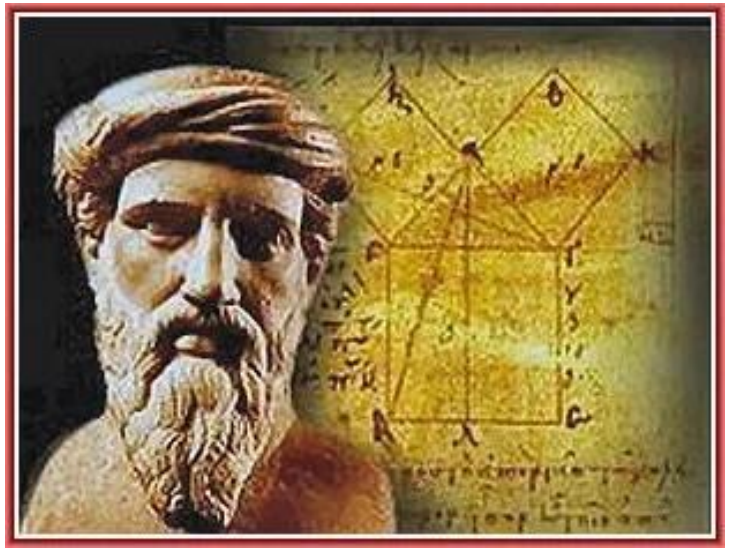
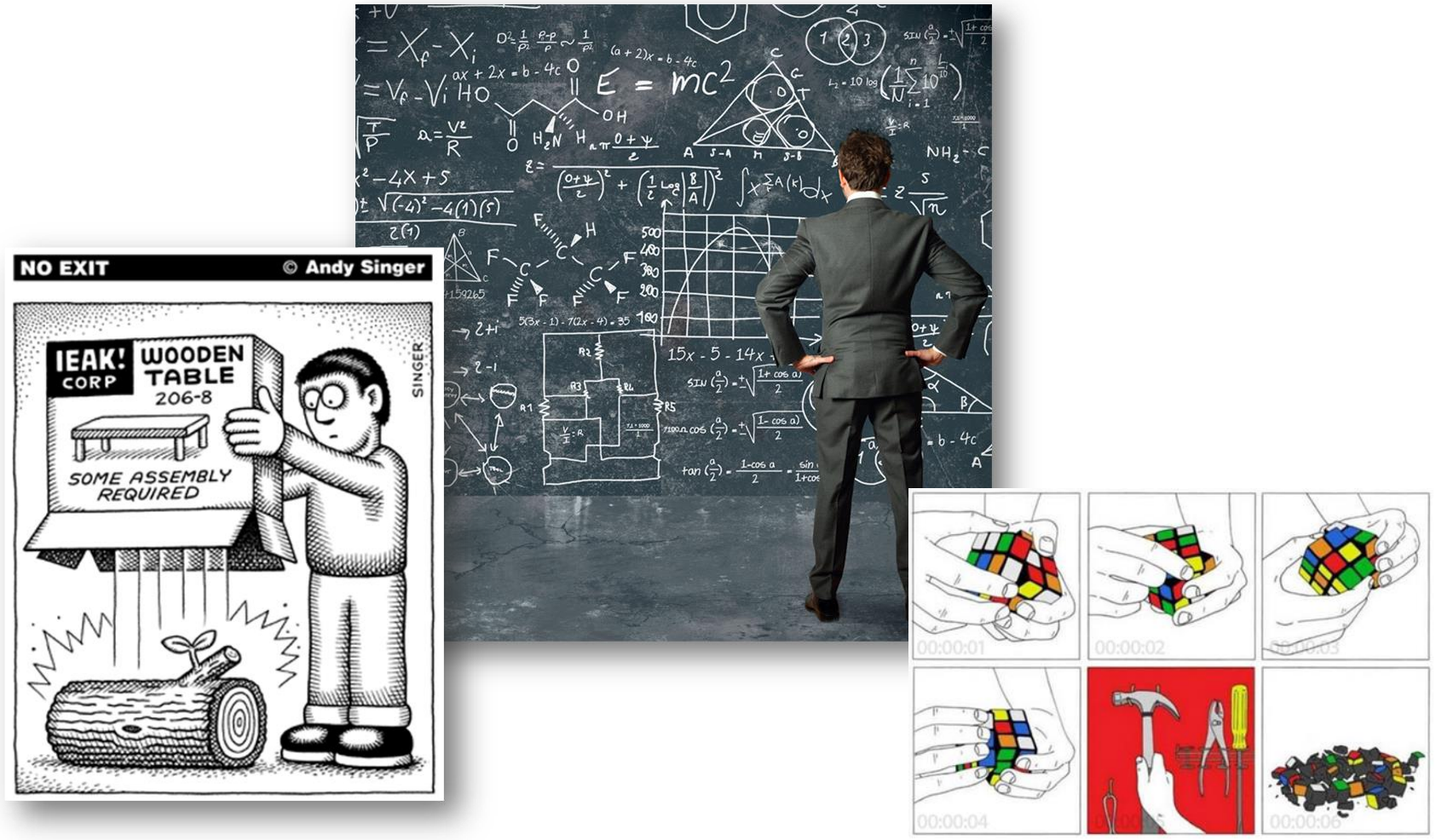
algoritmi

dall'analisi del problema
alla definizione dell'algoritmo

- risolvere un problema:
  - passare da uno stato iniziale
  - a uno stato finale
  - attraverso stati intermedi

