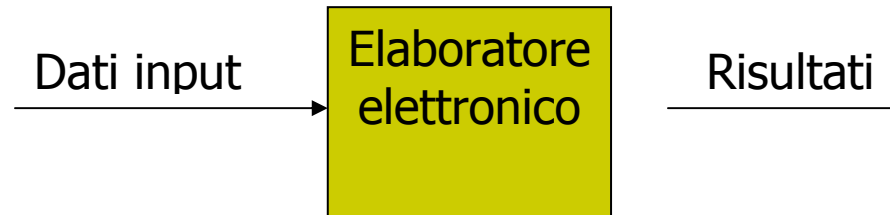
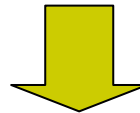


**Obiettivo:** risolvere problemi mediante l'uso di un elaboratore elettronico



Per risolvere un problema è necessario fornire una descrizione chiara e precisa del metodo risolutivo

La descrizione di un problema non fornisce un metodo risolutivo



Risolvere un problema vuol dire determinare un metodo generale che sia applicabile ad ogni istanza del problema

## Esempi di algoritmi descritti con "pseudocodice"

Esempio 1: **verifica se un numero è pari o dispari**

- a. prendi il numero
- b. calcola il resto della divisione intera del numero per 2
- c. se il resto è zero: il numero è pari, altrimenti è dispari

**a**, **b** e **c** sono istruzioni eseguite in *sequenza*. **a**, **b** sono strettamente operative, **c** è un'istruzione di *controllo decisionale*, implica una *selezione*.

Esempio 2: **determina il Massimo Comune Divisore (MCD)**

- a. prendi i due numeri
- b. calcola il resto della divisione intera del num. più grande per il più piccolo
- c. sostituisci il num. più piccolo al più grande e il resto al più piccolo
- d. finché tale resto è diverso da zero torna all'istruzione b
- e. il numero più grande (quello non nullo) è il MCD cercato

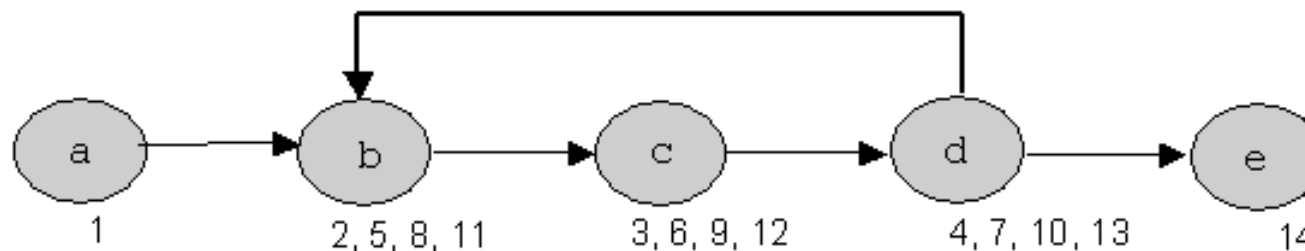
**d** è un'istruzione di *controllo iterativo*, in quanto implica una *ripetizione*.

# ALGORITMI

## Flusso di esecuzione dell'algoritmo di determinazione del MCD con i numeri 924 e 120

- passo 1**      **924 e 120**
- passo 2**      **84 è il resto della divisione intera di 924 per 120**
- passo 3**      **120 e 84**
- passo 4**      **il resto è diverso da zero, torna all'istruzione b**
- passo 5**      **36 è il resto della divisione intera di 120 per 84**
- passo 6**      **84 e 36**
- passo 7**      **il resto è diverso da zero, torna all'istruzione b**
- passo 8**      **12 è il resto della divisione intera di 84 per 36**
- passo 9**      **36 e 12**
- passo 10**     **il resto è diverso da zero, torna all'istruzione b**
- passo 11**     **0 è il resto della divisione intera di 36 per 12**
- passo 12**     **12 e 0**
- passo 13**     **il resto è uguale a zero, prosegui con l'istruzione successiva**
- passo 14**     **12 è il MCD**

L'esecuzione di ogni singola istruzione viene definita *passo*.



## Risoluzione di problemi con strumenti automatici

- La risoluzione di un problema è un **processo risolutivo** che trasforma i dati iniziali nei corrispondenti risultati finali
- Affinché possa essere realizzata attraverso l'uso di un elaboratore tale processo deve poter essere definito come **sequenza di azioni elementari**
- Non tutti i problemi sono risolvibili attraverso l'uso di un elaboratore
- Esistono classi di **problemi per cui la soluzione automatica non è proponibile**. In particolare si tratta di:
  - Problemi che presentano infinite soluzioni
  - Problemi per i quali non è stato individuato un metodo risolutivo
  - Problemi per i quali non esiste un metodo risolutivo automatizzabile


## Definizione di algoritmo


 sequenza finita di operazioni che risolve in un tempo finito una classe di problemi


*o anche*

 procedura di soluzione di una classe di problema per passi elementari

## Proprietà di un algoritmo

 **Comprensibilità:** ogni azione deve essere eseguibile dall'esecutore dell'algoritmo

 **Determinismo o *Non ambiguità*:** ogni azione deve essere univocamente interpretabile dall'esecutore

 **Efficienza o *Finitezza*** (nel tempo e nello spazio): il numero totale delle azioni da eseguire per ogni insieme dei dati di ingresso deve essere finito

## Problemi calcolabili

↪ I problemi risolvibili automaticamente sono definiti **calcolabili**

- Si tratta di funzioni per cui esiste un algoritmo che termina e che, fornito un input, produca un risultato
- Appartengono alla classe dei problemi calcolabili:
  - ↪ problemi di calcolo (funzioni): dato  $x$  determinare  $f(x)$
  - ↪ problemi decisionali
  - ↪ problemi di ricerca
  - ↪ problemi di enumerazione

## Tipi di algoritmi

### ALGORITMI NUMERICI

- ☞ operazioni aritmetiche
- ☞ calcolo espressioni
- ☞ ricerca del massimo di un insieme di numeri
- ☞ ordinamento di insiemi di numeri
- ☞ .....
- ☞ .....

### ALGORITMI NON NUMERICI

- ✗ installazione di un personal computer
- ✗ montaggio di una stazione di ricezione satellitare
- ✗ ricetta gastronomica
- ✗ scelta del percorso stradale tra due città
- ✗ localizzazione di un libro in una biblioteca
- ✗ .....

## Riflessioni

- ☹ Quasi sempre esistono algoritmi diversi per risolvere lo stesso problema.
- ☹ Importante risulta la scelta dell'algoritmo più efficiente.
- ☹ In alcuni casi la descrizione della soluzione di un problema può richiedere conoscenze o capacità o attitudini ritenute scontate da parte di un esecutore-uomo.
- ☹ La correttezza o determinismo dell'algoritmo può dipendere dalle assunzioni fatte sul tipo di esecutore.
- ☹ **La soluzione di un problema può essere ricondotta ad attività che a loro volta sono sottoproblemi, che richiedono una descrizione algoritmica.**



L'obiettivo della **programmazione strutturata** è di rendere un flusso ordinato il passaggio tra le istruzioni dall'inizio alla fine degli algoritmi risolutivi dei problemi calcolabili e, quindi, dei programmi che risolvono automaticamente i problemi.

