Laboratorio di Informatica

Esercitazione su algoritmi e diagrammi di flusso

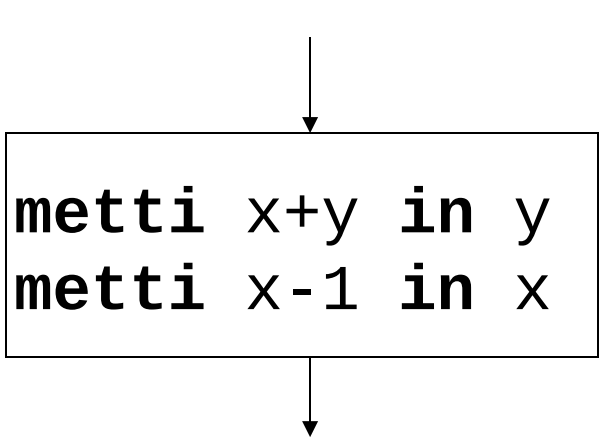
Algoritmi, programmi e dati

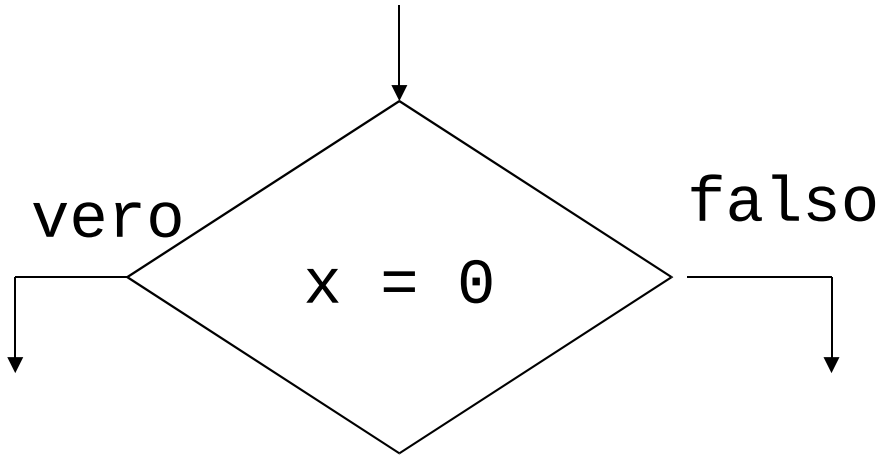
Algoritmo = insieme di istruzioni che indicano come svolgere operazioni complesse su dei dati attraverso successioni di operazioni elementari

Programma = algoritmo in un linguaggio "comprensibile" dal computer.

Dato = informazione da elaborare **rappresentata** in un formato che consenta al programma di operare su di essa

Diagrammi di flusso

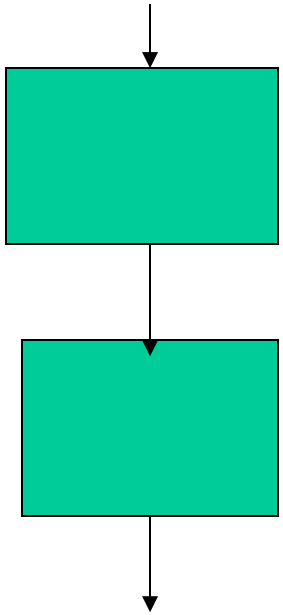




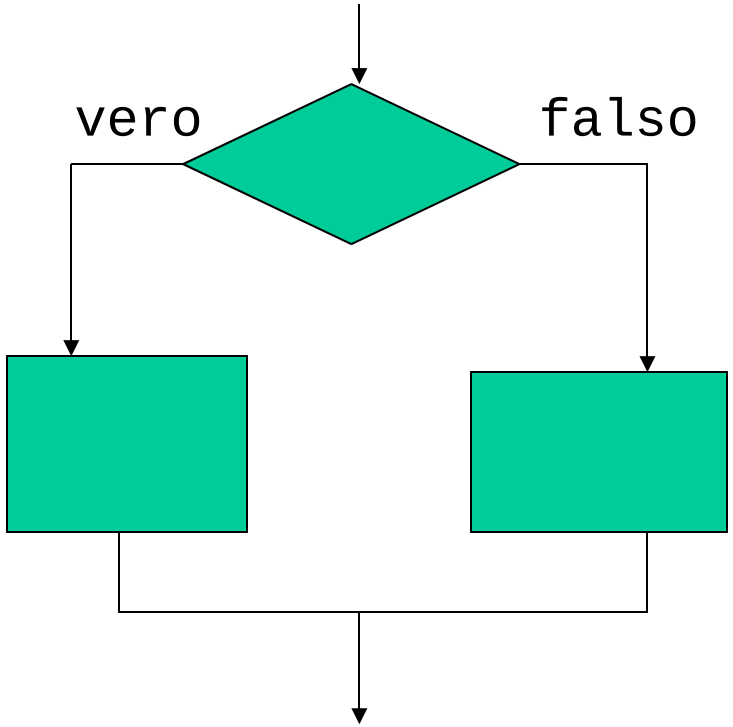
Blocchi di elaborazione
contengono sequenze di azioni

Blocchi decisionali
contengono una condizione booleana; se vera, si segue la freccia vero, se falsa si segue la freccia falso.

Strutture di controllo principali

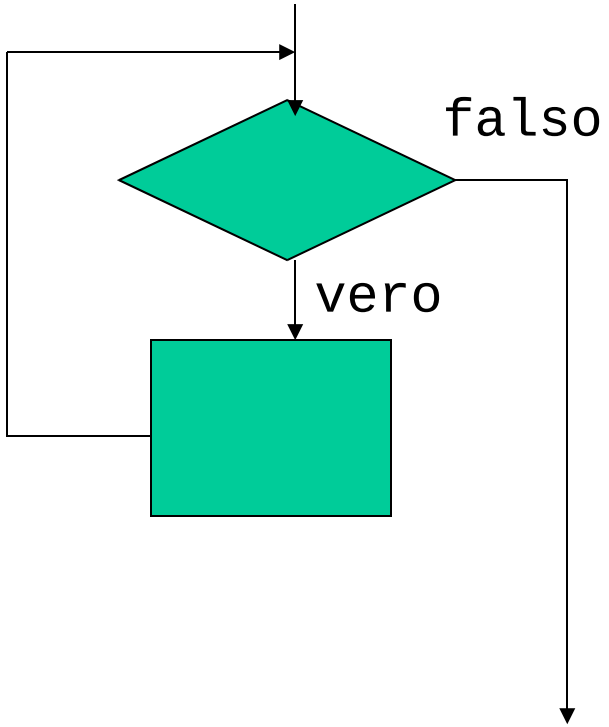


sequenza



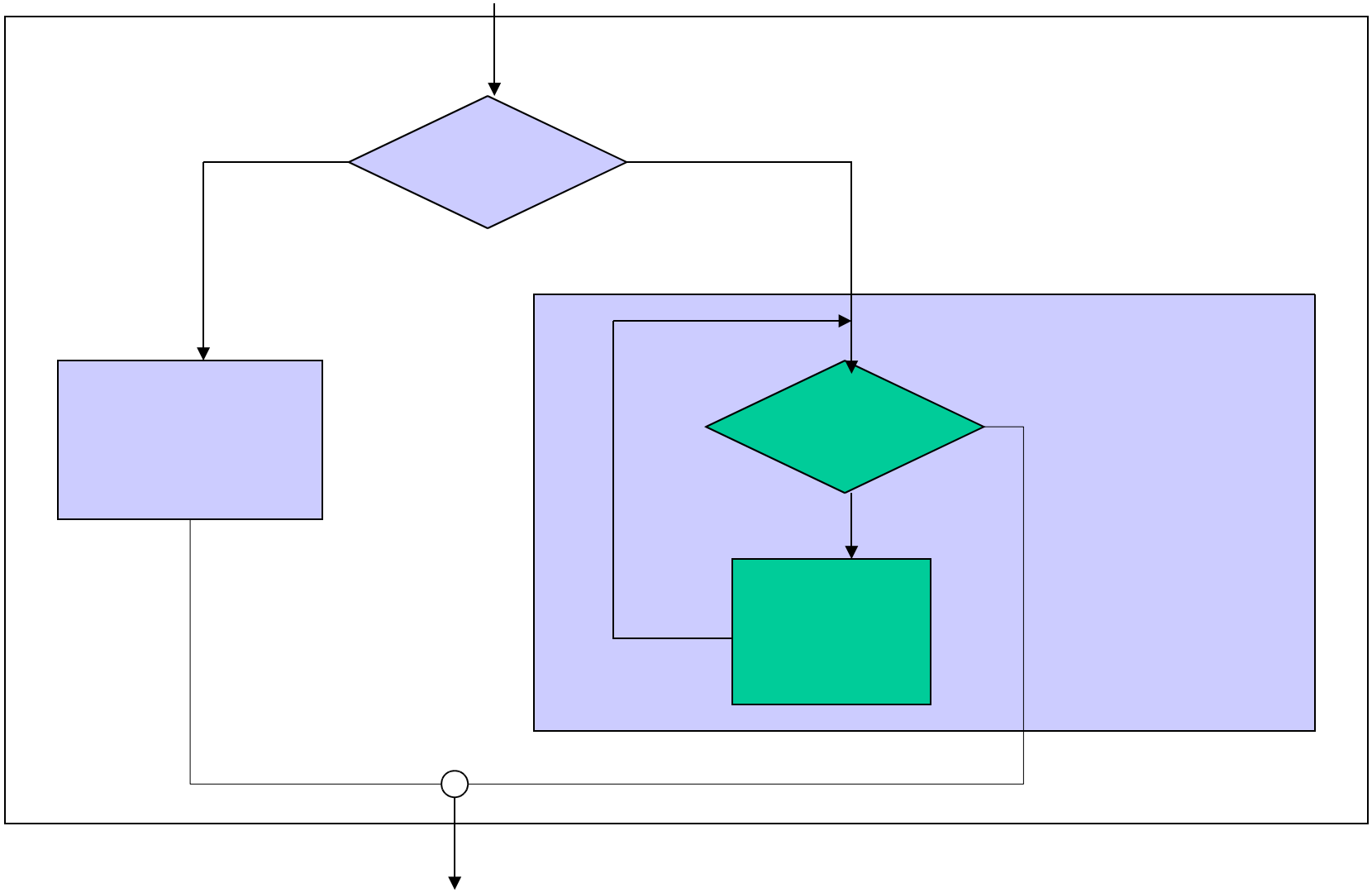
selezione

Strutture di controllo principali



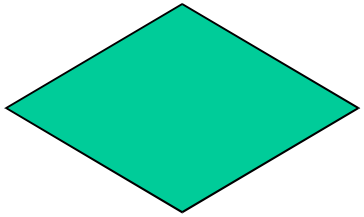
iterazione

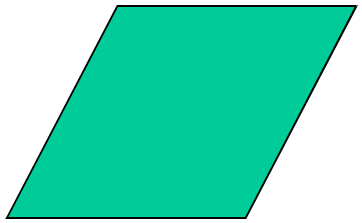
Esempio di decomposizione modulare



Simboli principali

- inizio o fine
- elaborazione
- decisione
- Connessione
- Operazione input/output





Processo di sviluppo di un programma

Il processo di sviluppo di un programma prevede le seguenti fasi:

- ***analisi*** del problema e ***specifica funzionale***
  - specificazione dei dati in ingresso e in uscita
- definizione dell'***algoritmo*** risolutivo
  - *identificazione* e *formalizzazione* di una soluzione, cioè dei *contenitori di dati* necessari e delle relative *operazioni*
- descrizione con un ***diagramma di flusso***
  - o con altro formalismo preciso e non ambiguo della successione di operazioni da eseguire.
- traduzione del diagramma di flusso in un ***programma***
  - in un linguaggio di programmazione "ad alto livello";
- ***compilazione***
  - traduzione in linguaggio macchina;
- ***verifica***
  - esecuzione del programma.

Esercizi di sviluppo di algoritmi orientati alla programmazione

- Partiamo dall'analisi del problema
- Scriviamo la specifica funzionale
- Introduciamo i contenitori di dati necessari e le relative operazioni elementari
- Scriviamo l'algoritmo che opera su tali dati con un diagramma di flusso

Esempio 1

A) Problema e analisi del problema

- L’analisi del problema è il primo passo; deve fornire
  - un ***nome*** e una ***breve descrizione*** di cosa si vuol fare;
  - un elenco di ***requisiti***: richieste a cui deve soddisfare il programma

Esempio di analisi del problema

Problema telefonata

- ***Descrizione***:
  - vogliamo chiamare un abbonato con il telefono.
- ***Requisiti, in cui prevediamo i diversi casi***:
  - la telefonata viene eseguita con successo
    - messaggio “telefonata riuscita”
  - la telefonata non può essere portata a termine
    - messaggio “telefonata non riuscita”

B) Specifica funzionale

Una specifica funzionale indica

- quali sono i ***dati iniziali***, cioè quelli da elaborare
  - detti anche ***ingressi*** all'algoritmo
- che ***risultato*** si vuole, *in funzione degli ingressi*
  - detto anche ***uscita*** dell'algoritmo

Esempio: specifica funzionale

- **TELEFONATA**: specifica funzionale
- ***Argomenti*** o ***ingressi***:
  - **N** : numero da comporre
- ***Risultati o uscite:***
  - messaggio “**telefonata riuscita**”
  - messaggio “**telefonata non riuscita**”

C) I contenitori di dati

- Un algoritmo ha bisogno di tener traccia di ingressi e risultati:
  - sia il risultato finale
  - sia eventuali risultati intermedi
- Allo scopo, usa dei ***contenitori di dati***

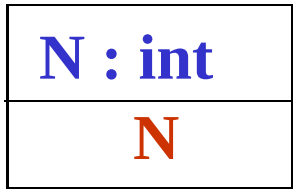
I contenitori di dati

- I contenitori di dati utilizzati per i risultati intermedi dipendono dall'algoritmo
  - quindi, a meno di casi assai elementari, è necessario avere già un'idea dell'algoritmo per determinarli
  - difficilmente sono TUTTI prevedibili sin dall'inizio; man mano che l'algoritmo prende forma, si possono aggiungere al volo nuovi contenitori

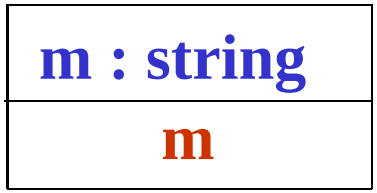
I contenitori utilizzati da TELEFONATA

- Di quali contenitori abbiamo bisogno per TELEFONATA?
  - Sicuramente di quello per contenere i dati di ingresso ed il risultato
    - ***1 contenitore per N*** (ingresso)
  - Eventuali contenitori per i risultati intermedi
  - Contenitore per il risultato finale

- Relativamente agli ingressi, abbiamo il contenitore:



- Relativamente all'uscita, abbiamo il contenitore:



In blu indichiamo il nome del **contenitore**, in rosso il numero **contenuto** in esso

Il tipo int corrisponde ai numeri interi; le operazioni che assumiamo disponibili per esso sono le solite: somma, prodotto, ….

