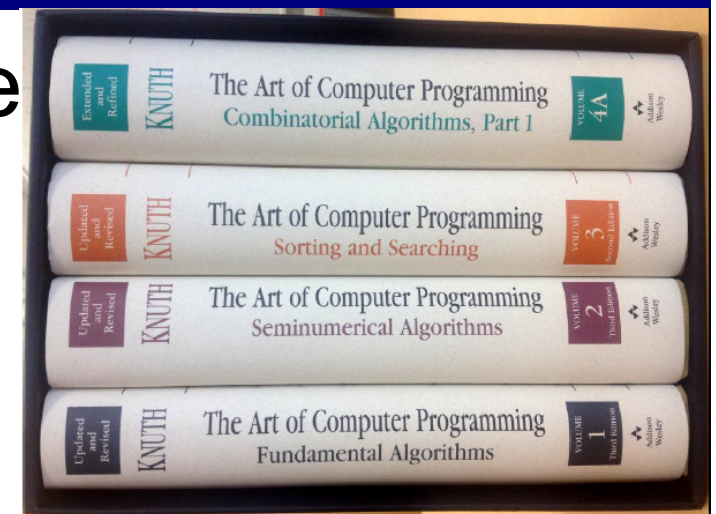


# Dal problema al programma

L'informatica e  
gli algoritmi





# Introduzione

- L'informatica può essere definita come la disciplina che si occupa dell'informazione e del suo trattamento in maniera automatica.
- Questa definizione evidenzia due elementi fondamentali:
  - Concetto di informazione
  - Mezzi per elaborarla
- Come mezzi per l'elaborazione dell'informazione, non intendiamo solo i mezzi fisici, come i computer, ma anche, e prima di tutto, i procedimenti di elaborazione, che prendono il nome di **algoritmi**.



# Introduzione (2)

- Il concetto centrale dell'informatica è appunto quello di **algoritmo**, come procedimento per la trasformazione delle informazioni.
- Lo studio degli algoritmi può essere svolto indipendentemente dagli strumenti fisici utilizzati per la manipolazione delle informazioni, i computer, e, in effetti, la sua nascita ha preceduto di molti secoli quella dei computer.
- In realtà la nozione di algoritmo non riguarda solo l'informatica e il trattamento dell'informazione, ma riguarda qualunque campo in cui si possono descrivere sequenze di operazioni finalizzate allo svolgimento di un compito. Es. di algoritmo:
  - Ricetta di cucina (esecutore: cuoco)
  - Spartito musicale (esecutore: pianista)
  - ...

# L'algoritmo di Euclide per il calcolo del MCD tra due numeri naturali



Euclide (ca. 325 a.C. – ca. 265 a.C.)



# Esempio di algoritmo

- Il primo esempio di algoritmo che presentiamo è in ambito matematico.
- Consideriamo il problema del calcolo del MCD tra due interi positivi.
- Algoritmo di Euclide:
  1. Siano  $x$  e  $y$  i due numeri
  2. Calcola il resto della divisione (intera) di  $x$  per  $y$
  3. Se il resto diverso da zero
    - Ricomincia dal passo 2 utilizzando come  $x$  il valore attuale di  $y$  e come  $y$  il valore del resto
    - Altrimenti prosegui con il passo successivo
  4. Il massimo comun divisore è uguale al valore attuale di  $y$



# Osservazioni

- Si osservi che chiunque sappia comprendere ed eseguire le operazioni che costituiscono l'alg. di Euclide, può calcolare il MCD tra due numeri, senza sapere cosa sia il MCD tra due numeri.
- L'esecutore può eseguire il procedimento senza avere la minima idea di quale sia lo scopo di tale procedimento.
- L'"intelligenza" necessaria per trovare la soluzione del problema (in questo caso per calcolare il MCD) è tutta codificata nell'algoritmo.

# Algoritmi

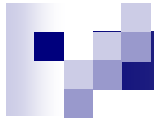
- Problema: Fare una telefonata
- Algoritmo:

## **INIZIO**

sollevare il ricevitore  
attendere il segnale di linea  
comporre il numero  
attendere la risposta  
condurre la conversazione  
deporre il ricevitore

## **FINE**





# Algoritmi

## ■ Osservazioni:

- **Struttura di sequenza** (struttura rigida, in quanto la sequenza di esecuzione è fissata a priori)
- Dati e risultati
- Esecutore





# Algoritmi

- Problema: Che cosa succede se non c'è il segnale di linea?
- Algoritmo:

**INIZIO**

sollevare il ricevitore

SE (c'è il segnale di linea)

ALLORA

**INIZIO**

comporre il numero

attendere la risposta

condurre la conversazione

deporre il ricevitore

**FINE**

ALTRIMENTI

**INIZIO**

deporre il ricevitore

avvisare la Telecom

**FINE**

FINE\_SE

**FINE**



# Algoritmi

## ■ Osservazioni:

- ☐ **Struttura di selezione** (binaria)
- ☐ Predicato o condizione
- ☐ Parole chiave o riservate
- ☐ Blocco (INIZIO ... FINE)
- ☐ Terminatore (FINE\_SE)
- ☐ Indentazione
- ☐ Clausola “ALTRIMENTI” opzionale



# Algoritmi

- **Sequenza di esecuzione:** insieme di azioni eseguite in una esecuzione dell'algoritmo.
- **Obiettivo:** avere un unico algoritmo che permetta di descrivere contemporaneamente più sequenze di esecuzione.
- **Problema:** Che cosa succede se il numero è occupato?



