ALGORITMI: PROPRIETÀ FONDAMENTALI

Non si può risolvere un problema senza prima fissare un insieme di “azioni”, di “mosse elementari” possibili per l'esecutore.
Bisogna conoscerne le caratteristiche, le mosse che sa eseguire ed il linguaggio che sa capire

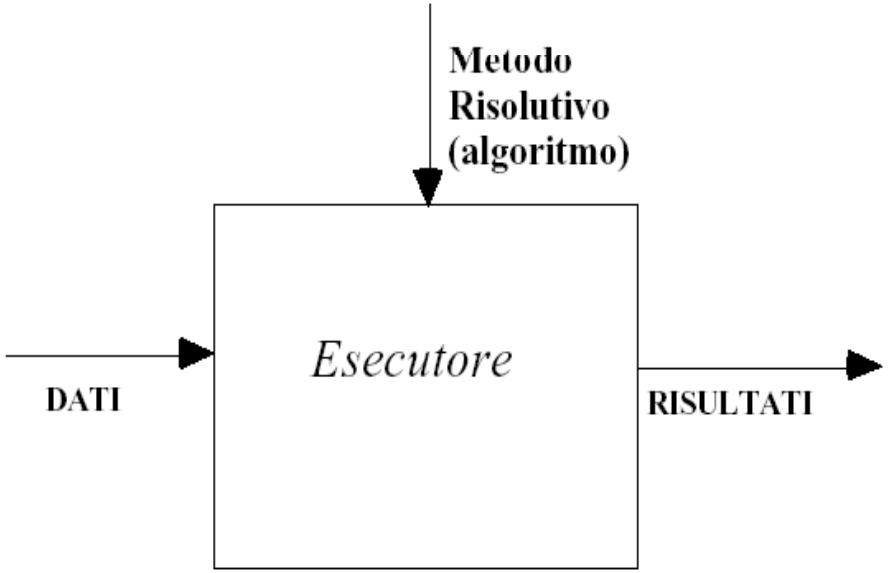
- **Non-ambiguità**: ogni azione deve essere *univocamente interpretabile* dall'esecutore,
- **Eseguibilità**: ogni azione deve essere *eseguibile in un tempo finito* da parte dell’esecutore dell’algoritmo,
- **Finitezza**: per ogni insieme di dati di ingresso, il numero totale di azioni da eseguire deve essere *finito*.

Proprietà desiderabile:

- **Efficienza**: deve risolvere il problema utilizzando al meglio le risorse a disposizione

ALGORITMI: ESECUTORE

Esecutore: una *macchina astratta* capace di *eseguire le azioni* specificate dall'algoritmo.



Rappresentazione degli algoritmi

1. Linguaggio naturale ← Linguaggi informali
2. Diagrammi di flusso ← Linguaggi semi-formali
3. Pseudo-codice ←
4. Linguaggio di programmazione ← Linguaggi formali

Variabili

- Un algoritmo descrive non una singola istanza di un problema, ma piuttosto un'intera classe di problemi (es. trovare il max in un insieme quali che siano i valori contentuti nell'insieme).

  Per questo motivo, le operazioni comprese nell'algoritmo fanno riferimento a variabili, vale a dire "contenitori" per dati.

- Ogni variabile ha
  - nome
  - valore (iniziale e corrente)

- Esempio:
  - N = 2 (il valore iniziale di N è 2)
  - N = N + 3 (il valore corrente di N è 5)

Esempio di algoritmo: calcolo del MCD

Problema: calcolo del Massimo Comun Divisore (MCD) fra due interi M ed N

Algoritmo n° 1

1. Calcola l'insieme A dei divisori di M
2. Calcola l'insieme B dei divisori di N
3. Calcola l'insieme C dei divisori comuni di M e N, C = A $\cap$ B
4. Il risultato è il massimo dell'insieme C

ESEMPIO: calcolo del MCD

Algoritmo n° 2

$$\mathbf{MCD\ (M,N)} = \begin{cases} \text{M (oppure N)} & \text{se } M=N \\ \mathbf{MCD\ (M-N,\ N)} & \text{se } M>N \\ \mathbf{MCD\ (M,\ N-M)} & \text{se } M<N \end{cases}$$

