

HOW DOES COMPUTER  
PROGRAMMING WORK?

MAGIC.



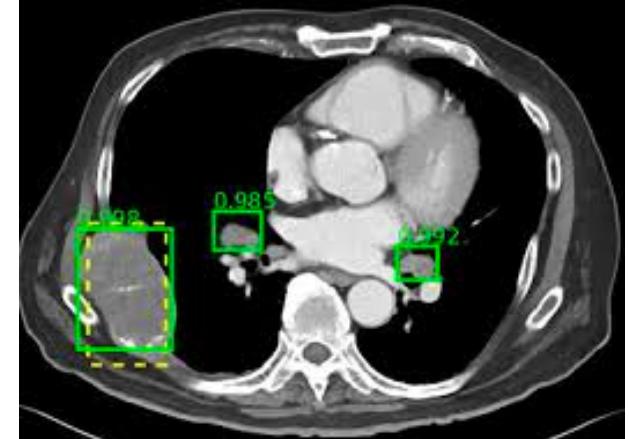
© 2012

# Introduzione alla Programmazione

Francesca Odone

# Dove vogliamo arrivare?...

- L'arte di programmare si può esprimere in migliaia di modi (e di mondi) diversi



# Punto di partenza (ossia: *i contenuti del corso*)

- **Concetti introduttivi:** problem solving e pensiero computazionale; il ciclo di vita del software; elementi di un sistema di calcolo
- **Introduzione al C++:** struttura base di un programma; identificatori; operatori aritmetici e logici; tipi di dato base; variabili e costanti; assegnazioni; standard I/O. Espressioni logiche; costrutti di selezione (if e switch); cicli (while, do-while, for)
- **Oltre i tipi base:** Sequenze di elementi omogenei (array e operazioni su array); strutture eterogenee (struct)
- **Funzioni;** passaggio dei parametri per valore, per riferimento, per costante. Funzioni parziali ed eccezioni
- **Puntatori;** i puntatori e allocazione dinamica della memoria
- Vector; principali funzioni membro e operatori.
- **Programmazione modulare** e compilazione separata
- **Liste** semplici ed esempi di applicazione (insiemi, pile, code, dizionari, ...)
- **Ricorsione**

# primi passi: il computer

- *Computer*: one that computes; specifically: a programmable usually electronic device that can store, retrieve, and process data [Merriam-Webster Dictionary]



- .. una definizione molto semplice e breve per qualcosa che ha cambiato profondamente la nostra società e il nostro modo di vivere!

# What is Programming?

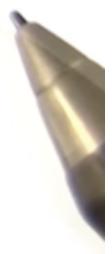


Let me  
rephrase  
that



I can  
do ANYTHING  
you tell  
me to...

... provided  
you speak  
my language



# primi passi: il programmatore

- Il programmatore analizza un **problema** e identifica la serie di passi che il computer dovrà eseguire
- E' suo il compito di descrivere i passi in modo che il computer possa comprenderli (e poi eseguirli)
- perche' lasciare che sia il computer a risolvere il problema?
  - le ragioni sono moltissime, per il momento pensiamo che un computer può svolgere lo stesso compito in modo consistente innumerevoli volte... senza annoiarsi

# primi passi: programmare

- molta parte del comportamento umano è caratterizzata da sequenze logiche.
  - sin da bambini siamo educati a seguire (o eseguire) istruzioni
- **programmare**: pianificare lo svolgimento di un'azione o un evento
- **programmare un computer**: pianificare una sequenza di passi che un computer possa eseguire
- **programma**: una sequenza di passi che un computer è in grado di eseguire

A che scopo? ... risolvere un problema

# primi passi: programmare

- Il termine **programmazione** fa riferimento all'insieme delle attività che portano allo sviluppo di applicazioni o programmi (il *software*) eseguibili su un sistema di calcolo per risolvere un problema predeterminato
- Caratteristiche di un “*buon programma*”:
  - rispondere alle specifiche (fare quello che deve)
  - correttezza e completezza
  - efficienza ed economia di uso delle risorse
  - scalabilità
  - robustezza, affidabilità, tolleranza ai guasti

... facciamo un balzo in fondo al corso

```
bool is_palindrome(string s)
{
    if (s.size() <= 1)
        return true;
    else if (s.front()!=s.back()) return false;
    else
    {
        s=s.substr(1,s.size()-2);
        cout << s<< endl;
        return is_palindrome(s);
    }
}
```





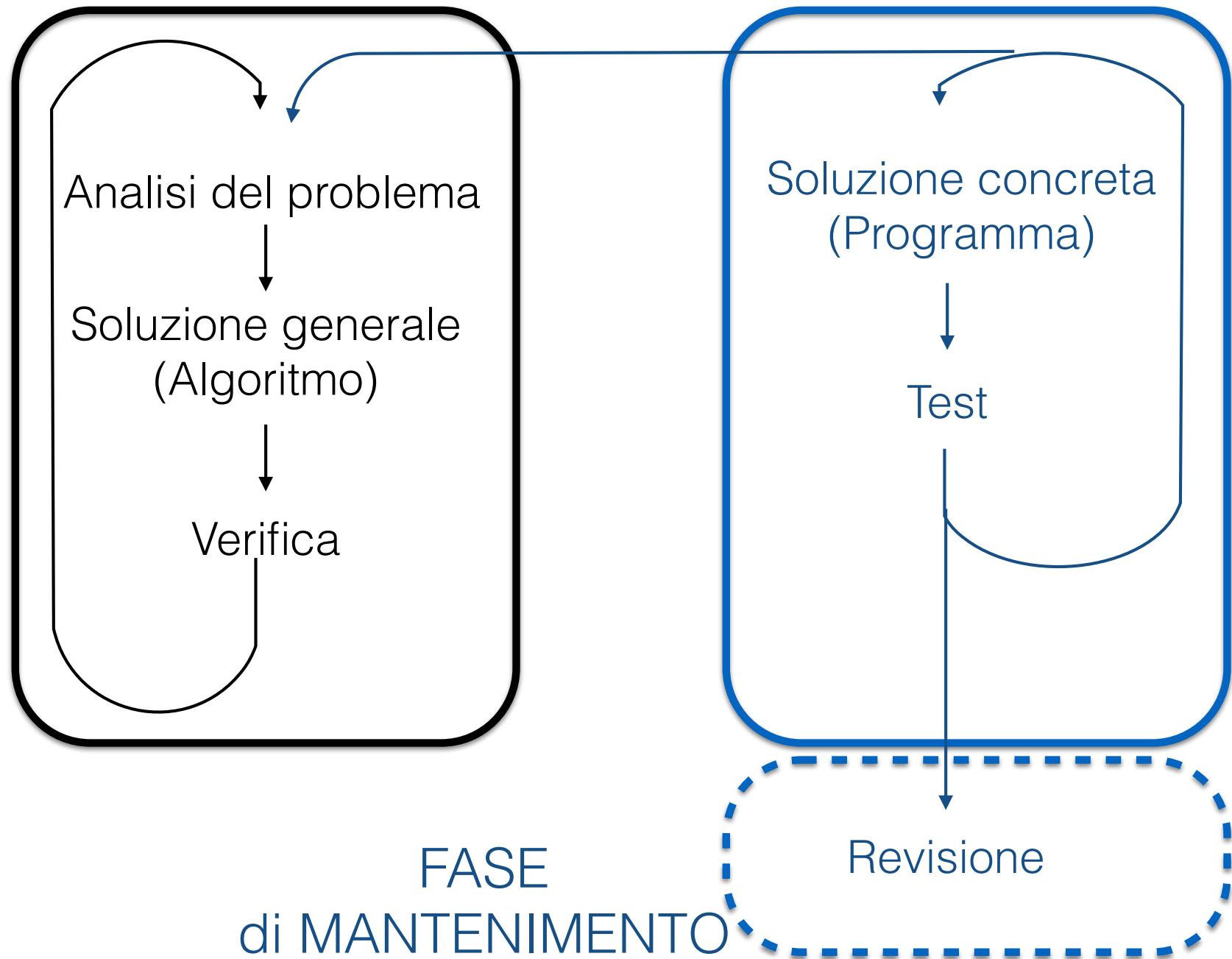
- ... cosa vuol dire “palindromo”?



