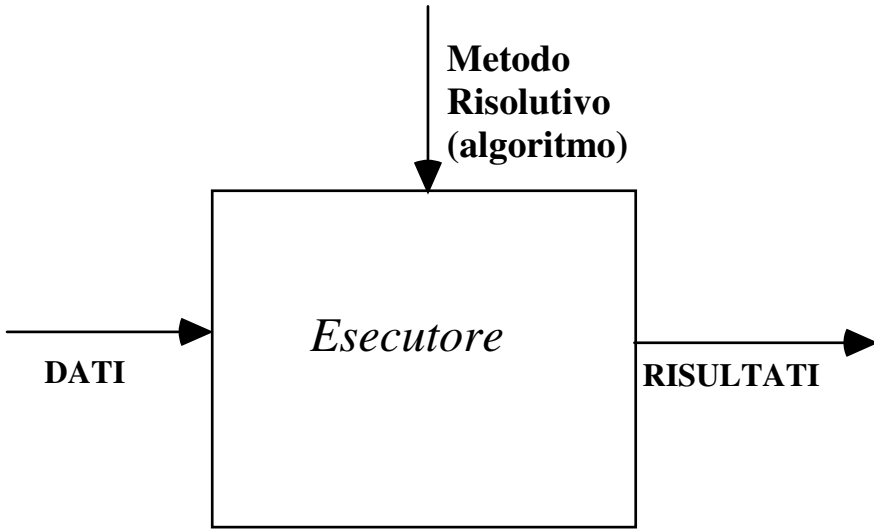
L'ELABORATORE ELETTRONICO

- Il calcolatore elettronico è uno strumento in grado di eseguire insiemi di *azioni* ("*mosse*") *elementari*
- le azioni vengono eseguite su oggetti (***dati***) per produrre altri oggetti (***risultati***)
- l'esecuzione di azioni viene richiesta all'elaboratore attraverso *frasi* scritte in un qualche *linguaggio* (*istruzioni*)

ALCUNE DOMANDE FONDAMENTALI

- **Quali istruzioni** esegue un elaboratore?
- **Quali problemi** può risolvere un elaboratore?
- ***Esistono problemi che un elaboratore non può risolvere?***
- **Che ruolo ha il linguaggio** di programmazione?

L'AUTOMA ESECUTORE

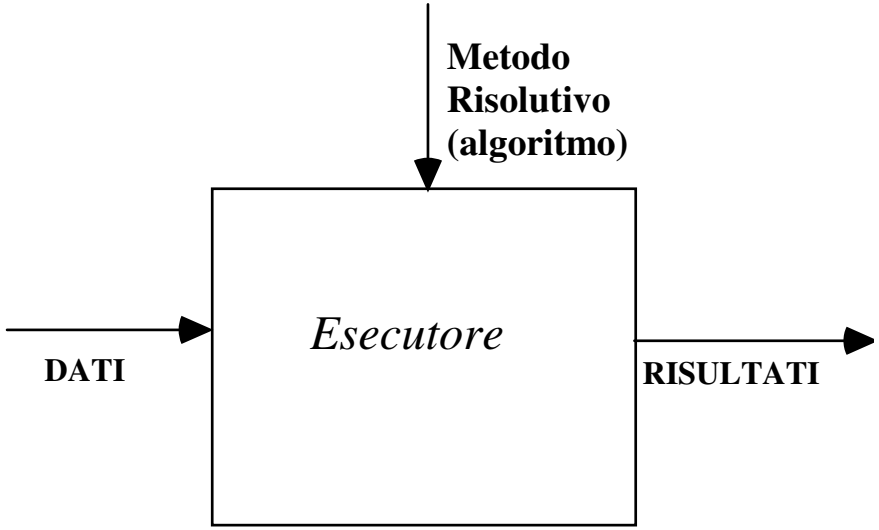


Un automa capace di **ricevere dall'esterno la descrizione dell'algoritmo richiesto**

cioè

capace di **interpretare un linguaggio** (***linguaggio macchina***)

