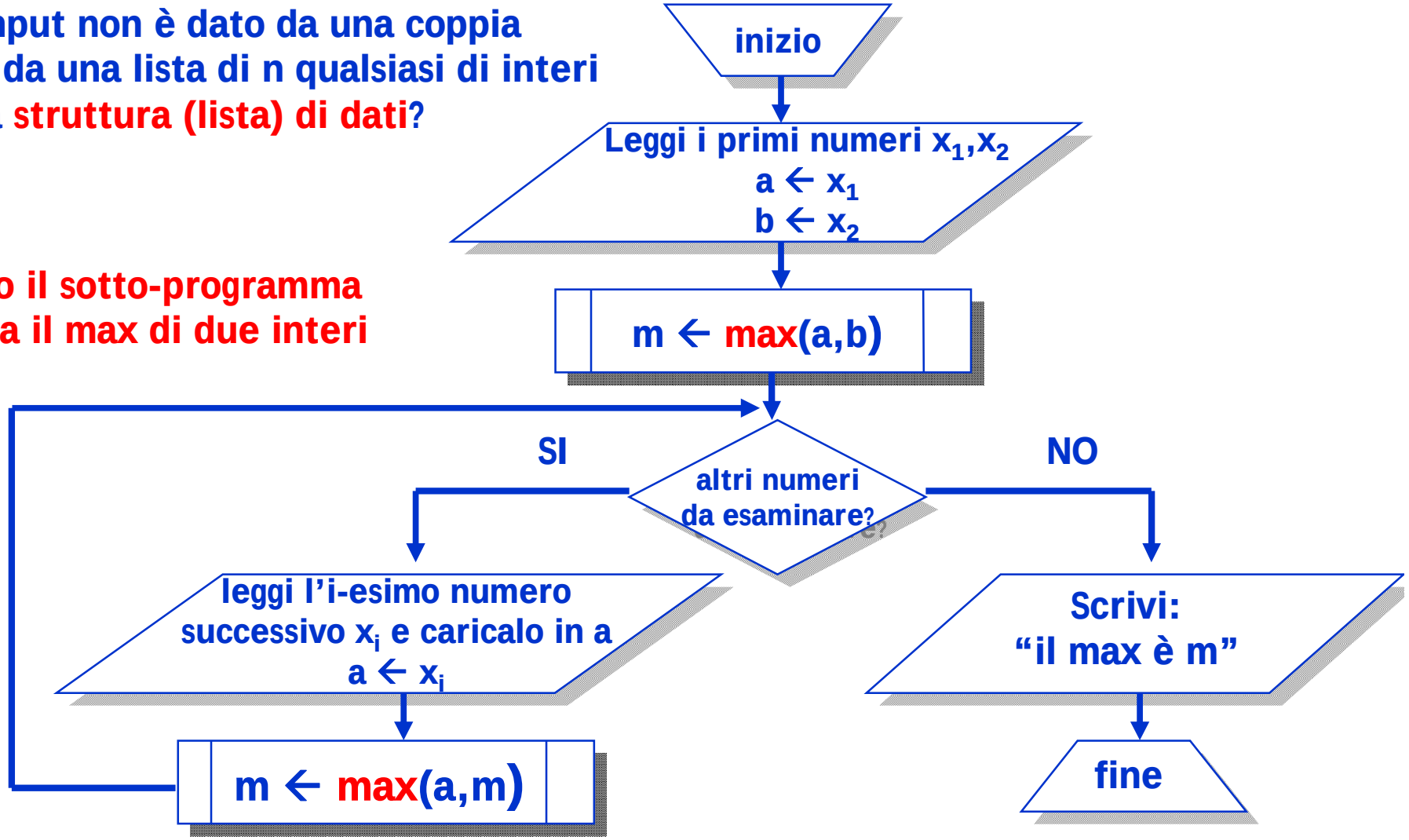
Algoritmi e Strutture dati

Esempio2: calcola il max di una lista di N interi $[x_1,x_2,x_3,\ldots,x_{n-1},x_n]$

problema come esprimere la lista di interi?

Osserva: l'input non è dato da una coppia di interi ma da una lista di n qualsiasi di interi
Occorre una struttura (lista) di dati?

Assumiamo il sotto-programma che calcola il max di due interi



Strutture dati

- **Fondamentali**
  - **strutture dati: problema duale** rispetto a quello della definizione di algoritmi
  - **algoritmi + strutture dati = programmi**”
  - l’uso di strutture dati adeguate semplifica la soluzione di un problema
- **Classificazione**
  - Strutture dati **statiche**
  - Strutture dati **dinamiche**

Strutture dati statiche

Struttura e dimensioni fisse

Array o Vettore: insieme ordinato di N **variabili omogenee** identificate dalla **posizione** all'interno della struttura (unidimensionale)

Esempio: una lista di 5 interi **[-3,7,-12,76,-4]**

- Variante M-dimensionale: matrice N x M

Record: insieme a struttura fissa di **variabili disomogenee** identificate da un nome detto **campo**

Esempio: le n-uple di una tabella di una Basi dati relazionale

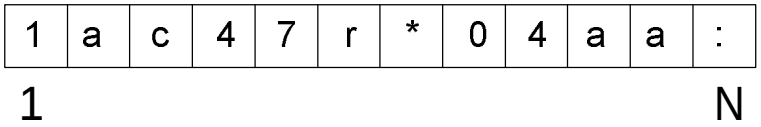
[Aldo, Rossi, 34, Roma]

Strutture dati dinamiche

- **Struttura e dimensioni variabili**
  - Lista
  - Coda (FIFO, First In First Out)
  - Pila (LIFO, Last In First Out)
  - Albero
  - Grafo
- **Composte da un elemento base con riferimenti ad altri elementi analoghi**
  - **Gli elementi vengono creati e inseriti o distrutti quando necessario**

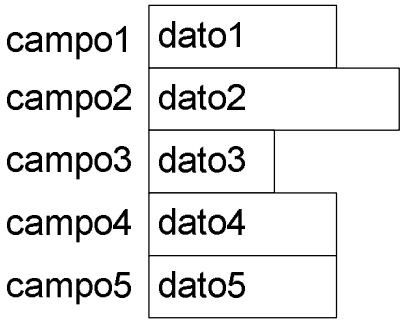
Rappresentazioni

Vettore (statico: lunghezza fissa)



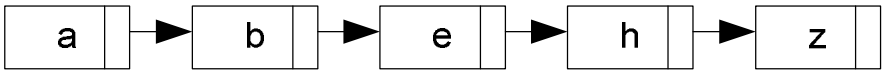
a)