# Algoritmi

```
INIZIO
  sollevare il ricevitore
  SE (c'è il segnale di linea)
    ALLORA
      INIZIO
        comporre il numero
        SE (c'è il segnale di libero)
          ALLORA
            INIZIO
              attendere la risposta
              condurre la conversazione
              deporre il ricevitore
            FINE
          ALTRIMENTI
            deporre il ricevitore
        FINE_SE
      FINE
    ALTRIMENTI
      INIZIO
        deporre il ricevitore
        avvisare la Telecom
      FINE
    FINE_SE
  FINE
```



# Algoritmi

- Osservazioni:
  - Struttura innestata o nidificata
  - Sequenze di esecuzione
  
- Problema: Che cosa succede se si sbaglia numero?



# Algoritmi

**INIZIO**

sollevare il ricevitore  
SE (c'è il segnale di linea)

ALLORA

**INIZIO**

comporre il numero  
SE (c'è il segnale di libero)

ALLORA

**INIZIO**

attendere la risposta  
chiedere con chi si parla  
SE (è il numero giusto)  
ALLORA  
condurre la conversazione

ALTRIMENTI

scusarsi

FINE\_SE

**FINE**

FINE\_SE  
deporre il ricevitore

**FINE**

ALTRIMENTI

**INIZIO**

deporre il ricevitore  
avvisare la Telecom

**FINE**

FINE\_SE

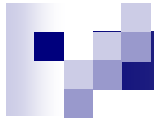
**FINE**



# Algoritmi

## ■ Osservazioni:

- ☐ Struttura nidificata complessa
- ☐ Sequenze di esecuzione
- ☐ “ALTRIMENTI” opzionale



# Algoritmi

- Dettagliamo un'azione particolare della telefonata, ad es.:
    - “comporre il numero”
- in azioni più semplici.





# Algoritmi

## ■ 1<sup>a</sup> forma

**RIPETI**

comporre una cifra

**FINO\_A\_CHE** (sono finite le cifre)

## ■ 2<sup>a</sup> forma

**FINTANTOCHE'** (ci sono cifre da comporre) **ESEGUI**

comporre una cifra



# Algoritmi

## ■ Osservazioni:

### □ **Struttura di iterazione**

- 1^a forma: con controllo in uscita (in coda)
- 2^a forma: con controllo in ingresso (in testa)

### □ Predicato o condizione



# Algoritmi – Struttura di iterazione

## ■ 1<sup>a</sup> forma:

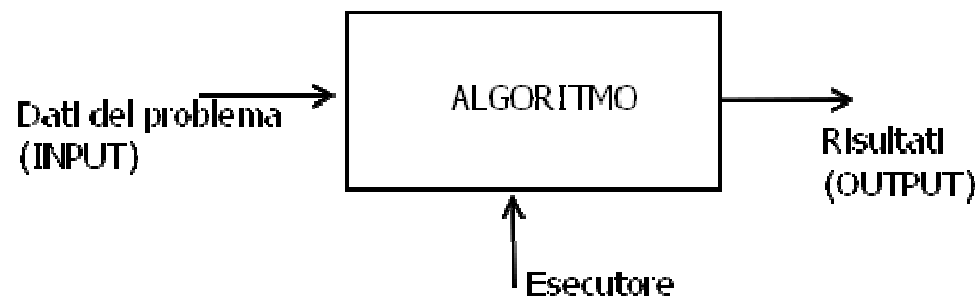
- ☐ Controllo in coda
- ☐ Si esce (dal ciclo) per condizione VERA
- ☐ Il blocco azioni viene eseguito almeno una volta

## ■ 2<sup>a</sup> forma:

- ☐ Controllo in testa
- ☐ Si esce (dal ciclo) per condizione FALSA
- ☐ Il blocco azioni può non essere eseguito (se la condizione è subito falsa)

# Algoritmi

- Dato un problema per risolverlo abbiamo indicato un insieme di azioni e regole da eseguire per ottenere il risultato.





# Definizione di algoritmo

- Una sequenza di azioni eseguendo le quali si perviene alla soluzione di un problema (classe di problemi).
- Azione = una frase che evoca l'esecuzione di un'operazione e che l'esecutore sa interpretare senza ulteriori spiegazioni.
- Azione elementare = azione direttamente eseguibile dall'esecutore

# Etimologia della parola algoritmo

- La parola *algoritmo* deriva dal nome di un matematico arabo del IX secolo d.C. di nome Al-Khuwarizmi.

Un francobollo commemorativo stampato il 6 settembre 1983 in Unione Sovietica per il 1200° anniversario (approssimativo) di nascita del grande matematico.





# Algoritmo

- Un algoritmo deve essere scritto in un linguaggio
  - non ambiguo
  - comprensibile all'esecutore.



# Proprietà di un algoritmo

- Finito: numero finito di azioni elementari ed inoltre l'elaborazione dell'alg. da parte dell'esecutore deve avere termine in un tempo finito.
- Univocità: l'alg. deve essere interpretato senza ambiguità e con precisione dall'esecutore.
- Generale: deve essere valido per tutti i problemi di una stessa classe.
- Completo: deve considerare tutti i casi possibili che si possono verificare durante l'esecuzione e, per ogni caso, indicare la soluzione da seguire.





# Proprietà di un algoritmo (2)

- Deve fornire almeno un risultato (un dato in uscita).
- Deterministico: partendo dagli stessi dati iniziali, l'esecuzione dell'alg. deve fornire sempre gli stessi risultati finali.
- Eseguibile: in un alg. non possono essere presenti istruzioni che non siano effettivamente eseguibili da un esecutore reale. Ad es. l'istruzione “produci l'espansione decimale di  $\pi$ ” non è un'istruzione effettivamente eseguibile, dal momento che la sua espansione è infinita.