D) L’algoritmo: primo passo

- Descrivere brevemente l’idea dell’algoritmo
  - cioè i passi da eseguire per giungere alla soluzione usando i contenitori di dati e le operazioni disponibili su di essi in base al tipo di dati, a grandi linee
- Può darsi che una prima idea sia già stata raggiunta per trovare i contenitori dati più appropriati
  - in tal caso si procede ad un eventuale raffinamento dell’idea

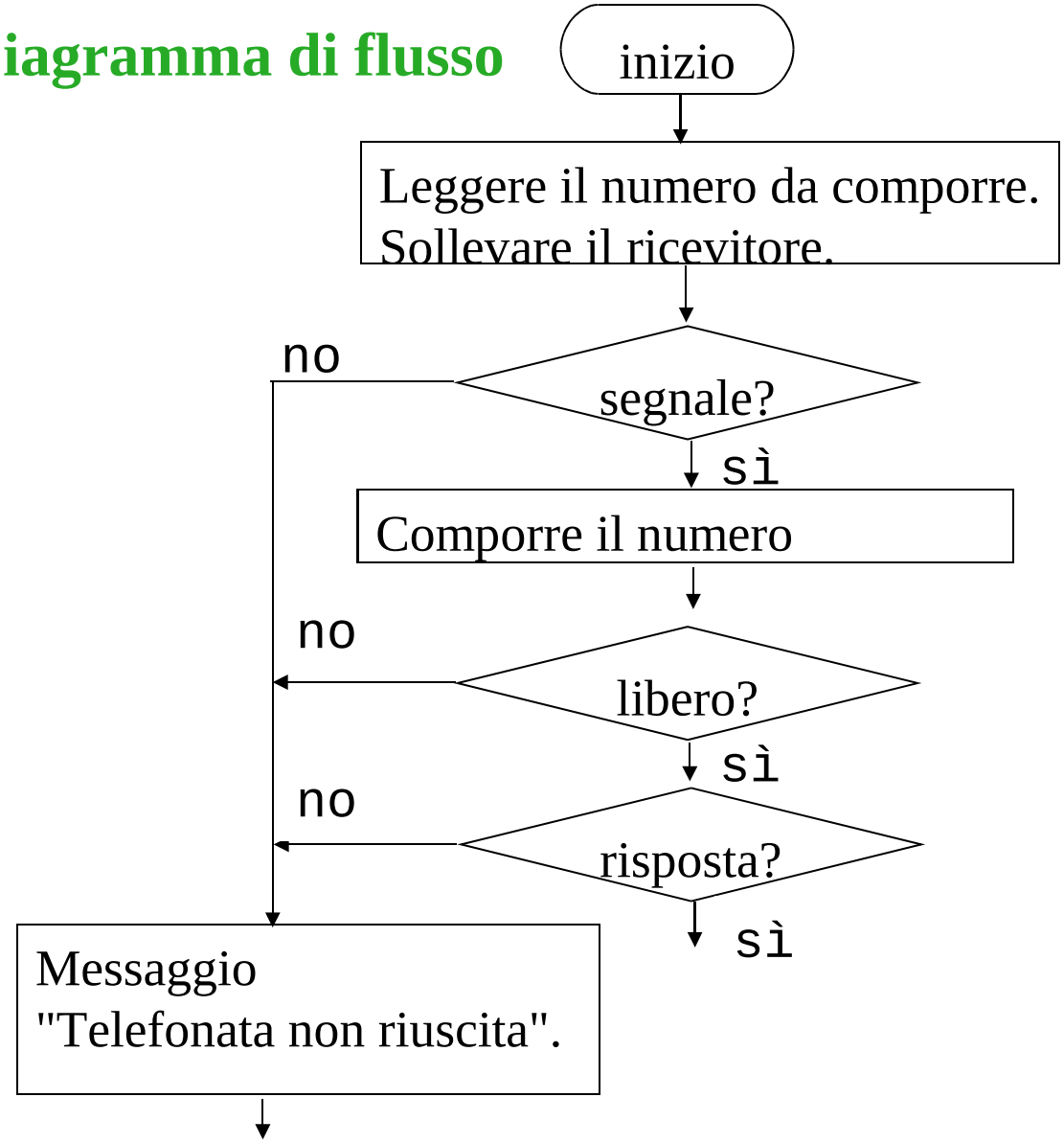
L’idea dell’algoritmo di soluzione di TELEFONATA

- Sollevo il ricevitore
- analizzo i vari casi di segnale:
  - assente
  - presente
- se posso, procedo componendo il numero
- di nuovo analizzo i vari casi:
- e caso per caso costruisco il messaggio da inviare in uscita.

L’idea dell’algoritmo di soluzione di TELEFONATA

- **Più in dettaglio**:
  - il telefono non dà segnale quando solleviamo la cornetta, non possiamo procedere e mettiamo giù
  - altrimenti possiamo comporre il numero
  - se il segnale è occupato o assente, non possiamo procedere e mettiamo giù
  - altrimenti, aspettiamo la risposta
  - se non arriva entro 10 squilli, mettiamo giù
  - altrimenti parliamo e poi mettiamo giù
- Passiamo al diagramma di flusso:

Diagramma di flusso

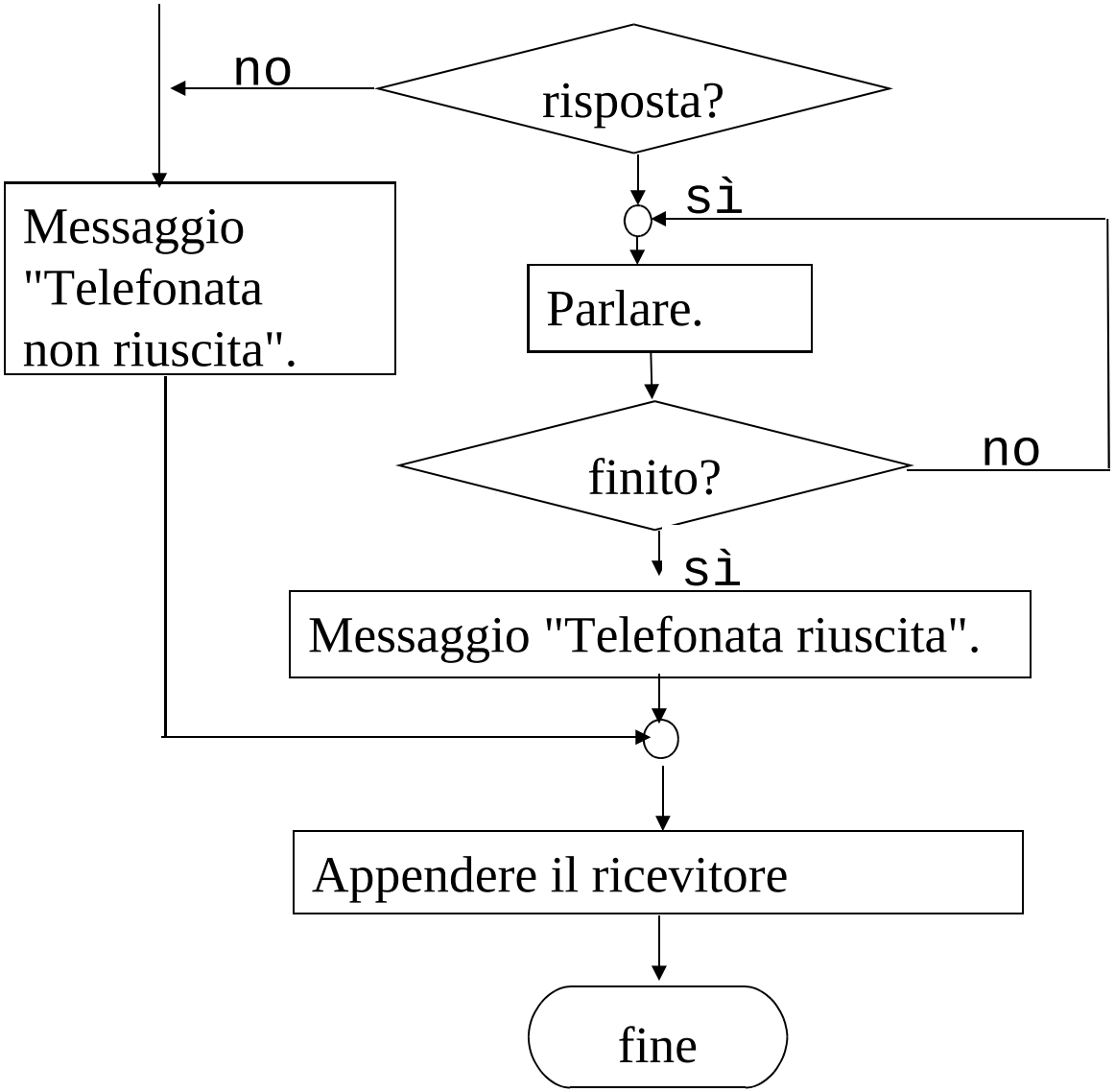


Contenitori di dati

N
N

messaggio
m

Diagramma di flusso

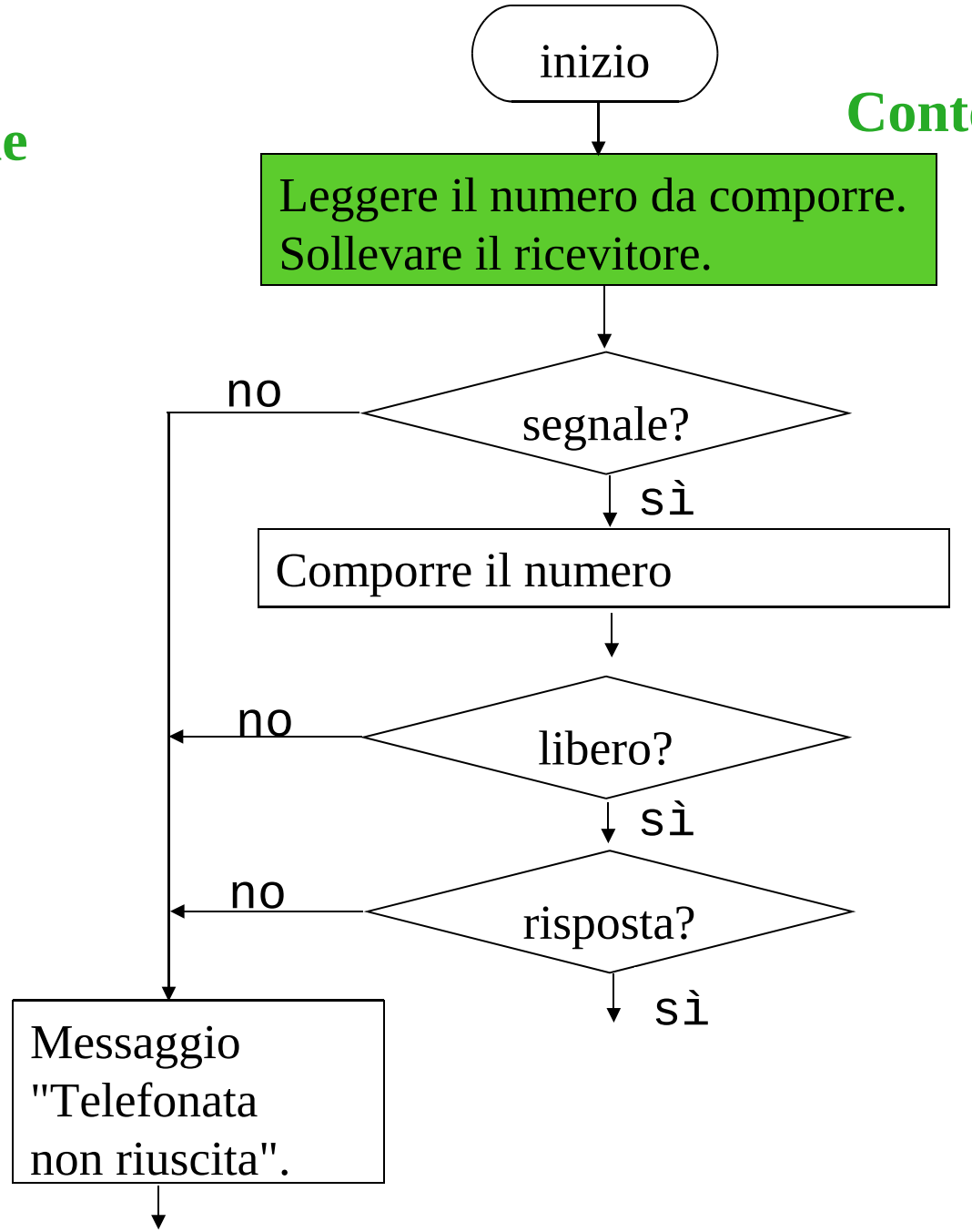


Contenitori di dati

N
N

messaggio
m

Esecuzione

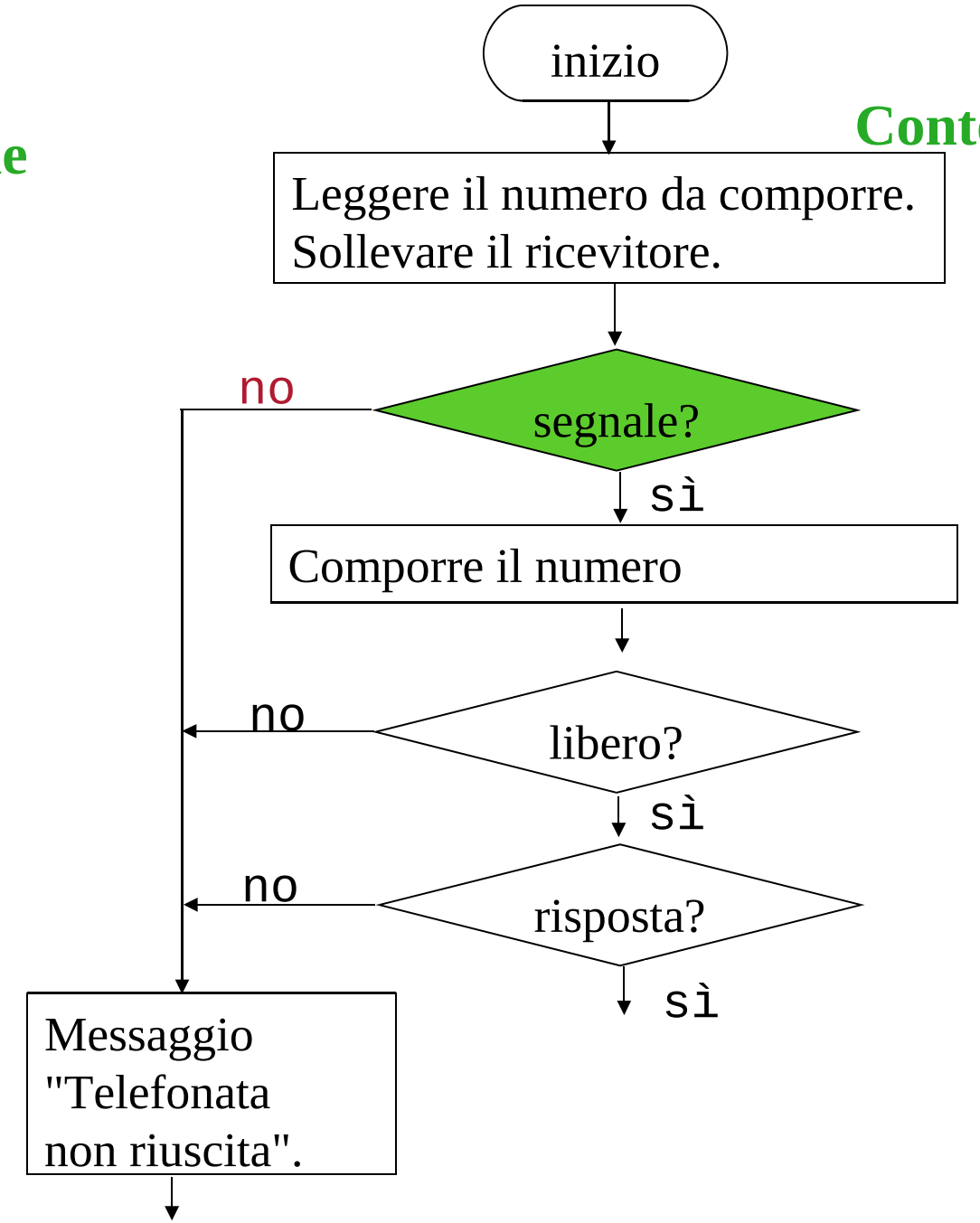


Contenitori di dati

N
025031

messaggio

Esecuzione

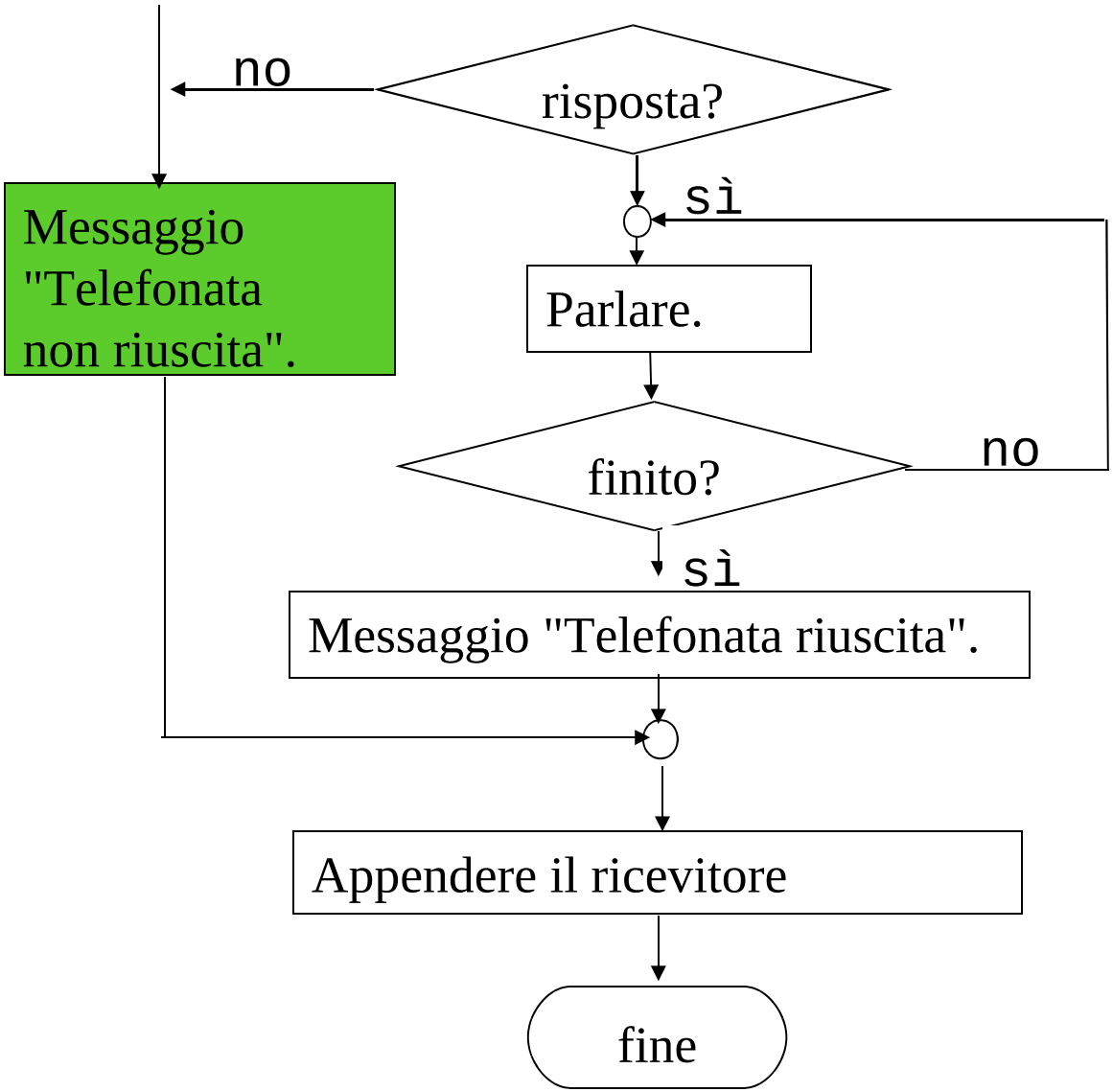


Contenitori di dati

N
025031

messaggio
m

Esecuzione

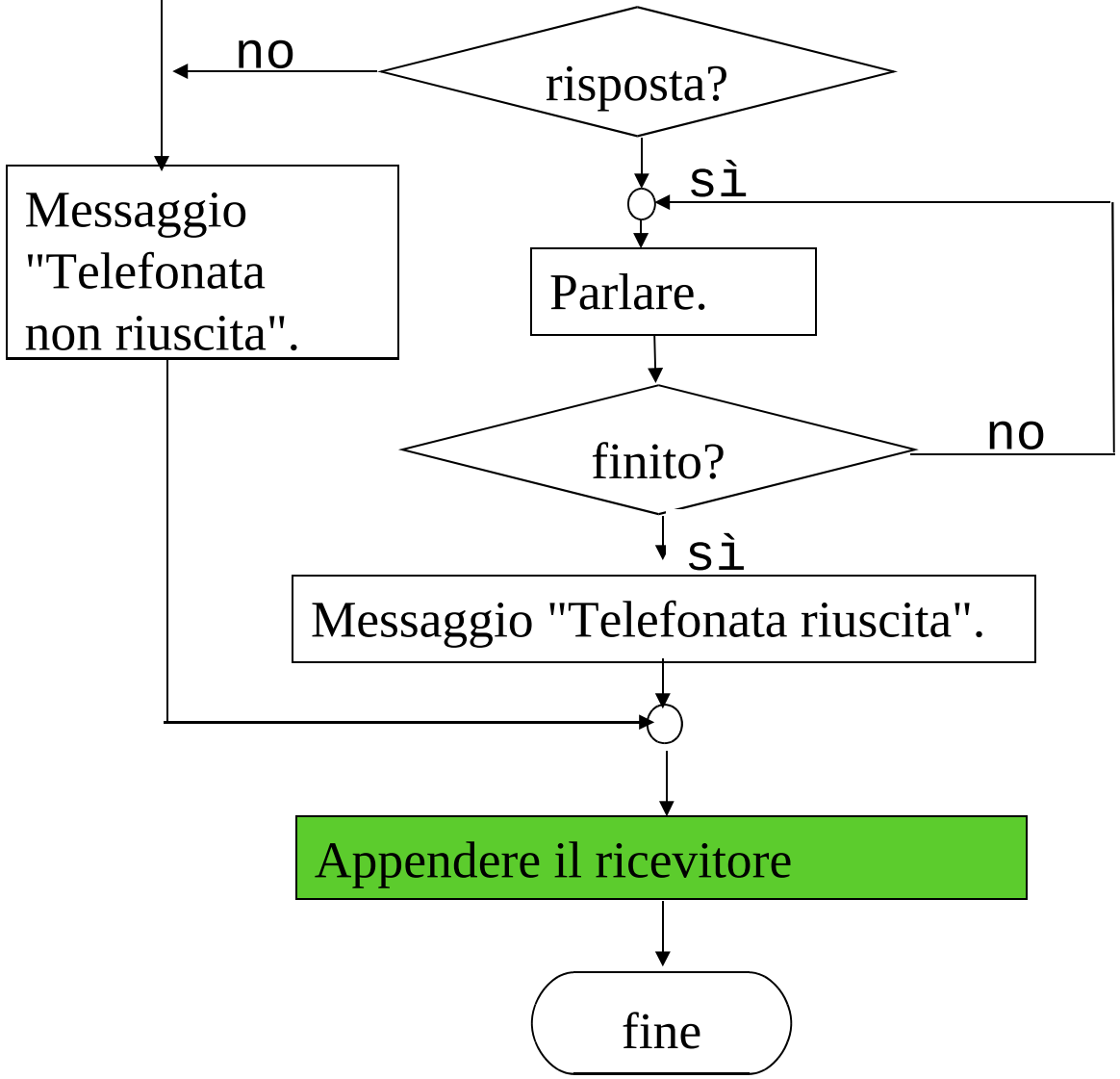


Contenitori di dati

N
025031

messaggio
T. non r.

Esecuzione

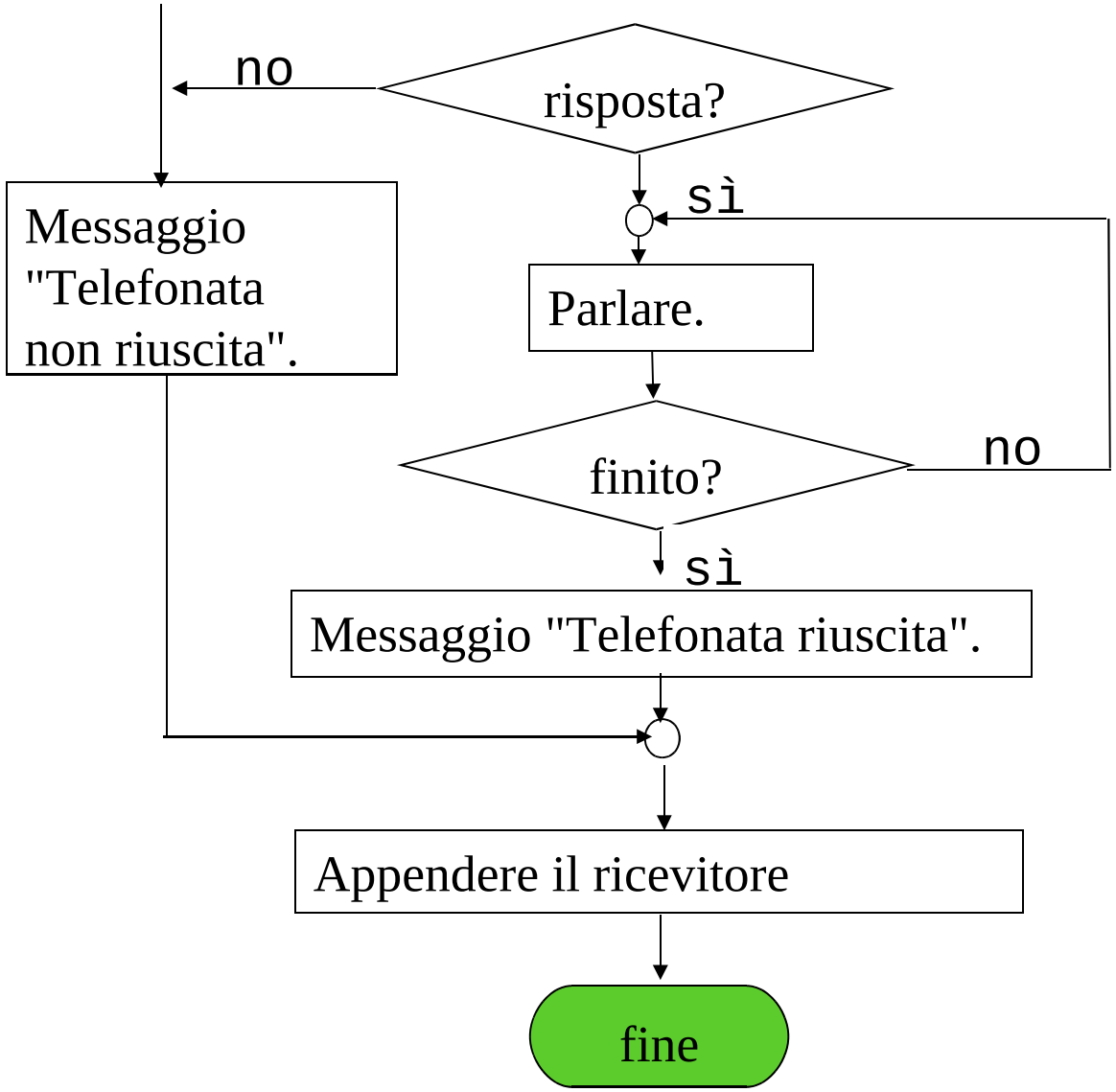


Contenitori di dati

N
025031

messaggio
T. non r.

Esecuzione



Contenitori di dati

N
025031

messaggio
T. non r.

Esempio 3

A) Analisi del problema

Problema MAGGIORE

- ***Descrizione***: vogliamo il maggiore tra due numeri interi, x e y.
- ***Requisiti in cui prevediamo i diversi casi***:
  - se x - y > 0, il maggiore è **x**
  - se x - y < 0, il maggiore è **y**
  - se x - y = 0, x e y sono uguali

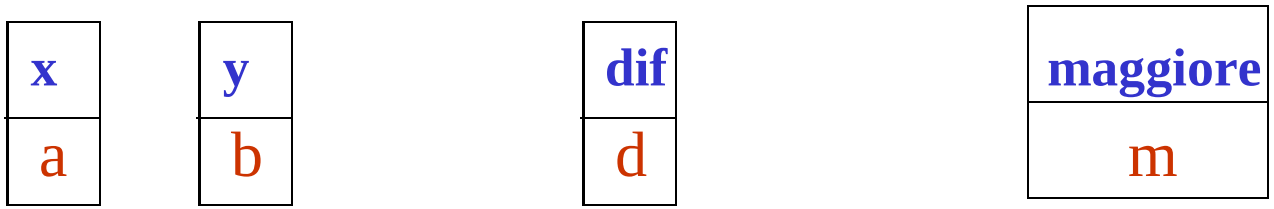
B) Specifica funzionale

- Una specifica funzionale indica
  - quali sono gli argomenti o ingressi
  - che risultato si vuole, in funzione degli ingressi
- ***Argomenti*** o ***ingressi***:
  - **x** : numero intero
  - **y** : numero intero
- Il ***risultato*** o uscita è
  - maggiore = **x**, se **x-y** ≥ 0
  - maggiore = **y**, se **x-y** < 0

I contenitori utilizzati dal nostro algoritmo

- Di quali contenitori abbiamo bisogno?
- Sicuramente di quelli per contenere i dati di ingresso ed il risultato
  - ***Contenitore del primo numero*** (ingresso)
  - ***Contenitore del secondo numero*** (ingresso)
  - ***Contenitore del risultato*** (uscita)
- Dei contenitori per i risultati intermedi
  - ***Contenitore della differenza***

Abbiamo bisogno di x, y, dif, maggiore:



ingressi ris. intermedi ris. finali

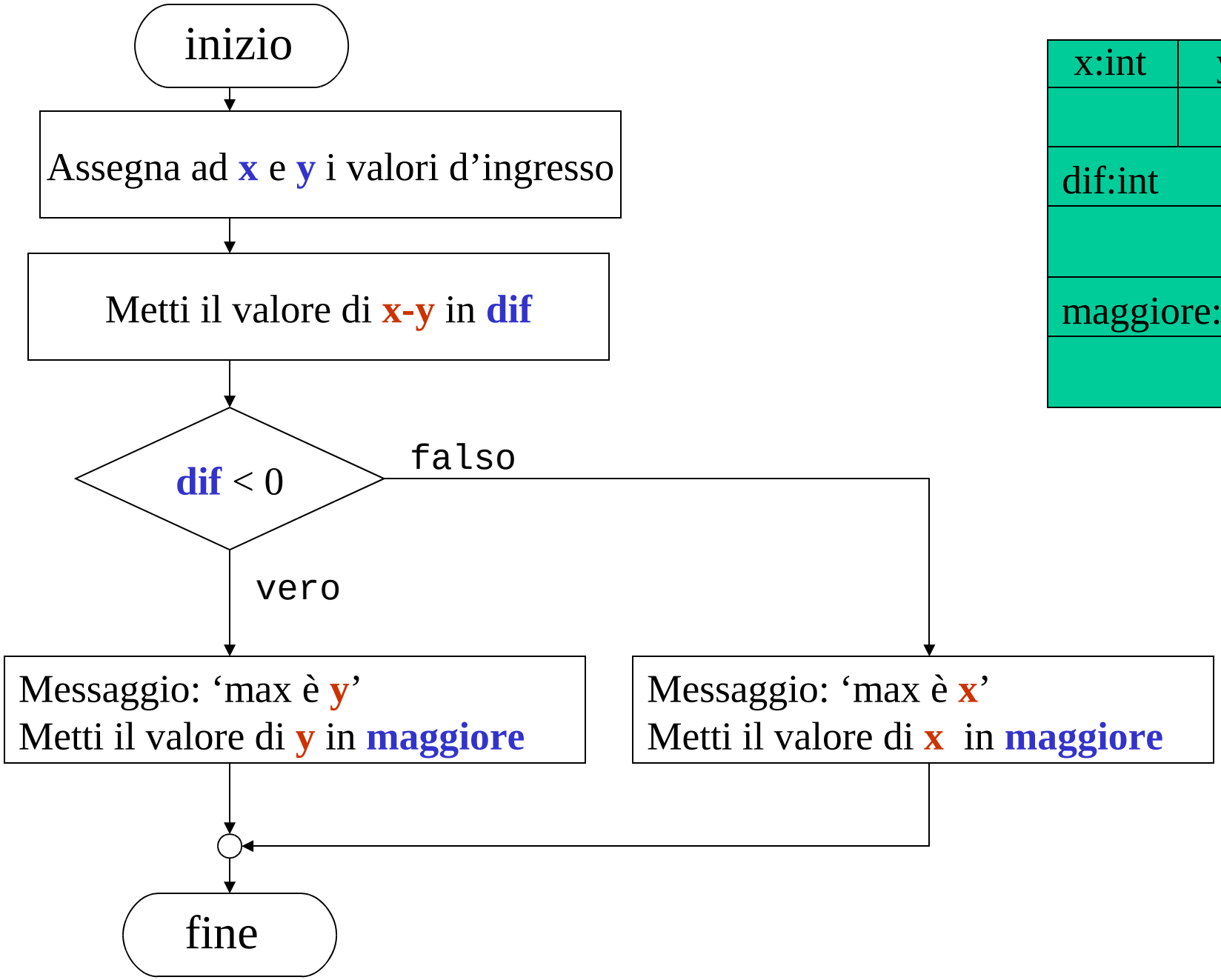
- Sono tutti ***contenitori di dati;***
- i loro contenuti sono di tipo ***intero*** e su di essi possiamo applicare le operazioni sugli interi ***somma, prodotto***, ecc.