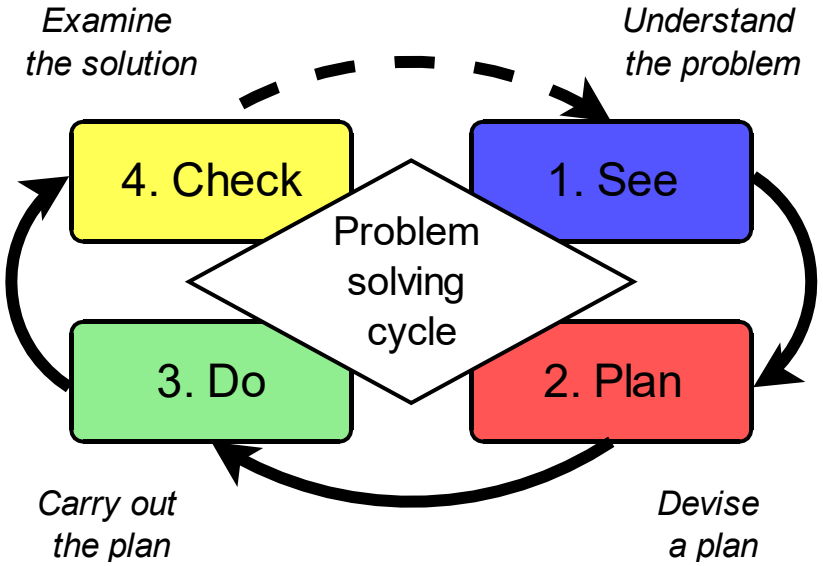


non solo problemi matematici



(1) See

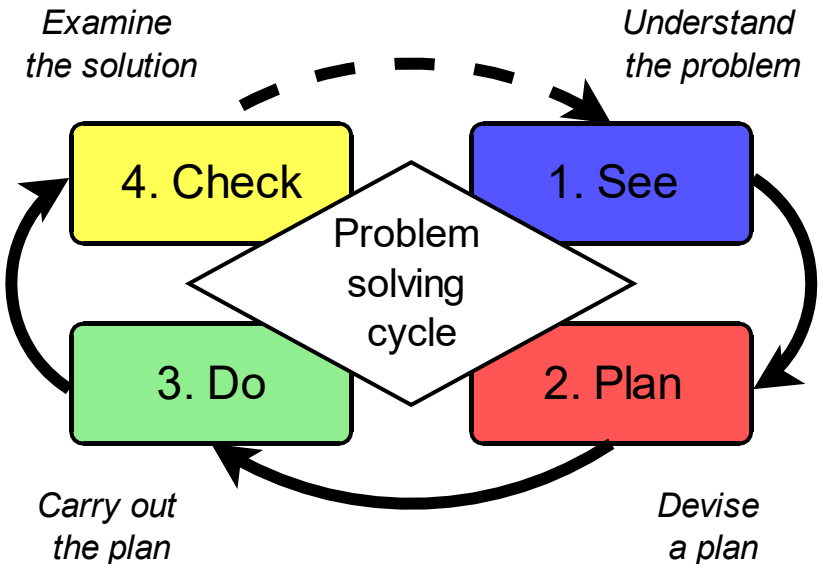
- capire il problema
- quali sono i dati, quali le incognite?
- quali sono le condizioni? sono soddisfacibili, ridondanti, contraddittorie?
- figure, notazione



Make things as simple as possible, but not simpler. (A. Einstein)
For every complex problem there is an answer that is clear, simple, and wrong.
(H.L. Mencken)

(2) Plan

- elaborare un piano
- mettere in relazione dati e incognite
- metodologie: divide et impera, composizione, astrazione...
- computational thinking
- cominciare a risolvere un problema più semplice
  - (vincoli rilassati)