Record (lung. fissa)



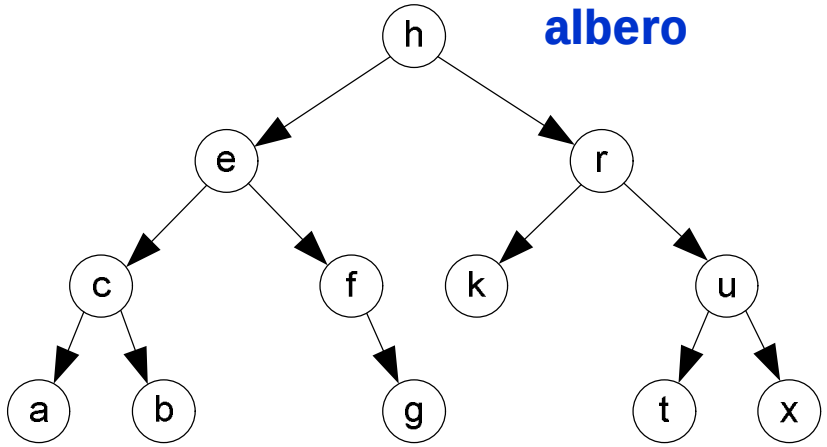
b)

Lista (dinamica: lunghezza variabile)



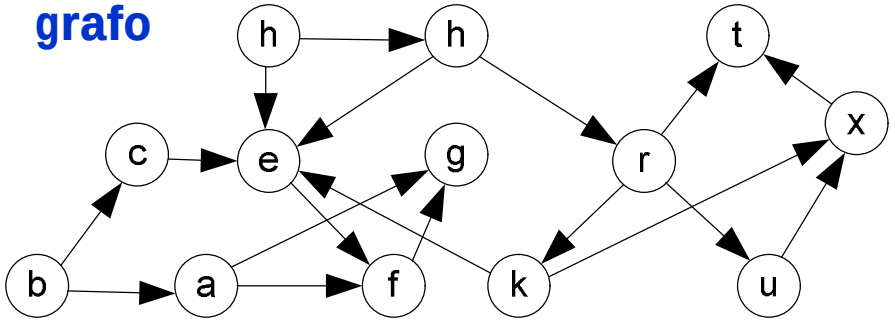
c)

albero



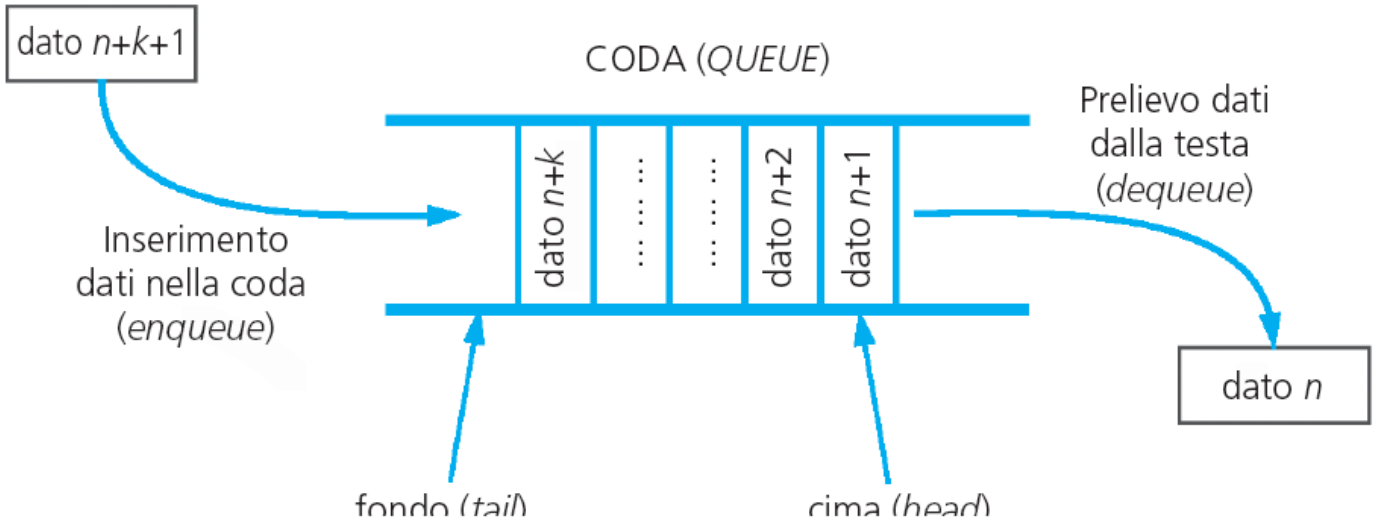
d)

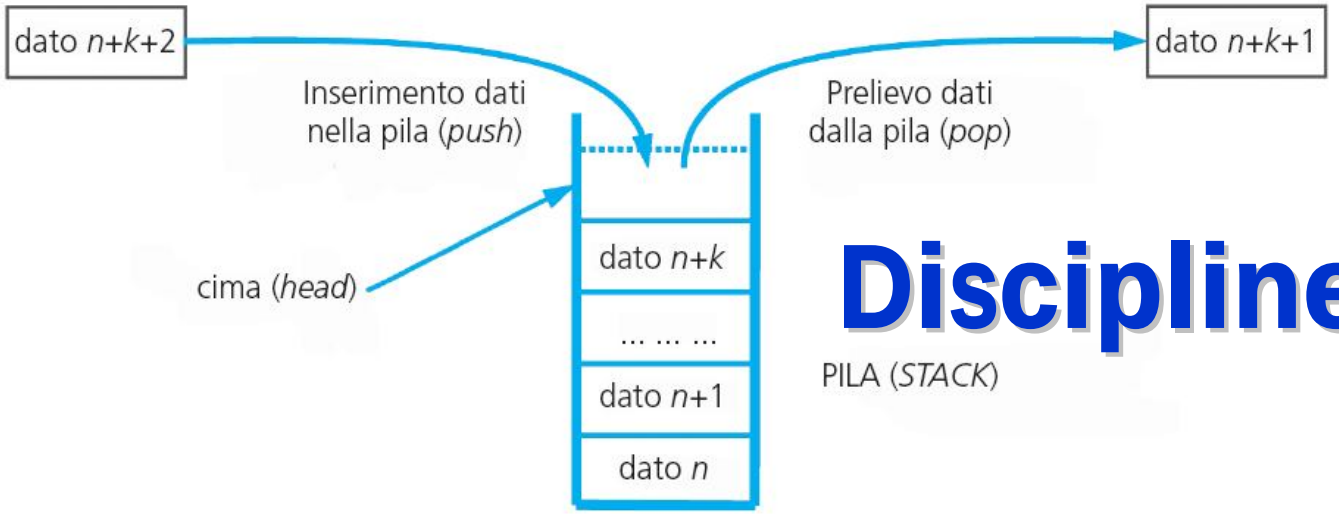
grafo



e)