Abbiamo bisogno di operazioni (somme, confronti, ...)

Passiamo all'algoritmo: l'idea

- Leggo gli ingressi e li metto in **x** e **y**
- Calcolo la differenza **d** = **x** - **y** e la metto in **dif**
- Confronto **dif** con 0
- Se **dif** $\geq$ 0, scrivo "max è **x**" e metto **x** in **maggiore**
- Se **dif** < 0, scrivo "max è **y**" e metto **y** in **maggiore**
- Fine
  - a questo punto ho finito e **maggiore** contiene il ***risultato finale***

Diagramma di flusso



Contenitori di dati

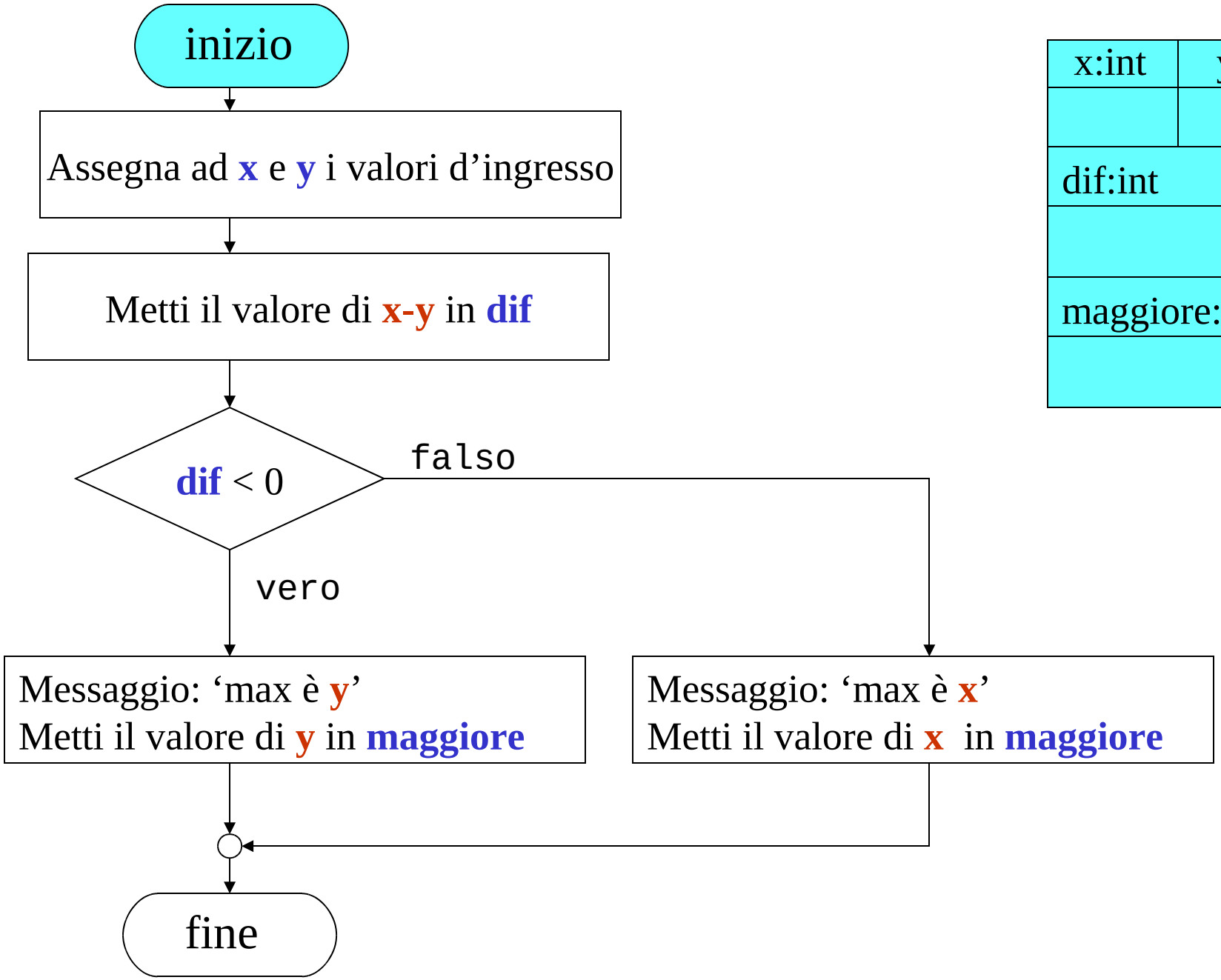
Esecuzione a mano

- Valori in ingresso:

  x: 10

  y: 10

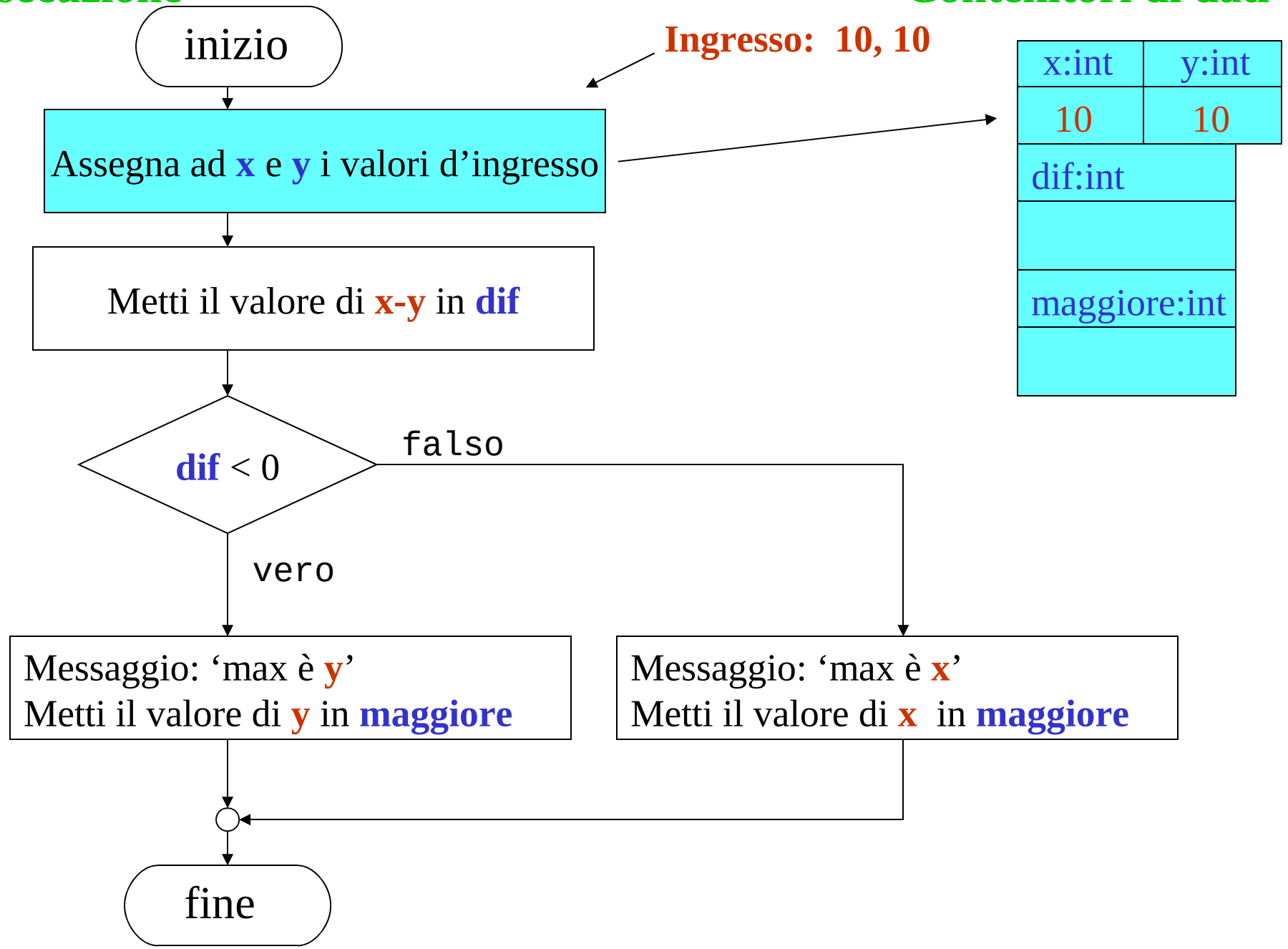
Esecuzione



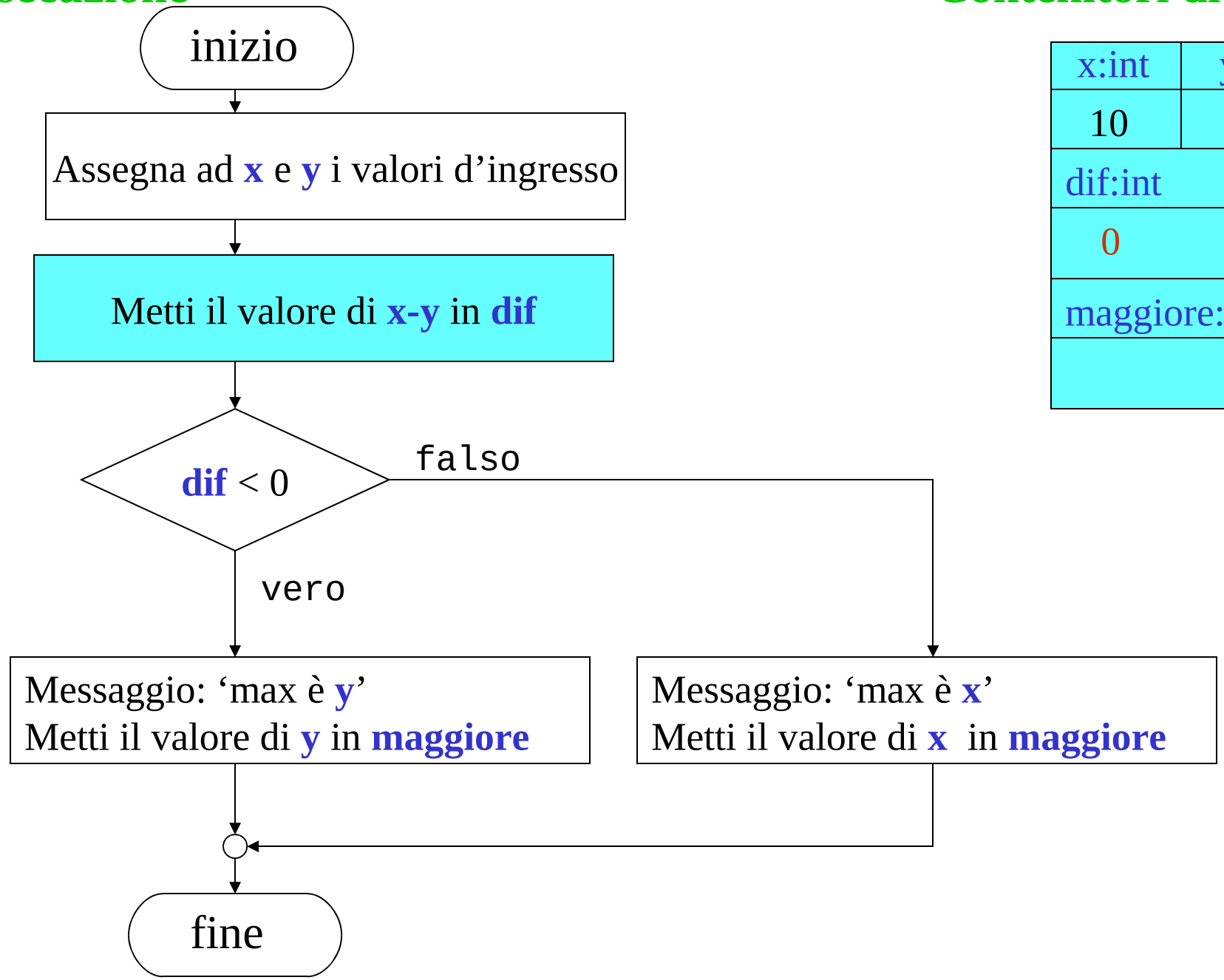
Contenitori di dati

Esecuzione

Contenitori di dati



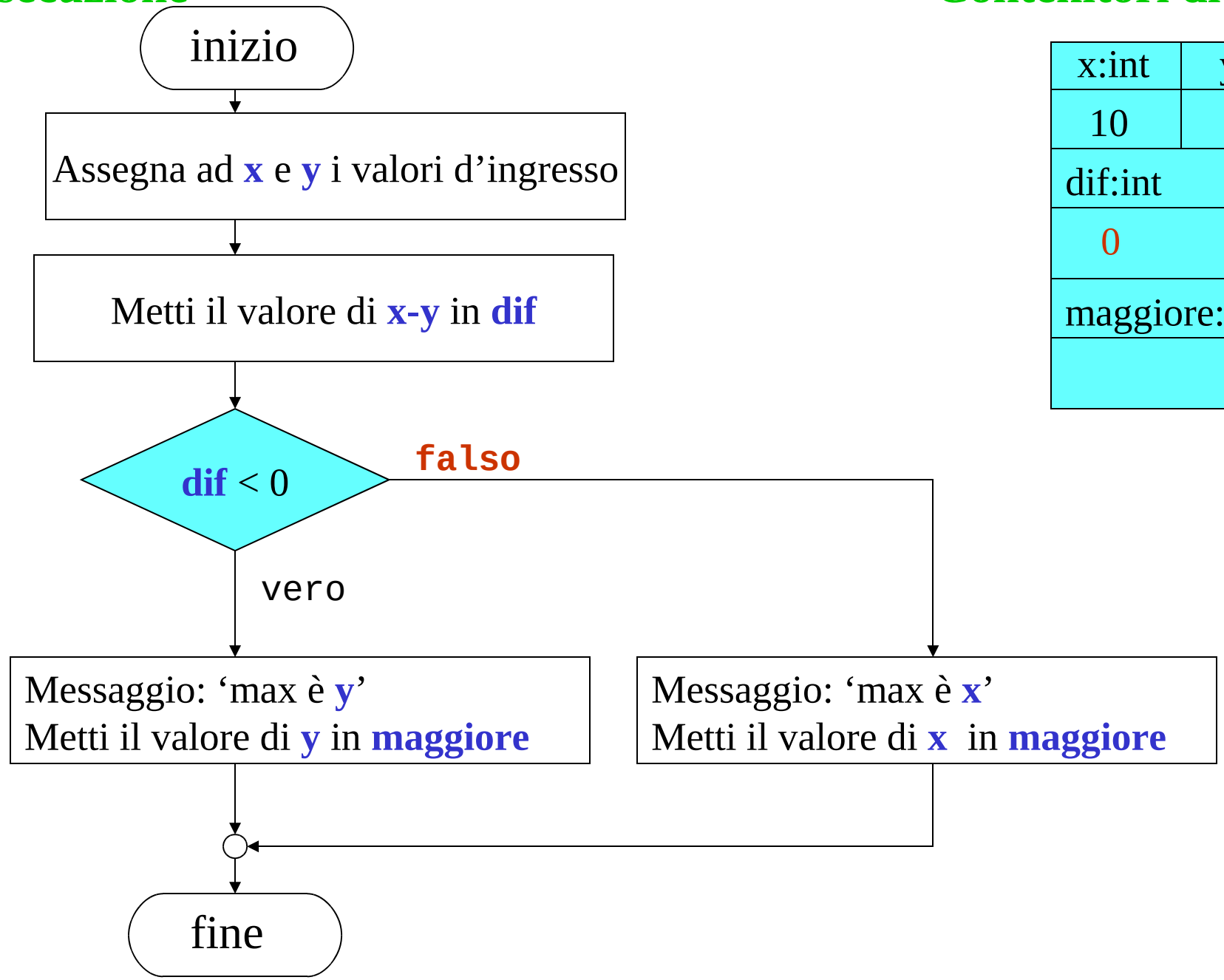
Esecuzione



Contenitori di dati

x:int	y:int
10	10
dif:int	
0	
maggiore:int	

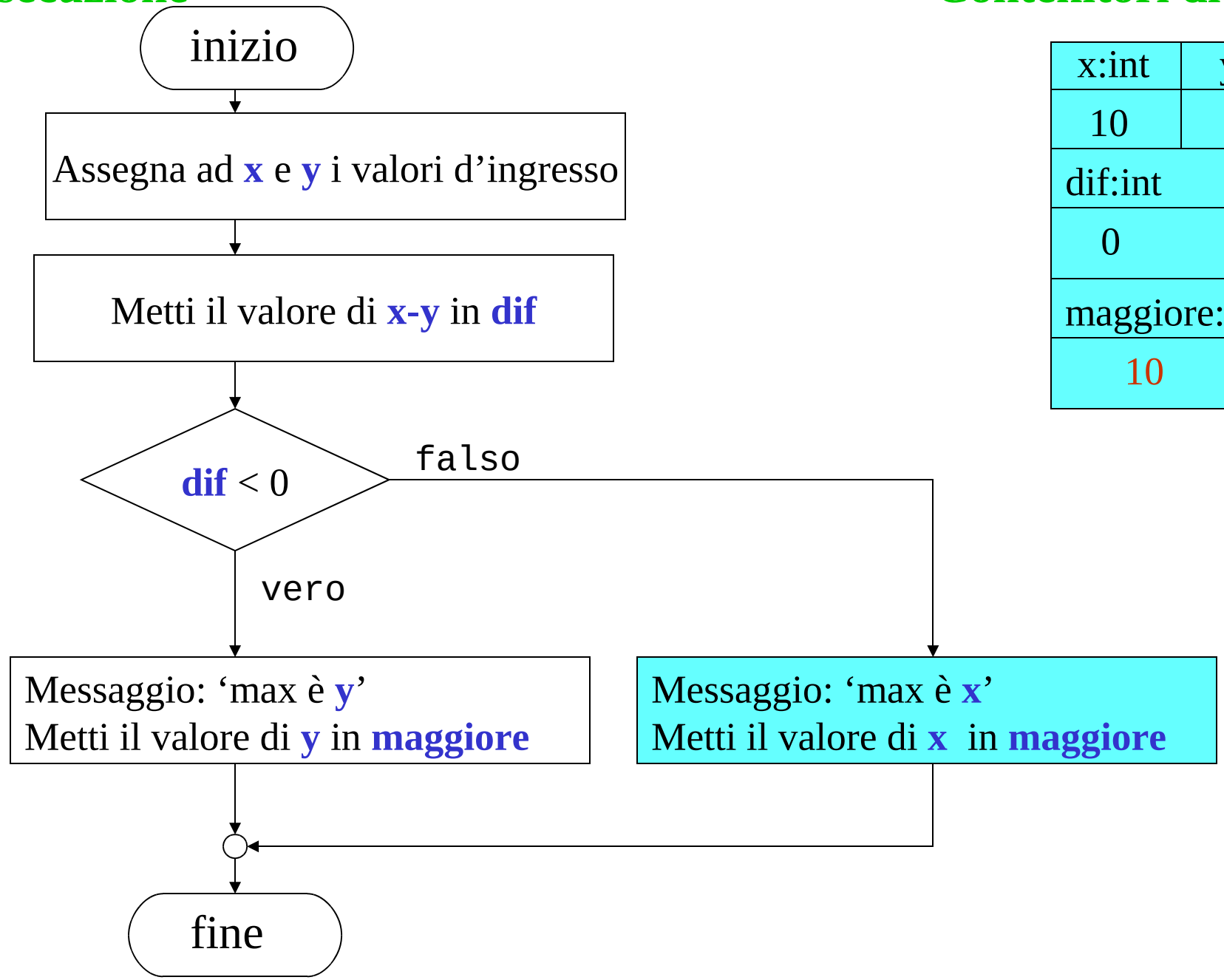
Esecuzione



Contenitori di dati

x:int	y:int
10	10
dif:int	
0	
maggiore:int	

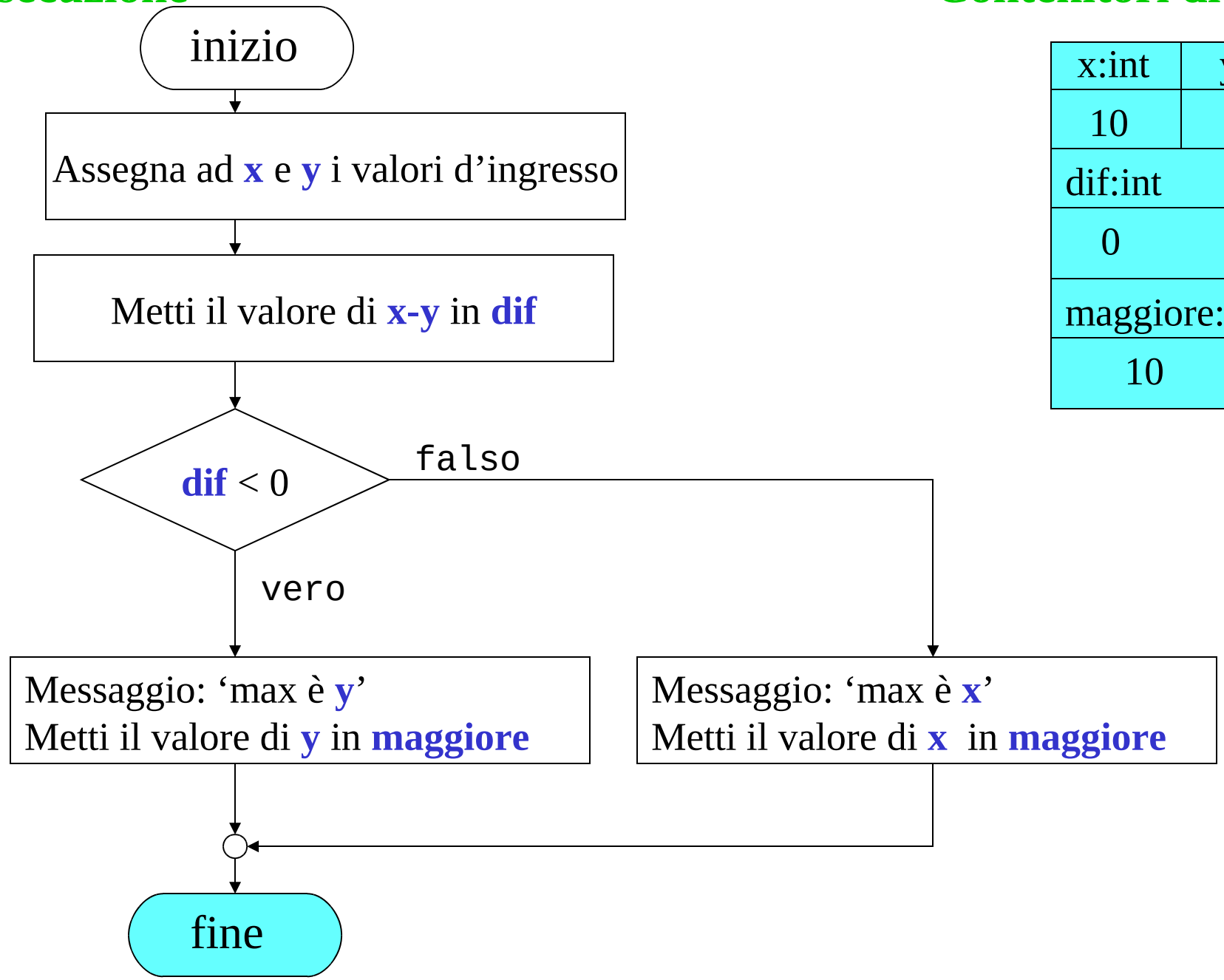
Esecuzione



Contenitori di dati

x:int	y:int
10	10
dif:int	
0	
maggiore:int	
10	

Esecuzione



Contenitori di dati

x:int	y:int
10	10

dif:int
0

maggiore:int
10

Esempio 4

A) analisi del problema

Problema prodotto

- ***Descrizione:*** vogliamo calcolare il prodotto di due numeri.
- ***Requisiti in cui prevediamo i diversi casi*** : dati due numeri x e y, il programma calcolerà il prodotto x * y, utilizzando l'addizione
  - se y = 0, il prodotto è 0
  - se y > 0, il prodotto è x + x + … + x (*y volte*)

B) Specifica funzionale

- Una specifica funzionale indica
  - quali sono gli argomenti o ingressi
  - che risultato si vuole, in funzione degli ingressi
- ***Argomenti*** o ***ingressi***:
  - **x** : numero intero
  - **y** : numero intero
- Il ***risultato*** o uscita è
  - prodotto = 0, se **y** = 0
  - prodotto = x * **y**, se **y** > 0

I contenitori utilizzati dal nostro algoritmo

- Di quali contenitori abbiamo bisogno?
- Sicuramente di quelli per contenere i dati di ingresso ed il risultato
  - ***Contenitore del primo numero*** (ingresso)
  - ***Contenitore del secondo numero*** (ingresso)
  - ***Contenitore del risultato*** (uscita)

Abbiamo bisogno di x, y, prodotto:

ingressi ris. finali

- Sono tutti ***contenitori di dati;***
- i loro contenuti sono di tipo ***intero*** e su di essi possiamo applicare le operazioni sugli interi ***somma, confronto***, ecc.

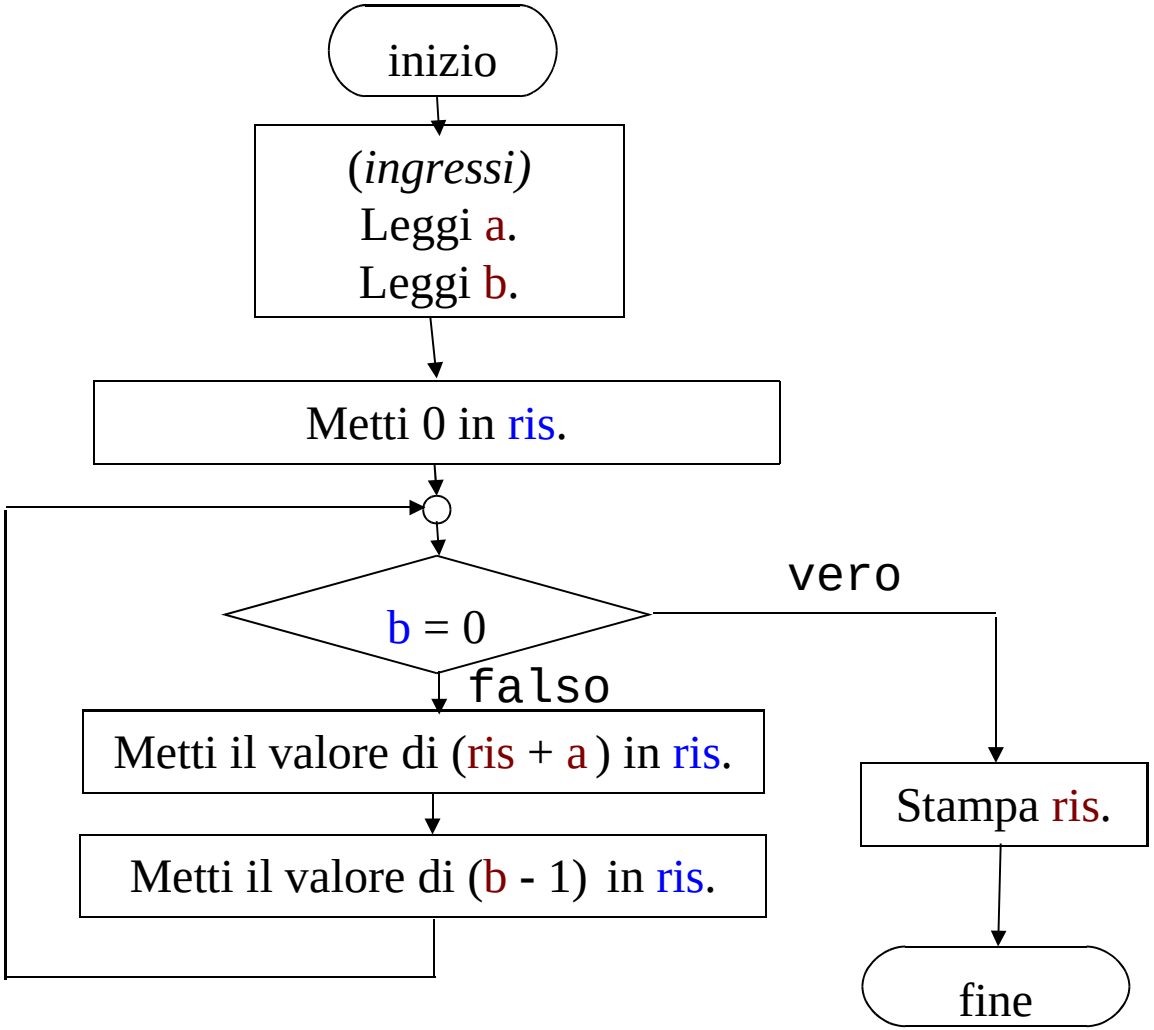
Noi assumeremo di disporre della somma, ma non del prodotto.