# Algoritmo

- Componenti di un algoritmo:
  - Azioni
  - Strutture di controllo (del flusso)



# Strutture di controllo

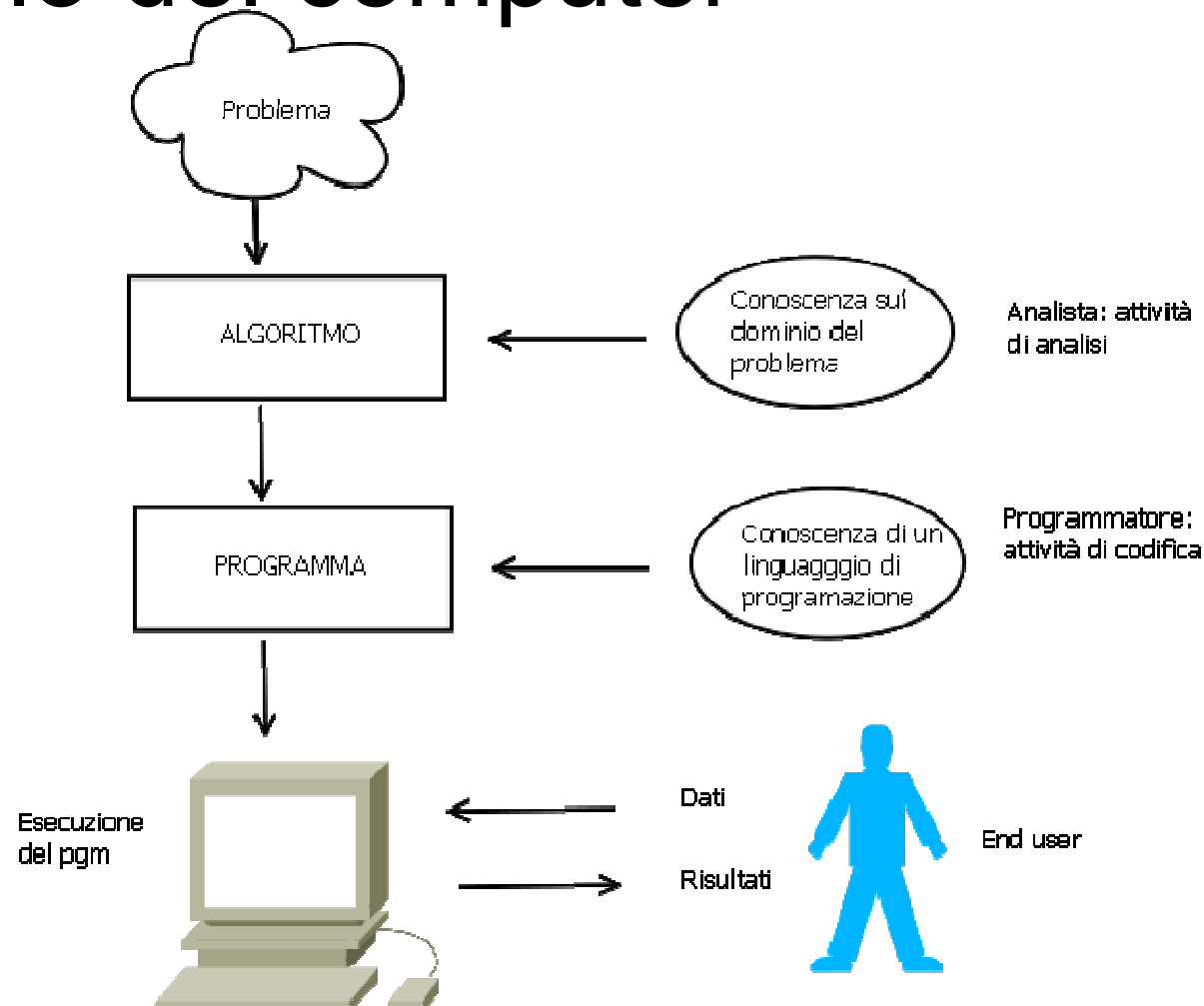
- Le strutture

- ☐ Sequenza
- ☐ Selezione
- ☐ Iterazione

sono degli strumenti linguistici che vengono affiancati alle azioni, per selezionare differenti vie dell'algoritmo (sequenze di esecuzione) controllandone l'esecuzione.

- Le strutture di controllo sono pensate come schemi di composizione e quindi si adattano ad essere mescolate tra di loro.