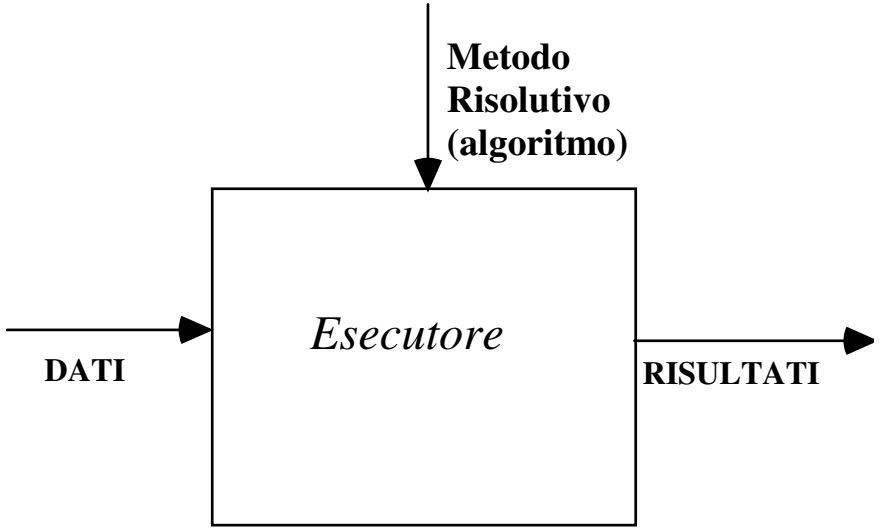
L'AUTOMA ESECUTORE



Vincolo di realizzabilità:

- se l'automa è fatto di parti, queste sono in numero finito
- ingresso e uscita devono essere denotabili attraverso un insieme finito di simboli.

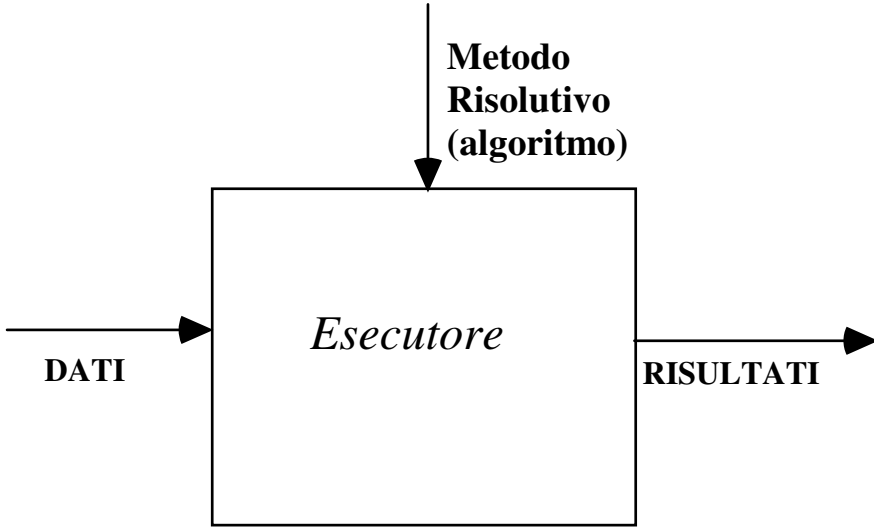
L'AUTOMA ESECUTORE



Realizzazione:

1) mediante **congegni meccanici**
   - macchina aritmetica (1649) di Blaise Pascal
   - macchina analitica di Charles Babbage (1792-1871)