Corrado Böhm e Giuseppe Jacopini hanno dimostrato (**teorema di Boehm-Jacopini**) che

C. Böhm, G. Jacopini "Flow Diagrams, Turing Machines, and Languages with only two Formation Rules",  
*Communications of the ACM*, Vol. 9, No. 5, May 1966, pp.336-371

**un qualunque algoritmo può essere descritto (e, perciò, tutti i programmi possano essere scritti) con l'utilizzo solo delle tre strutture: *sequenza*, *selezione* e *iterazione* (senza l'uso del salto *goto*).**

**Selezione** e **Iterazione** costituiscono le **strutture o istruzioni di controllo**.

Si dice quindi che **le tre strutture**

- ➡ **Sequenza**
- ➡ **Diramazione o Selezione**
- ➡ **Ciclo o Iterazione**

sono **complete**.

## Diagramma di flusso

- ❏ È una rappresentazione grafica di un algoritmo.
- ❏ Mostra con chiarezza il modo in cui operano le strutture di controllo
- ❏ È preferibile allo pseudocodice

## Pseudocodice

- ❏ Linguaggio informale per la descrizione degli algoritmi
- ❏ È simile al linguaggio naturale
- ❏ Non è un linguaggio di programmazione

## Lo schema a blocchi o flow chart

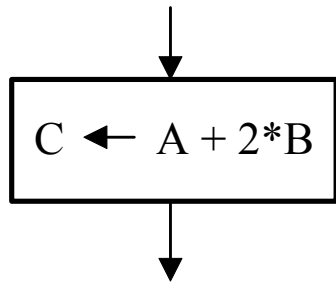
È uno strumento rudimentale, anche se molto diffuso, per rappresentare il flusso di esecuzione di un programma, o meglio per ***descrivere graficamente un algoritmo***.

I singoli blocchi indicano operazioni elementari (su variabili).

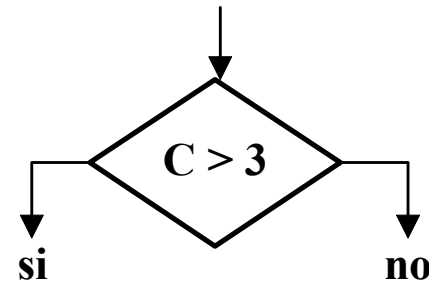
I blocchi che possono essere presenti in un flow chart sono:

- ✓ il **blocco di inizio** della descrizione dell'algoritmo
- ✓ il **blocco di fine** della descrizione dell'algoritmo
- ✓ il **blocco di assegnazione** ad una variabile del risultato di un'espressione
- ✓ il **blocco di test** di verità di un'espressione logica (o predicato o condizione); in base al risultato del test sono indicati due differenti percorsi di avanzamento dell'algoritmo
- ✓ il **blocco di lettura** che assegna ad una variabile un valore ricavato da un dispositivo di input (per esempio dalla tastiera)
- ✓ il **blocco di scrittura** che riporta su un dispositivo di output (per esempio il video) il valore di una variabile

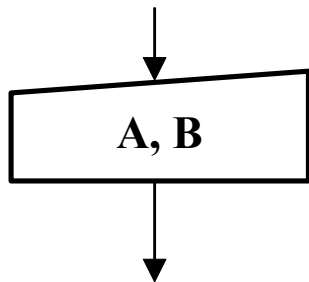
## I blocchi di un flow chart



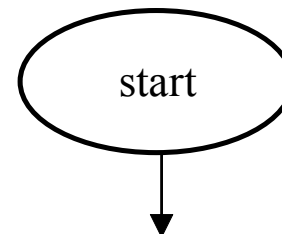
Blocco di **assegnazione**



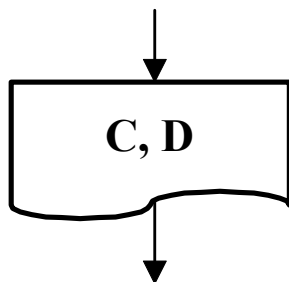
Blocco di **test**



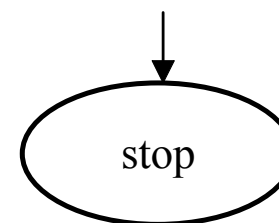
Blocco di **lettura**



Blocco di **inizio**



Blocco di **scrittura**



Blocco di **fine**

# PROGRAMMAZIONE STRUTTURATA

## Il flow chart delle strutture fondamentali

### Sequenza

Istruzioni semplici di Ingresso, Uscita, Assegnazione

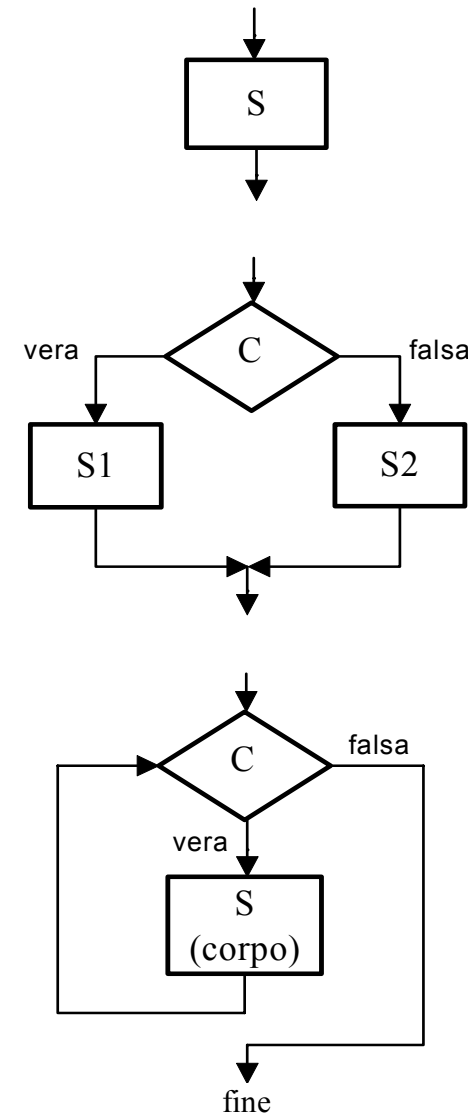
### Istruzioni di controllo

#### Diramazione o Selezione

- Esprime la scelta tra due possibili azioni mutuamente esclusive
- Si valuta l'espressione relazionale o logica specificata all'interno del blocco:
- Se il risultato dell'espressione è vero si prosegue verso un ramo della diramazione altrimenti si segue l'altro ramo

#### Ciclo o Iterazione

- Esprime la ripetizione di un'azione
- Ad ogni iterazione si valuta l'espressione relazionale o logica specificata all'interno del blocco
- Finché il risultato dell'espressione è vero si ripete l'azione. L'iterazione si interrompe non appena la condizione non è più soddisfatta