- ... cosa vuol dire “palindromo”?
- Come verificare se una parola (sequenza di caratteri, o *stringa*, senza spazi) è palindroma?
  - a, ada, abba, ingegni, tenet sono palindrome

# FASE del PROBLEM SOLVING

# FASE dell'IMPLEMENTAZIONE



# primi passi: algoritmo

- **Algoritmo:** una procedura in più passi per la risoluzione di un problema in una quantità di tempo *finita*
- *Algoritmo e programma* hanno definizioni molto simili. In termini generali potremmo dire che il programma è un algoritmo “riscritto per il computer”

assunzione: programmatore e calcolatore hanno un linguaggio in comune (ossia si capiscono)

# esempio di algoritmo - spaghetti all'olio

- prendo la pentola
  - riempio d'acqua la pentola
  - accendo il fuoco
  - metto la pentola sul fuoco
  - **aspetto fino a che l'acqua non bolle**
  - butto nell'acqua una manciata di sale grosso
  - butto gli spaghetti
  - **fino a che gli spaghetti non sono cotti rimescolo "occasionalmente"**
  - scolo la pasta
  - la verso in una zuppiera
  - condisco con olio
- nota: qui sto assumendo che il “calcolatore” sappia come funziona il rubinetto!
- nota: questo passo coinvolge un periodo di tempo più lungo (e non noto a priori)
- nota: qui un passo è implicito (spengo il fuoco)

# esempio di algoritmo - prima di uscire di casa

- mi alzo dal letto
  - vado in bagno
  - preparo la colazione
  - mangio la colazione
  - lavo i denti
  - mi vesto
  - prendo la borsa
  - **se piove prendo l'ombrelllo**
- in questo caso l'ordine di alcuni passi può cambiare  
(non sempre la soluzione è unica!)

# fase di problem solving

## il signor Rossi ad un colloquio

- Il signor Rossi ha appena terminato un colloquio di lavoro e ha ricevuto un'offerta di 25.000 euro lordi annui per 13 mensilità
- il signor Rossi non sa se accettare perche' non riesce a capire quale sia il corrispondente stipendio mensile netto
- Il calcolo preciso dello stipendio mensile netto è piuttosto complicato e dipende, tra le altre cose, dai contributi previdenziali, le imposte regionali e comunali, eventuali familiari a carico....

# fase di problem solving il signor Rossi ad un colloquio

- Però può fare un conto approssimativo che gli dà rapidamente un'idea di quanto potrà percepire al mese
- Questa è la procedura (algoritmo):
  - stipendio mensile lordo = stipendio annuo lordo diviso il numero di mensilità
  - detrazioni circa il 25% dello stipendio lordo
  - lo stipendio mensile netto è lo stipendio mensile lordo meno le detrazioni

# fase di problem solving il signor Rossi ad un colloquio

- Però può fare un conto approssimativo che gli dà rapidamente un'idea di quanto potrà percepire al mese

il signor Rossi fa i suoi conti...