L'AUTOMA ESECUTORE

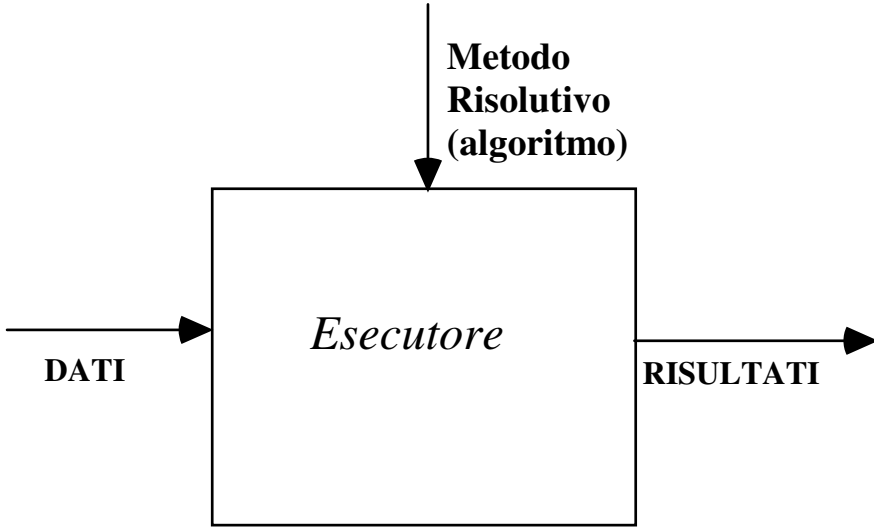


Realizzazione:

2) mediante **un modello matematico**

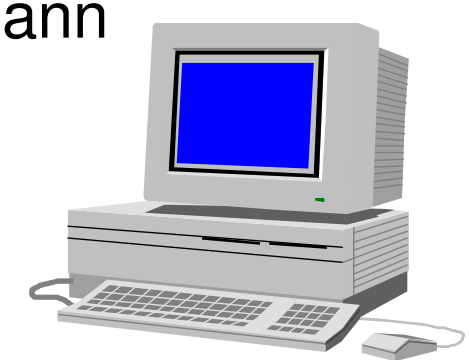
- funzionale (Hilbert, (1842-1943), Church, Kleene)
- **operazionale (Turing, 1912-1954)**
- sistemi di riscrittura (Post, Markov,..).

L'AUTOMA ESECUTORE



Realizzazione:

Mediante calcolatore elettronico basato sull'architettura di Von Neumann



PERCHÉ I MODELLI MATEMATICI

Macchine diverse potrebbero avere **diversa capacità di risolvere problemi**

Una macchina potrebbe essere "più potente" di un'altra
È NECESSARIO SAPERLO

Se neanche la macchina "più potente" riesce a risolvere un problema, **potrebbero esserci PROBLEMI NON RISOLUBILI**

GERARCHIA DI MACCHINE

- macchine combinatorie
- macchine (automi) a stati finiti
- macchina a stack
- **macchina di Turing**

TESI DI CHURCH-TURING
Non esiste alcun formalismo capace di risolvere una classe di problemi più ampia della Macchina di Turing

Alan Turing

- Definizione il più possibile precisa e generale del concetto di "procedura di calcolo”
- Alan Turing a partire dal 1930 iniziò ad analizzare le attività rilevanti che un essere umano faceva durante l’esecuzione di un processo di calcolo.
  - utilizza fogli di carta come memoria;
  - legge dati dal foglio di carta;
  - scrive dati a partire da altri dati, eventualmente cancellando dati precedenti;
  - decide cosa scrivere in base a ciò che è presente sul foglio e al proprio stato mentale.

La macchina di Turing (TM)

- La macchina di Turing (TM) è costituita da:
  - un **nastro**, che gioca il ruolo del dispositivo di memoria e viene supposto illimitato in entrambe le direzioni; il nastro è suddiviso in celle tutte uguali e ognuna di esse può contenere un simbolo.
  - una **testina** che permette di leggere e scrivere sulle celle del nastro; la testina esplora una cella alla volta, quella cioè che a un dato istante viene a trovarsi in sua corrispondenza. **Il simbolo letto dalla testina può essere cancellato e sostituito con un nuovo simbolo**.
  - un **dispositivo di controllo** che regola il movimento della testina rispetto al nastro, spostandola ogni volta di una cella verso sinistra o verso destra secondo i casi. **Il dispositivo di controllo è dotato di una memoria interna costituita da un certo numero finito di stati**.

- Il comportamento della TM dipende:
  - dal simbolo letto sul nastro mediante la testina;
  - dallo stato interno in cui si trova il dispositivo di controllo in un dato momento.

La Macchina di Turing

- In questo modo, egli inventò la cosiddetta **macchina di Turing** (che egli chiamava **A-machine**), che costituisce un modello di calcolatore molto semplice ma al tempo stesso sufficientemente potente per studiare le proprietà matematiche di ciò che è calcolabile.