**Sequenza**

**Diramazione  
o  
Selezione**

**Ciclo o Iterazione  
(a condizione iniziale)**

## Le istruzioni di una sequenza

### Istruzione di ingresso

#### lettura

riceve dati dall'unità in ingresso e li assegna alle variabili

### Istruzione di uscita

#### scrittura

trasmette all'unità di uscita il valore di una espressione

### Istruzione di assegnazione

#### Istruzione della forma $V=E$

Calcola il valore dell'espressione E e lo assegna alla variabile V

Un'espressione può essere

- ✓ **Aritmetica**: una sequenza di variabili e costanti combinate mediante operatori aritmetici (+, -, \*, /)  
Ad esempio:  $a+b*8/10$
- ✓ **Relazionale o logica**: fornisce un risultato binario vero o falso, usano operatori logici o relazionali (==, <, >, !=, &&, ||)  
Ad esempio:  $a != b$

## Altre strutture di controllo

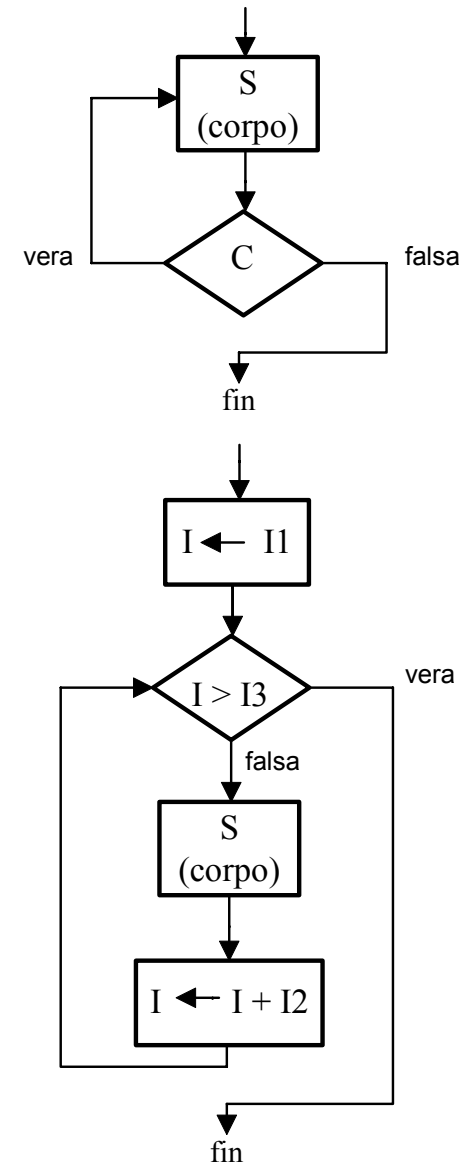
**I** rappresenta l'**indice** del ciclo

**I1** è il **valore iniziale** dell'indice del ciclo

**I2** è l'**incremento** (step) dell'indice dopo ogni iterazione

**I3** è il **valore finale** dell'indice del ciclo

Il corpo del ciclo viene eseguito  $(I3 - I1 + 1)$  volte

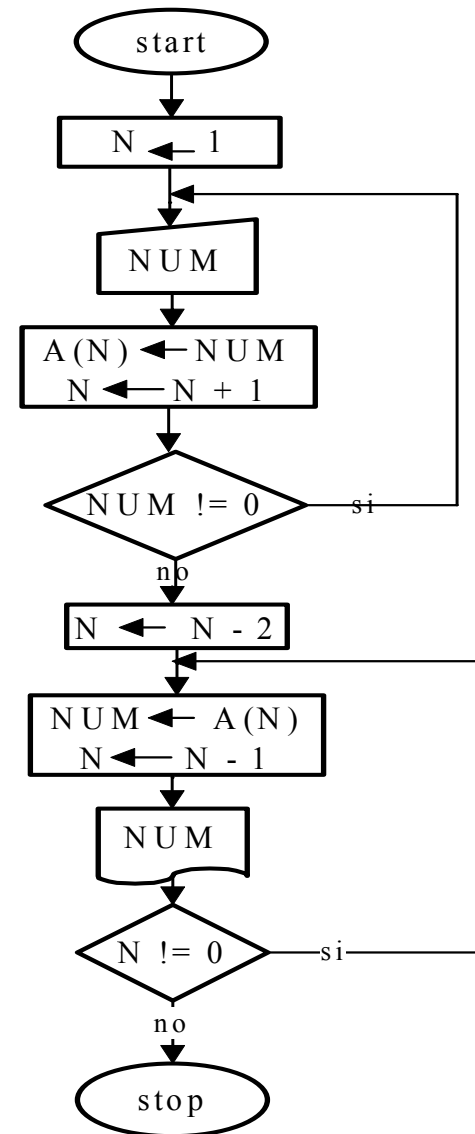


**Ciclo  
a condizione  
finale**

**Ciclo  
enumerativo**

## Un esempio di flow chart

*Algoritmo: inversione di una sequenza di numeri chiusa da uno zero*





## Progettazione e programmazione strutturata

I vantaggi del progettare e programmare facendo uso delle sole strutture fondamentali:

- ***migliore chiarezza o leggibilità;***
- ***maggior facilità di modifica;***
- ***maggior facilità di codifica;***
- ***minore probabilità di errore.***

Le istruzioni sono eseguite in maniera sequenziale: **non sono utilizzate le istruzioni che consentono un trasferimento del controllo**