# Dal problema al programma: il ruolo del computer

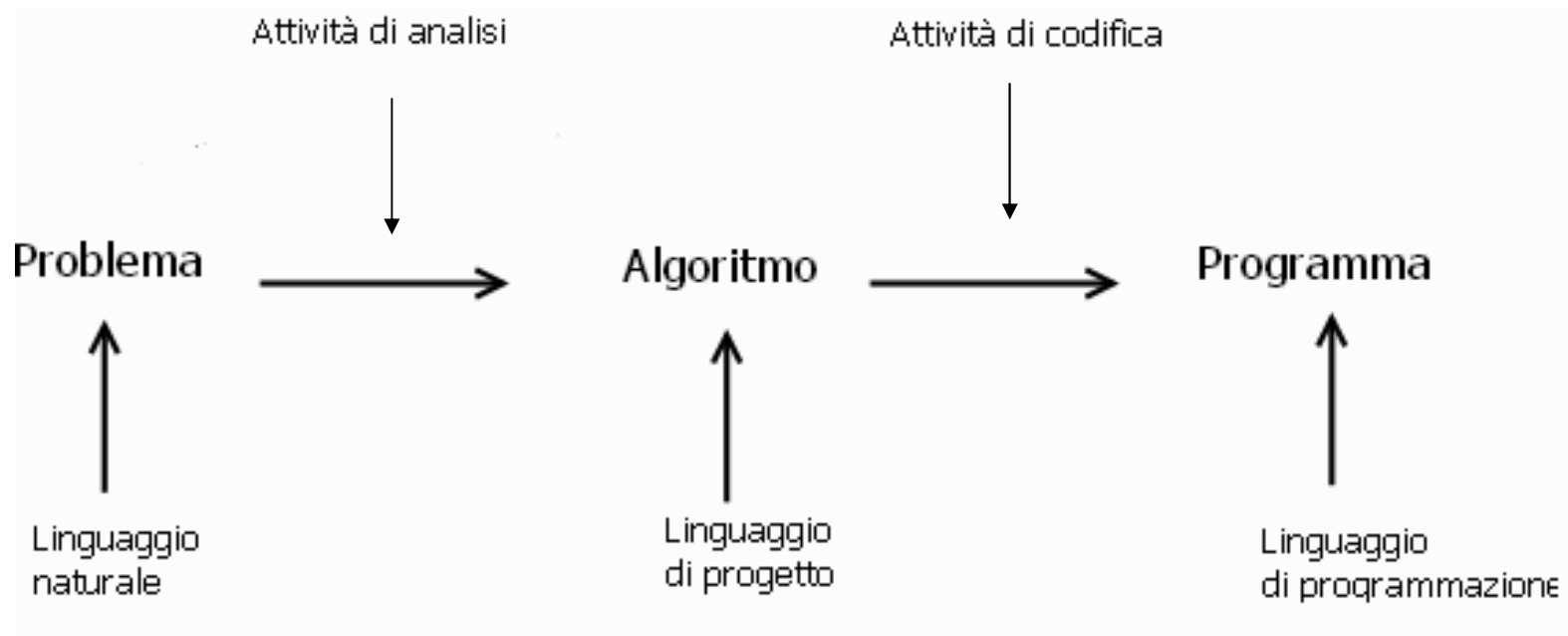




# Alcune definizioni

- Il computer è una macchina concepita per l'elaborazione automatica dei dati per mezzo di programmi scritti dall'uomo.
- Programma = è una sequenza di istruzioni eseguibili dall'elaboratore.
- Istruzione = è una frase (stringa) che evoca l'esecuzione di un'operazione e che l'esecutore (computer) sa interpretare senza ulteriori spiegazioni.

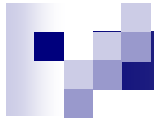
# Dal problema al programma





# Linguaggi utilizzati per descrivere gli algoritmi

- Lingua italiana
- Flow-chart (diagrammi di flusso, diagrammi a blocchi)
- Pseudolinguaggio (o linguaggio di progetto)



# I flow-chart

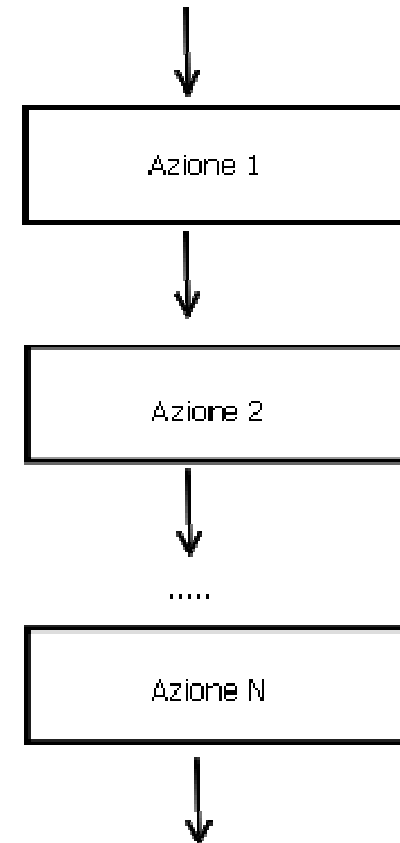
## ■ Azioni

Nome azione



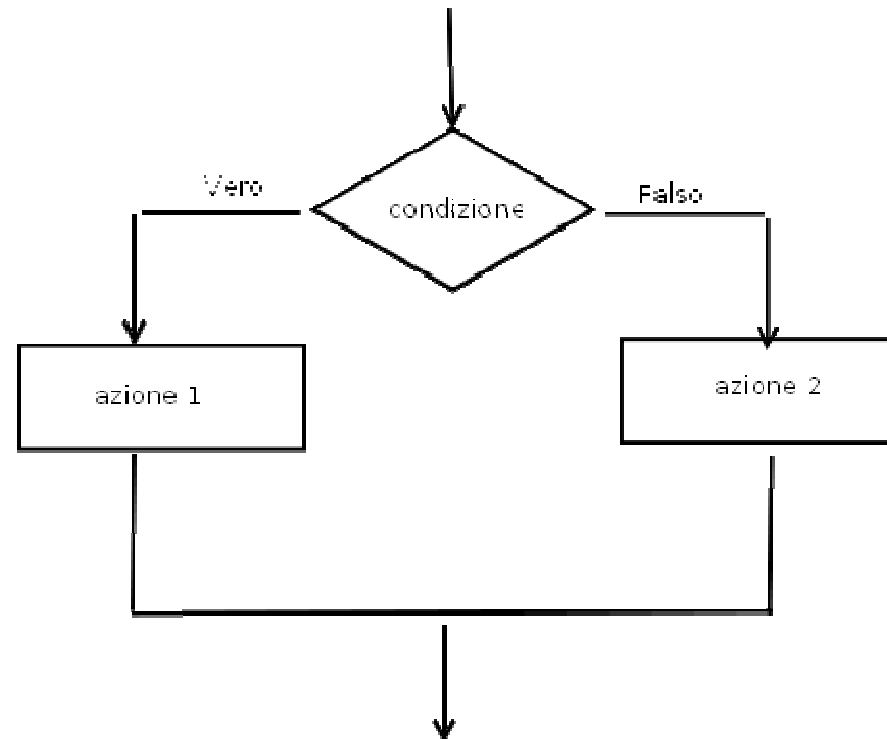
# I flow-chart (2)