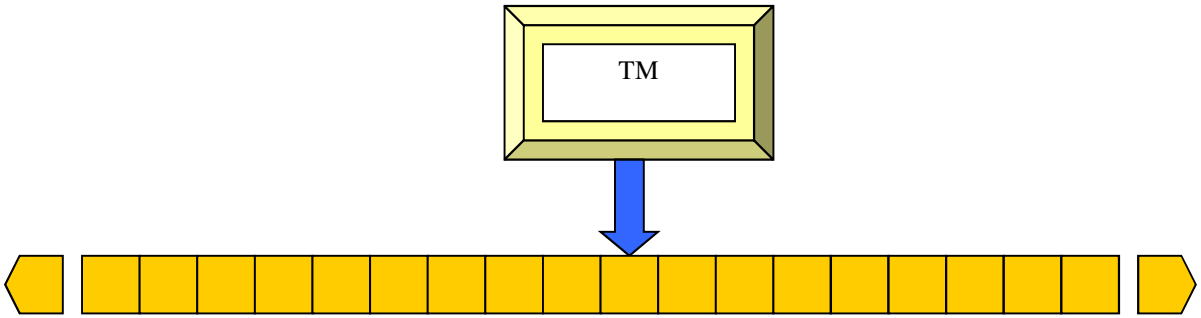
- Turing concepì **questo dispositivo come modello ideale di calcolatore** per studiarne le caratteristiche in termini teorici e non lo costruì fisicamente con dei dispositivi fisici. Si rese però conto che le azioni dell'uomo potevano essere facilmente sostituite da un meccanismo fisico.

LA MACCHINA DI TURING

La TM è una macchina capace di:

- *leggere un simbolo* dal nastro
- *scrivere sul nastro un simbolo*
- transitare in un nuovo stato interno specificato
- spostarsi sul nastro di una posizione in una certa direzione

Quando raggiunge lo stato HALT, la macchina si ferma



LA MACCHINA DI TURING E CPU REALI

In pratica:	corrisponde a:
• leggere/scrivere un simbolo dal/sul nastro	• lettura/scrittura memoria RAM/ROM
• transitare in un nuovo stato interno	• nuova configurazione dei registri della CPU
• spostarsi sul nastro di una (o più) posizioni	• scelta della cella di memoria su cui operare

RISOLVERE PROBLEMI CON LA T.M.

Risolvere un problema con la Turing Machine richiede quindi di:

- **definire** una opportuna ***rappresentazione dei dati iniziali sul nastro***
- **definire la** ***parte di controllo*** mediante delle funzioni in modo da **rendere disponibile sul nastro**, ***alla fine*****,** la ***rappresentazione della soluzione***

MACCHINE SPECIFICHE

- Una volta definita la parte di controllo, la Macchina di Turing è capace di risolvere un dato problema (risolubile)..... **ma così facendo, essa è *specifica di quel problema!***
- **Siamo circondati** da macchine specifiche:
  - calcolatrici
  - lavastoviglie,
  - videoregistratori, videogiochi
  - orologi, telecamere, ...

MACCHINE SPECIFICHE

Conviene fare macchine specifiche?

- sì, per usi particolari e mercati di massa…
- **no, se vogliamo una macchina di uso generale con cui risolvere ogni problema (risolubile)**

MACCHINE UNIVERSALI

- **Una volta definita la parte di controllo**, cioè **l'algoritmo necessario**, la Macchina di Turing è capace di risolvere un dato problema (risolubile)
- **Finora**, l'algoritmo era ***cablato nella macchina***
- **..e se invece fosse *sul nastro*, e la macchina se lo andasse a prendere?**

MACCHINA DI TURING UNIVERSALE

- **Una Macchina di Turing** la cui parte di controllo (cioè il cui algoritmo "cablato") consiste nel **leggere *dal nastro* una *descrizione dell'algoritmo* richiesto**
- **È una macchina UNIVERSALE: *senza modifiche alla sua struttura*, può essere *istruita per risolvere un qualunque problema (risolubile)***

- **Una macchina *programmabile***

MACCHINA DI TURING UNIVERSALE

Ma leggere *dal nastro* una *descrizione dell'algoritmo* richiesto richiede di:

- **saper descrivere tale algoritmo**
- **il che richiede un qualche *linguaggio***
- **e una macchina che lo *interpreti***

Dunque...