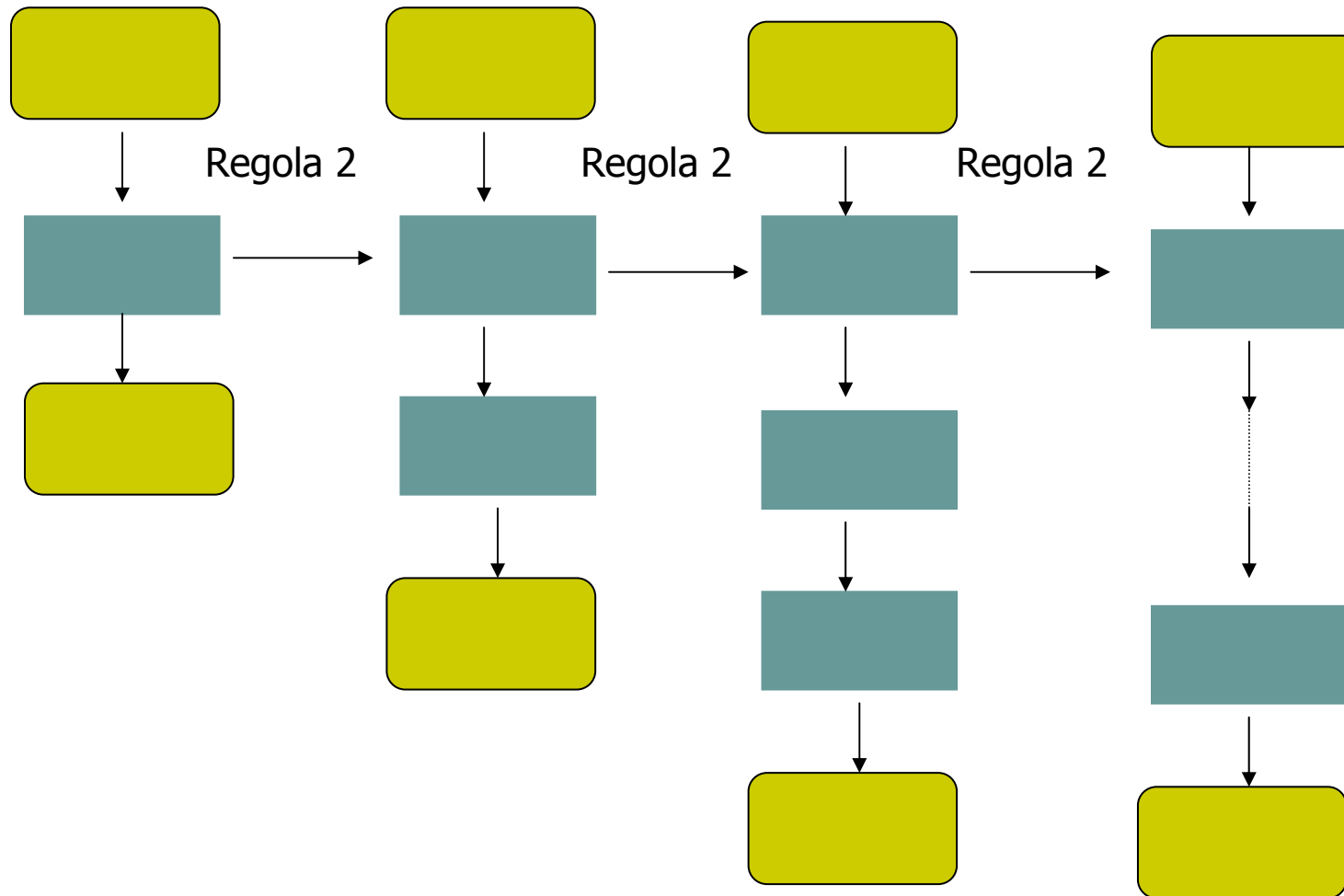
“Eliminazione dei goto”

## Regole per la creazione di programmi strutturati

Nel flow chart la programmazione strutturata è garantita mediante le seguenti regole:

- 1) Iniziare un diagramma con il flusso di **azioni anche non elementari**
- 2) Ogni **azione** (rettangolo) può essere **sostituita da due azioni in sequenza**
- 3) Ogni **azione** può essere **sostituita da una qualsiasi struttura di controllo** (sequenza, selezione, iterazione)
- 4) Regola di **nidificazione**: le regole 2 e 3 possono essere applicate ripetutamente ed in qualsiasi ordine
- 5) Un diagramma di flusso ha **un unico punto di ingresso ed un unico punto di uscita**

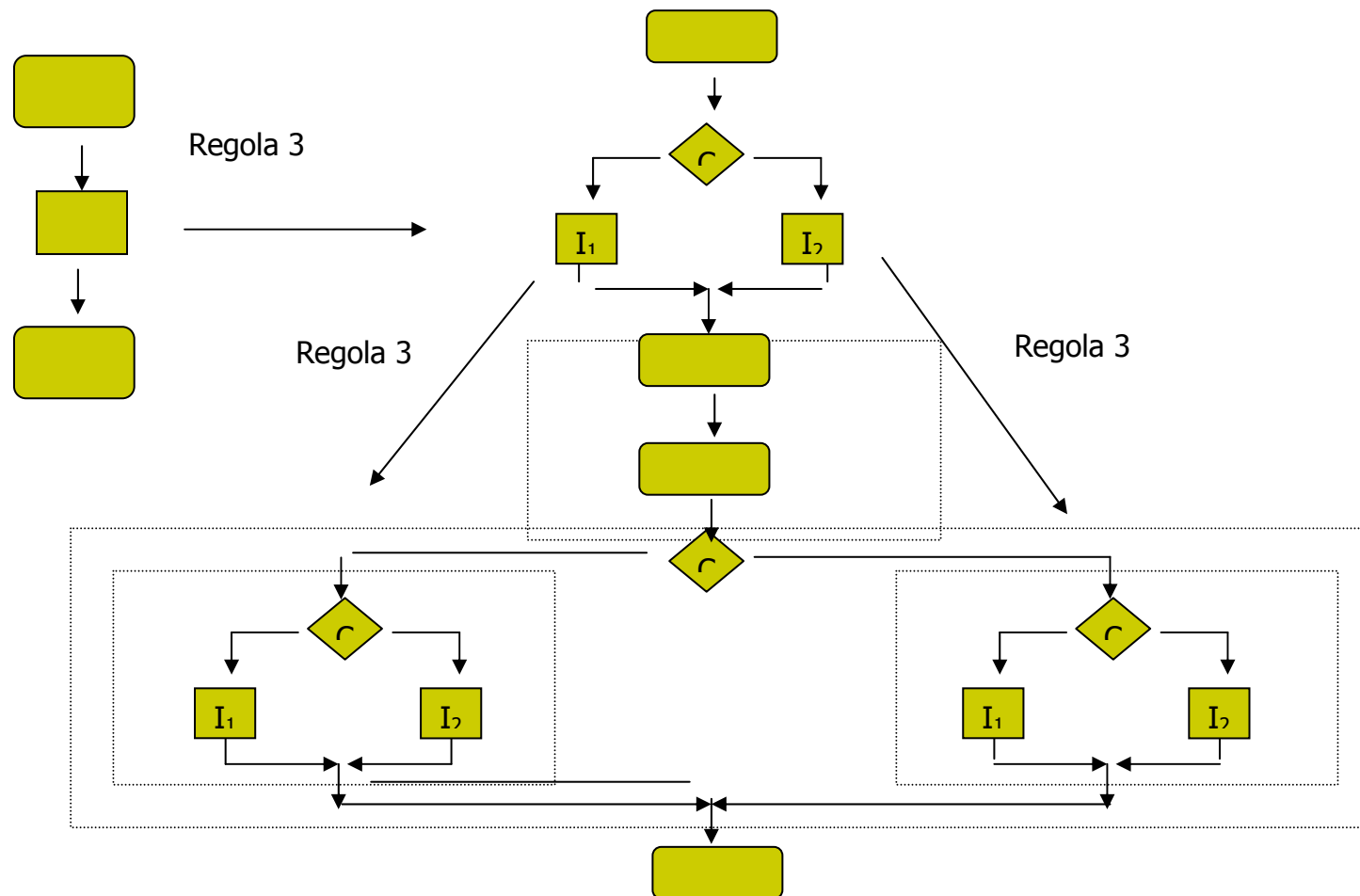
## Applicazione della Regola 2



# PROGRAMMAZIONE STRUTTURATA

## Applicazione della Regola 3

Applicando ripetutamente la regola 3 al diagramma di flusso elementare si otterrà un diagramma con strutture di controllo nidificate in modo ordinato