Strategia risolutiva:

- Finchè $M \neq N$:

  se $M>N$, sostituisci a M il valore $M' = M-N$

  altrimenti sostituisci a N il valore $N' = N-M$

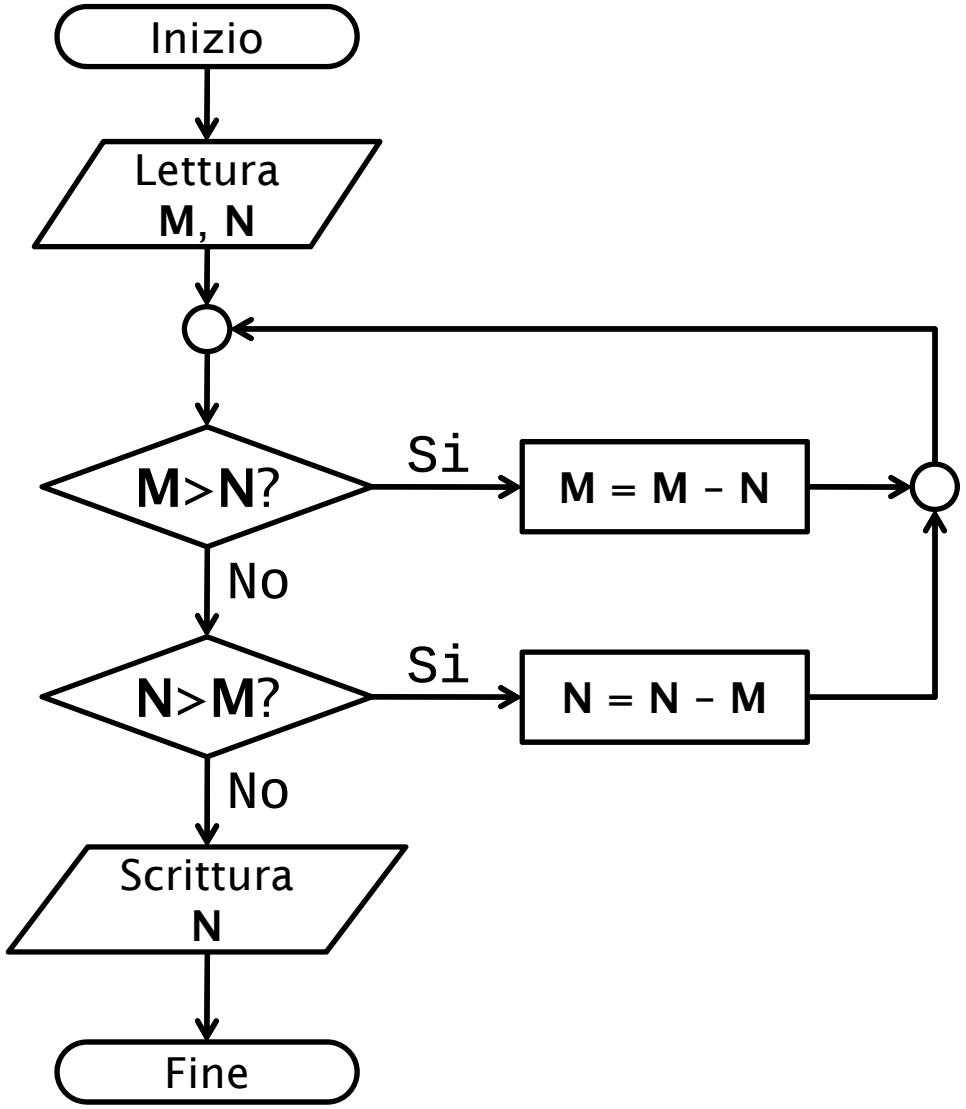
- Il Massimo Comun Divisore è il valore finale ottenuto quando M e N diventano uguali

ESEMPIO: calcolo del MCD

Calcoliamo il MCD di M = 24 e N = 14.