- **la Universal Turing Machine (UTM) è *l'interprete di un linguaggio!***

MACCHINA DI TURING UNIVERSALE

Una Universal Turing Machine (UTM)
***modella il concetto di elaboratore di uso generale* (“general purpose”)**

- Una macchina che *va a cercare le “istruzioni” da compiere*… (**fetch**)
- ... le interpreta… (**decode**)
- ... e le esegue (**execute**)

MACCHINA DI TURING UNIVERSALE E MACCHINA DI VON NEUMAN

- **Una Universal Turing Machine (UTM) è in grado di *elaborare***
  - prendendo *dati* e *algoritmo* **dal nastro**
  - e scrivendo **sul nastro** i *risultati*
- Dunque, **una UTM opera solo da/verso il nastro** (astrazione della memoria)**: non esiste il concetto di "*mondo esterno*"**
- **Ingresso/Uscita**

MACCHINA DI TURING UNIVERSALE E MACCHINA DI VON NEUMAN

Dunque,

- **la macchina di Von Neumann è modellata dalla UTM per ciò che attiene alla *computazione...***
- **ma prevede anche la dimensione dell' *interazione***

PROBLEMI RISOLUBILI E COMPUTABILITÀ

Secondo la Tesi di Church-Turing, **non esiste un formalismo "più potente" di TM**, ossia capace di risolvere una classe più ampia di problemi

Dunque...

se neanche la macchina di Turing riesce a risolvere un problema, **quel problema non è risolubile**

PROBLEMA RISOLUBILE

Un problema la cui soluzione **può essere espressa da una Macchina di Turing** o formalismo equivalente

QUALCHE DEFINIZIONE

FUNZIONE CARATTERISTICA DI UN PROBLEMA

- Dato un **problema P**,
  l'insieme X dei suoi dati di ingresso,
  l'insieme Y delle risposte corrette,
  si dice ***funzione caratteristica*** **del problema P** la funzione:

$$f_P: X \rightarrow Y$$

  che associa a ogni dato d'ingresso la corrispondente risposta corretta

QUALCHE DEFINIZIONE

FUNZIONE CARATTERISTICA DI UN PROBLEMA

- **Perché questo artificio?**
- Perché grazie a questa funzione, **decidere *se un problema è risolubile*** **equivale a chiedersi** ***se la funzione f_P è computabile***

D'ora in poi parleremo quindi solo di ***funzioni computabili***, sapendo che ciò equivale a parlare di ***problemi risolubili***

QUALCHE DEFINIZIONE

FUNZIONE COMPUTABILE

Una funzione f: A→B per la quale esiste una Macchina di Turing che

- data sul nastro una rappresentazione di **x∈ A**

dopo un numero finito di passi

- produce sul nastro una rappresentazione del risultato **f(x)∈ B**

FUNZIONI NON COMPUTABILI

È facile dimostrare che **esistono funzioni *definibili ma non computabili***

e, quindi, **problemi *non risolubili***

ESEMPIO:

Problema dell'*halt* della macchina di Turing

Dire se una data macchina di Turing T, con un generico ingresso X, si ferma oppure no

FUNZIONI NON COMPUTABILI

La funzione caratteristica f_H di questo problema può essere così definita:

$$f_H(m,x) = \begin{cases} \textbf{1, se m con ingresso x si ferma} \\ \textbf{0, se m con ingresso x non si ferma} \end{cases}$$

Si può dimostrare che **questa funzione è *definita* ma *non computabile***, in quanto tentare di calcolarla conduce a un assurdo.

Osserva che non è possibile semplicemente provare ad eseguire m con ingresso x ed aspettare: questo basterebbe per rispondere 1 (quando effettivamente l'esecuzione si ferma), ma quando rispondere 0? Anche dopo milioni di istruzioni eseguite, la terminazione potrebbe essere proprio l'effetto della prossima istruzione, come invece l'esecuzione potrebbe andare avanti all'infinito.