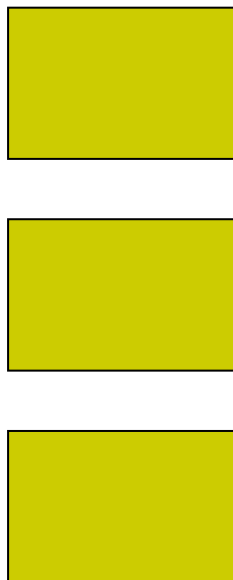


## Applicazione della Regola 4

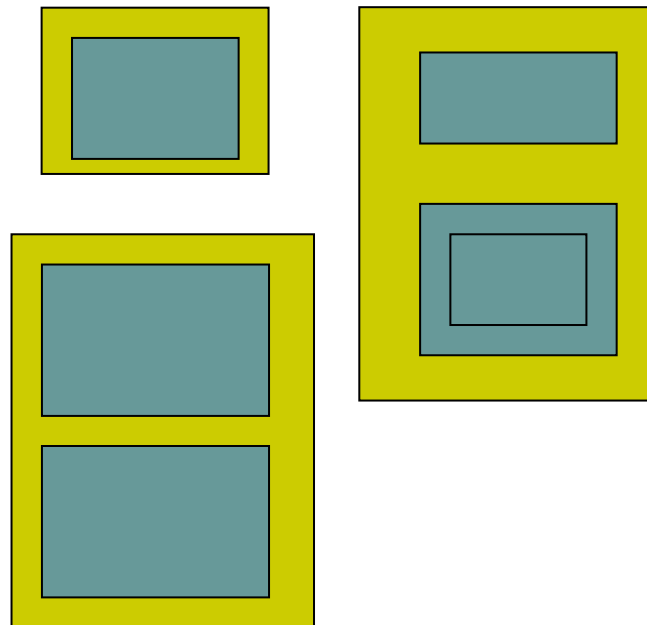
La regola 4 genererà strutture più grandi, più complesse e nidificate a più livelli

I diagrammi di flusso che possono essere formati applicando le 4 regole costituiscono l'insieme di tutti i possibili diagrammi di flusso strutturati e, quindi, l'insieme di tutti i programmi strutturati