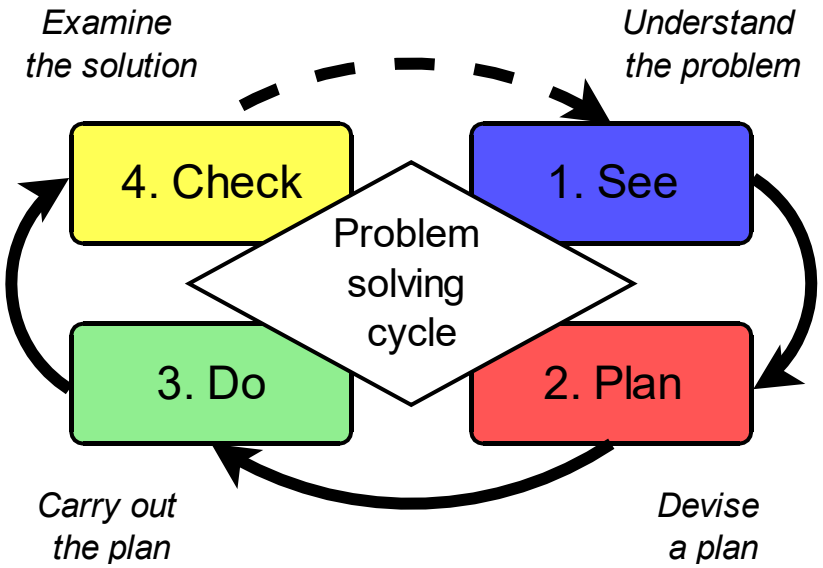
If you can't solve a problem…
then there is an easier problem you can solve: find it. (G. Polya)

(3) Do

- eseguire il piano
- controllare ogni passo
  - è corretto?

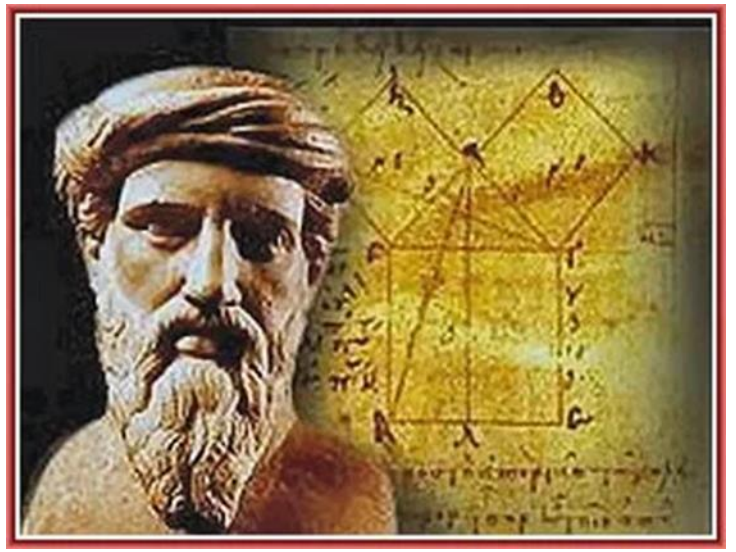
(4) Check

- controllare la soluzione
- è corretta?
- è ottenibile in altro modo?
- il risultato è utilizzabile per altri problemi?



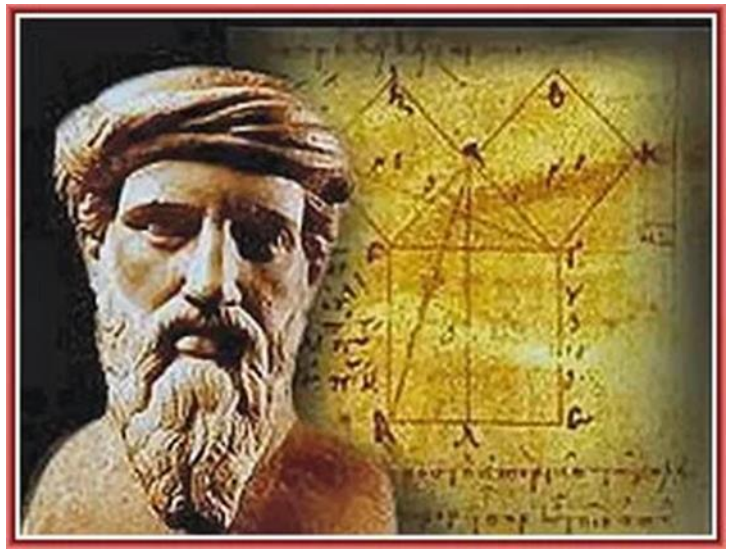
esempio: analisi

- l'***analista*** deve raccogliere le informazioni necessarie per definire il problema
- individua le ***informazioni iniziali*** significative
- individuare le ***informazioni finali*** (risultato)
- *esempio*: Pitagora identifica come obiettivo la ricerca del valore dell'ipotenusa di un triangolo rettangolo e come dati iniziali significativi i valori dei due cateti



esempio: progettazione

- il ***progettista*** fornisce una descrizione del procedimento che porta alla soluzione del problema (*algoritmo*)
- specifica le azioni da eseguire per passare dai dati iniziali ai dati intermedi ai risultati finali
- esempio:
  - calcola il quadrato del primo cateto
  - calcola il quadrato del secondo cateto
  - somma i due valori ottenuti
  - calcola la radice quadrata del valore ottenuto