PROGRAMMAZIONE

L'attività con cui si predispone l'elaboratore a **eseguire un *particolare insieme di azioni* su *particolari dati***, allo scopo di *risolvere un problema*

PROBLEMI DA RISOLVERE

I problemi che siamo interessati a risolvere con l'elaboratore sono di ***natura molto varia***:

- Dati due numeri trovare il ***maggiore***
- Dato un elenco di nomi e relativi numeri di telefono ***trovare*** il numero di telefono di una determinata persona
- Dati a e b, ***risolvere l'equazione*** ax+b=0
- Stabilire se una parola viene ***alfabeticamente*** prima di un'altra
- ***Somma*** di due numeri interi
- Scrivere tutti gli n per cui l'equazione: $X^n + Y^n = Z^n$ ha soluzioni intere positive ("Non esistono soluzioni intere positive se N > 2" - ***Teorema di Fermat-Wiles 1995***)
- ***Ordinare*** una lista di elementi
- Calcolare il ***massimo comune divisore*** fra due numeri dati
- Calcolare il ***massimo*** in un insieme
- ……

RISOLUZIONE DI PROBLEMI

- La descrizione del problema non fornisce (in generale) un metodo per risolverlo
  - Affinché un problema sia risolvibile è necessario che la sua definizione sia chiara e completa

- ***Non tutti*** i problemi sono risolvibili attraverso l'uso del calcolatore. Esistono classi di problemi per le quali la soluzione automatica non è proponibile. Ad esempio:
  - se il problema presenta infinite soluzioni
  - per alcuni dei problemi ***non è stato trovato un metodo risolutivo***
  - per alcuni problemi è stato dimostrato che ***non esiste un metodo risolutivo automatizzabile (funzioni non computabili)***

RISOLUZIONE DI PROBLEMI

- Noi ci concentreremo sui problemi che, ragionevolmente, ***ammettono un metodo risolutivo***

  ➡ ***funzioni calcolabili***

- Uno degli obiettivi del corso è presentare le ***tecnologie*** e le ***metodologie di programmazione***
  - **Tecnologie**: strumenti per lo sviluppo di programmi
  - **Metodologie**: metodi per l'utilizzo corretto ed efficace delle tecnologie di programmazione

RISOLUZIONE DI PROBLEMI

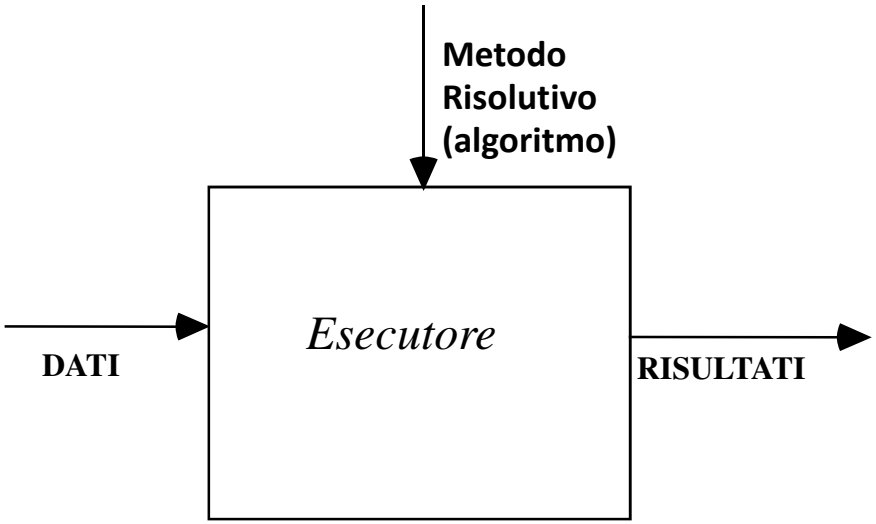
La risoluzione di un problema è il processo che dato un problema e individuato un opportuno metodo risolutivo, trasforma i dati iniziali nei corrispondenti risultati finali

Affinché la risoluzione di un problema possa essere realizzata attraverso l'uso del calcolatore, tale processo deve poter essere definito come *sequenza di azioni elementari*

ALGORITMO

Un algoritmo è una sequenza **finita** di mosse che risolve ***in un tempo finito*** una *classe* di problemi

L'esecuzione delle azioni *nell'ordine specificato dall'algoritmo* consente di ottenere, a partire dai dati di ingresso, i risultati che risolvono il problema



ESECUTORE
una ***macchina astratta***
capace di ***eseguire le azioni***
specificate dall'algoritmo

ALGORITMI: PROPRIETÀ

- **Eseguibilità**: ogni azione deve essere *eseguibile* dall'esecutore *in un tempo finito.*

- **Non-ambiguità**: ogni azione deve essere *univocamente interpretabile* dall'esecutore.