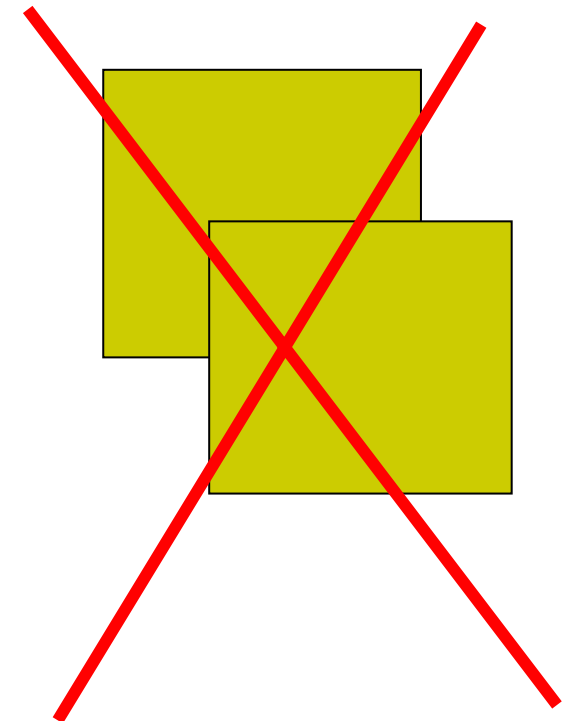
Mattoncini accatastati



Mattoncini nidificati



Mattoncini sovrapposti

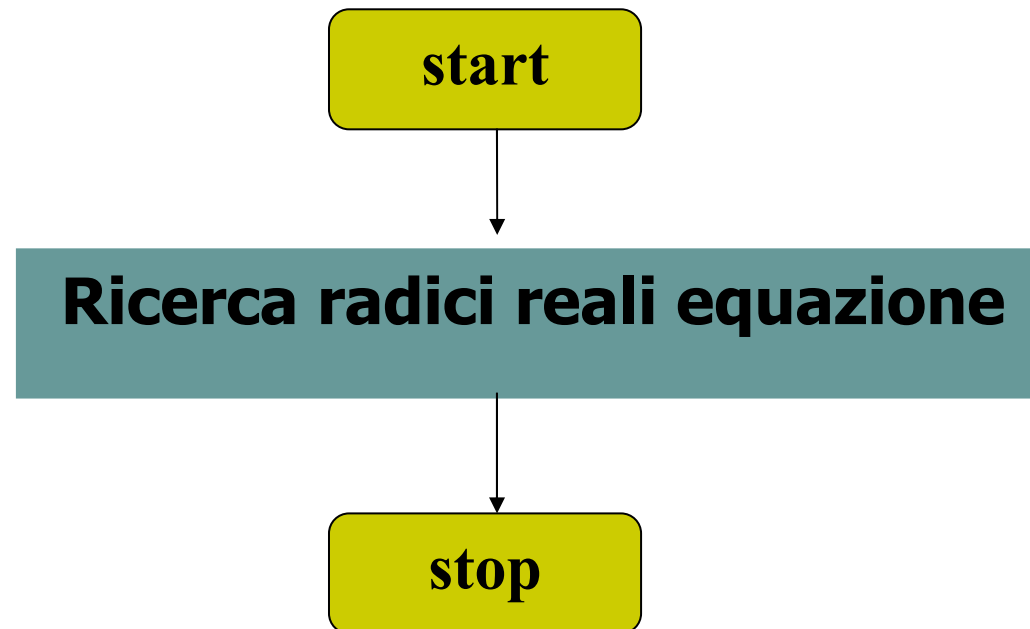


## Regole per la creazione di programmi strutturati: Un esempio (1/5)

**Problema:** Calcolo delle radici reali di un'equazione di 2° grado.

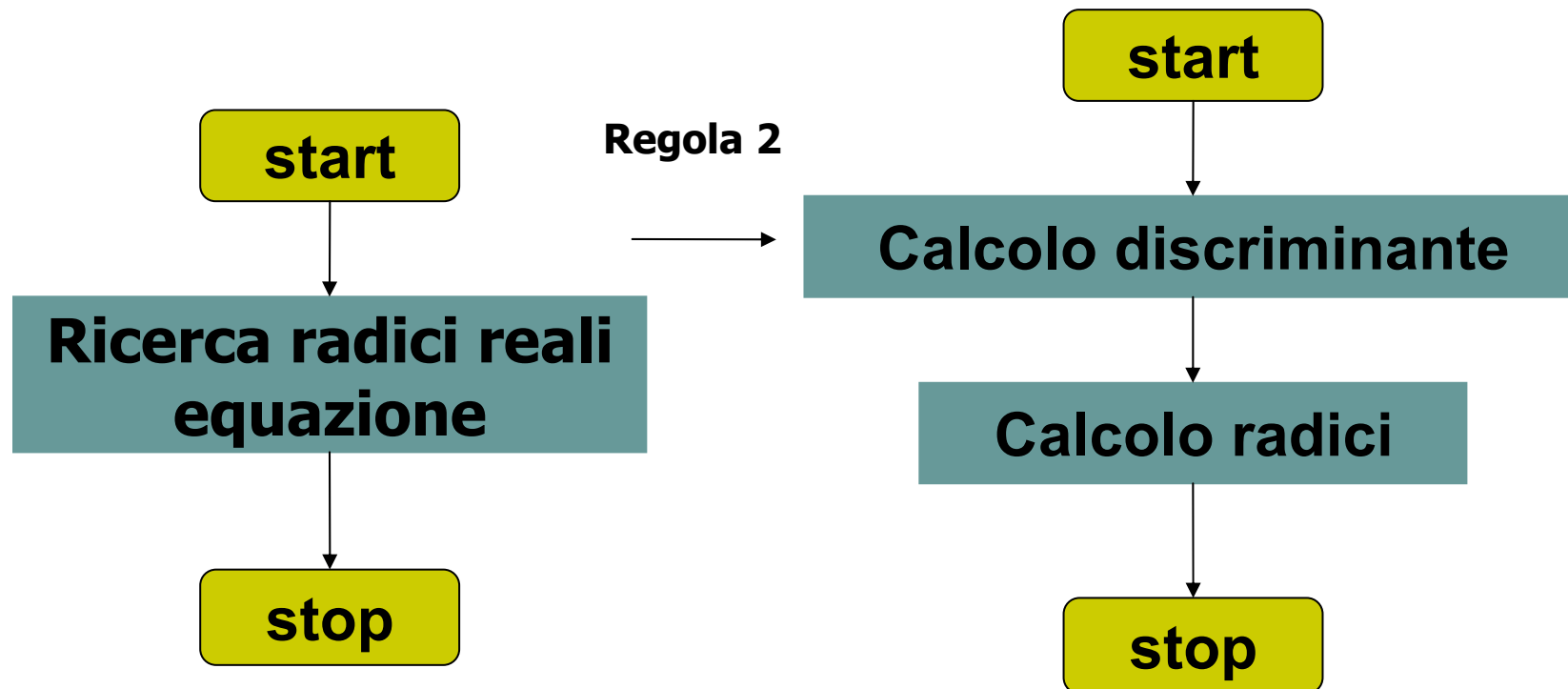
**Soluzione:** Data un'equazione di 2° grado  $ax^2 + bx + c = 0$  la determinazione delle radici reali passa attraverso il calcolo del discriminante  $\Delta = b^2 - 4ac$ . Se  $\Delta \geq 0$  le radici reali sono  $x_1 = (-b + \sqrt{\Delta}) / 2a$  e  $x_2 = (-b - \sqrt{\Delta}) / 2a$ .

### Applicazione Della Regola 1: Diagramma Di Flusso Elementare



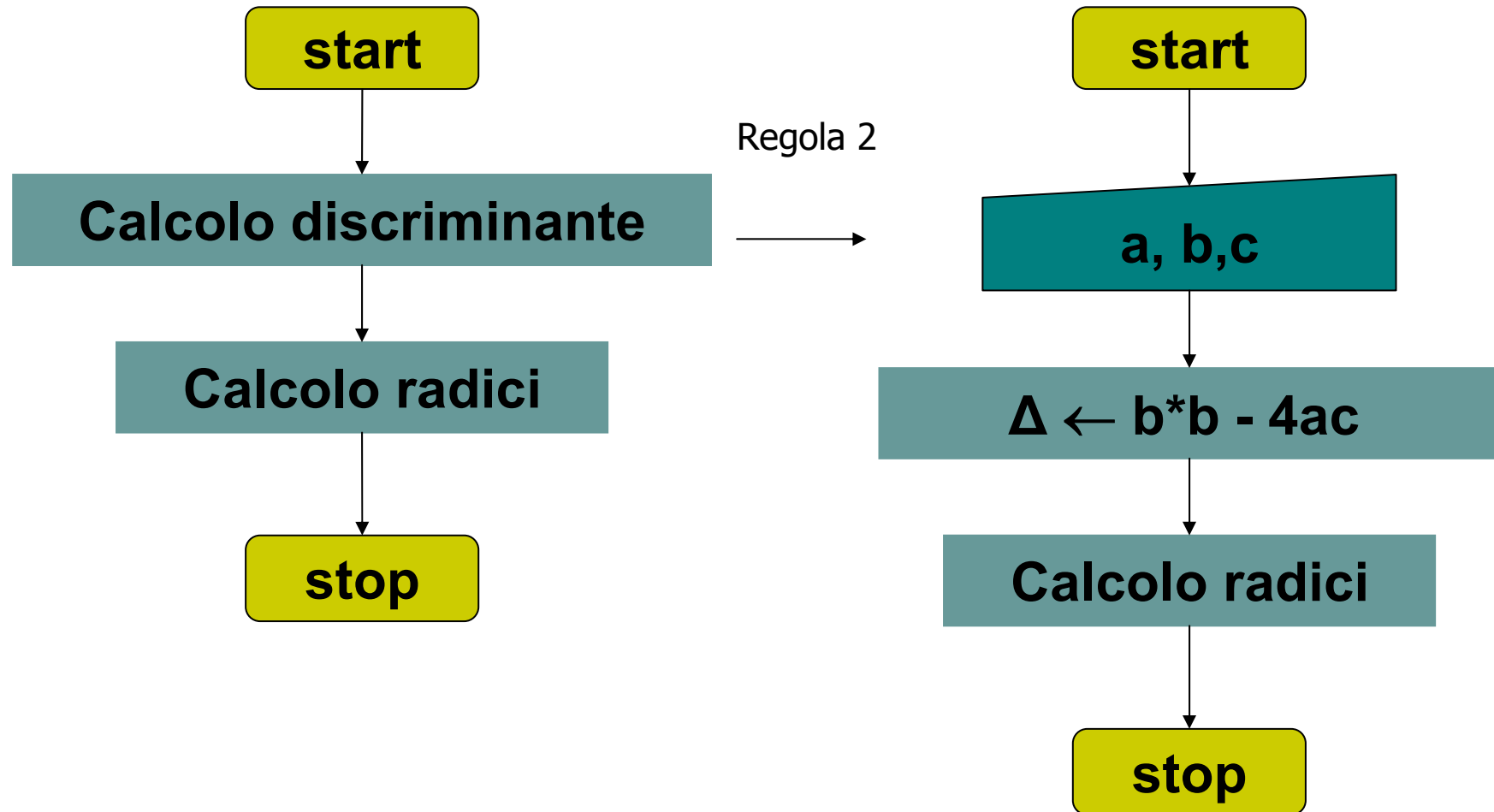
## Regole per la creazione di programmi strutturati: Un esempio (2/5)