Passiamo all'algoritmo: l'idea

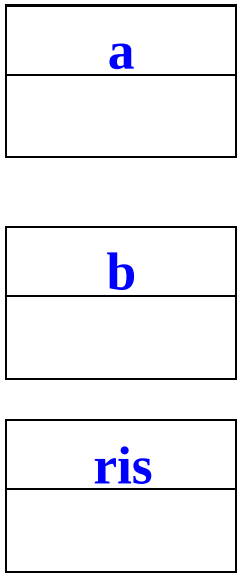
- Leggo gli ingressi e li metto in **x** e **y**
- Calcolo la differenza **d** = **x** - **y** e la metto in **dif**
- Confronto **y** con 0
- Se **y** = 0, metto 0 in **prodotto** e scrivo "il prodotto è 0"
- Se **y** > 0, calcolo il prodotto con un numero opportuno di addizioni, metto il risultato in **prodotto** e scrivo "il prodotto è **prodotto** "
- Fine

L’idea si traduce nel seguente diagramma di flusso

Diagramma di flusso per il prodotto a * b (b ≥ 0)



Contenitori di dati



Nota: 'b' svolge il ruolo di ***contatore.***

Esempio 5

Analisi del problema

Problema CONVERSIONE BASE

- Vogliamo un algoritmo che operi la conversione di un numero da una rappresentazione in base B alla rappresentazione in base 10.
- L'algoritmo sarà programmato in un linguaggio di programmazione.

Specifica funzionale

- Una specifica funzionale indica
  - quali sono gli argomenti o ingressi
  - che risultato si vuole, in funzione degli ingressi.
- ***Argomenti*** o ***ingressi***:
  - **B** : base compresa fra 2 e 16
  - **R** : una rappresentazione in base **B**
- Il ***risultato*** o uscita dev'essere un numero **N** tale che
  - **N** è la rappresentazione in base 10 del numero rappresentato da **R** nella base **B**

Contenitori di dati

- Di quali contenitori avrà bisogno il nostro algoritmo?
- Sicuramente di quelli per contenere i dati di ingresso ed il risultato:
  - ***Contenitore della base*** (ingresso)
  - ***Contenitore della rappresentazione*** (ingresso)
  - ***Contenitore del risultato*** (uscita)

Contenitori di dati

- Se non teniamo conto delle operazioni, abbiamo:

rappresentazione	base	risultato
$c_m\, c_{m-1} \ldots c_1\, c_0$	B	N

Dalla specifica:

$$N = c_m \cdot B^m + c_{m-1} \cdot B^{m-1} + \ldots + c_1 \cdot B^1 + c_0 \cdot B^0$$

Abbiamo bisogno, per i: 0,...n, del numero c_i rappresentato dalla cifra c_i

Abbiamo bisogno di somme, prodotti (e potenze?)

Contenitori di dati

- Per quanto riguarda **rappresentazione**, lo consideriamo ***contenitore di dati;***
- il suo contenuto è di tipo ***array di interi*** e sugli elementi dell'array possiamo applicare le operazioni sugli interi ***somma, prodotto***, ecc.
- Per quanto riguarda **risultato**, il suo contenuto è di tipo ***intero*** e su di esso possiamo applicare le operazioni sugli interi.

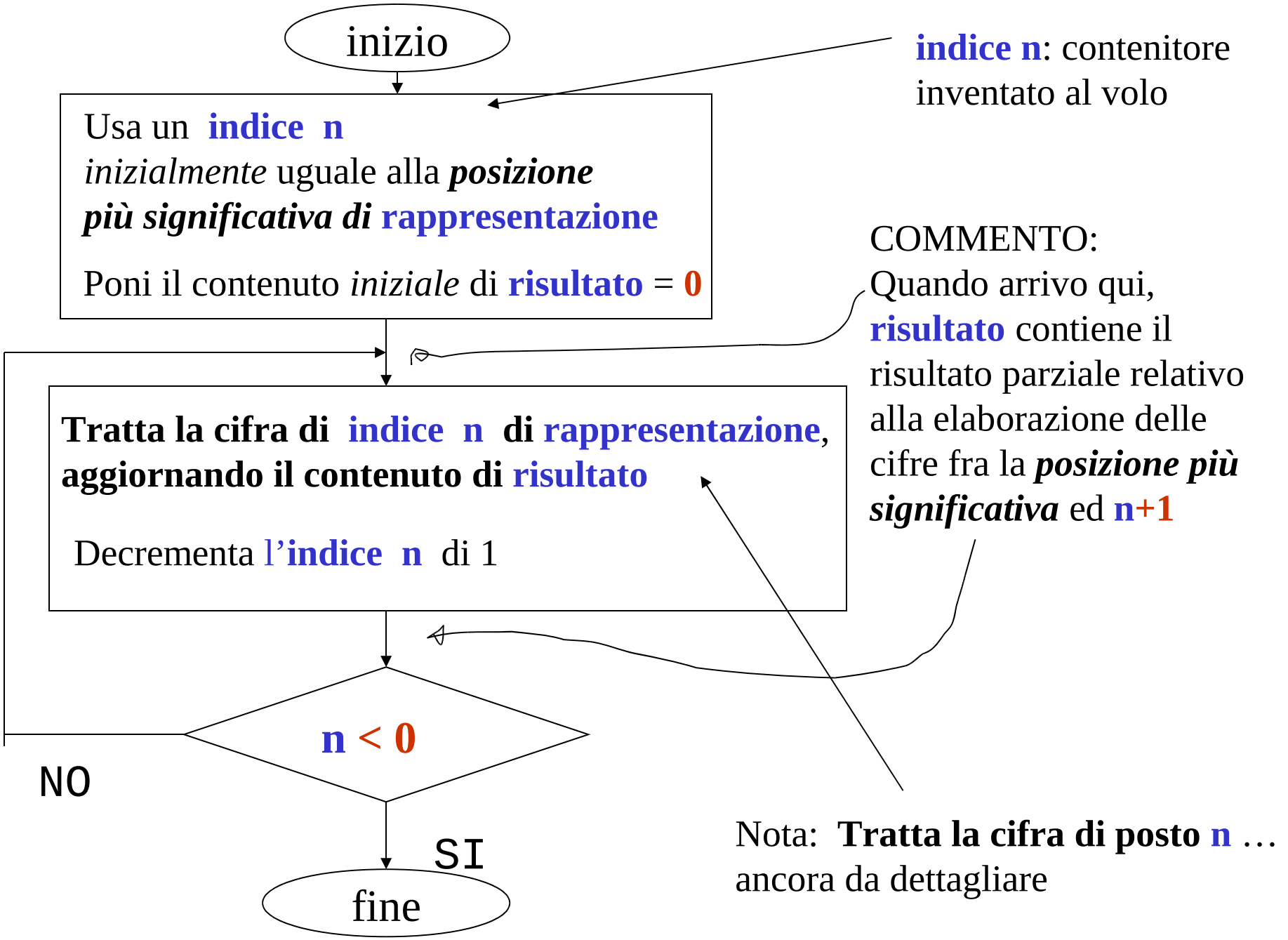
Passiamo all'algoritmo: l'idea

- Parto col contenuto ***iniziale*** di **risultato** = 0
- considero la cifra di **rappresentazione** più significativa, c_m, ed uso il suo valore per aggiornare **risultato**
  - Esempio: sommo $\mathbf{c_m \cdot B^m}$ al valore iniziale **0** di **risultato** e **risultato** ora contiene $\mathbf{c_m \cdot B^m}$

Passiamo all'algoritmo: l'idea

- Considero la cifra immediatamente successiva, c_{m-1}, ed uso il suo valore per aggiornare **risultato**
  - sommo $\mathbf{c_{m-1} \cdot B^{m-1}}$ al valore precedente $\mathbf{c_m \cdot B^m}$ di **risultato** e **risultato** ora contiene $\mathbf{c_m \cdot B^m + c_{m-1} \cdot B^{m-1}}$
- Proseguo così fino a trattare la cifra c_0
  - a questo punto ho finito e **risultato** contiene il ***risultato finale***

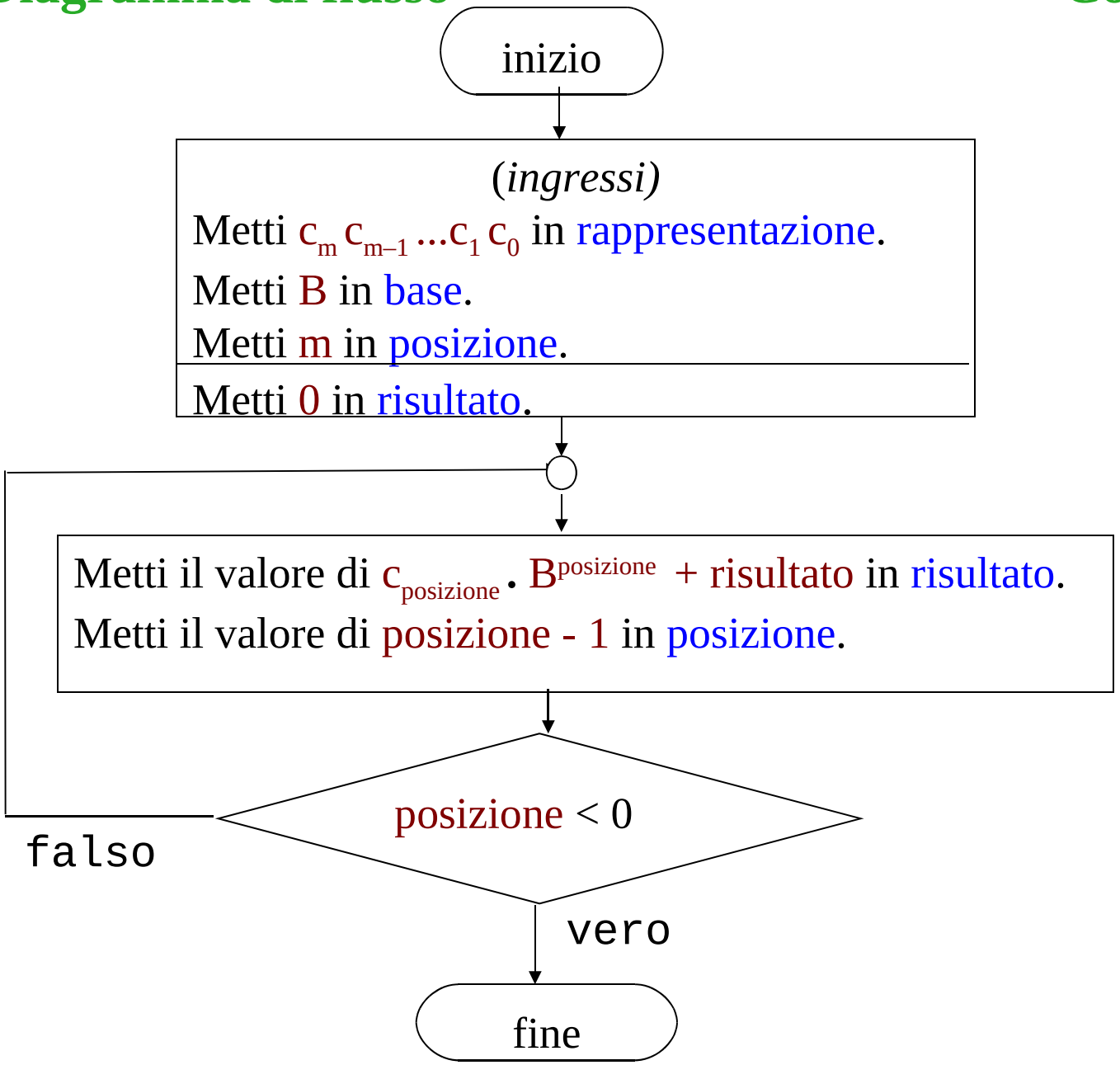
L’idea si traduce nel seguente diagramma di flusso


inizio
Usa un indice n
inizialmente uguale alla posizione più significativa di rappresentazione
Poni il contenuto iniziale di risultato = 0
Tratta la cifra di indice n di rappresentazione, aggiornando il contenuto di risultato
Decrementa l'indice n di 1
n < 0
NO
SI
fine
indice n: contenitore inventato al volo
COMMENTO:
Quando arrivo qui, risultato contiene il risultato parziale relativo alla elaborazione delle cifre fra la posizione più significativa ed n+1
Nota: Tratta la cifra di posto n … ancora da dettagliare

Un modo possibile di dettagliare “Tratta …”

- **Tratta la cifra di indice n di rappresentazione, aggiornando il contenuto di risultato** si realizza come segue
- si somma a **risultato** il ***valore della cifra di posto n*** della **rappresentazione,** moltiplicato per la **base B** della **rappresentazione *elevata ad* n**

Diagramma di flusso



Contenitori di dati

rappresentazione
$c_m\, c_{m-1} \ldots c_1\, c_0$

base
B

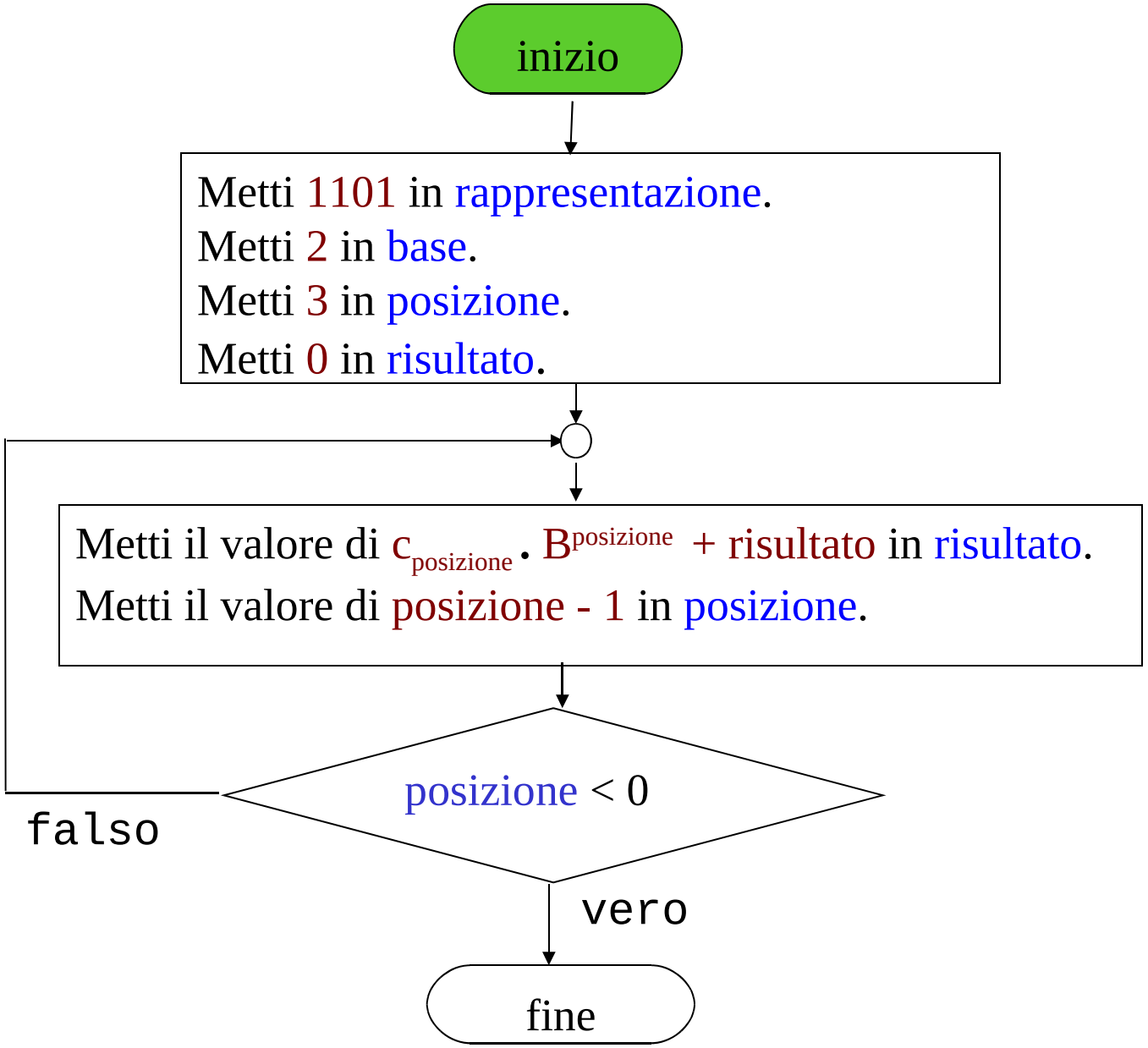
posizione
m

risultato
N

Esecuzione a mano

- Convertire 1101_2 in base 10

Esecuzione. 1.