## ■ Struttura di sequenza



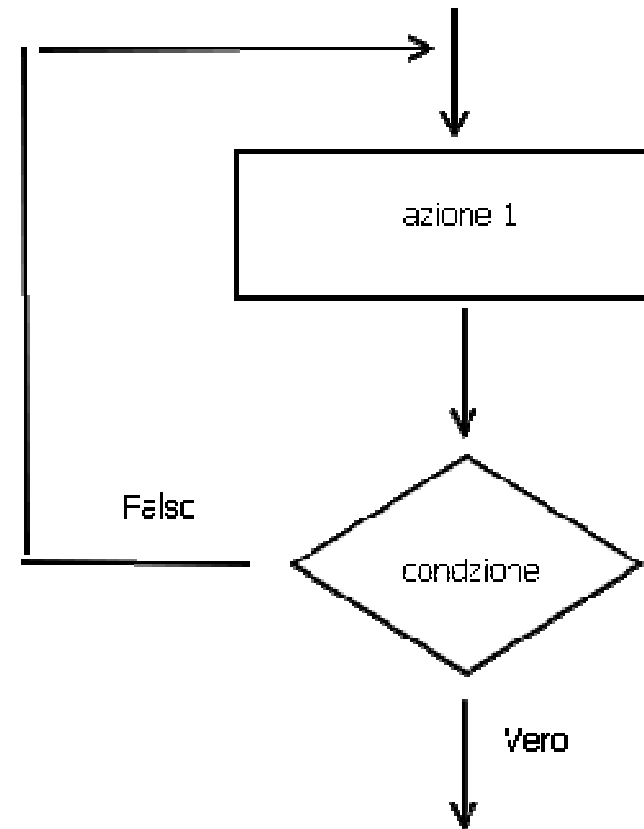
# I flow-chart (3)

## ■ Struttura di selezione



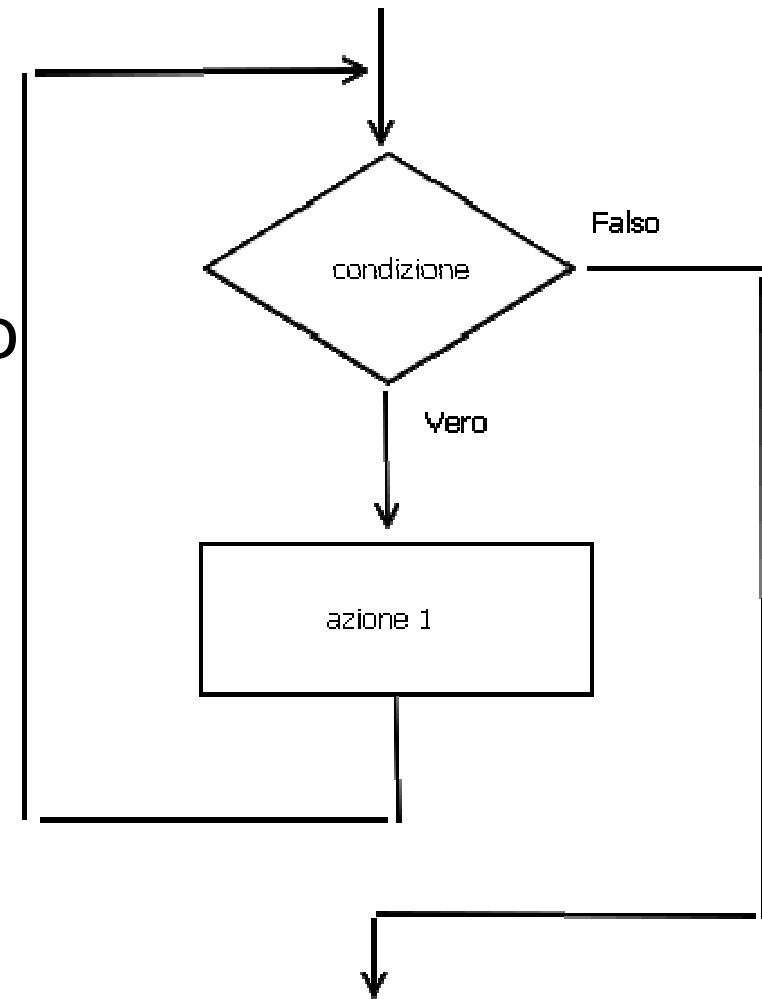
# I flow-chart (4)

- Struttura di iterazione
  - 1^forma: ciclo ad uscita controllata



# I flow-chart (5)

- Struttura di iterazione
  - 2^a forma: ciclo ad ingresso controllato





# Pseudolinguaggio

- Linguaggio per progettare algoritmi
  - È in italiano: unisce i pregi del linguaggio naturale (facilità di lettura) a quelli dei linguaggi di programmazione (precisione e sinteticità)
- Linguaggio di comunicazione uomo-uomo
- Linguaggio di documentazione
- Orientato alla codifica in un linguaggio di alto livello