### Prima Applicazione Della Regola 2



## Regole per la creazione di programmi strutturati: Un esempio (3/5)

### Seconda Applicazione Della Regola 2

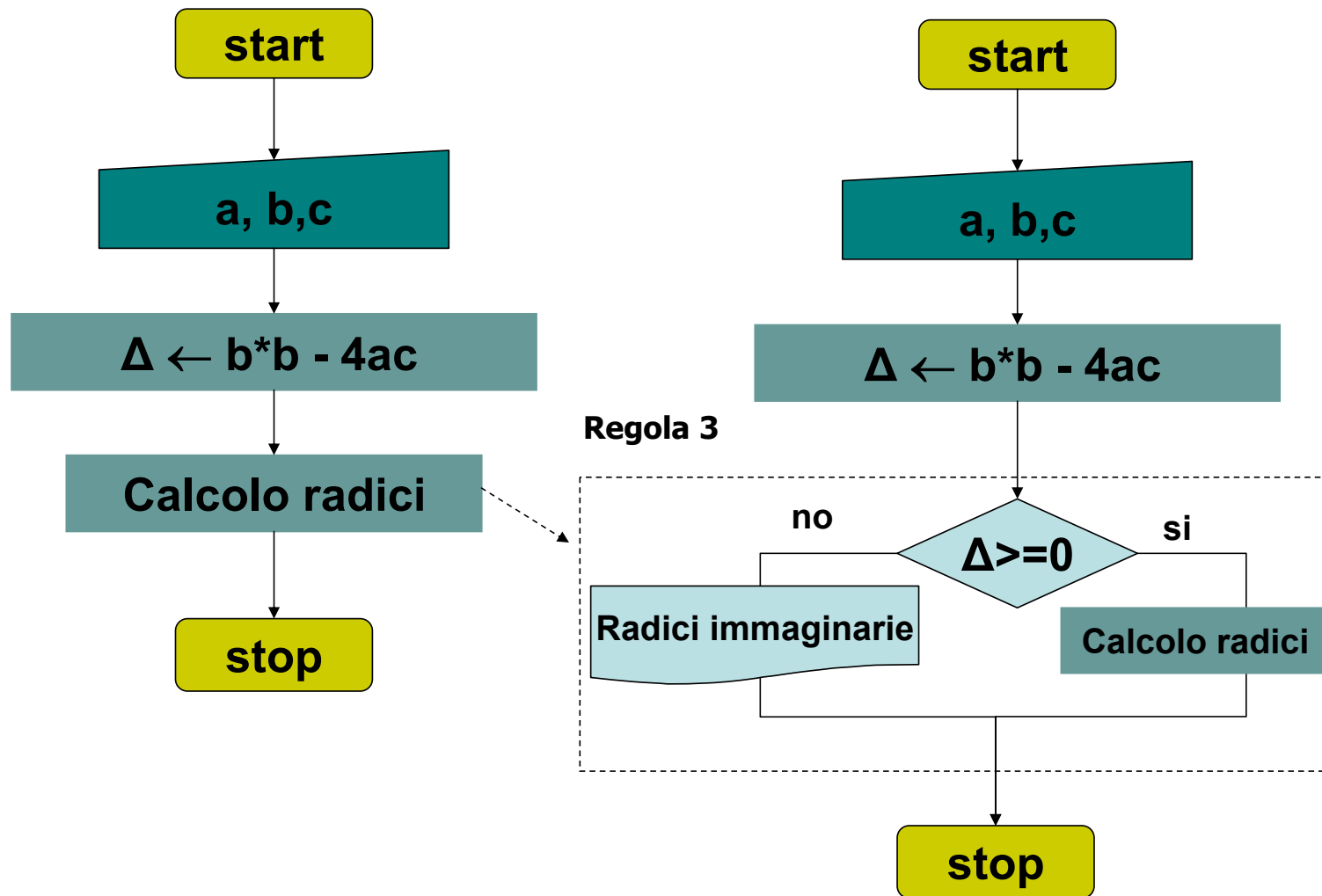




# PROGRAMMAZIONE STRUTTURATA

## Regole per la creazione di programmi strutturati: Un esempio (4/5)

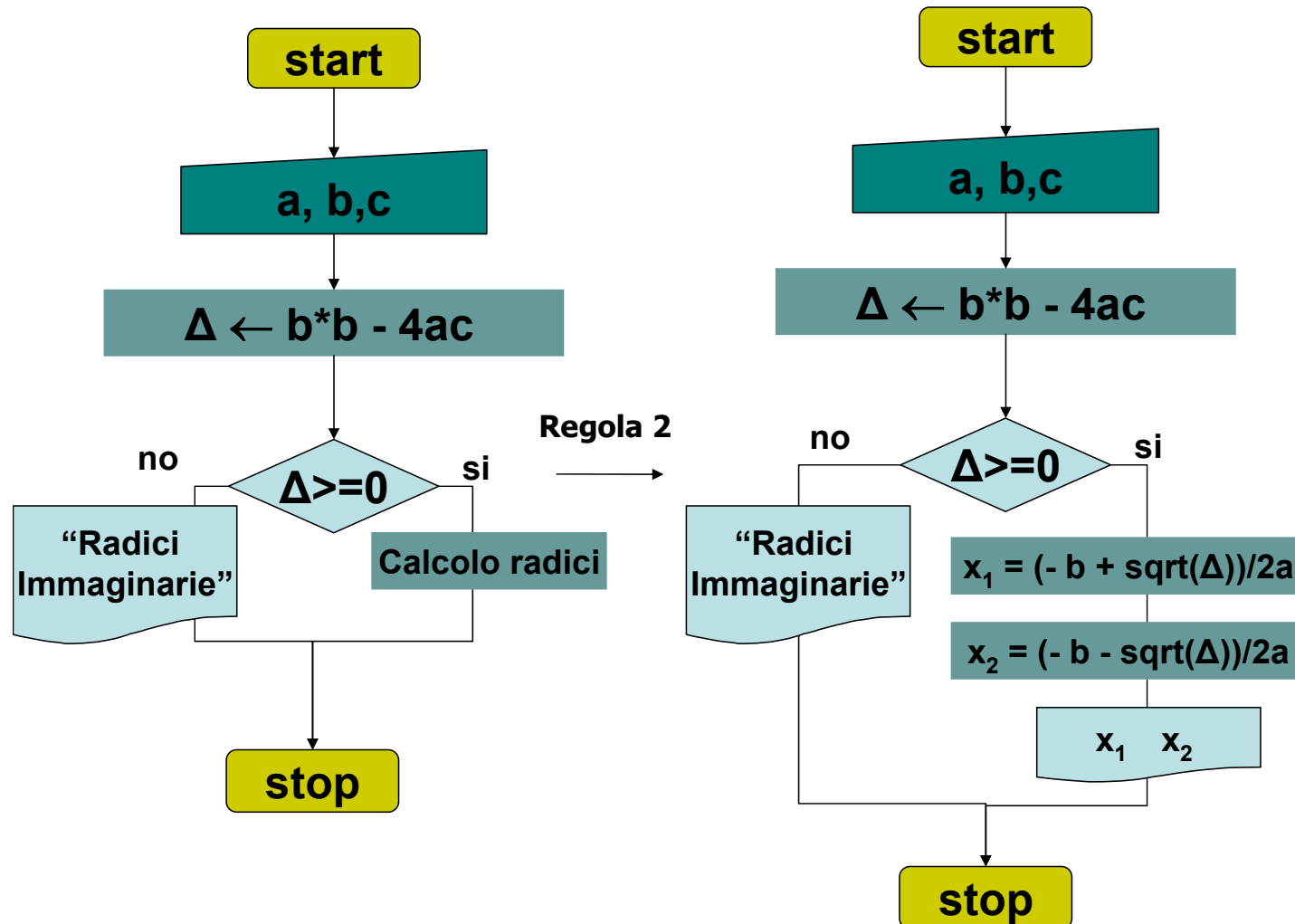
### Applicazione Della Regola 3



# PROGRAMMAZIONE STRUTTURATA

## Regole per la creazione di programmi strutturati: Un esempio (5/5)

### Terza Applicazione Della Regola 2: Diagramma Finale



## Secondo esempio (1/2)

Una classe di 10 studenti sostiene un esame. Le votazioni possono assumere valori da 1 a 100. Descrivere un algoritmo per il calcolo della media dei voti della classe nell'esame considerato