$$25000/13 = 1923.08$$

$$1923.08 * 0.25 = 480.8$$

$$1923.08 - 480.8 = 1442.28$$

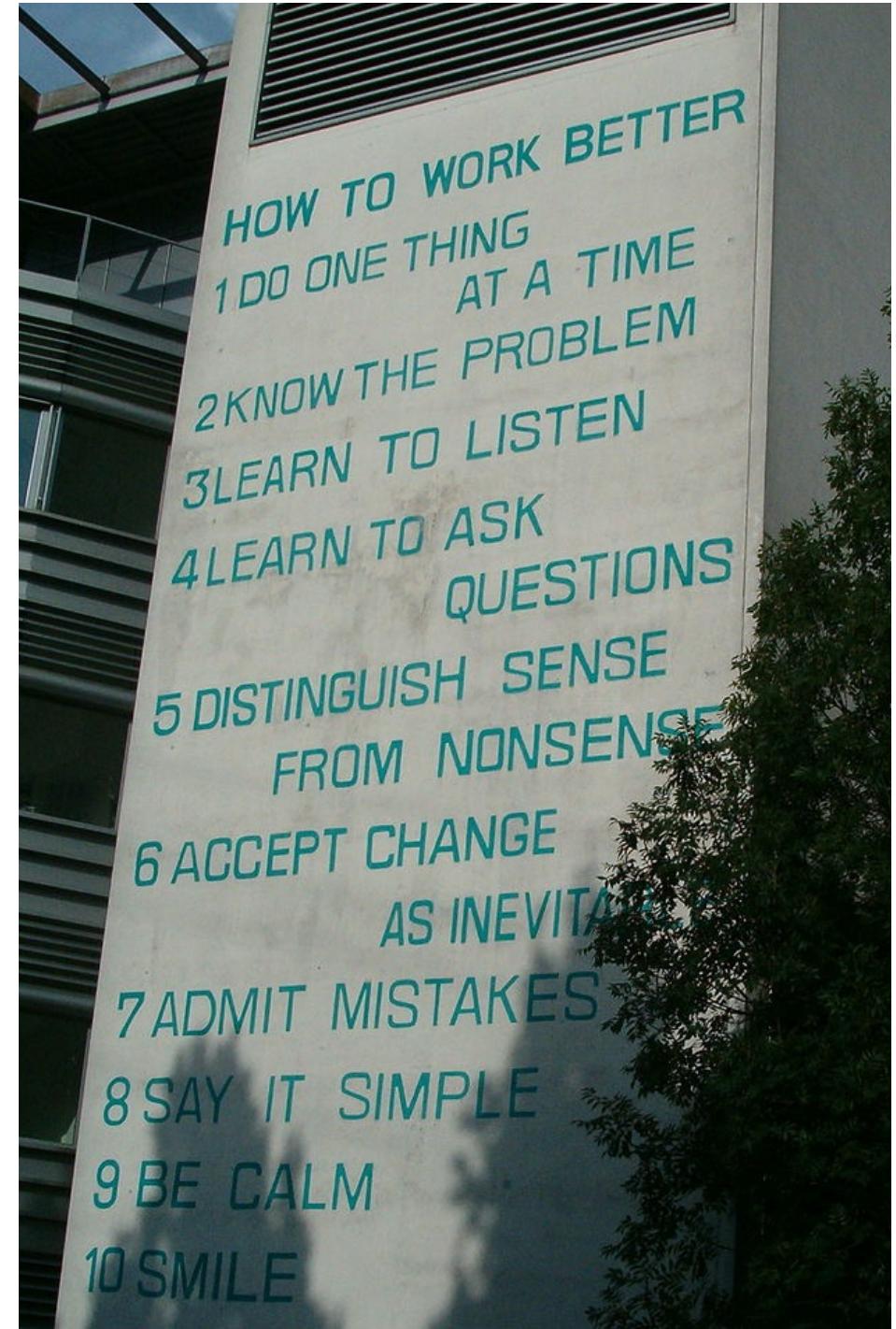
# fase di problem solving

## il signor Rossi ad un colloquio

- Se invece di usare la calcolatrice decide di (ri)scrivere un piccolo algoritmo che gli fa il calcolo, potrà usare la stessa procedura in eventuali colloqui futuri (semplicemente cambiando i dati in ingresso)
  - $\text{SAL} \leftarrow \text{inserisci}$  lo stipendio annuo lordo
  - $\text{NM} \leftarrow \text{inserisci}$  il numero di mensilità
  - $\text{DET} \leftarrow \text{inserisci}$  la percentuale di detrazione
  - $\text{SML} \leftarrow \text{SAL/NM}$
  - $\text{detrazioni} \leftarrow \text{DET} * \text{SML}$
  - $\text{SMN} \leftarrow \text{SML} - \text{detrazioni}$
  - **stampa** SMN

# il problem solving

- un'attività del pensiero che un organismo o un dispositivo di intelligenza artificiale mettono in atto per raggiungere una condizione desiderata a partire da una condizione data [wikipedia it].
- Problem solving consists of using generic or ad hoc methods, in an orderly manner, for finding solutions to problems.
- Some of the problem-solving techniques developed and used in artificial intelligence, computer science, engineering, mathematics, medicine, etc. are related to mental problem-solving techniques studied in psychology [wikipedia en].



# Pensiero computazionale

Coinvolge una serie di abilità di problem solving che sono tipiche dell'attività del programmatore:

- *Decomposizione*: spezzare un problema in passi atomici più semplici da trasferire ad un altro “agente” (persona o computer)
- *Riconoscimento di pattern*: notare similitudini o differenze con altri problemi/attività noti
- *Generalizzazione e astrazione*: filtrare informazioni non necessarie e ottenere rappresentazioni del problema che siano il più generali possibile (e quindi maggiormente riusabili)
- *Progettazione di algoritmi*: riassumere quanto compreso dai passi precedenti in una procedura passo-passo che aiuti l’agente a risolvere il problema

# Dal problema al programma

- in questo corso, all'inizio, vi sarà dato un problema e vi saranno chiariti tutti gli elementi necessari a trovare una soluzione
- mentre il corso procede i problemi prenderanno una forma sempre più astratta; dovrete imparare a passare dall'inquadramento generale ai dettagli, formulando le domande giuste per ridurre le *ambiguità* intrinseche nella formulazione di un problema

# esempio - algoritmo dell'anno bisestile

- Problema: dato un anno vogliamo verificare se sia bisestile
- Regola
  - l'anno deve essere divisibile per 4
  - ma non deve essere multiplo di 100 (non divisibile per 100)
  - ma se è un multiplo di 400 allora è bisestile (divisibile per 400)

# esempio - algoritmo dell'anno bisestile

- Divide et impera - tre passi fondamentali
  - chiedi un anno
  - verifica se l'anno è bisestile
  - produci il risultato

# esempio - algoritmo dell'anno bisestile

- Divide et impera - tre passi fondamentali
  - **chiedi un anno**
  - verifica se l'anno è bisestile
  - produci il risultato

come fare?

- chiarimento: ci interessano solo i numeri di 4 cifre
- chiediamo all'utente di inserirli da tastiera

# esempio - algoritmo dell'anno bisestile

- Divide et impera - tre passi fondamentali
  - chiedi un anno
  - verifica se l'anno è bisestile
  - **produci il risultato**

come fare?

- “stampiamo” un messaggio sul monitor

# esempio - algoritmo dell'anno bisestile

- Divide et impera - tre passi fondamentali
  - chiedi un anno
  - **verifica se l'anno è bisestile**
  - produci il risultato

# esempio - algoritmo dell'anno bisestile

**verifica se l'anno N è bisestile:**

**Versione 1:**

- se N non è divisibile per 4 allora non è bisestile
- altrimenti verifica le eccezioni

# esempio - algoritmo dell'anno bisestile

**verifica se l'anno N è bisestile:**

**Versione 2:**

- se “non è divisibile per 4”:

1. dividi N per 4

2. se il resto è diverso da 0 allora N non è divisibile per 4 e quindi N non è bisestile

**se (il resto di  $N/4$  è diverso da 0)**

**allora N non è bisestile**

# esempio - algoritmo dell'anno bisestile

**verifica se l'anno N è bisestile:**

**Versione 2:**

- se ((N mod 4) not 0)      **se** (il resto di N/4 è diverso da 0)

**allora N non è bisestile**

**altrimenti verifico le eccezioni**

# esempio - algoritmo dell'anno bisestile

**verifica se l'anno N è bisestile:**

**Versione 2:**

a questo punto N è divisibile per 4

- **altrimenti** “verifica le eccezioni”:

1. **se**  $((N \bmod 100)=0) **allora** non è bisestile$

2. **Se**  $((N \bmod 400)=0) **allora** è bisestile$

# esempio - algoritmo dell'anno bisestile

**verifica se l'anno N è bisestile:**

**Versione 2:**

a questo punto N è divisibile per 4

- **altrimenti** “verifica le eccezioni”:

1. **se**  $((N \bmod 100)=0)$  **allora** non è bisestile
2. **Se**  $((N \bmod 400)=0)$  **allora** è bisestile

c'è qualcosa che non va!!!