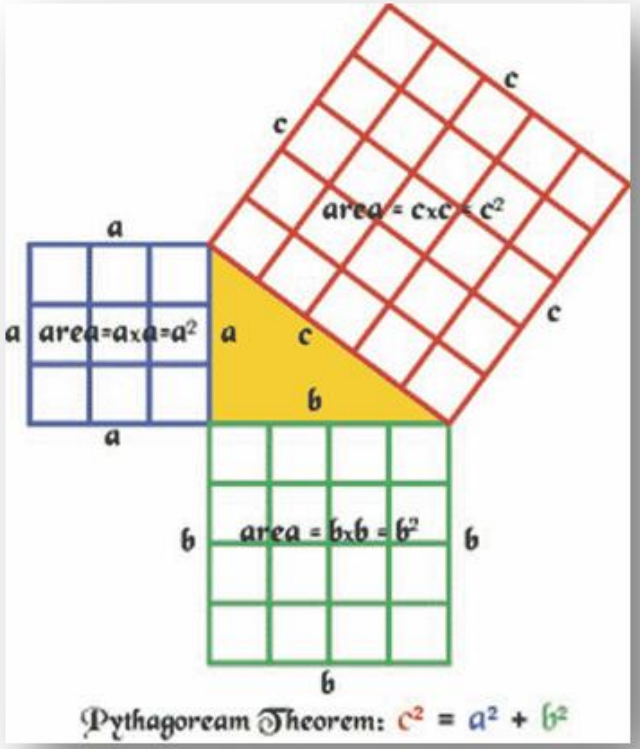
esempio: programmazione

- se il ***risolutore*** è un computer l'algoritmo deve essere tradotto in un ***linguaggio di programmazione***

```
""" pitagora """
import math
# dati di input
c1 = float(input("primo cateto: "))
c2 = float(input("secondo cateto: "))
# calcola il quadrato del primo cateto
q1 = math.pow(c1,2)
# calcola il quadrato del secondo cateto
q2 = math.pow(c2,2)
# somma i due valori ottenuti
s = q1 + q2
# calcola la radice quadrata del valore ottenuto
ip = math.sqrt(s)
# dati di output
print("ipotenusa",ip)
```

esempio: verifica della soluzione

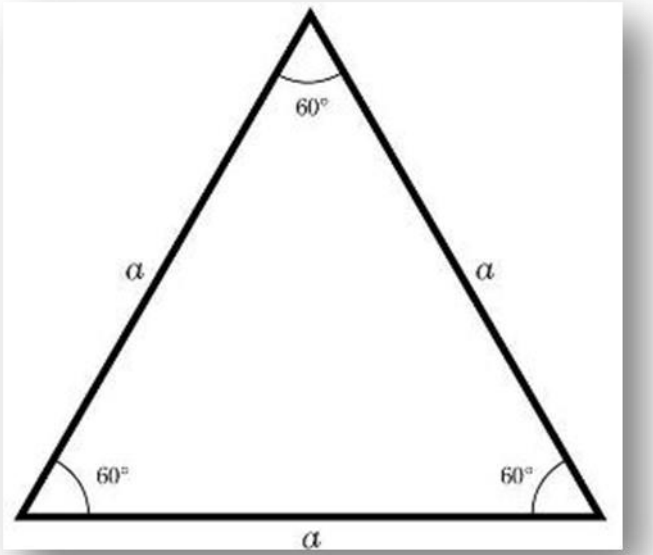
- il ***tester*** verifica che i risultati ottenuti non generino alcuna contraddizione con i dati iniziali
- in caso contrario si deve ripartire dall'analisi per poi passare di nuovo alla progettazione finché la verifica della soluzione non ha dato esito positivo

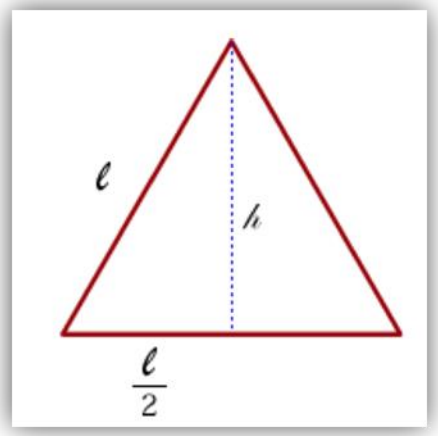


definizioni

- ***algoritmo***: procedimento che risolve un determinato problema attraverso un numero *finito* di passi *elementari* (al-Khwarizmi, ابو جعفر محمد خوارزمی ~800)
- ***dati***: *iniziali* (istanza problema), *intermedi*, *finali* (soluzione)
- ***passi elementari***: azioni *atomiche* non scomponibili in azioni più semplici
- ***processo*** (esecuzione): sequenza ordinata di passi
- *proprietà dell'algoritmo*: finitezza, non ambiguità, realizzabilità, efficienza…

- data la lunghezza di un lato di un triangolo equilatero trovare il perimetro e l'area





classi di problemi

- molti problemi hanno ***radice comune***, appartengono alla stessa classe
- uno stesso elenco di istruzioni può servire per la soluzione di problemi specifici che ***differiscono*** solo per le informazioni iniziali
- la sequenza di istruzioni che permette di trovare l'ipotenusa del triangolo con cateti di cm 3 e cm 4 (***problema specifico***) è la stesa che permette di trovare l'ipotenusa di un qualsiasi triangolo rettangolo con cateti di dimensione x, y.
  - i cateti x, y sono i ***parametri*** che caratterizzano questa classe di problemi
- è importante quindi non trattare un problema specifico ma una classe di problemi

esempi di algoritmi

- sono esempi di ***algoritmi*** le procedure che permettono di:
  - effettuare le quattro operazioni matematiche
  - ordinare di una sequenza di numeri
  - verificare la presenza di una parola in un testo
  - simulare il volo di un aereo
  - far diventare il computer un grande giocatore di scacchi
- e …
  - la ricetta per la torta al cioccolato ?
  - le istruzioni di IKEA per montare la libreria MALSJÖ ?