### L'algoritmo dovrà:

- ✓ prendere in input ciascun voto
- ✓ eseguire il calcolo della media
- ✓ visualizzare il risultato

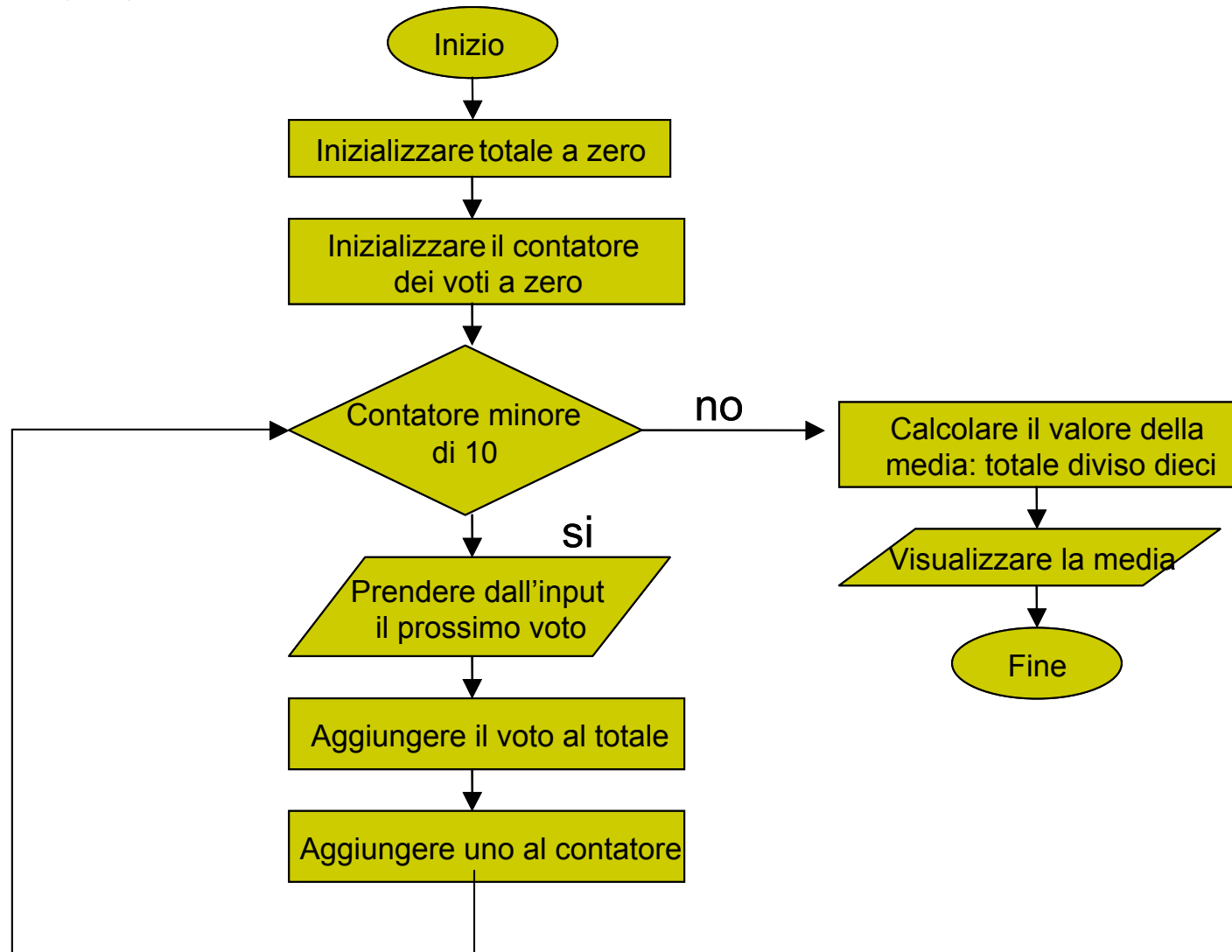
### Strategia di soluzione

- ✓ È noto il numero di studenti, quindi il numero di voti da prendere in input:
- ✓ È possibile usare una iterazione controllata da un contatore per prendere in input i voti

### Variabili

- ✓ È necessario usare:
  - ☞ Una variabile **contatore** che specifichi il numero di volte che l'iterazione deve essere eseguita (è usata per contare il numero di voti immessi)
  - ☞ Una variabile **totale** per accumulare la somma della serie di valori

## Secondo esempio (2/2)



## Terzo esempio (1/3)

Descrivere un algoritmo per il calcolo della media dei voti di una classe. L'algoritmo elaborerà un numero arbitrario di votazioni ogni volta che sarà eseguito.

### L'algoritmo dovrà:

- ✓ prendere in input ciascun voto
- ✓ eseguire il calcolo della media
- ✓ visualizzare il risultato

### Strategia di soluzione

- ✓ Non è data alcuna indicazione sul numero delle votazioni.
  - ☞ In che modo il programma deve terminare l'immissione delle votazioni?
  - ☞ In che modo il programma può sapere quando calcolare e visualizzare la media dei voti?



Usare un **valore speciale fittizio**, o valore **flag** (bandiera)

Le iterazioni controllate da un valore flag sono indefinite il numero delle iterazioni non è noto prima che inizi l'esecuzione del programma

## Terzo esempio (2/3)

### Variabili

- ✓ Una variabile usata per indicare la fine dell'immissione dei dati (**flag**): L'utente immetterà un valore sentinella per indicare che l'ultima votazione è stata immessa  
Il valore di sentinella dovrà essere scelto in modo che non possa essere confuso con un valore di input accettabile: ad esempio si può usare il valore -1 poiché le valutazioni degli esami sono generalmente numeri interi non negativi.
- ✓ Una variabile che contenga il **totale** progressivo dei numeri
- ✓ Un **contatore** del numero di voti elaborati
- ✓ Una variabile per il **valore** di ogni valutazione immessa nell'input
- ✓ Una variabile per conservare la **media** calcolata

### Iterazione

- ✓ È necessario fare uso di una struttura di iterazione che prenda in input ogni votazione.
- ✓ Poiché non è noto in anticipo il numero delle votazioni (e quindi delle iterazioni) si usa una iterazione controllata da un valore sentinella.
  - ☞ L'utente immetterà una per volta tutte le votazioni legittime.
  - ☞ Terminata l'immissione delle votazioni legittime l'utente immetterà il valore sentinella
  - ☞ Per ognuna delle votazioni immesse il programma controllerà l'immissione del valore sentinella
  - ☞ Il programma terminerà il ciclo quando verrà immesso il valore sentinella

## Terzo esempio (3/3)