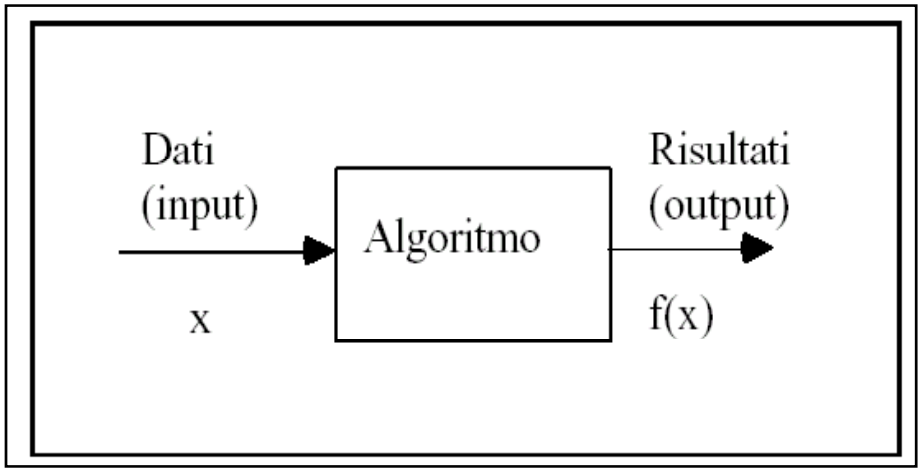
1. **M=24, N=14**
   24>14 → M = 24-14 = 10
2. **M=10, N=14**
   10<14 → N = 14-10 = 4
3. **M=10, N=4**
   10>4 → M = 10-4 = 6
4. **M=6, N=4**
   6>4 → M = 6-4 = 2
5. **M=2, N=4**
   2<4 → N = 4-2 = 2
6. **M=2, N=2**
   2=2 → "il MCD di 24 e 14 è 2"

Esempio di Grafo di flusso: MCD



ALGORITMI EQUIVALENTI

- In generale un algoritimo può essere visto come una **funzione** da un dominio di ingresso (***input)*** ad dominio di uscita (***output***)



- Due algoritmi si dicono **equivalenti** quando:
  - hanno stesso dominio di ingresso e stesso dominio di uscita;
  - in corrispondenza degli stessi valori nel dominio di ingresso producono gli stessi valori nel dominio di uscita.

ESEMPIO: calcolo del MCD

Algoritmo n° 3

Dati due interi M e N ($M \geq N$)

1. Dividi M per N, e sia R il resto della divisione;
2. Se R=0 allora termina. Il Massimo Comun Divisore è N;
3. Altrimenti assegna a M il valore di N ed a N il valore del resto R
   e torna al punto 1.

ESEMPIO: calcolo del MCD

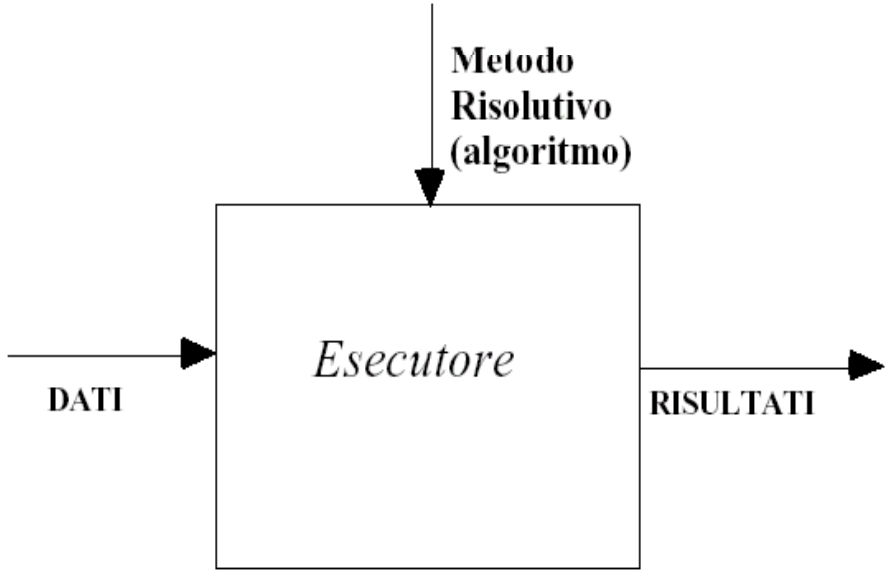
Calcoliamo il MCD di M = 24 e N = 14.