Discipline FIFO





Discipline LIFO

Esempio 2bis (soluzione con vettore di lunghezza n>0):

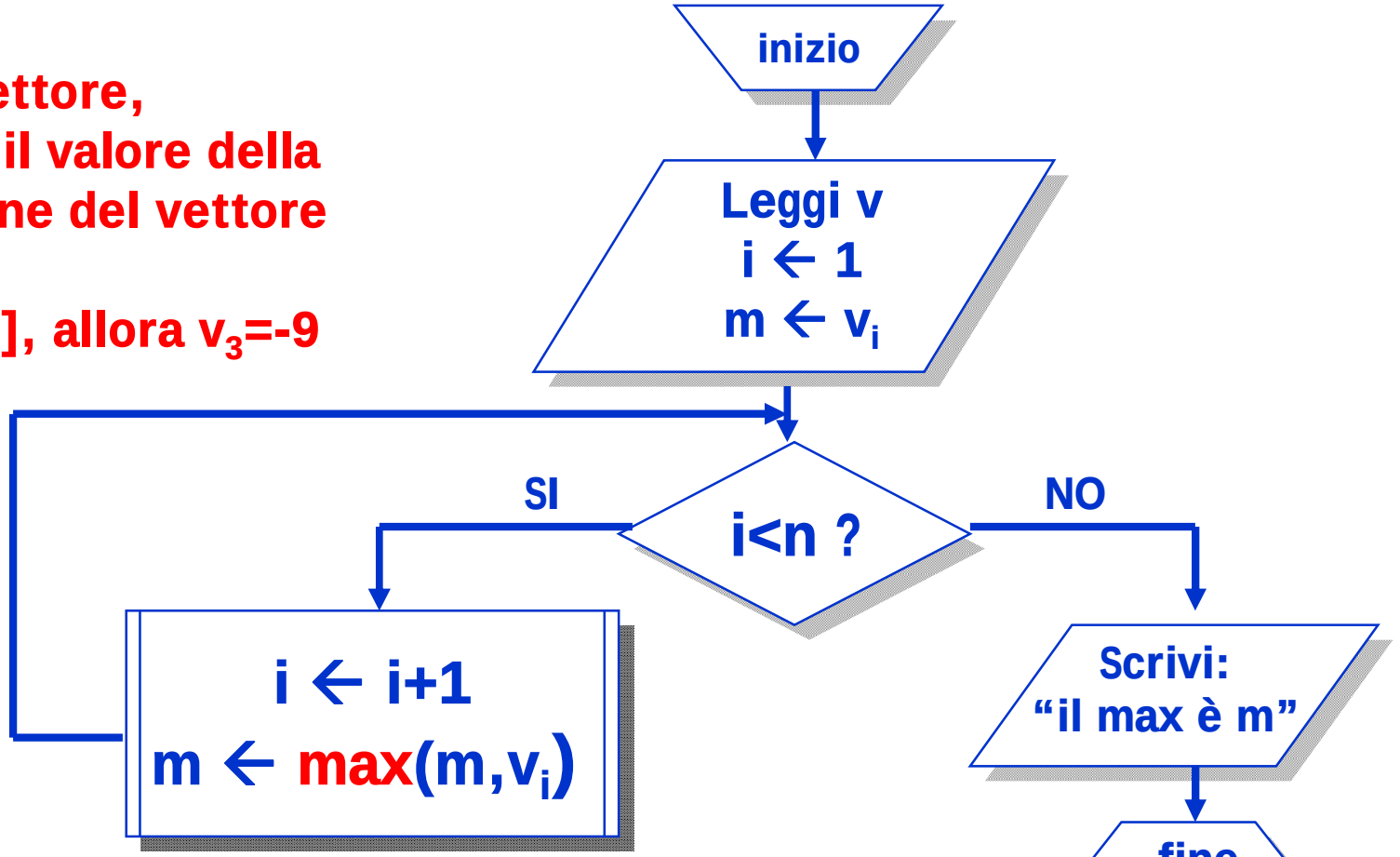
calcolare il max di un vettore di N>0 interi $v=[x_1,x_2,x_3,\ldots,x_{n-1},x_n]$

Notazione:

se v indica il vettore,
allora v_i indica il valore della
i-esima posizione del vettore

Esempio:

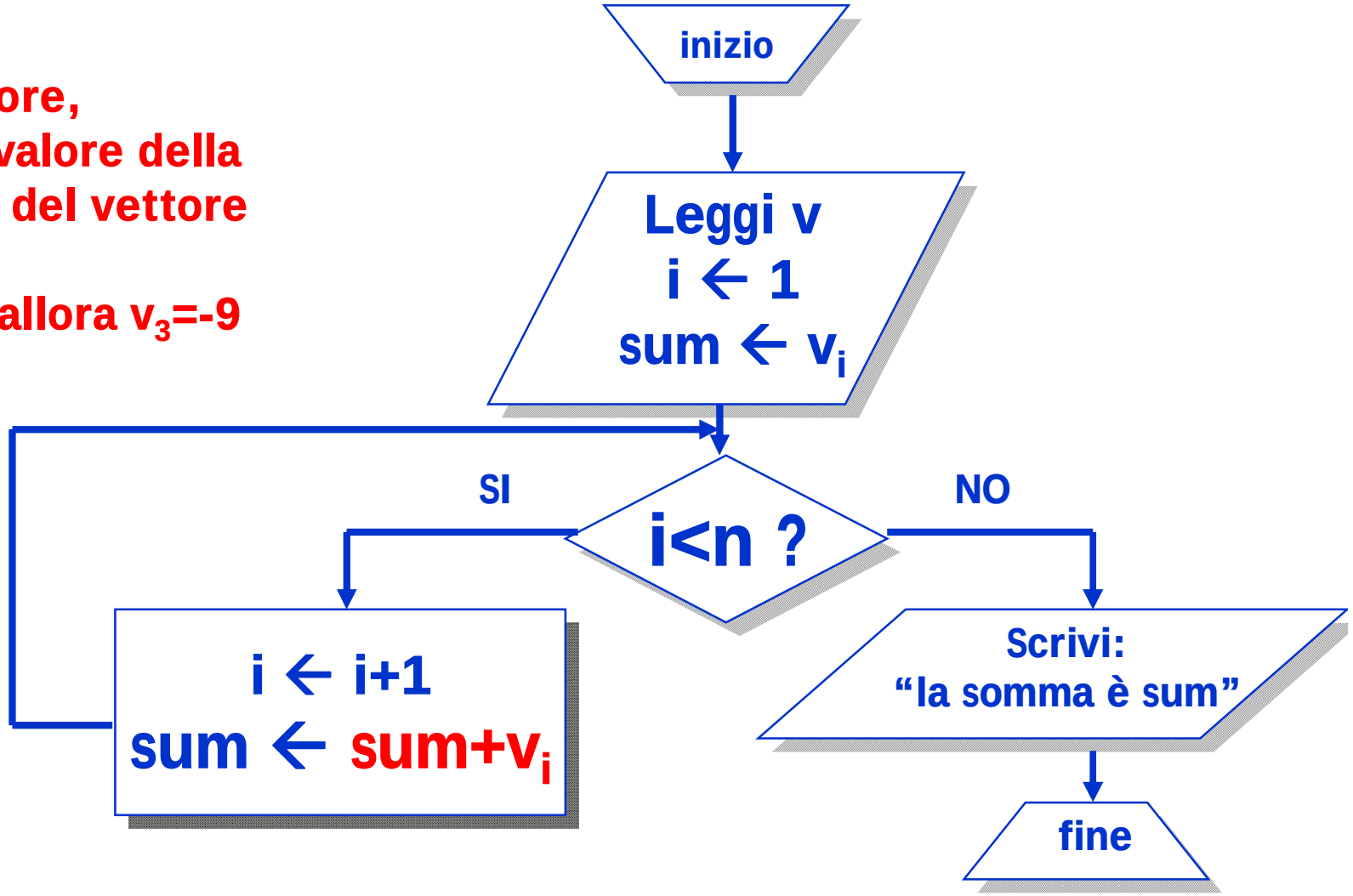
se $v=[2,-1,-9,5]$, allora $v_3=-9$



Assumiamo il sotto-programma
che calcola il max di due interi

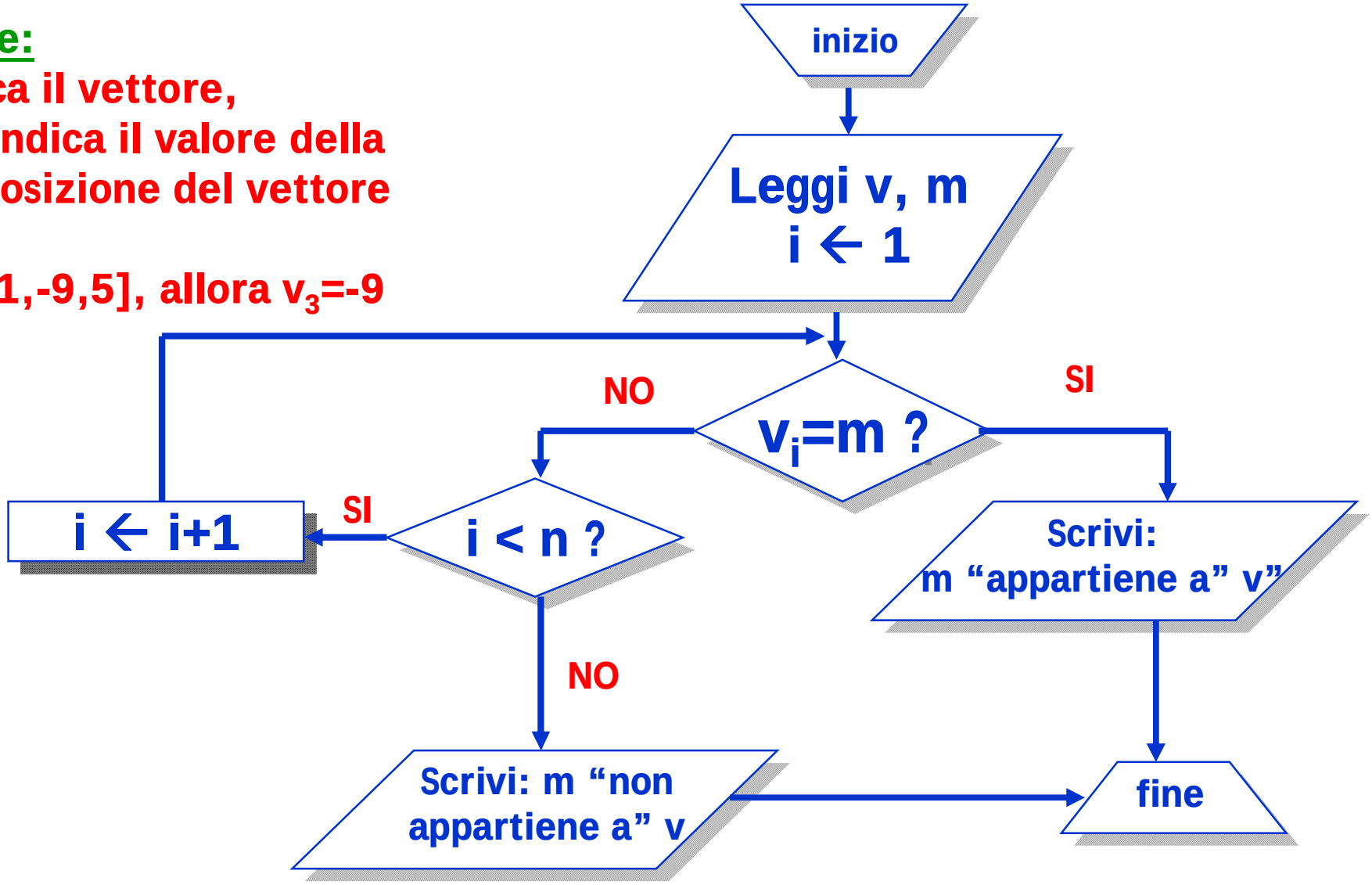
Esercizio 7: calcolare la somma degli interi contenuti in un vettore v di lunghezza n>0

Notazione:
se v indica il vettore,
allora v_i indica il valore della
i-esima posizione del vettore
Esempio:
se v=[2,-1,-9,5], allora v_3=-9