# esempio - algoritmo dell'anno bisestile

**verifica se l'anno N è bisestile:**

**Versione 3:**

a questo punto N è divisibile per 4

- altrimenti “verifica le eccezioni”:

1. **Se**  $((N \text{ mod } 100) \text{ not } 0)$  **allora** è bisestile

2. **Se**  $((N \text{ mod } 100) = 0)$  e  $((N \text{ mod } 400) \text{ not } 0)$  allora non è bisestile

3. **se**  $((N \text{ mod } 400) = 0)$  allora è bisestile

inutile!

ora va bene ma si può ancora semplificare

# esempio - algoritmo dell'anno bisestile

**verifica se l'anno N è bisestile:**

**Versione 4 completa:**

**Se** ((N mod 4) not 0)

**allora** N non è bisestile [esco dal passo]

**altrimenti** // arrivano qui gli N divisibili per 4

**Se** ((N mod 100) not 0)

**allora** N è bisestile [esco dal passo]

**altrimenti** // arrivano qui gli N divisibili per 4 e per 100

**se** ((N mod 400) not 0) **allora** N non è bisestile [esco dal passo]

**altrimenti** N è bisestile [fine del passo] // (N divisibile per 4, per 100, per 400)

per implementare questo flusso di istruzioni un linguaggio di programmazione dovrà contenere operatori matematici e istruzioni che permettano di fare la selezione tra due strade possibili

**verifica se l'anno N è bisestile:**

**Versione 4 completa:**

**Se**  $((N \text{ mod } 4) \text{ not } 0)$

**allora** N non è bisestile [esco dal passo]

**altrimenti** // arrivano qui gli N divisibili per 4

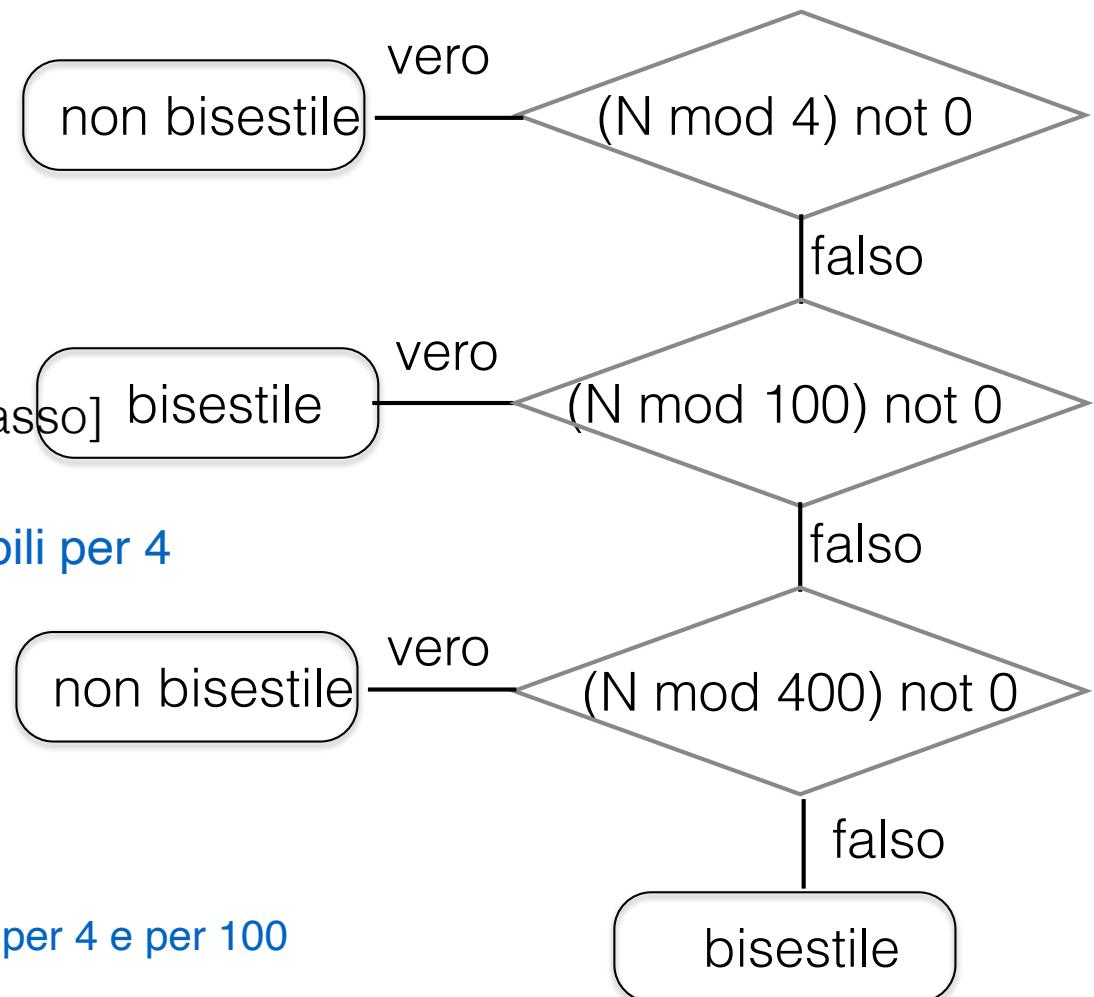
**Se**  $((N \text{ mod } 100) \text{ not } 0)$