Per preparare la torta al cioccolato per prima cosa sciogliete al microonde o a bagnomaria il cioccolato fondente (1) spezzettato e lasciatelo intiepidire mescolando di tanto in tanto. In una ciotola capiente, ponete il burro tagliato a cubetti e lasciatelo ammorbidire a temperatura ambiente. Sbattetelo con una frusta (2) (o con lo sbattitore) fino a ridurlo a crema, poi aggiungete lo zucchero (3) e continuate a sbattere per amalgamarlo bene.

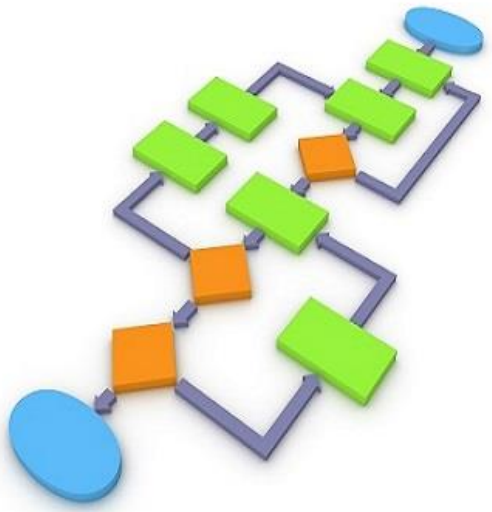
Unite un uovo alla volta (4) sempre sbattendo e via via le altre, avendo l'accortezza di fare assorbire completamente l'uovo prima di aggiungerne un altro: in questo modo avrete ottenuto una crema soffice e omogenea. Incorporate il cioccolato fondente sciolto (5) e ormai tiepido e il pizzico di sale. In una ciotola unite assieme alla farina, il lievito e il cacao amaro (6), quindi setacciateli.

Aggiungete al composto di burro, uova e cioccolato gli ingredienti (7) setacciati alternando l'aggiunta del latte (8) (non troppo freddo). Una volta ottenuto il composto finale, imburrate e infarinate una tortiera del diametro di 22-24 cm quindi versatelo al suo interno (9), livellando poi la superficie. Infornate in forno statico già caldo a 180° per circa 60 minuti, quindi pungetela con uno stecchino per vedere se la torta al cioccolato è cotta. Lasciatela leggermente intiepidire poi sformatela e ponetela su di una gratella per farla raffreddare definitivamente.

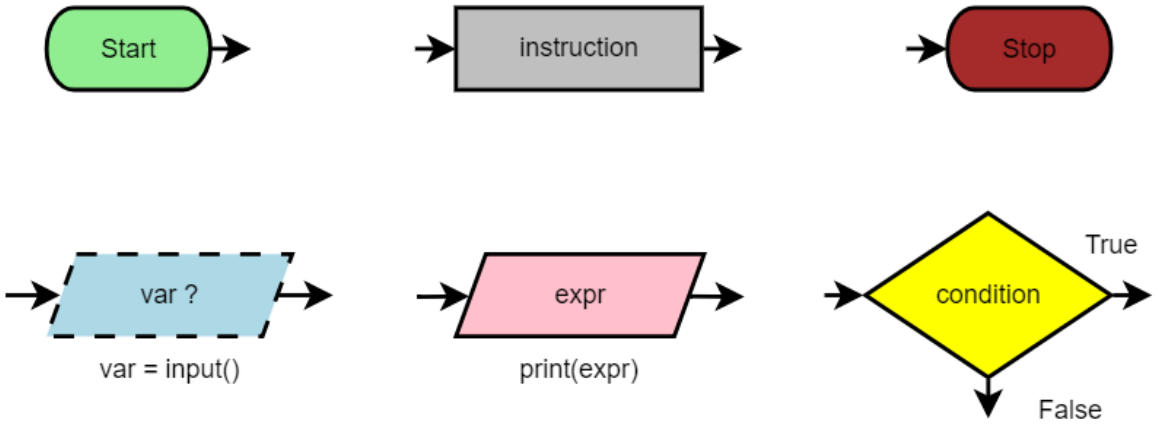
- caratteristiche di un linguaggio algoritmico
  - ***non ambiguità***
  - capacità di esplicitare il ***flusso*** di esecuzione delle istruzioni
- deve contenere istruzioni di tipo:
  - ***operativo*** (fare qualcosa)
  - ***input/output*** (comunicare con il mondo esterno)
  - ***decisionale*** (variare il flusso di esecuzione)