# Pseudolinguaggio (2)

- Azione      azione i-esima

- Struttura di sequenza

**INIZIO**

azione 1

azione 2

...

azione N

**FINE**



# Pseudolinguaggio (3)

- Struttura di selezione

**SE** (predicato)

**ALLORA**

**INIZIO**

Azioni per predicato vero

**FINE**

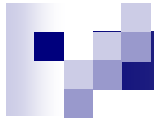
**[ ALTRIMENTI**

**INIZIO**

Azioni per predicato falso

**FINE ]**

**FINE\_SE**



# Pseudolinguaggio (4)

- Struttura di iterazione: ciclo con controllo in coda

**RIPETI**

azioni da ripetere

**FINO\_A\_CHE** (predicato)





# Pseudolinguaggio (5)

- Struttura di iterazione: ciclo con controllo in testa

**FINTANTOCHE'** (predicato) **ESEGUI**

**INIZIO**

azioni da ripetere

**FINE**



# Pseudolinguaggio (6)

- Lo **pseudolinguaggio** è un linguaggio formale, ossia un linguaggio che utilizza simboli ai quali corrisponde uno ed un solo significato in qualsiasi contesto.
- La descrizione formale dell'algoritmo in pseudolinguaggio viene detta **pseudocodice**.
- L'attività di scrittura dello pseudocodice prende il nome di **pseudocodifica**.



# Teorema delle strutture (o di Böhm-Jacopini 1966)

C.Böhm (1923 - 2017, matematico e informatico italiano)

Jacopini ( , matematico e informatico italiano)

- Un qualunque algoritmo può essere realizzato utilizzando le sole tre strutture fondamentali:
  - ☐ **SEQUENZA**
  - ☐ **SELEZIONE**
  - ☐ **ITERAZIONE**



# Azioni

- Eseguiamo il seguente algoritmo:

INIZIO

leggi due numeri

moltiplica i due numeri letti fra loro

SE (il prodotto è  $> 20$ )

ALLORA

aggiungi al prodotto 10

ALTRIMENTI

sottrai al prodotto 5

FINE\_SE

scrivi il risultato

FINE



## Azioni (2)

- Nel semplice algoritmo possiamo notare tre tipi di azioni:
  - Lettura
  - Calcolo/confronto
  - Scrittura
- Un'elaborazione di dati comprende tre azioni principali:  
Lettura → Calcolo/confronto → Scrittura



# Costruzione top-down di un algoritmo


(o sviluppo di un algoritmo per raffinamenti successivi)

1. Si procede specificando l'algoritmo come costituito da sequenze di macro-operazioni.
  2. Ogni macro-operazione viene poi scomposta in una o più sottosequenze di operazioni più semplici.
  3. Si procede ad un ulteriore passo di scomposizione e così via fino a che l'alg. risulta costituito da sole operazioni elementari.
- Operazioni elementari = operazioni direttamente comprensibili dall'esecutore.



# Esempio di costruzione top-down di un algoritmo

- Problema: Calcolare le soluzioni di un'equazione di 2° della forma  $ax^2+bx+c=0$ .
- Soluzione: applicazione di una semplice formula (vedi libro di algebra).
- Ora proviamo a descrivere una sequenza di azioni (algoritmo) che portino alla ricerca delle radici di un'equazione di 2°.



# Esempio di costruzione top-down di un algoritmo (2)

## ■ 1° livello di scomposizione

INIZIO

leggi i valori  $a, b, c$

calcola il discriminante

→ operazione non elementare

SE (discriminante negativo)

ALLORA

scrivi “nessuna soluzione”

ALTRIMENTI

calcola e stampa le soluzioni → operazione non elementare

FINE\_SE

FINE



# Esempio di costruzione top-down di un algoritmo (3)

## ■ 2° livello di scomposizione

|

INIZIO

leggi a,b,c

calcola  $b^2-4ac$

SE(discriminante < 0)

ALLORA

scrivi "nessuna soluzione in R"

ALTRIMENTI

SE (discriminante = 0)

ALLORA

INIZIO

scrivi "soluzioni coincidenti"

calcola  $(-b/(2a))$

scrivi il risultato

FINE

ALTRIMENTI

INIZIO

scrivi "due soluzioni reali e distinte"

calcola  $((-b+\sqrt{b^2-4ac})/(2a))$

calcola  $((-b-\sqrt{b^2-4ac})/(2a))$

scrivi i risultati

FINE

FINE\_SE

FINE\_SE

FINE



# Osservazioni

- Osservando l'algoritmo precedente notiamo i seguenti tipi di azioni/istruzioni:
  - ☐ Istruzione di lettura
  - ☐ Istruzioni di calcolo/confronto
  - ☐ Istruzioni di scrittura
  - ☐ Istruzioni di inizio e fine (indicano l'inizio e la fine di un blocco di istruzioni)
  - ☐ Istruzioni di controllo (sequenza, selezione e iterazione)
- Tali classi di istruzioni sono presenti in tutti i linguaggi per descrivere gli algoritmi (e anche nei linguaggi di programmazione).