**allora** N è bisestile [esco dal passo]

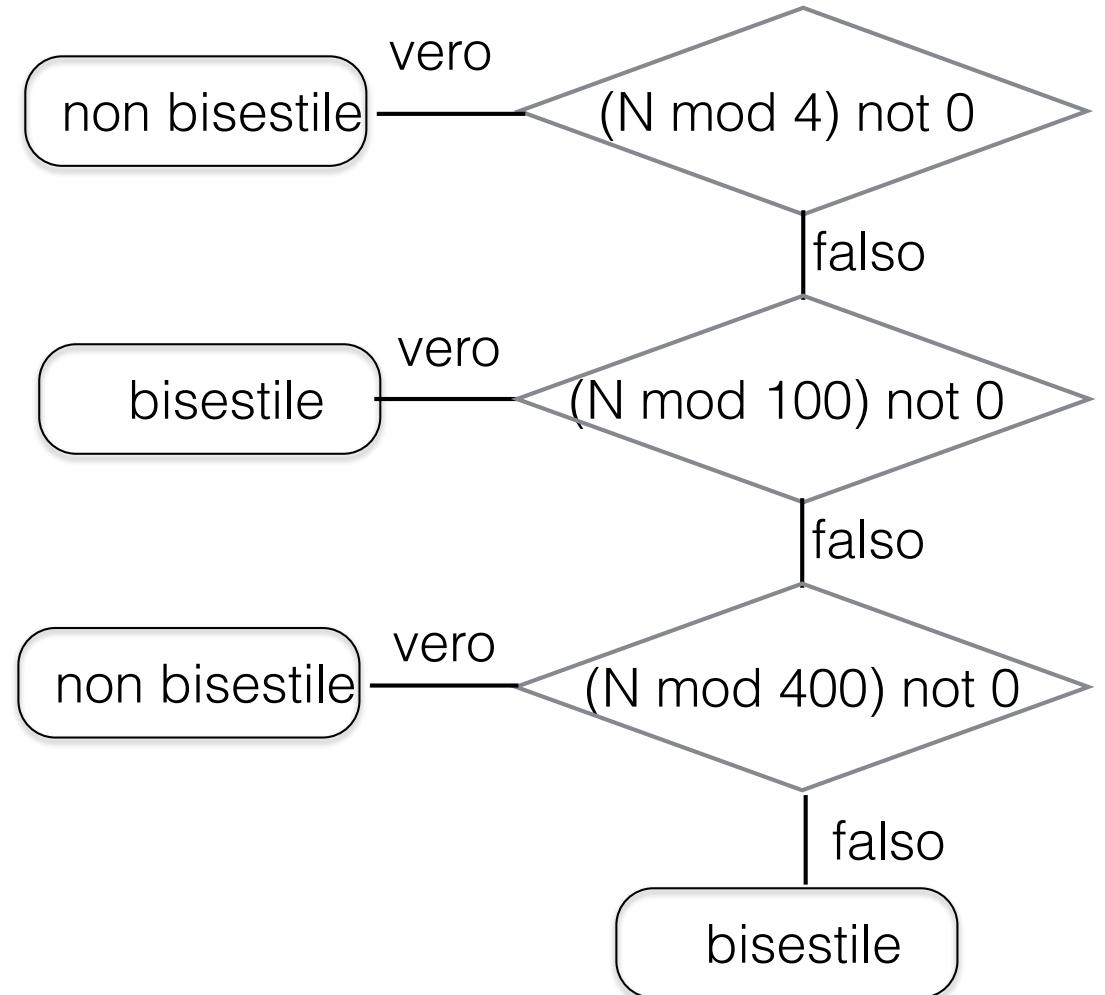
**altrimenti** // arrivano qui gli N divisibili per 4 e per 100

**se**  $((N \text{ mod } 400) \text{ not } 0)$  **allora** N non è bisestile [esco dal passo]

**altrimenti** N è bisestile [fine del passo] // $(N \text{ divisibile per } 4, \text{ per } 100, \text{ per } 400)$



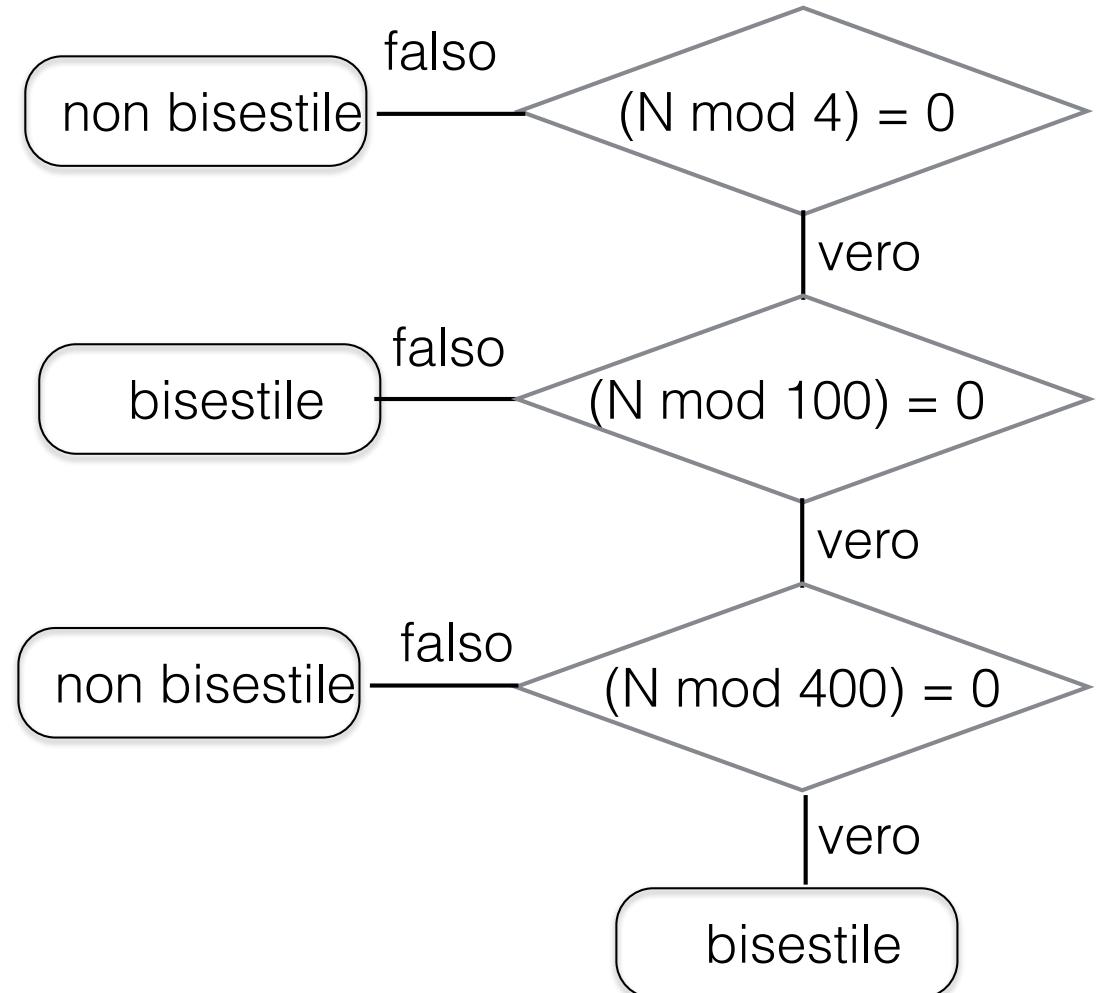
# esempio - algoritmo dell'anno bisestile



# esempio - algoritmo dell'anno bisestile

se siete ancora in dubbio,  
**verificate** che tutto torni con  
gli anni

- 2010 e 2017 non bisestili
- 1896, 1996, 2016 bisestili
- 1800 e 1900 non bisestili
- 1600 e 2000 bisestile



NB - in questa slide e nella precedente i diagrammi di  
flusso fanno verifiche opposte (careful!)