1. **M=24, N=14**
   24/14 = 1,R=10 → M=N=14, N=R=10
2. **M=14, N=10**
   14/10 = 1,R=4 → M=N=10, N=R=4
3. **M=10, N=4**
   10/4 = 2, R=2 → M=N=4, N=R=2
4. **M=4, N=2**
   4/2 = 2, R=0 → "il MCD di 24 e 14 è 2"

ALGORITMI: ESECUZIONE

- L'**esecuzione** delle azioni *nell'ordine specificato dall'algoritmo* consente di ottenere, a partire dai dati di ingresso, i risultati che risolvono la particolare *istanza* del problema.

Esecutore: una *macchina astratta* capace di *eseguire le azioni* specificate dall'algoritmo.



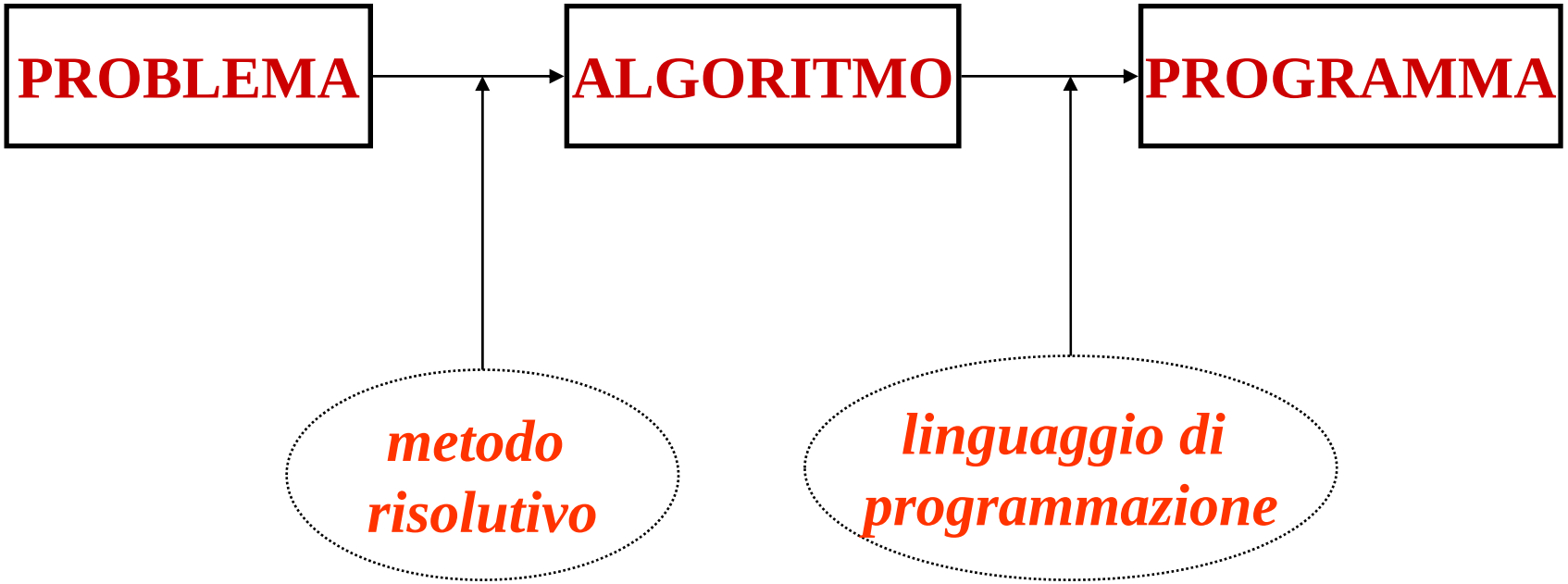
ALGORITMI: ESECUZIONE TRAMITE COMPUTER

La **programmazione** è l'attività con cui si predispone l'elaboratore ad eseguire un *particolare insieme di azioni* su una *particolare tipologia di dati*, allo scopo di *risolvere un problema*.