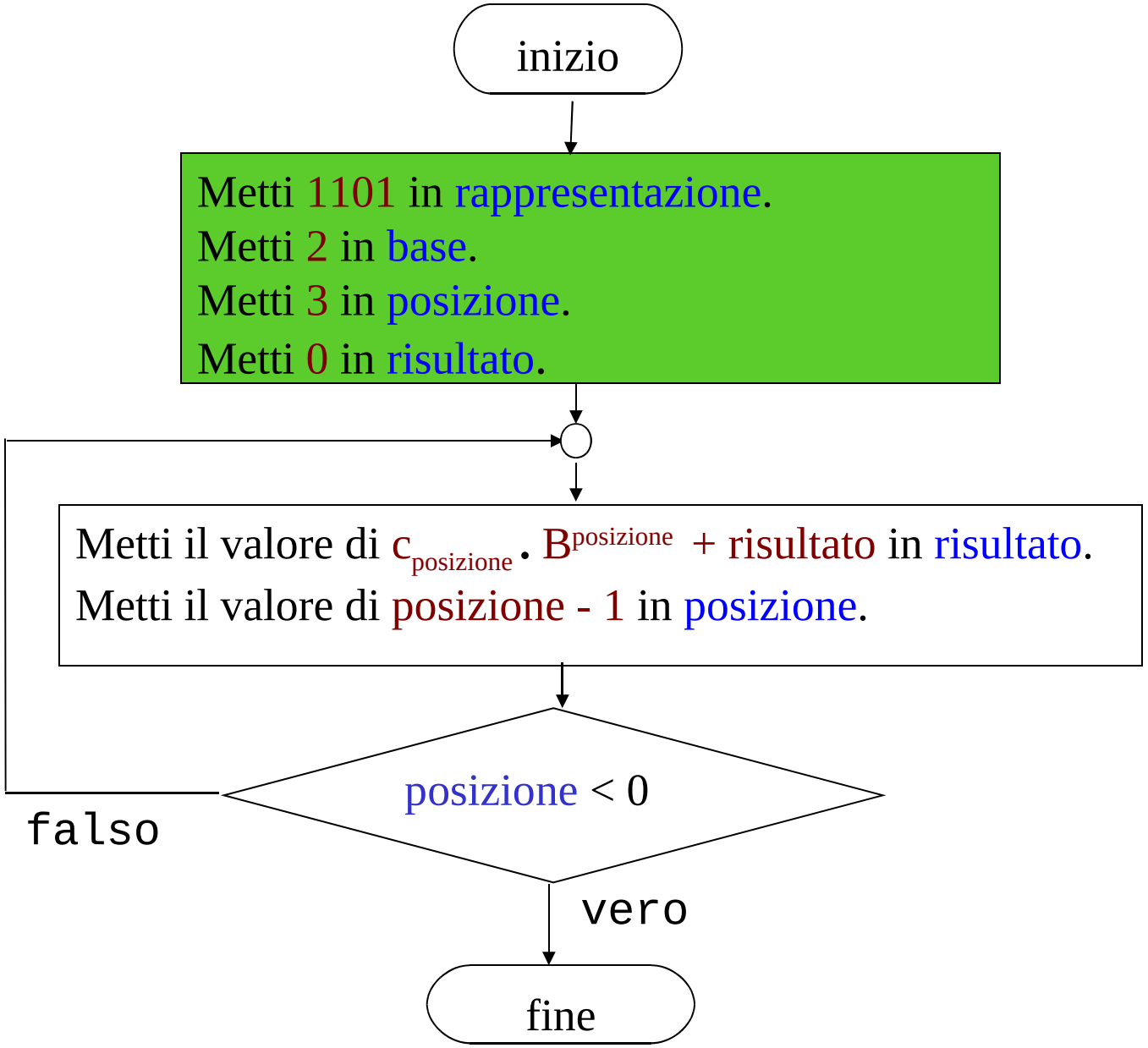
rappresentazione

base

posizione

risultato

Esecuzione. 2.



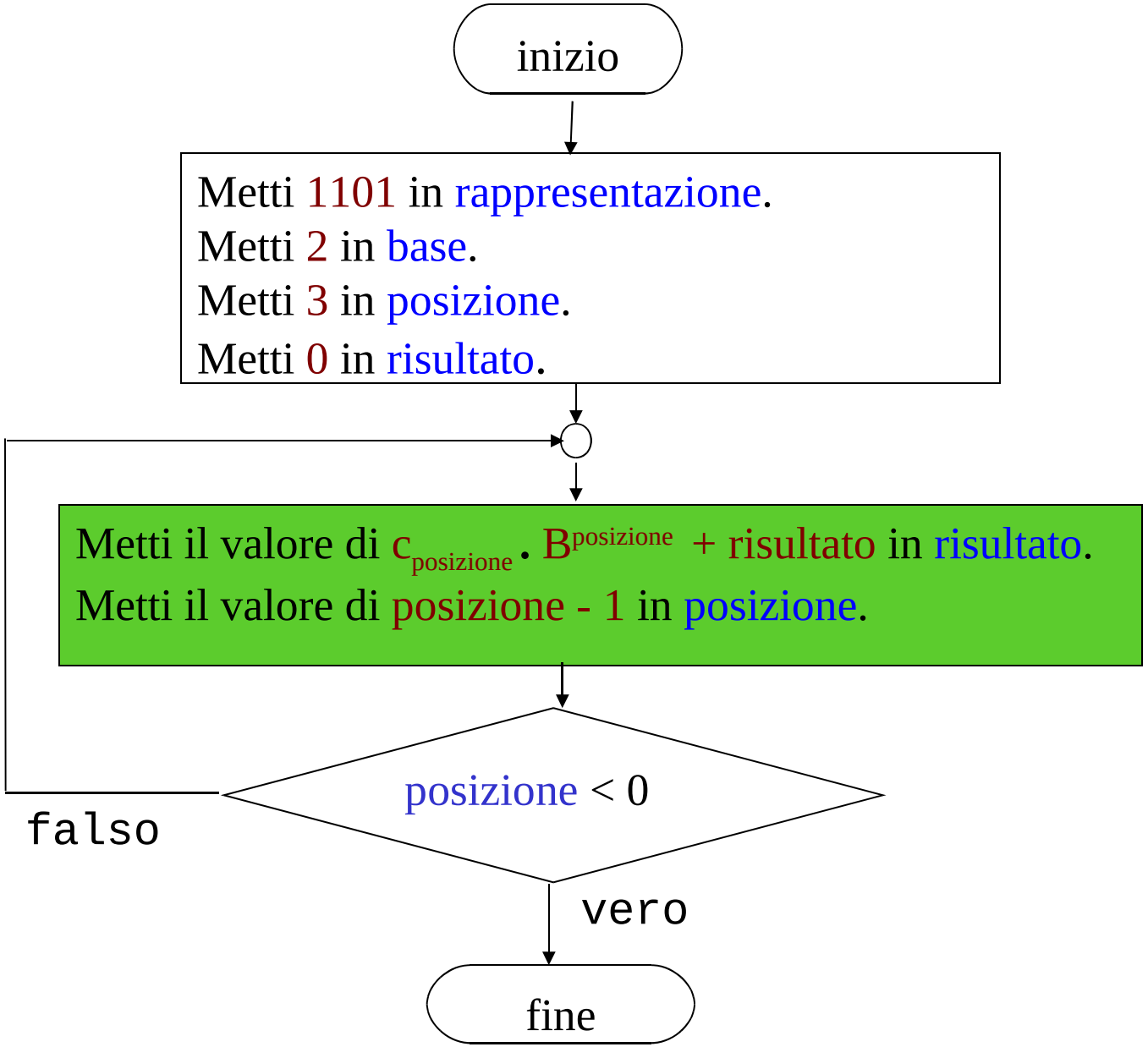
rappresentazione
1101

base
2

posizione
3

risultato
0

Esecuzione. 3.



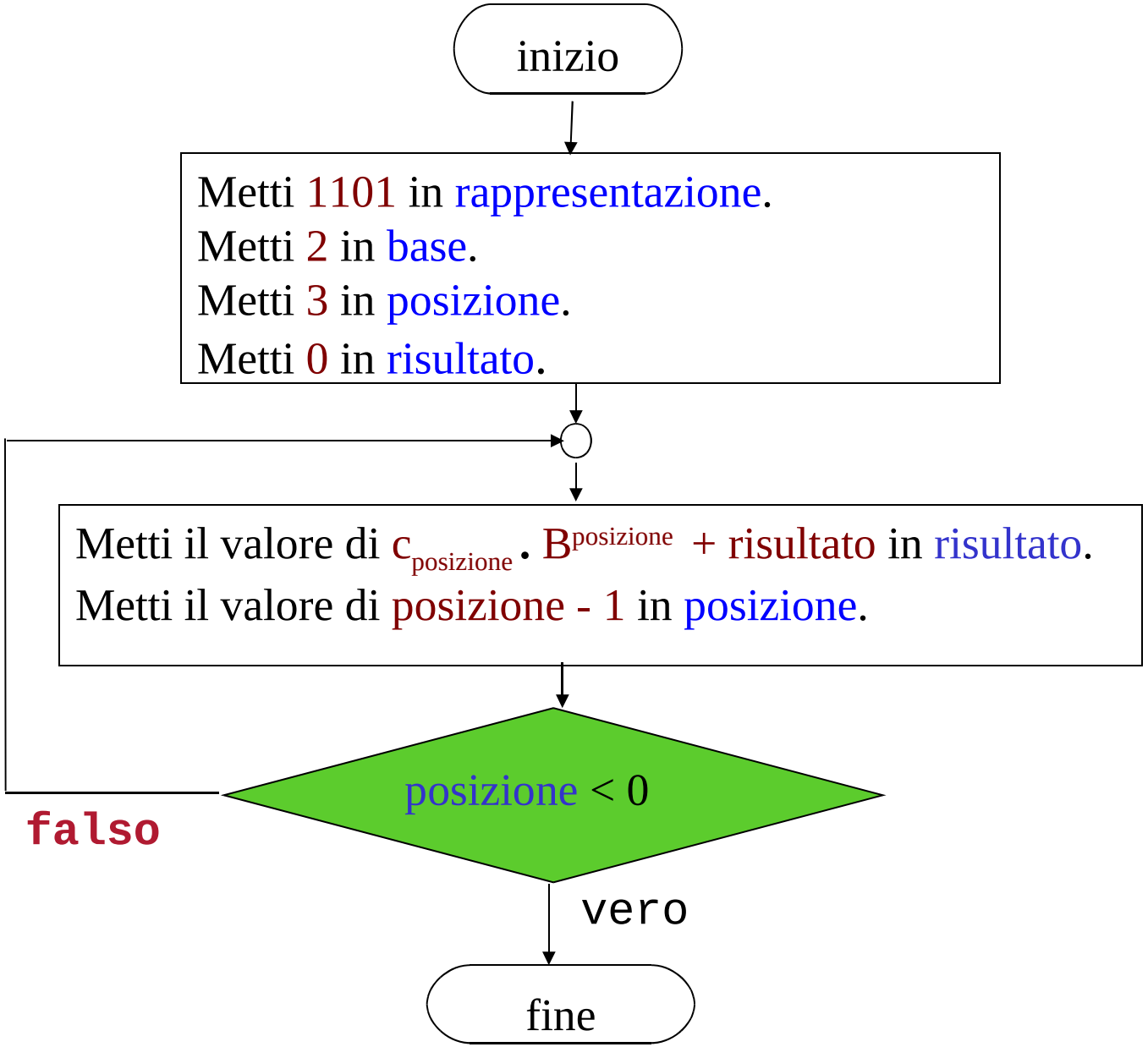
rappresentazione
1101

base
2

posizione
2

risultato
8

Esecuzione. 4.



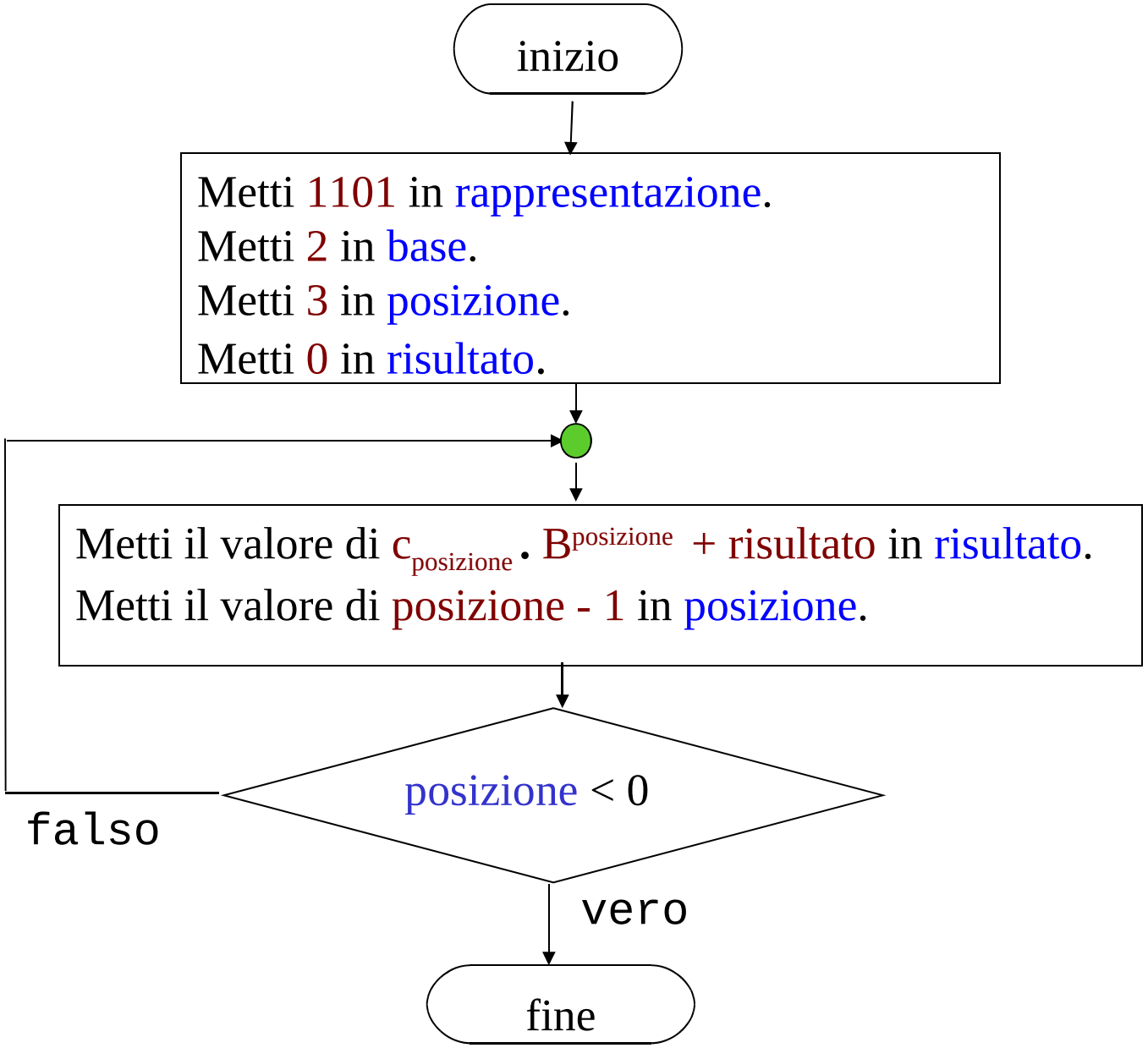
rappresentazione
1101

base
2

posizione
2

risultato
8

Esecuzione. 5.