# Istruzione di assegnamento: le variabili

- Esaminiamo nel dettaglio le seguenti istruzioni

leggi a,b,c

...

calcola  $(-b/(2a))$

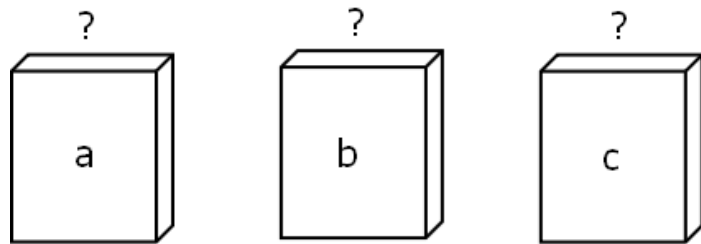
- Il concetto di variabile



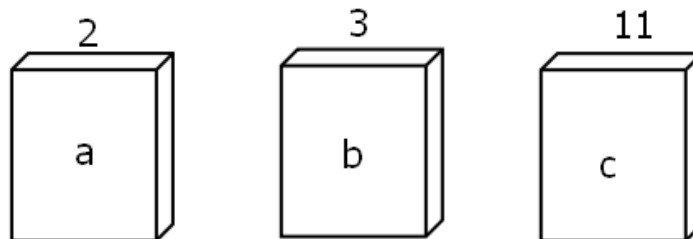
# Il concetto di variabile

- Pensiamo alla MEMORIA del computer come formata da scatole.
- Possiamo pensare una VARIABILE come una “scatola” identificata da un nome e contenente un valore che può essere modificato durante l'esecuzione dell'algoritmo.
- Una variabile è un nome associato ad una locazione di memoria

# Il concetto di variabile



leggi *a*, *b*, *c*



- *a*, *b*, *c* sono variabili
- All'inizio le variabili contengono un valore indefinito (indicato con ?)



# Istruzione di assegnamento

- Le istruzioni di calcolo presuppongono istruzioni di assegnamento

Esempio: calcola  $(-b/(2a))$

- Ma quando è stato eseguito il calcolo, il risultato dove lo metto?
- In una variabile
- Più precisamente scriveremo:  
 $x \leftarrow -b/(2a)$



# Concetti di sintassi e semantica di un'istruzione

- **Sintassi** = rappresenta l'insieme di regole che consentono di costruire correttamente le frasi del linguaggio
- **Semantica** = specifica il significato delle frasi



# Istruzione di assegnamento

## ■ Sintassi:

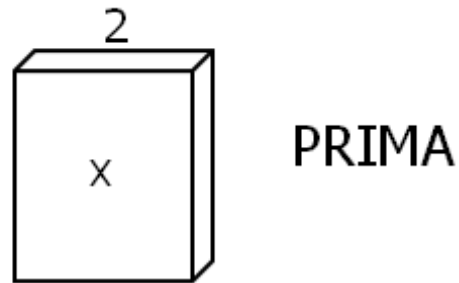
□ `<variabile> ← <espressione>`

## ■ Semantica:

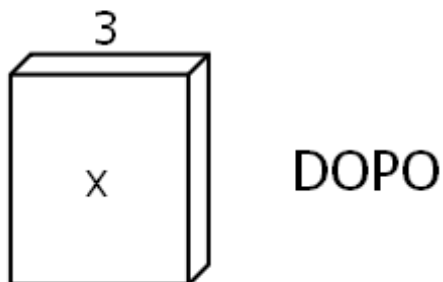
□ Calcola il valore dell'espressione posta a destra dell'operatore `←` e assegna (nel senso di scrive) il risultato nella variabile posta alla sinistra dell'operatore.



# Esempio

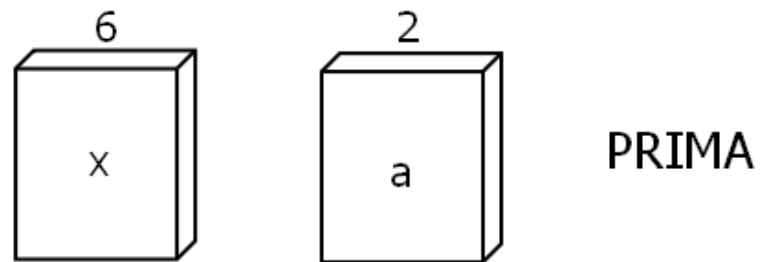


$x \leftarrow x + 1$

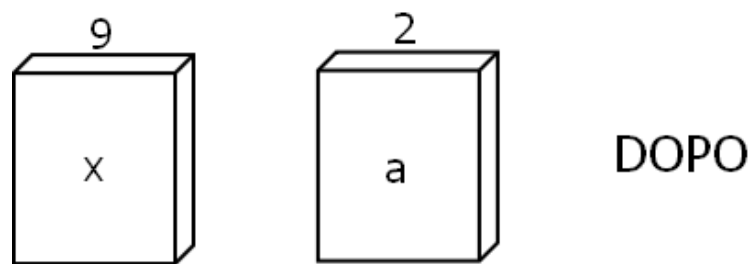


- Quando assegno un valore (o più in generale il risultato di un'espressione) ad una variabile, il contenuto precedente viene perso.
- Si dice:
  - La scrittura in memoria è distruttiva.
  - La lettura (in memoria) invece non distrugge il valore della variabile.

## Esempio 2



$x \leftarrow a + x + 1$

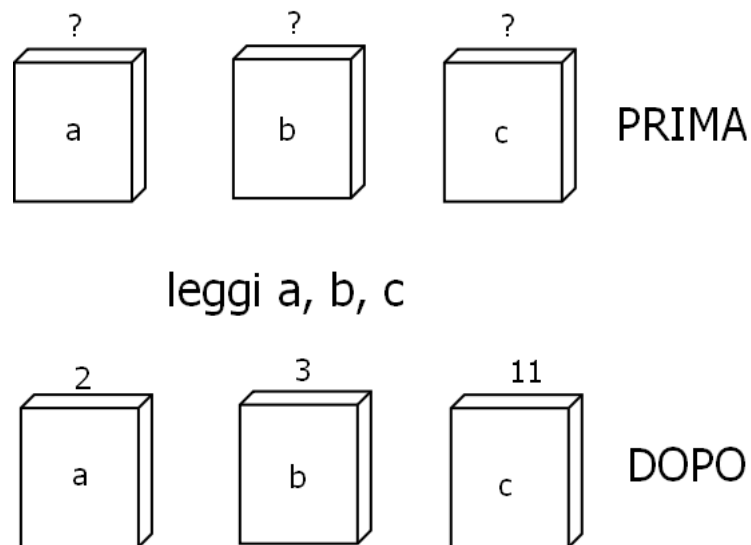




# Istruzione di calcolo

- Possiamo quindi dire che un'istruzione di calcolo prevede in generale:
  - Una o più letture di dati in memoria
  - Delle operazioni di calcolo
  - La scrittura del risultato in memoria