Esercizio 8: dato un vettore v di interi di lunghezza n>0 ed un intero m, dire se m appartiene a v

Notazione:
se v indica il vettore,
allora v_i indica il valore della
i-esima posizione del vettore
Esempio:
se v=[2,-1,-9,5], allora v_3=-9



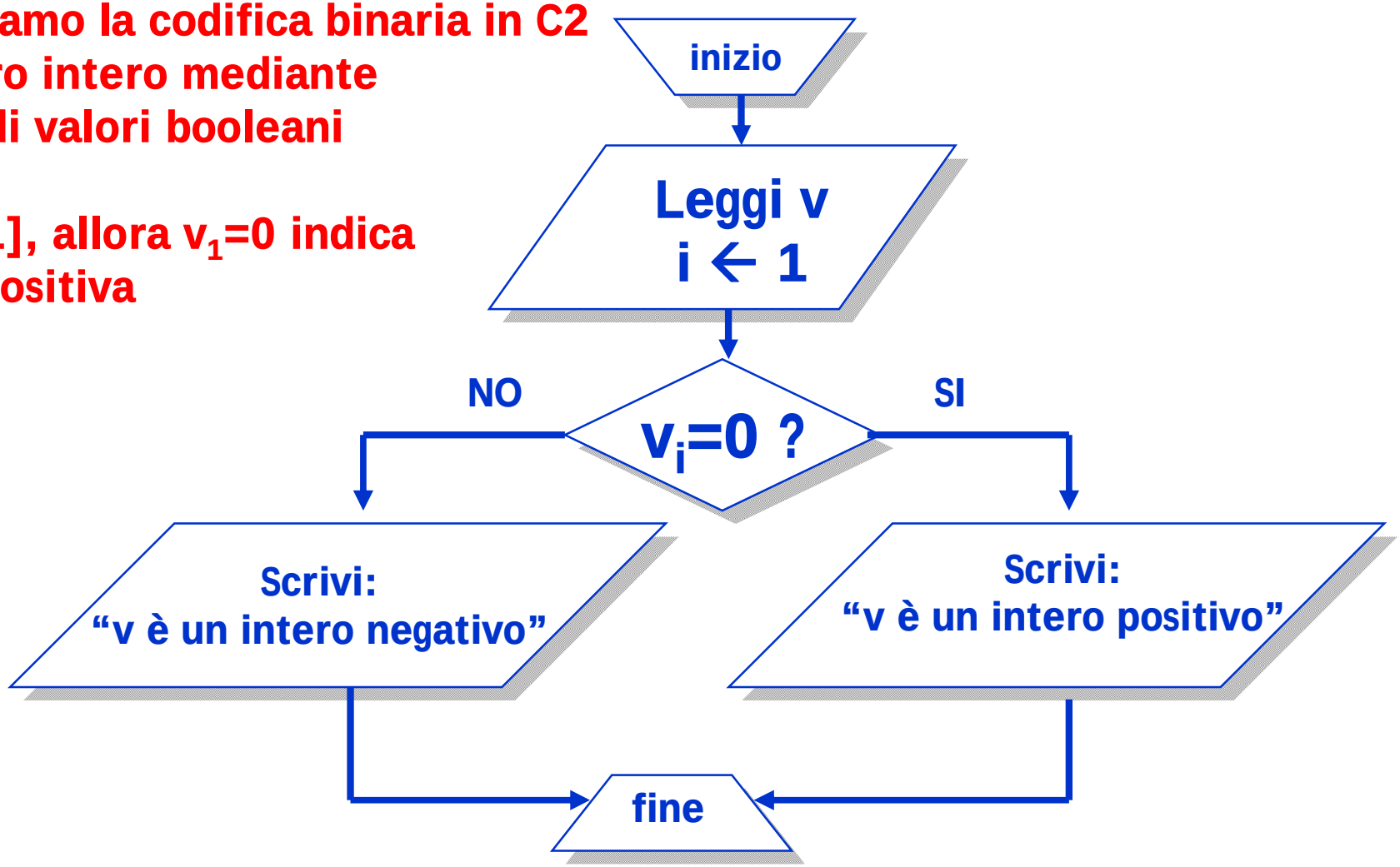
Esercizio 9: descrivere un algoritmo tale che, data la codifica binaria in C2 di un intero n, restituisca la polarità (positiva o negativa) di n

Notazione:

Rappresentiamo la codifica binaria in C2 di un numero intero mediante un vettore di valori booleani

Esempio:

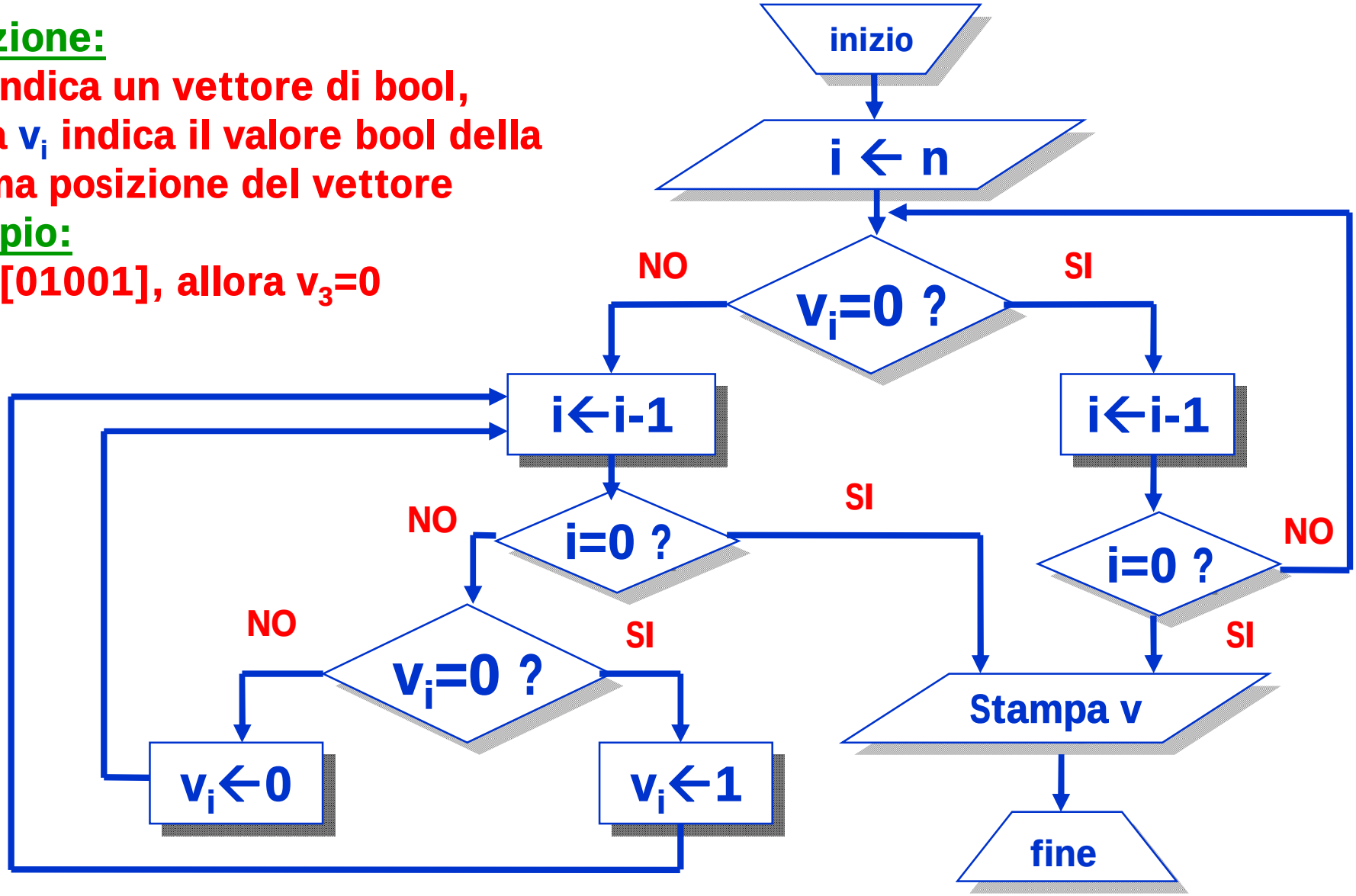
se v=[01001], allora v_1=0 indica la polarità positiva



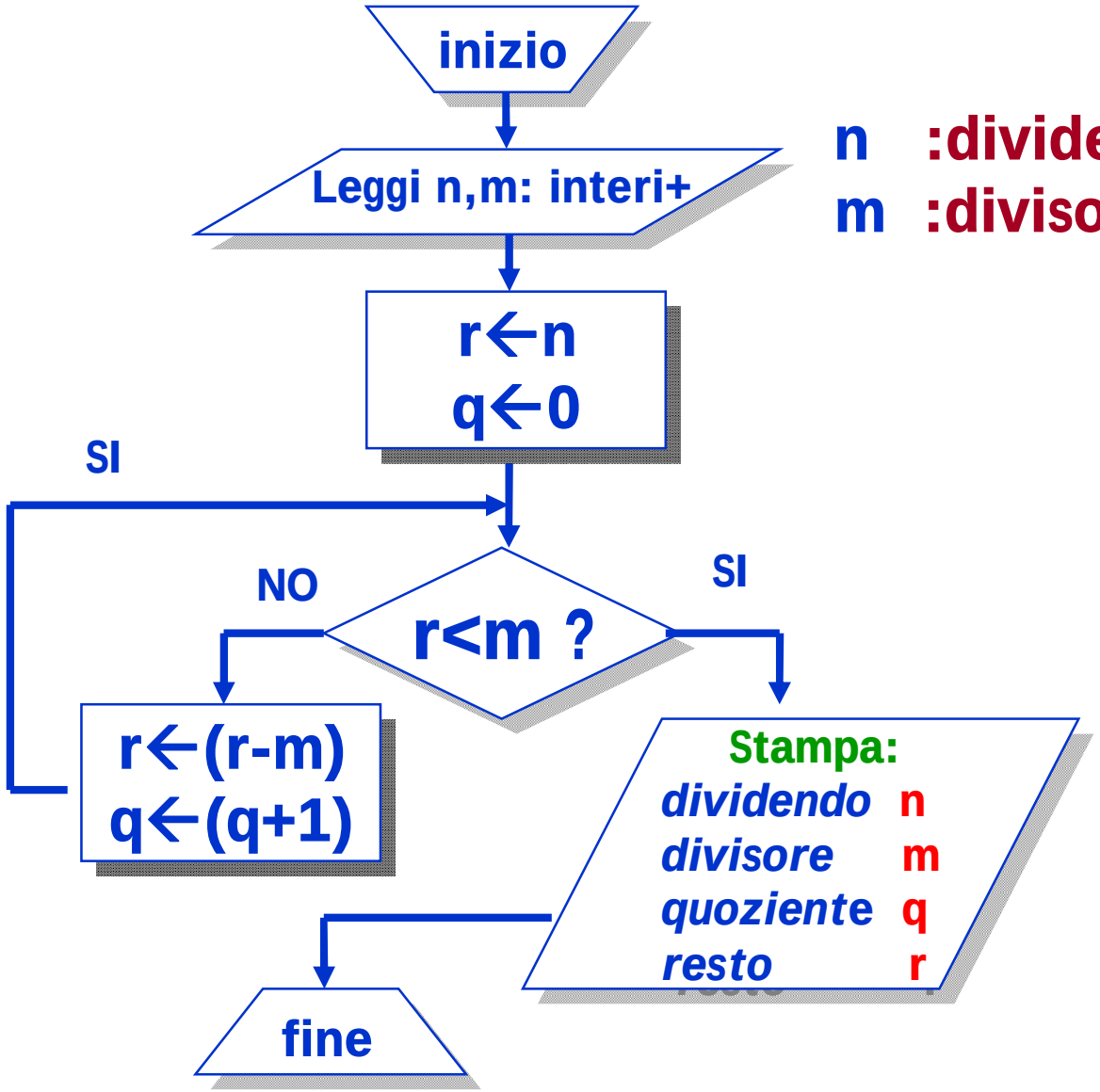
Esercizio 10: data in input la codifica in C2 di un intero N, calcolare in output l'opposto il suo opposto -N

Notazione:
se v indica un vettore di bool, allora v_i indica il valore bool della i-esima posizione del vettore

Esempio:
se v=[01001], allora v_3=0



Esercizio 11: assumendo che l'esecutore sappia eseguire solo somma e sottrazione, scrivere l'algoritmo della divisione tra interi positivi, calcolando quoziente e resto



Operazioni Logiche (Algebra Booleana)