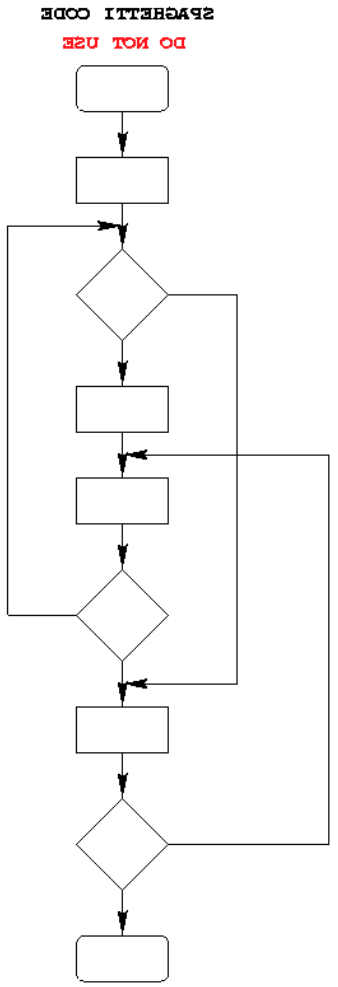
diagrammi di flusso

- diagramma di flusso (***flow-chart***):
  - rappresentazione grafica di algoritmi
  - più ***efficace*** e ***meno ambigua*** di una descrizione a parole
- è un grafo orientato
- due tipi di entità:
  - ***nodi***
  - ***archi***

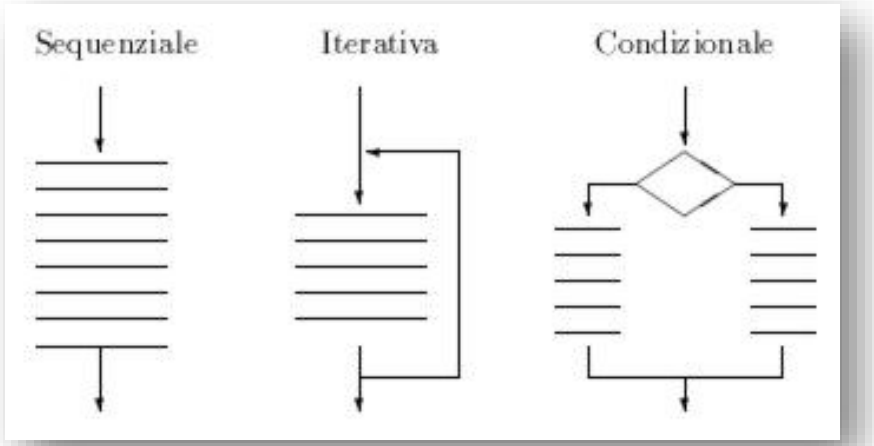


tipi di nodi

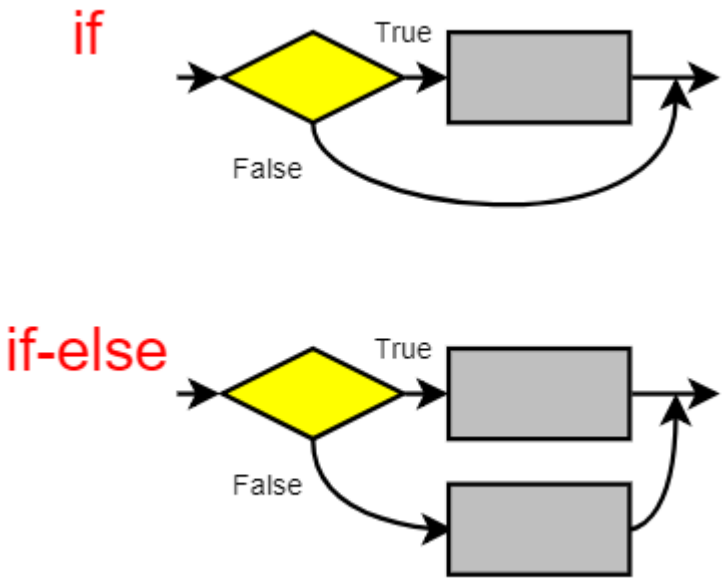

Start
instruction
Stop
var ?
var = input()
expr
print(expr)
condition
True
False

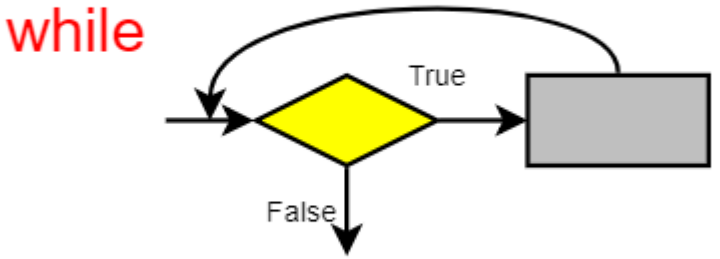

SPAGHETTI CODE
DO NOT USE

- strutture di controllo:
  - ***sequenza***
  - ***selezione***
  - ***iterazione***