# esempio - algoritmo dell'anno bisestile

pseudo-codifica un po' più rigorosa

## versione 5

**if** ((N mod 4) not 0)

**then return** false

**else**

**if** ((N mod 100) not 0)

**then return** true

**else**

**if** ((N mod 400) not 0) **then return** false

**return** true // (N divisibile per 4, per 100, per 400)



true e false sono valori di tipo logico (o booleani)

# esempio - algoritmo dell'anno bisestile

Il programma completo

1.  $N \leftarrow$  Chiedi all'utente di inserire un numero naturale di 4 cifre
  2.  $A \leftarrow \text{Bisextile}(N)$
  3. **IF**  $A$  è true  
    Stampa "Anno bisestile"  
**ELSE**  
    Stampa "Anno non bisestile"
- questo nome "nasconde" il  
"pezzo" di algoritmo realizzato  
prima  
(funzione)*

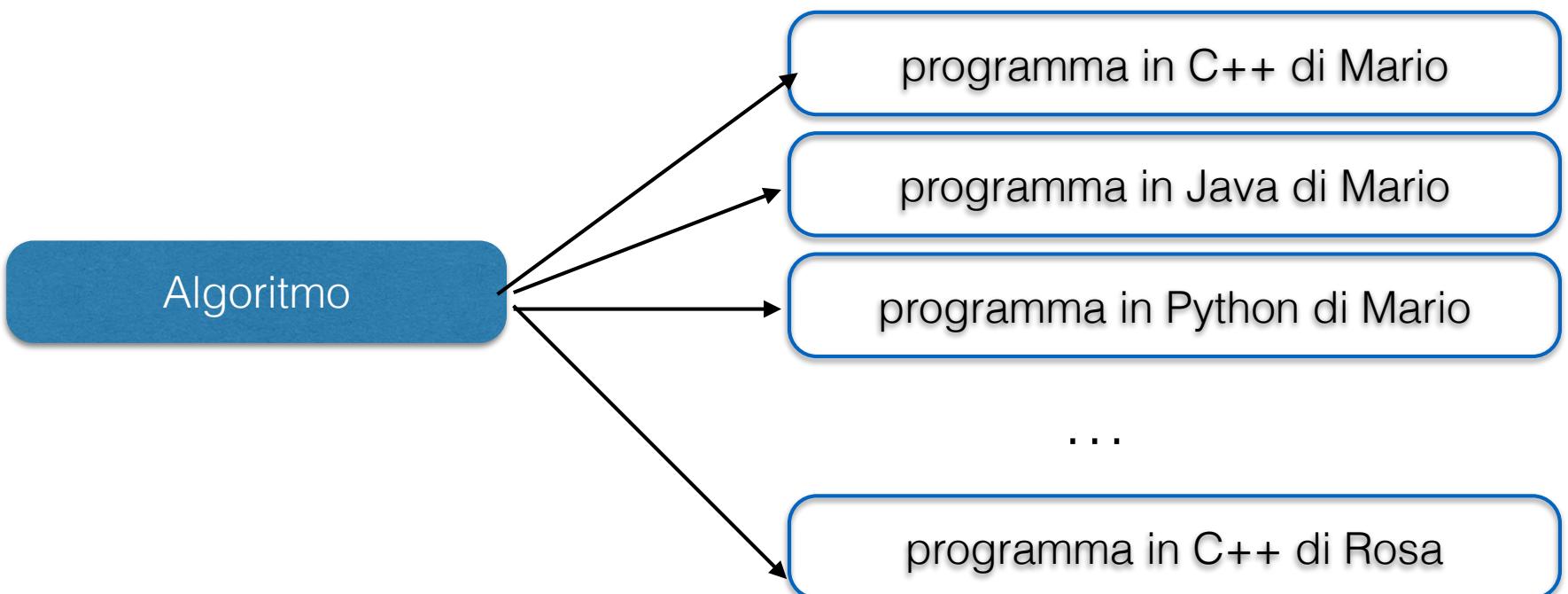
*N è un numero naturale - A è un booleano*

# primi passi: implementazione

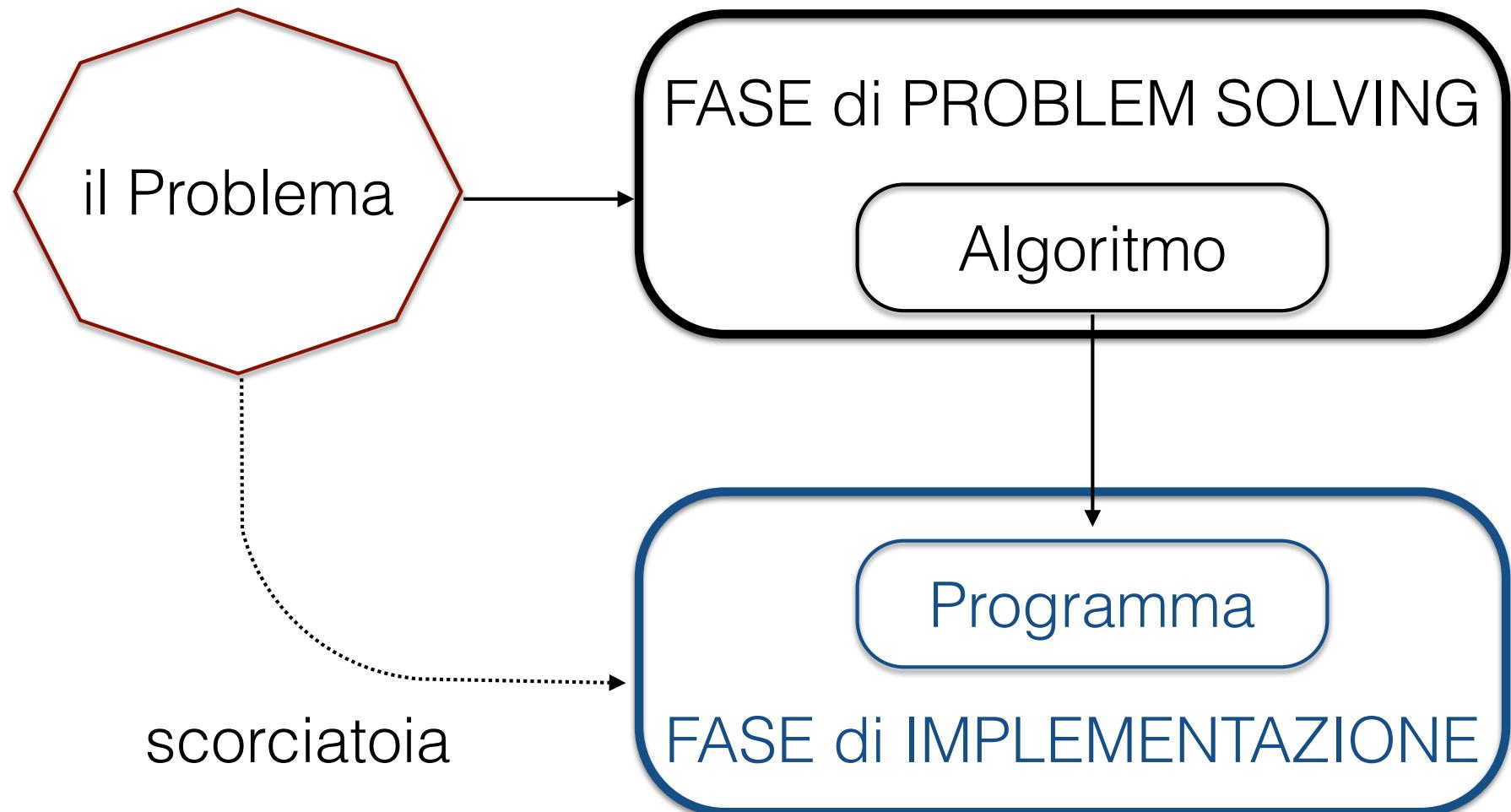
- quando il programmatore è soddisfatto dell'algoritmo scritto, deve tradurlo in un *linguaggio appropriato* in modo che il computer possa comprenderlo
- un **linguaggio di programmazione** è un insieme di regole, simboli e parole speciali usati per costruire un programma per il calcolatore
- in prima battuta lo possiamo pensare come una forma molto semplificata e rigida di inglese (con formule matematiche) che segue regole grammaticali molto rigide