- **Finitezza**: il numero totale di azioni da eseguire, per ogni insieme di dati di ingresso, deve essere finito

ALGORITMI: PROPRIETÀ (2)

Quindi, l'algoritmo deve:

- essere *applicabile a qualsiasi insieme di dati di ingresso* appartenenti al **dominio di definizione** dell'algoritmo.
- essere costituito da operazioni appartenenti ad un determinato **insieme di operazioni fondamentali.**
- essere costituito da **regole non ambigue**, cioè interpretabili in modo **univoco** qualunque sia l'esecutore (persona o "macchina") che le legge.

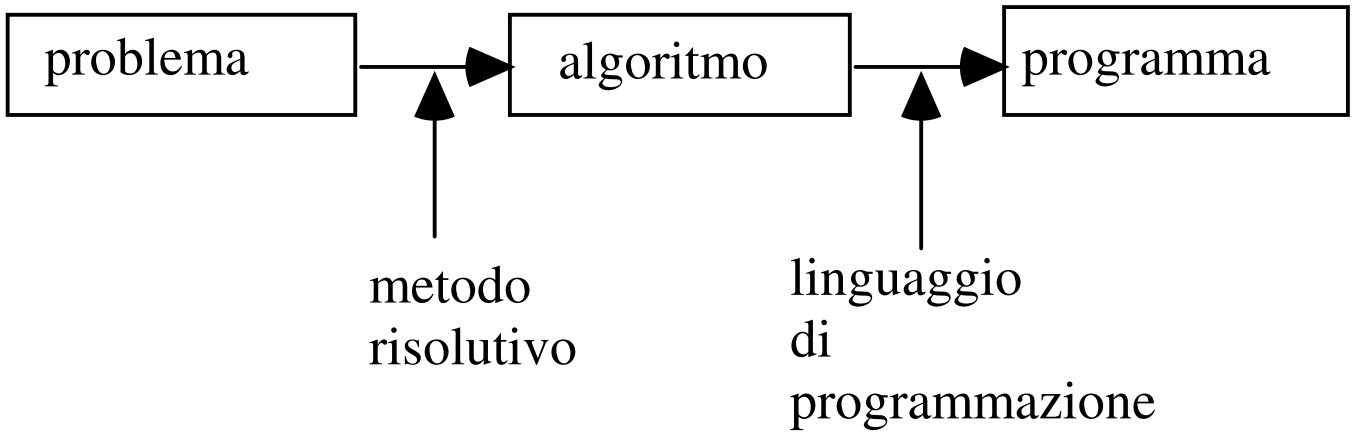
ALGORITMI E PROGRAMMI

- Ogni elaboratore è una macchina in grado di eseguire azioni elementari su oggetti detti **DATI.**
- L'esecuzione delle azioni è richiesta all'elaboratore tramite comandi elementari chiamati **ISTRUZIONI** espresse attraverso un opportuno formalismo: il **LINGUAGGIO di PROGRAMMAZIONE**
- La formulazione testuale di un algoritmo in un linguaggio comprensibile a un elaboratore è detta **PROGRAMMA.**
- **Un programma è un testo** scritto in accordo alla **sintassi** e alla **semantica** di un linguaggio di programmazione.