Algebra di Boole

- **L'algebra booleana (George Boole, 1815) serve a descrivere le operazioni logiche.**
- **Componenti dell'algebra di Boole:**
  - Operatori booleani (connettivi logici)
  - Regole di trasformazione ed equivalenza tra operatori booleani
- **Gli operandi booleani hanno solo due valori:**

  Vero/Falso True/False 1/0 Sì/No …

Operatori e tavole di verità

A	not A
0	1
1	0

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A ≡ B
0	0	1
0	1	0
1	0	0
1	1	1

A	B	A nand B
0	0	1
0	1	1
1	0	1
1	1	0

A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0

Notazione

- **Esistono convenzioni diverse:**
  - **Negazione** **not A** **¬ A** **A !** **- A**
  - **Congiunzione** **A and B** **A ∧ B** **A & B** **A × B**
  - **Disgiunzione** **A or B** **A ∨ B** **A | B** **A + B**
  - **Disgiunzione esclusiva**
    A xor B **A ^ B** **A ⊕ B**
    [equivale a **(A and (not B)) or ((not A) and B)]**

  - **Implicazione** **A → B** **A ⊂ B** **A ⇒ B**
    (se … allora)
  - **Doppia implicazione** **A ↔ B** **A ≡ B** **A ⇔ B**
    (se e solo se)

Tabella della verità

a	b	c	a × b	-a × c	(a × b) + (-a × c)
F	F	F	F	F	F
F	F	T	F	T	T
F	T	F	F	F	F
F	T	T	F	T	T
T	F	F	F	F	F
T	F	T	F	F	F
T	T	F	T	F	T
T	T	T	T	F	T

Tabella della verità

a	b	c	a + c	a + (–b)	(a + c) × (a + (–b))
F	F	F	F	T	F
F	F	T	T	T	T
F	T	F	F	F	F
F	T	T	T	F	F
T	F	F	T	T	T
T	F	T	T	T	T
T	T	F	T	T	T
T	T	T	T	T	T

Espressioni booleane

- **Equivalenza**

  Due espressioni booleane sono **equivalenti** se hanno la medesima tavola di verità

  Esempio: (not A or B) = (A --> B)

- **Tautologia**

  Un'espressione booleana è una **tautologia** se è sempre vera

  Esempio: **A or (not A)**

- **Contraddizione**

  Un'espressione booleana è una **contraddizione** se è sempre falsa

  Esempio: **A and (not A)**

Proprietà degli operatori booleani

Proprietà	AND (×)	OR (+)
Identità	$A \times 1 = A$	$A + 0 = A$
Elemento nullo	$A \times 0 = 0$	$A + 1 = 1$
Idempotenza	$A \times A = A$	$A + A = A$
Inverso	$A \times (-A) = 0$	$A + (-A) = 1$
Commutativa	$A \times B = B \times A$	$A + B = B + A$
Associativa	$A \times (B \times C) = (A \times B) \times C$	$A + (B + C) = (A + B) + C$
Distributiva	$A \times (B + C) = (A \times B) + (A \times C)$	$A + (B \times C) = (A + B) \times (A + C)$
Assorbimento	$A \times (A + B) = A$	$A + (A \times B) = A$
De Morgan	$-(A \times B) = (-A) + (-B)$	$-(A + B) = (-A) \times (-B)$

Dalla Tavola di Verità alla Proposizione Logica

- **Problema:** conosciamo la tabella, ma non sappiamo qual è l'espressione logica corrispondente
- **Soluzione:** ci basta trovare **una** delle espressioni equivalenti che hanno questa tavola di verità

a	b	c	Espressione?
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	T
T	T	F	T
T	T	T	T

Dalla tabella all'espressione

1. **identificazione di tutte le righe che hanno valore T;**
2. **per ogni riga identificata si costruisce una sottoespressione prodotto (AND) di tutte le lettere che sono prese nella loro forma naturale o negata seguendo i seguenti principi:**
   - le lettere che nella riga in esame hanno valore T sono prese nella forma naturale;
   - le lettere che nella riga in esame hanno valore F sono prese nella forma negata (duale);
3. **le sottoespressioni prodotto così ottenute vengono sommate (OR) tra loro per realizzare l'espressione desiderata.**

Dalla tabella all'espressione (1)

identificazione di tutte le righe che hanno valore T;

	a	b	c	espressione	
riga 0	F	F	F	F	
riga 1	F	F	T	F	
riga 2	F	T	F	F	
riga 3	**F**	**T**	**T**	**T**	⇜
riga 4	T	F	F	F	
riga 5	**T**	**F**	**T**	**T**	⇜
riga 6	**T**	**T**	**F**	**T**	⇜
riga 7	**T**	**T**	**T**	**T**	⇜

Dalla tabella all'espressione (2)

per ogni riga identificata si costruisce una sottoespressione prodotto (and) di tutte le lettere prese seguendo i seguenti principi:

1) le lettere che in una riga hanno valore T sono prese nella forma naturale;

2) le lettere che in una riga nella riga hanno valore F sono prese nella forma complementata (duale);

	a	b	c	espressione		
riga 0	F	F	F	F		
riga 1	F	F	T	F		
riga 2	F	T	F	F		
riga 3	**F**	**T**	**T**	**T**	⇜	**(–a) × b × c**
riga 4	T	F	F	F		
riga 5	**T**	**F**	**T**	**T**	⇜	**a × (–b) × c**
riga 6	**T**	**T**	**F**	**T**	⇜	**a × b × (–c)**
riga 7	**T**	**T**	**T**	**T**	⇜	**a × b × c**

Dalla tabella all'espressione (3)

le sottoespressioni prodotto così ottenute vengono sommate (or) tra loro per realizzare l'espressione desiderata.

	a	b	c	espressione	
riga 0	F	F	F	F	
riga 1	F	F	T	F	
riga 2	F	T	F	F	
riga 3	**F**	**T**	**T**	**T**	$m_1 = (-a) \times b \times c$
riga 4	T	F	F	F	
riga 5	**T**	**F**	**T**	**T**	$m_2 = a \times (-b) \times c$
riga 6	**T**	**T**	**F**	**T**	$m_3 = a \times b \times (-c)$
riga 7	**T**	**T**	**T**	**T**	$m_4 = a \times b \times c$

$$expr = m_1 + m_2 + m_3 + m_4$$

$$expr = ((-a) \times b \times c) + (a \times (-b) \times c) + (a \times b \times (-c)) + (a \times b \times c)$$