RISOLUZIONE DI PROBLEMI CON L'ELABORATORE ELETTRONICO

- Ogni elaboratore è una macchina in grado di eseguire **azioni** elementari su **dati**
- L'esecuzione delle azioni elementari è richiesta all'elaboratore tramite comandi chiamati **istruzioni**
- Le istruzioni sono espresse attraverso **frasi** di un opportuno **linguaggio di programmazione**
- Un **programma** non è altro che la formulazione testuale di un algoritmo in un linguaggio di programmazione

ALGORITMI E PROGRAMMI



LINGUAGGI AD ALTO LIVELLO

È opportuno impostare la soluzione di un problema a partire dalle "mosse elementari" del **linguaggio macchina**?

- SI, per risolvere il problema *con efficienza*
- NO, se la macchina di partenza ha mosse di livello *troppo basso* (difficile progettare un algoritmo)

Linguaggi di Programmazione ad Alto Livello (di astrazione)

- le istruzioni corrispondono ad operazioni più complesse
- esempi: Pascal, Basic, C, C++, Java

Fondamenti di Informatica

Francesco Folino

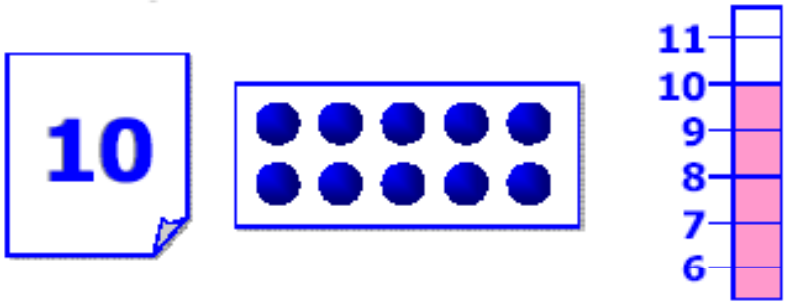
Codifica dell'Informazione

CODIFICA DI DATI E ISTRUZIONI

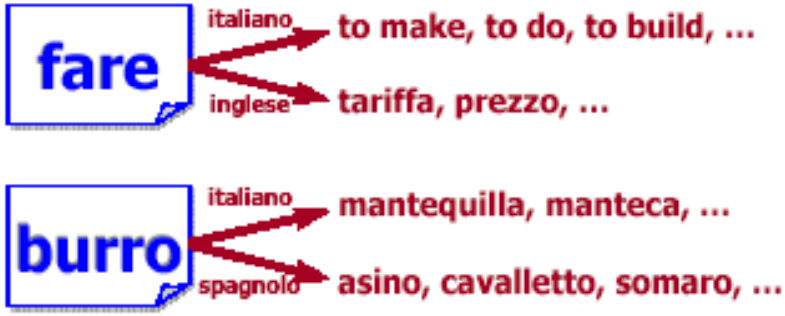
- Algoritmi
  - Istruzioni che operano su dati
- Per scrivere un programma è necessario rappresentare dati e istruzioni in un formato tale che l'esecutore automatico sia in grado di
  - Memorizzare istruzioni e dati
  - Manipolare istruzioni e dati
- Le informazioni gestite dai sistemi di elaborazione devono essere codificate
  - per poter essere memorizzate, elaborate, scambiate,…

CODIFICA DELL'INFORMAZIONE

- La stessa informazione si può rappresentare in modi differenti



- Stessa rappresentazione per informazioni differenti



SISTEMI DI CODIFICA

Sistema di codifica (o ***codifica***, o ***codice***)