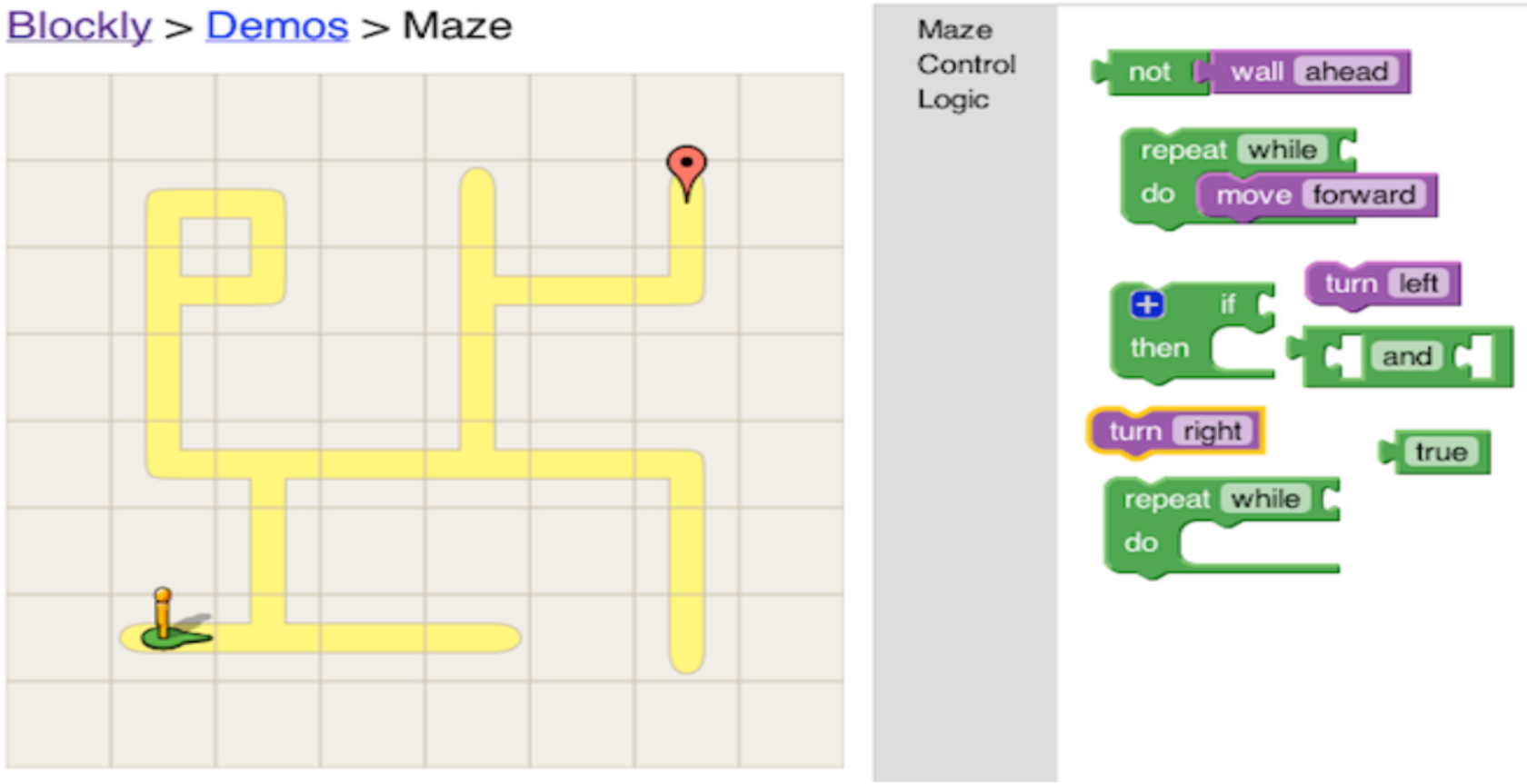


Qualunque algoritmo può essere implementato utilizzando queste tre sole strutture (Teorema di Böhm-Jacopini, 1966)





esempi quotidiani di if e while:
“se non c'è il lievito, usare due cucchiaini di bicarbonato”
“battere gli albumi finché non montano”