Verifica

a	b	c	m_1 = (-a) × b × c	m_2 = a × (-b) × c	m_3 = a × b × (-c)	m_4 = a × b × c	$m_1 + m_2 + m_3 + m_4$
F	F	F	F	F	F	F	F
F	F	T	F	F	F	F	F
F	T	F	F	F	F	F	F
F	T	T	T	F	F	F	T
T	F	F	F	F	F	F	F
T	F	T	F	T	F	F	T
T	T	F	F	F	T	F	T
T	T	T	F	F	F	T	T

Un’altra espressione equivalente

a	b	c	a × b	a × c	b × c	(a × b) + (a × c) + (b × c)
F	F	F	F	F	F	F
F	F	T	F	F	F	F
F	T	F	F	F	F	F
F	T	T	F	F	T	T
T	F	F	F	F	F	F
T	F	T	F	T	F	T
T	T	F	T	F	F	T
T	T	T	T	T	T	T

L’esecutore
(Cenni sulla computabilità)

Il problema dell'esecutore

- **Due domande fondamentali:**
  - È sempre possibile trovare una soluzione algoritmica ad un problema?
  - Esiste un esecutore automatico in grado di eseguire un algoritmo e se sì come è fatto?

- **A queste domande risponde la**

teoria della computabilità

La computabilità

- **La teoria della computabilità**
  - E' parte fondamentale dell'*informatica teorica*
  - Definisce quali caratteristiche un problema deve avere per ammettere una soluzione computabile
- **Necessario definire un criterio**
- **Computabilità secondo Turing**
  - Un problema è *computabile secondo Turing* se esiste una *Macchina di Turing* che lo risolve
- **Risultati analoghi forniti anche da Church**

Tesi di Church-Turing

- **Risultato fondamentale:**
  - ***Un problema è computabile se è computabile secondo Turing***
  - **Tesi *non dimostrata***, ma dedotta dalla sostanziale equivalenza delle varie definizioni proposte per la computabilità e *mai contraddetta finora*
- **Conseguenze:**
  - Tutti gli esecutori sono equivalenti alla Macchina di Turing
  - Gli esecutori differiscono tra loro solo nella velocità di risoluzione dei problemi, non nella capacità di risolverli

La Macchina di Turing

- **Primo modello di esecutore automatico**
  - E' un modello teorico (ovvero non realizzabile praticamente) che risolve automaticamente un determinato problema
  - Composta da un nastro infinito ed una unità di controllo con stato che può scrivere, leggere e cancellare simboli sul nastro
- **Idea fondamentale: l'"impiegato diligente"**
  - La macchina opera eseguendo istruzioni del tipo

    "se è vero A allora esegui B"
  - Set di istruzioni minimo (una!) e completo (tesi)

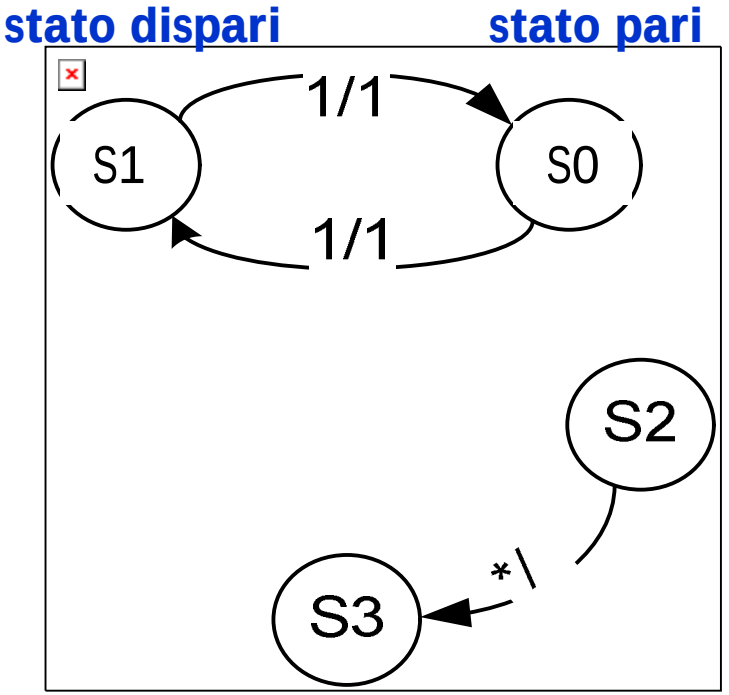
Rappresentazione

MdT che determina la parità di un intero

In uscita aggiungi un 1 al nastro se l'intero è pari

Soluzione 1 (notazione unaria): un simbolo (1), * nastro vuoto

Stato	Letto	Vai a	Scrivi	Sposta
S0 (inizio)	1	S1	1	Sinistra
S0	*	S2	*	Sinistra
S1	1	S0	1	Sinistra
S1	*	S3	*	Sinistra
S2	*	S3	1	Sinistra
S3 (fine)	-	-	-	-



Rappresentazione

MdT che determina la parità di un intero

In uscita aggiungi un 1 al nastro se l'intero è pari

Soluzione 2 (notazione binaria): due simboli (0,1), * nastro vuoto

Stato	Letto	Vai a	Scrivi	Sposta
S0 (inizio)	1	S0	1	Sinistra
S0	0	S1	0	Sinistra
S0	*	S3	*	Sinistra
S1	0	S1	0	Sinistra
S1	1	S0	1	Sinistra
S1	*	S2	*	Sinistra
S2	*	S3	1	Sinistra
S3 (fine)	-	-	-	-

stato dispari **stato pari**

La MdT universale

- **Ulteriore risultato fondamentale**
  - E' possibile definire una MdT detta "**MdT universale**" che è in grado di:
    - leggere dal nastro il comportamento di un'altra MdT
    - leggere i dati su cui detta MdT opera
    - eseguire tale comportamento sui dati letti
  - Ciò dimostra che è possibile definire *macchine programmabili*
    - Istruzioni e dati *sulla stessa memoria*

Limiti

- **Troppo elementare**
  - La disponibilità di una sola istruzione rende complesso realizzare anche algoritmi semplici
  - La soluzione dipende molto da una buona codifica e quindi dalla scelta dell'alfabeto di simboli del nastro
- **Non realizzabile praticamente**
  - Nastro infinito (si potrebbe rilassare l'ipotesi)
  - Lentezza (problema di carattere pratico)
  - Macchinosità della programmazione

Modello di Von Neumann

- **Più pragmatico**
  - Equivalente alla Macchina di Turing
  - Architettura orientata alla realizzazione pratica
  - Alla base dei calcolatori odierni