# linguaggi di programmazione

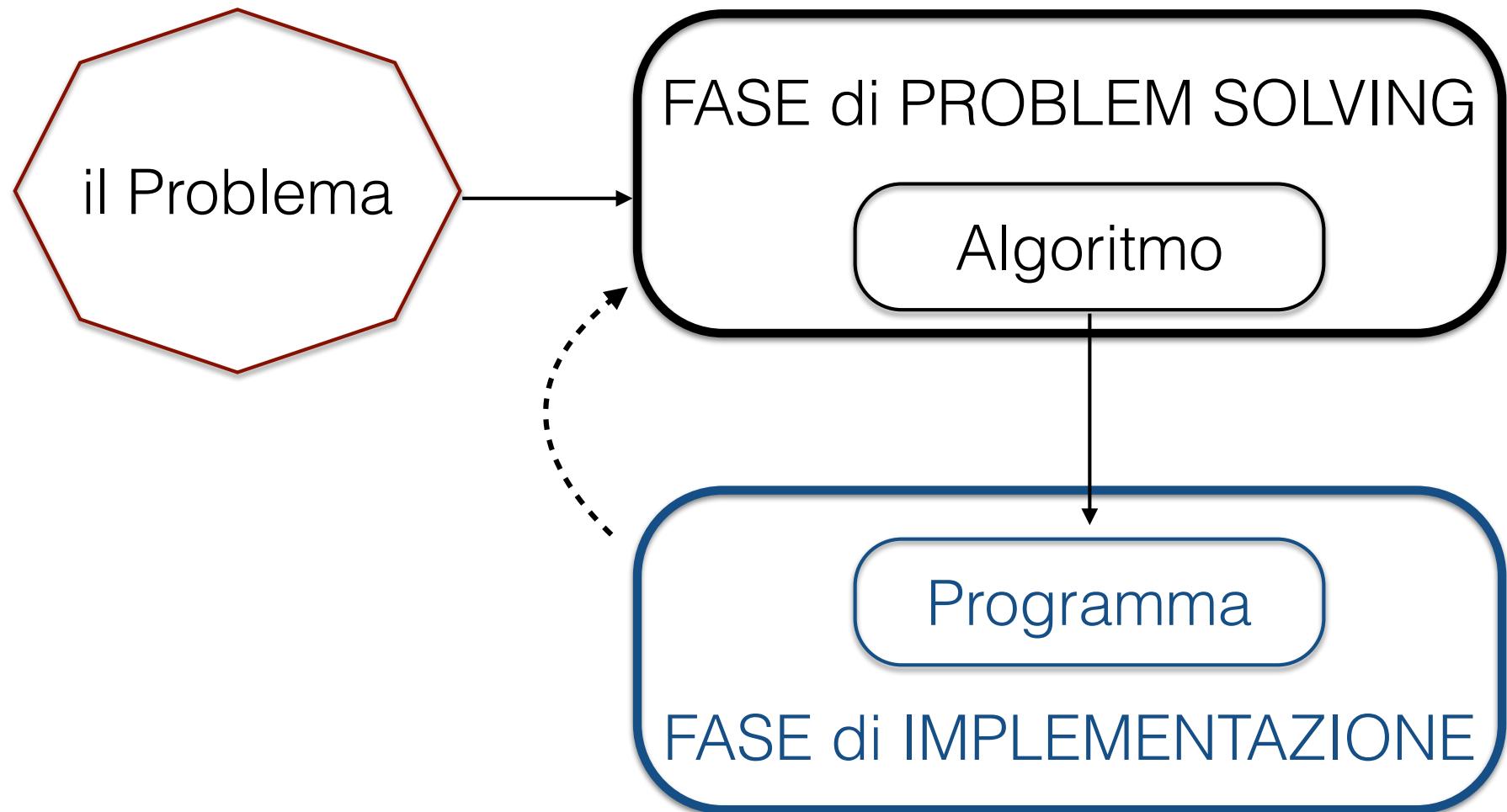
- uno stesso algoritmo può essere tradotto in modi diversi a seconda del linguaggio di programmazione scelto ...
- in realtà anche fissando il linguaggio di programmazione possiamo ottenere diverse implementazioni



# il ciclo di vita del programma



# il ciclo di vita del programma



Mai usare  
la scorciatoia!

# “parlare la lingua del computer”

- i sistemi di calcolo (o computer) sono dispositivi elettronici: segnali elettrici si propagano lungo canali al suo interno
- un concetto chiave, che ha permesso di ridurre (o gestire) la grande complessità interna dei computer è la strutturazione in vari *livelli di astrazione*
  - semplificazione tramite stratificazione
- la stratificazione “base” comprende il livello “uomo” (il programmatore che parla il linguaggio L1) e il livello “macchina” (il sistema di calcolo che parla il linguaggio L0)

# la macchina virtuale

- nella pratica all'interno di un sistema di calcolo si possono identificare vari livelli
- ogni livello può essere rappresentato in modo astratto tramite il concetto di macchina virtuale
- una macchina virtuale non fa necessariamente riferimento ad una macchina reale — potrebbe essere realizzata da un insieme di dispositivi diversi oppure essere una persona, ...