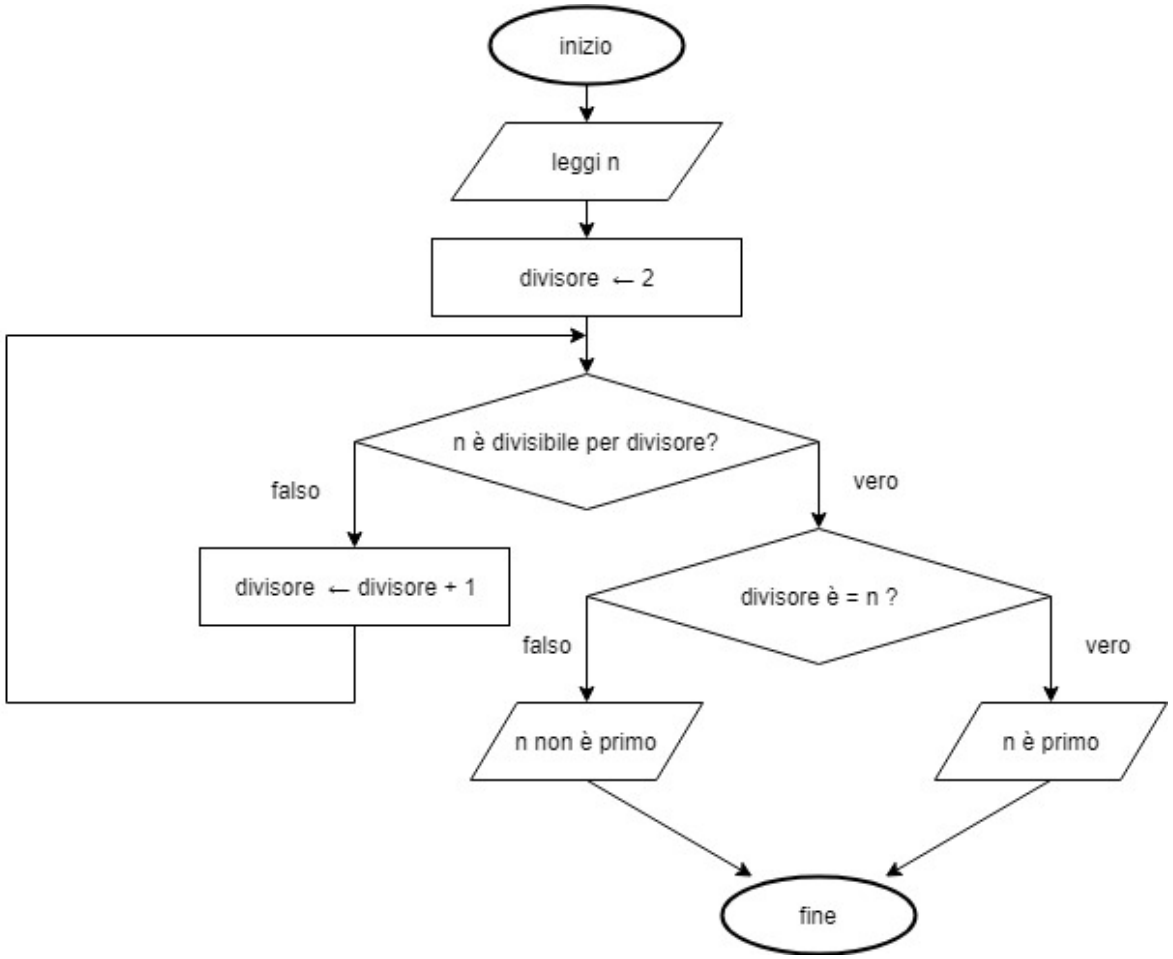
http://blockly-games.appspot.com/maze

http://www.ce.unipr.it/~aferrari/codowood/maze.html

- ***determinare se un numero è primo***
- ***analisi***:
  - un numero è primo se è divisibile esattamente solo per 1 e per se stesso
  - si cerca il minimo divisore intero maggiore di 1 del numero
  - se è uguale al numero stesso allora questo è primo
- ***dato iniziale***:
  - un qualsiasi numero intero
- ***dato finale***:
  - «primo» o «non primo»

- provare a dividere il numero per 2, per 3, per 4 e così fino a che il resto della divisione intera è diverso da zero
- i tentativi si esauriscono quando il resto è uguale a zero (si è individuato un divisore esatto del numero)
- se il divisore è uguale al numero stesso allora questo è un numero primo

flow chart


inizio
leggi n
divisore ← 2
n è divisibile per divisore?
falso
vero
divisore ← divisore + 1
divisore è = n ?
falso
vero
n non è primo
n è primo
fine

- per poter eseguire le istruzioni che compongono l'algoritmo è necessario poter ***memorizzare***
  - i ***dati*** iniziali
  - i ***dati*** intermedi
  - i ***risultati*** finali
  - ma anche le ***istruzioni*** stesse
- è necessaria una ***memoria***, indipendentemente dal fatto che l'esecutore sia umano o una macchina

risolvere i problemi

1. dato il raggio calcolare la circonferenza e l'area del cerchio
2. date le coordinate di due punti A e B trovare le coordinate del punto medio del segmento AB
3. per il lavoro di un operaio sono registrati l'orario di entrata e l'orario di uscita sia al mattino che al pomeriggio: calcolare il totale delle ore e dei minuti lavorati e, data la paga oraria, calcolare la paga giornaliera
4. per la vendita di un prodotto si deve applicare uno sconto progressivo in base al numero di pezzi ordinati in base alla regola: fino a 3 pezzi 5%, fino a 5 pezzi 10%, fino a 10 pezzi 20%, oltre 10 pezzi 30%. Dato il prezzo del prodotto e il numero di pezzi ordinati calcolare il prezzo da pagare.