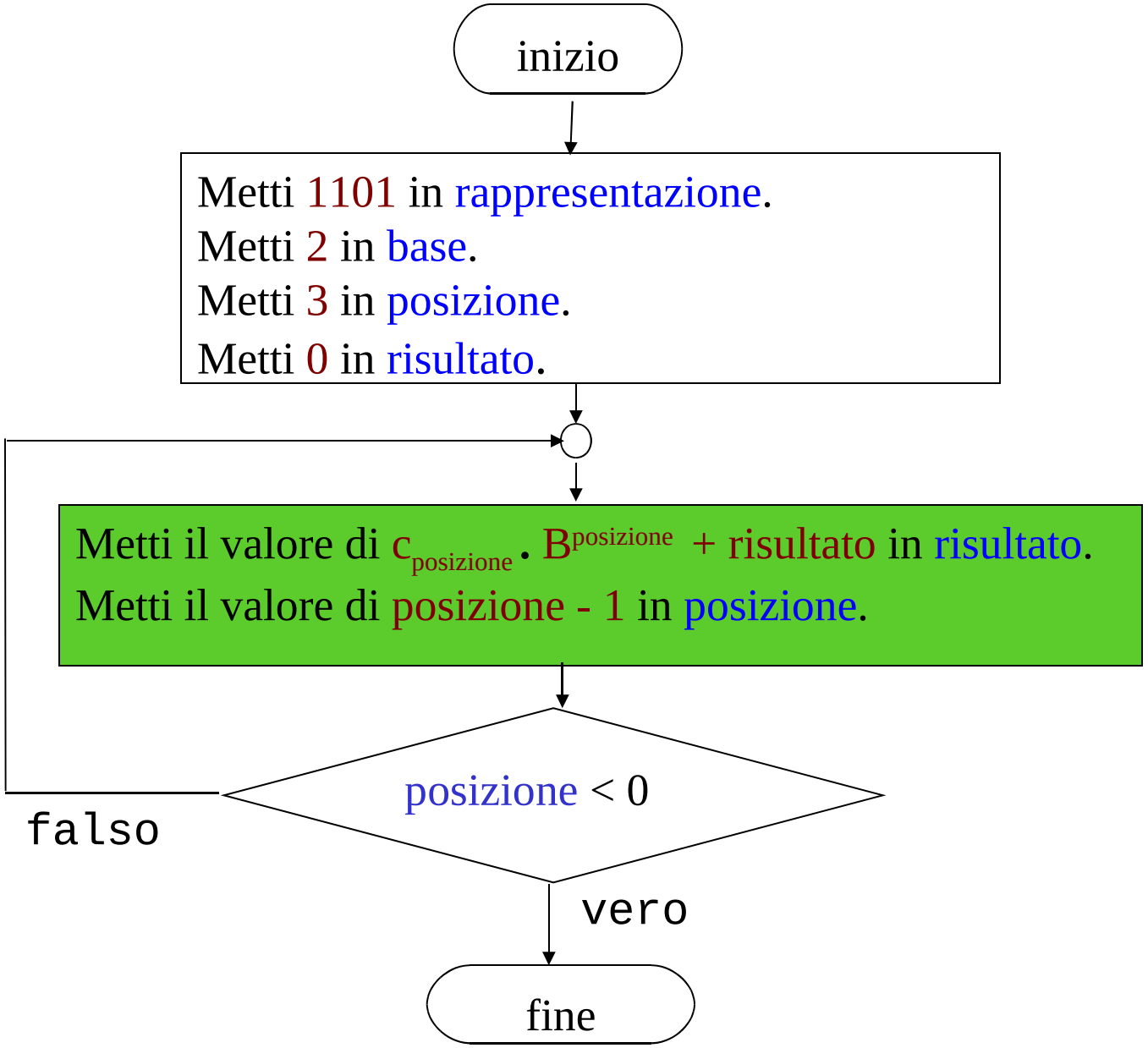
rappresentazione
1101

base
2

posizione
2

risultato
8

Esecuzione. 6.



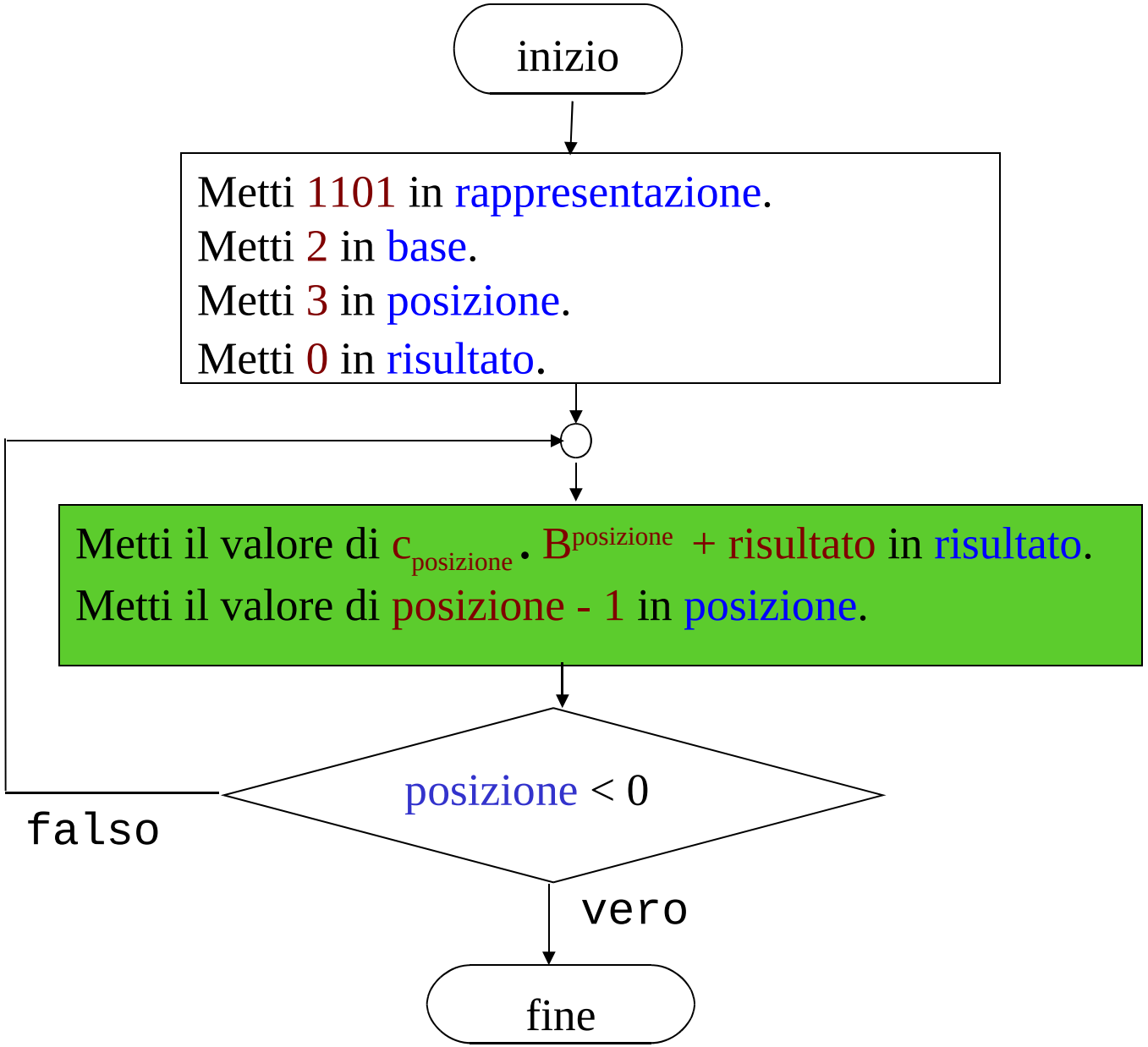
rappresentazione
1101

base
2

posizione
1

risultato
12

Esecuzione. 12.



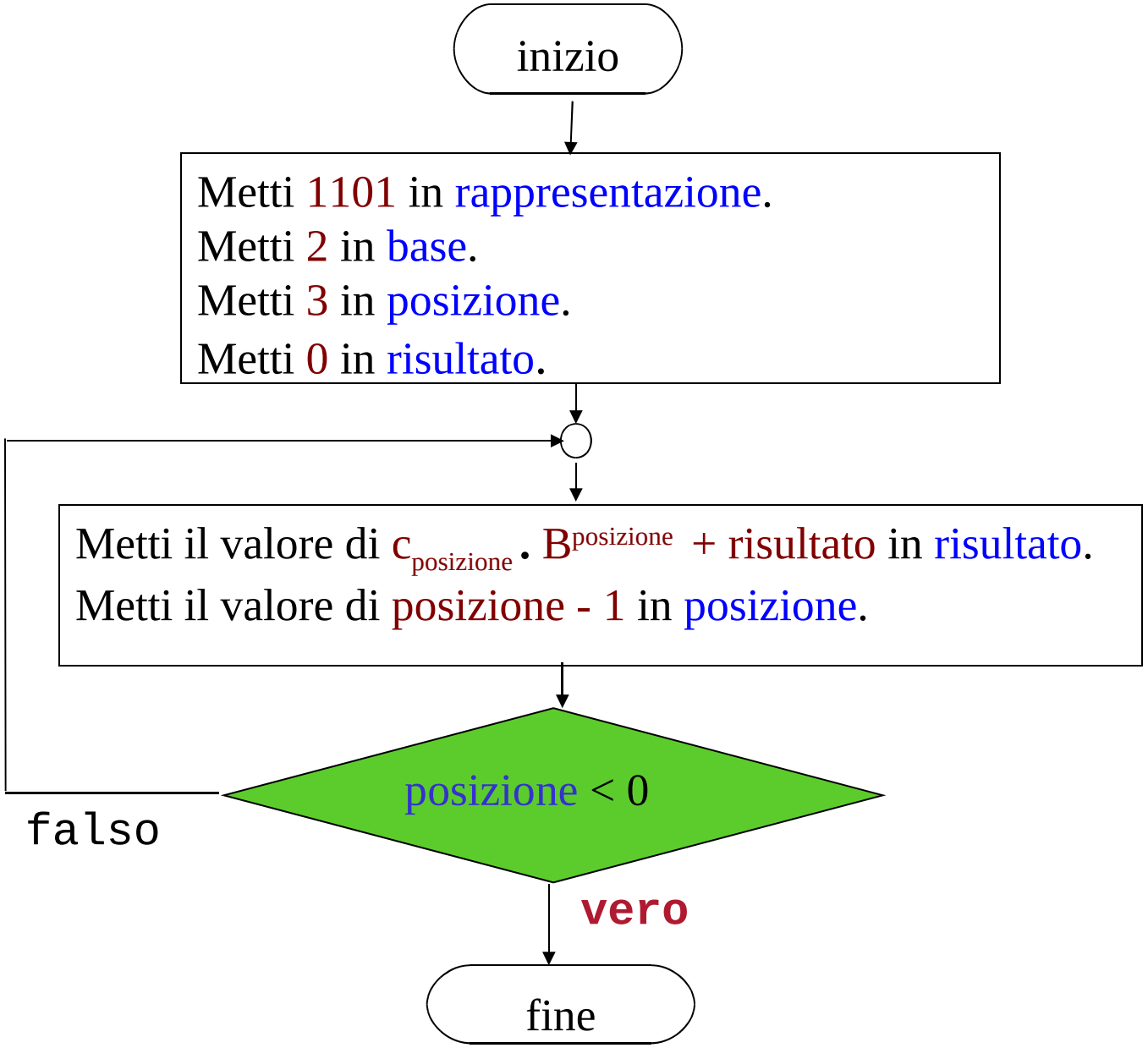
rappresentazione
1101

base
2

posizione
-1

risultato
13

Esecuzione. 13.



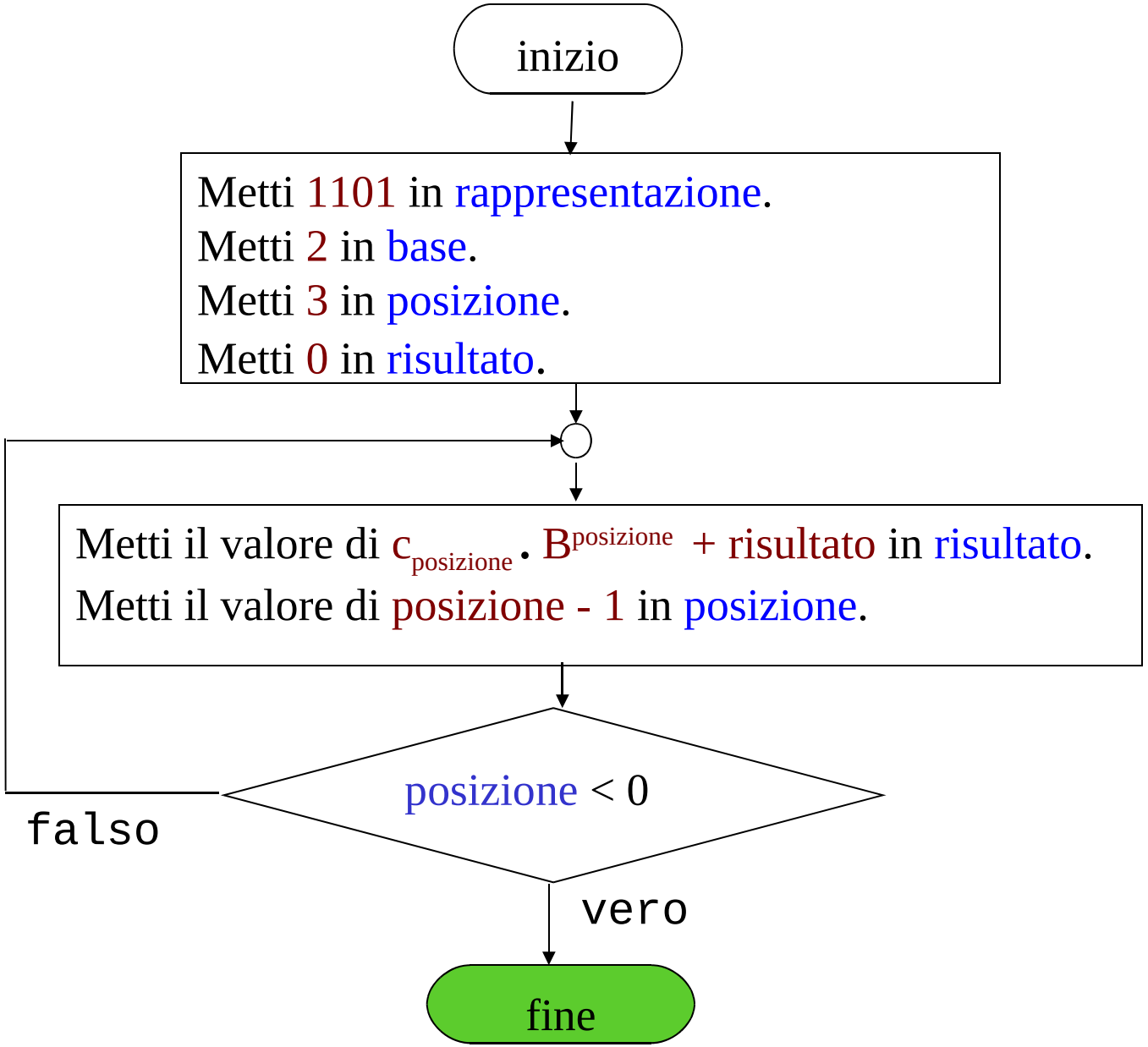
rappresentazione
1101

base
2

posizione
-1

risultato
13

Esecuzione. 14.



rappresentazione
1101

base
2

posizione
-1

risultato
13