# la macchina virtuale della pizza

questa sera ho voglia di pizza

1. cucino la pizza!

mi procuro

- ingredienti
- la ricetta
- gli attrezzi

eseguo la ricetta...

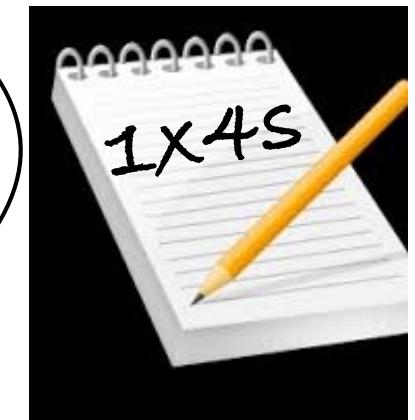
la ricetta è il programma  
la macchina virtuale  
che lo realizza sono io

2. vado in pizzeria

cliente

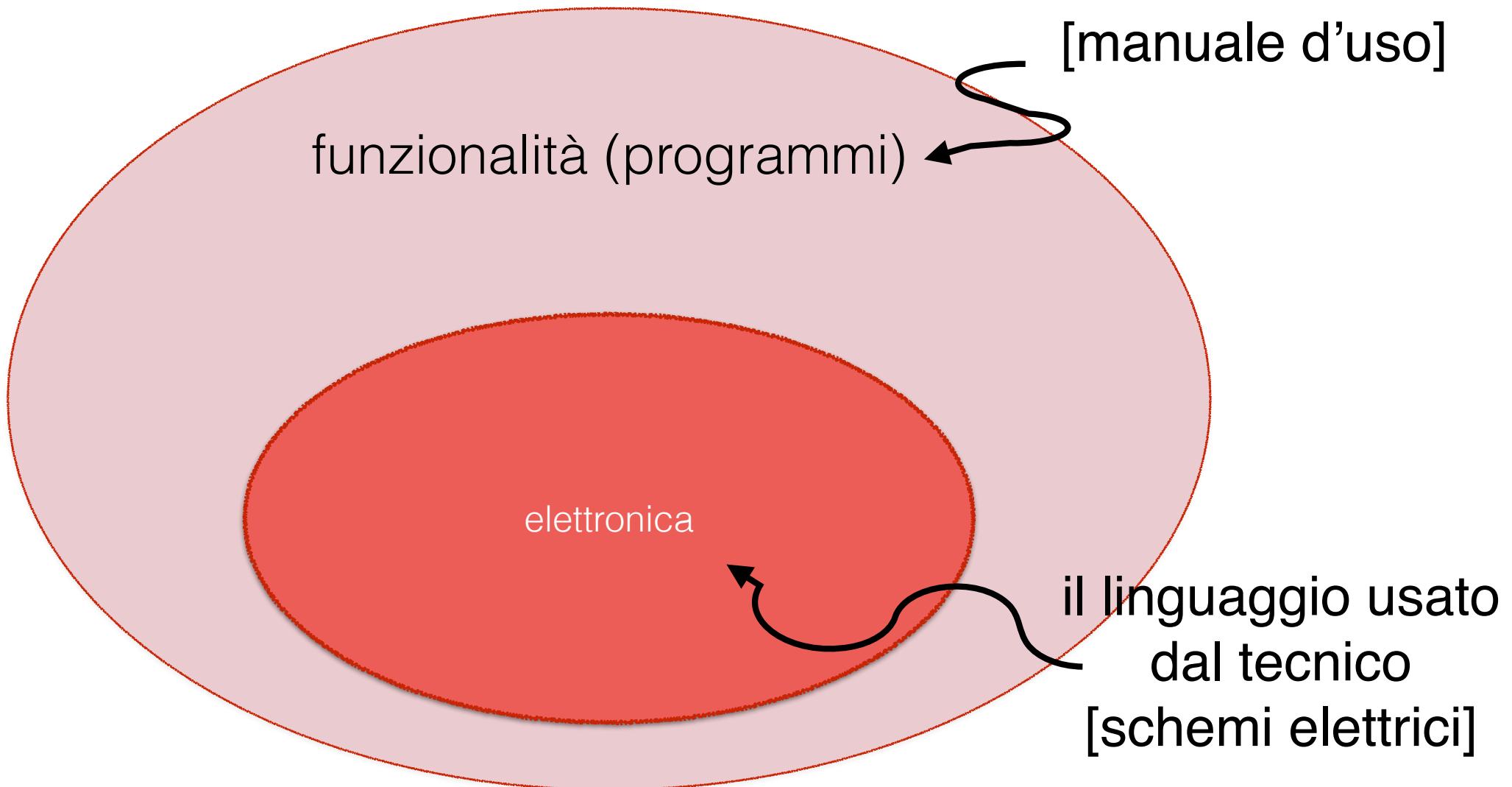
cameriere

cuoco



# la macchina virtuale lavabiancheria

2 livelli di astrazione diversi!



# la macchina virtuale - definizioni

- **l'alfabeto** della macchina virtuale è un insieme di simboli diversi tra loro e riconoscibili, utilizzati dalla macchina virtuale stessa
- il **linguaggio** della macchina virtuale è definito come l'insieme di tutte le sequenze di simboli dell'alfabeto che identificano **comandi** eseguibili oppure **dati** che vengono usati o prodotti dalla macchina virtuale
- la **macchina virtuale** è in grado di *interpretare* il linguaggio, ossia manipolare i dati eseguendo i comandi scritti nel linguaggio stesso.
- la macchina virtuale è indipendente da come l'effettiva macchina reale sia realizzata fisicamente

# Un po' di esercizi

progetta un algoritmo e scrivi il relativo pseudo-codice (al livello di dettaglio a cui riesci ad arrivare) per:

- calcolare l'età media degli studenti di IP
- calcolare il perimetro e l'area di un rettangolo (quali dati servono?)
- calcolare il perimetro e l'area di un triangolo
- giocare a forbice-carta-sasso (quali dati servono?)
- calcolare i numeri primi tra 1 e 120

# Un po' di esercizi

Partire da uno schema noto per risolvere altri problemi.

Progetta un algoritmo e scrivi il relativo pseudo-codice (al livello di dettaglio a cui riesci ad arrivare) per:

- calcolare l'altezza media degli studenti di IP
- calcolare l'età dello studente più giovane
- calcolare l'età dello studente più vecchio