- Usa un insieme di simboli di base (**alfabeto**)
- I simboli dell'alfabeto possono essere combinati ottenendo differenti **configurazioni** (o ***codici***, o ***stati***), distinguibili l'una dall'altra
- Associa ogni configurazione ad una particolare entità di informazione (la configurazione diventa un modo per rappresentarla)

SISTEMI DI CODIFICA: NUMERI INTERI (DECIMALI)

- Alfabeto
  - Cifre “0”, “1”, “2”, …, “9”
  - separatore decimale (“,”)
  - separatore delle migliaia (“.”)
  - Segni positivo (“+”) e negativo (“-”)
- Regole di composizione (sintassi)
  - Definiscono le combinazioni ben formate
    - 12.318,43
    - 12,318.43
- Codice (semantica)
  - Associano ad ogni configurazione un’entità di informazione
    - $12.318,43 = 1\times10^{4}+ 2\times10^{3}+ 3\times10^{2}+ 1\times10+ 1\times10^{0}+ 4\times10^{-1}+ 3\times10^{-2}$
- Lo stesso alfabeto può essere usato per codici diversi
  - $123,456 = 1\times10^{2}+ 2\times10+ 3\times10^{0}+ 4\times10^{-1}+ 5\times10^{-2} + 6\times10^{-3}$ [IT]
  - $123,456 = 1\times10^{5}+ 2\times10^{4}+ 3\times10^{3}+ 4\times10^{2}+ 5\times10^{1} + 6\times10^{0}$ [UK]

CODIFICA BINARIA

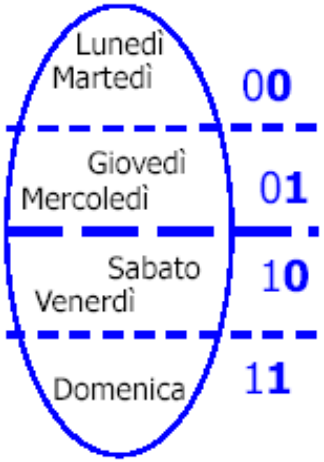
- Codifica binaria: usa un alfabeto di **2** simboli
- Utilizzata nei sistemi informatici
  - Si utilizza una grandezza fisica (luminosità, tensione elettrica, la corrente elettrica), per rappresentare informazione
  - Si divide in due intervalli l'insieme dei valori che la grandezza può assumere: ogni intervallo corrisponde ad un simbolo
- Solo 2 simboli al fine di ridurre la probabilità di errore
  - Tanto più simboli si devono distinguere e tanto meno la rivelazione sarà affidabile (gli intervalli della grandezza fisica saranno meno ampi)

CODIFICA BINARIA

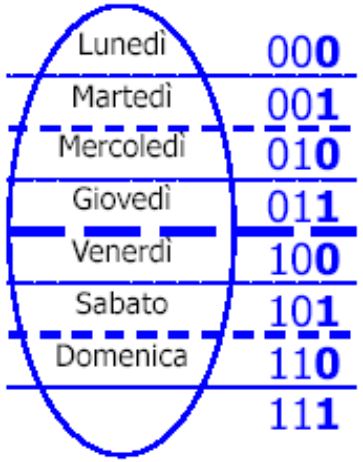
- **BIT (BInary digiT)**
  - unità elementare di informazione rappresentabile con dispositivi elettronici
  - con 1 bit si possono rappresentare 2 stati
    - 0/1, on/off, si/no
- Combinando più bit si può codificare un numero maggiore di stati
  - con 2 bit possono rappresentare 4 stati
  - con **K** bit si possono rappresentare $\mathbf{2^K}$ stati
- Quanti bit sono necessari per codificare N oggetti?
  - $N \leq 2^K \rightarrow K \geq \log_2 N \rightarrow \mathbf{K = \lceil \log_2 N \rceil}$

I GIORNI DELLA SETTIMANA IN BINARIO

1 bit
2 "gruppi"



2 bit
4 "gruppi"



3 bit
8 "gruppi"