# Istruzione di lettura (dall'esterno)



- Corrisponde ad una scrittura in memoria.
- Le variabili vengono modificate.
- Se una variabile non l'ho mai utilizzata il suo valore è indefinito.
- Concetto di inizializzazione delle variabili.



# Algoritmo

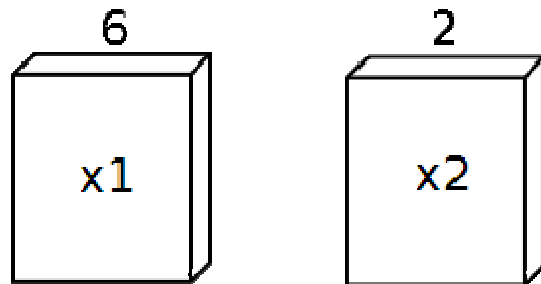
```
INIZIO
  leggi a,b,c
  delta  $\leftarrow b^2-4ac$ 
  SE(delta < 0)
    ALLORA
      scrivi "nessuna soluzione in R"
    ALTRIMENTI
      SE (delta = 0)
        ALLORA
          INIZIO
            scrivi "soluzioni coincidenti"
             $x \leftarrow (-b/(2a))$ 
            scrivi x
          FINE
        ALTRIMENTI
          INIZIO
            scrivi "due soluzioni reali e distinte"
             $x1 \leftarrow ((-b+\text{sqrt}(b^2-4ac))/(2a))$ 
             $x2 \leftarrow ((-b-\text{sqrt}(b^2-4ac))/(2a))$ 
            scrivi x1, x2
          FINE
        FINE_SE
      FINE_SE
    FINE_SE
  FINE
```



# Sintassi azione “leggi”

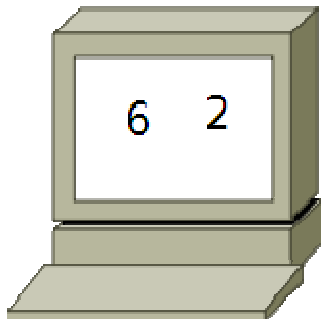
- leggi <variabile> {,<variabile>}

# Istruzione di scrittura (verso l'esterno)



PRIMA

scrivi x1,x2



DOPO

- Corrisponde ad una lettura in memoria.
- Le variabili non vengono modificate.



# Istruzione di scrittura (2)

scrivi "nessuna soluzione"





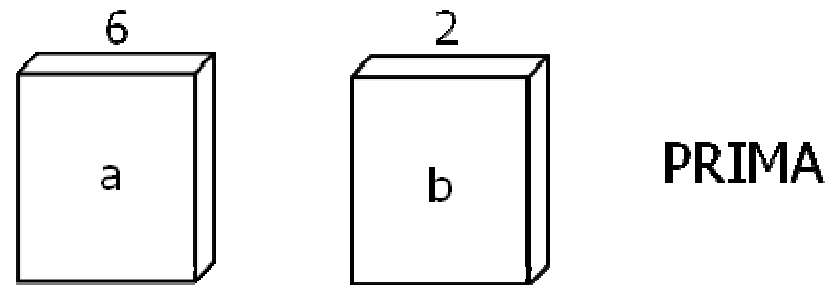


# Sintassi azione “scrivi”

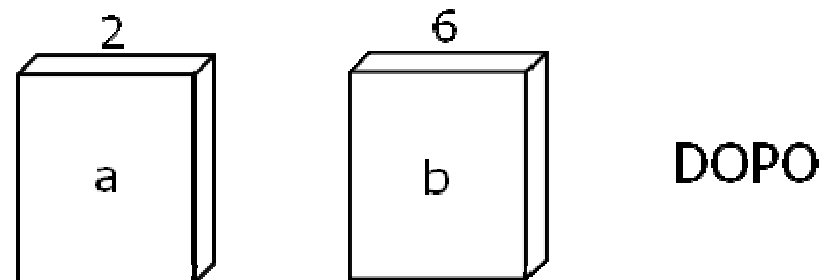
- scrivi [<messaggio>] [,] {<variabile>[,]}

# Problema

- Date due variabili a e b voglio scambiare i loro valori.



algoritmo





# Algoritmo di scambio

■ 1° tentativo

$a \leftarrow b$

NO! Perdo il valore di a.

■ 2° tentativo

$b \leftarrow a$

NO! Perdo il valore di b.



# Algoritmo di scambio (2)

Idea!

- Utilizziamo un'altra variabile che chiamiamo temp.
- Algoritmo:  
     $\text{temp} \leftarrow a$   
     $a \leftarrow b$   
     $b \leftarrow \text{temp}$
- La variabile temp viene chiamata variabile temporanea o di comodo.



# Correttezza ed efficienza

Vogliamo progettare algoritmi che:

- Producano correttamente il risultato desiderato
- Siano efficienti in termini di tempo di esecuzione ed occupazione di memoria