ALGORITMO & PROGRAMMA

Passi per la risoluzione di un problema:

- individuazione di un procedimento risolutivo
- scomposizione del procedimento in un insieme ordinato di azioni ➡ **ALGORITMO**
- rappresentazione dei dati e dell'algoritmo attraverso un formalismo comprensibile dal calcolatore ➡ **LINGUAGGIO DI PROGRAMMAZIONE**



UN ESEMPIO DI PROGRAMMA (in linguaggio C)

```
main()  {
 int A, B;
 printf("Immettere due numeri: ");
 scanf("%d %d", &A, &B);
 printf("Somma: %d\n", A+B);
}
```

ALGORITMI: ESEMPI

- **Soluzione dell'equazione ax+b=0**
  - *leggi* i valori di *a* e *b*
  - *calcola -b*
  - *dividi* quello che hai ottenuto per *a* e chiama *x* il risultato
  - *stampa x*

- **Calcolo del massimo di un insieme**
  - *Leggi* i valori dell'insieme *Ins*
  - *Scegli* un elemento come massimo provvisorio *max*
  - *Per ogni elemento i dell'insieme*: se *i>max* eleggi *i* come nuovo massimo provvisorio *max*
  - Il risultato è *max*

NOTA: si utilizzano **VARIABILI**, ossia nomi simbolici usati nell'algoritmo per denotare dati

ALGORITMI: ESEMPI

Stabilire se una parola P viene alfabeticamente prima di una parola Qdi

- **leggi** *P* e *Q*
- **ripeti quanto segue:**
  - **se** prima lettera di *P* < prima lettera di *Q*
  - **allora** scrivi vero
  - **altrimenti se** prima lettera *P* > prima lettera di *Q*
  - **allora** scrivi falso
  - **altrimenti** (le prime lettere sono =)
  - togli da *P* e *Q* la prima lettera
- **fino** a quando hai trovato le prime lettere di *P* e *Q* diverse

- *Nota: funziona solo con parole P e Q di uguale lunghezza e con parole diverse*
- *Esercizio proposto: rilassare tali condizioni*

ALGORITMI: ESEMPI

- **Somma degli elementi dispari di un insieme:** Detto *INS* l'insieme di elementi (letto) considero un elemento *X* di *INS* alla volta senza ripetizioni. Se *X* è dispari, sommo *X* a un valore *S* inizialmente posto uguale a *0*. Se *X* è pari non compio alcuna azione. Termino quando ho considerato tutti gli elementi di *INS* e stampo il risultato contenuto in *S*.

- **Somma di due numeri: X + Y = Z** (ovvero incrementare il valore di Z, inizialmente posto uguale a X per Y volte) con un esecutore che non sa calcolare espressioni direttamente, ma sa solo decrementare, incrementare e confrontare valori.
  - Leggi i dati e inseriscili in *X* e *Y*
  - poni *Z = X*
  - poni *U = 0*
  - finché *U* è diverso da *Y*
    - incrementa *Z* (*Z=Z+1*)
    - incrementa *U* (*U=U+1*)
  - Stampa il risultato Z

ALGORITMI EQUIVALENTI

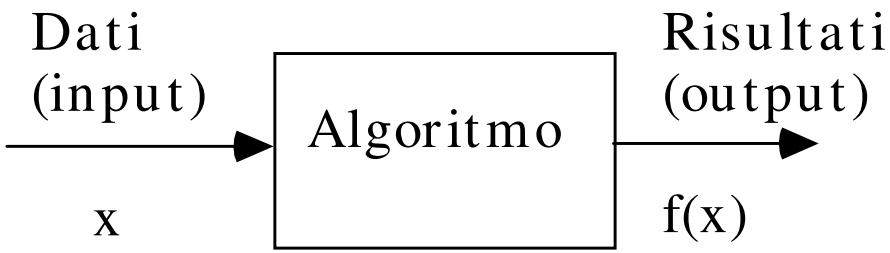
Due algoritmi si dicono ***equivalenti*** quando:

- hanno lo stesso **dominio di ingresso**
- hanno lo stesso **dominio di uscita**
- in corrispondenza degli **stessi valori del dominio di ingresso *producono gli stessi valori* nel dominio di uscita.**

ALGORITMI EQUIVALENTI (2)

Due algoritmi ***equivalenti***

- forniscono lo **stesso risultato**
- ma possono avere ***diversa efficienza***
- e possono essere **profondamente diversi!**



ALGORITMI EQUIVALENTI (3)

ESEMPIO: calcolo del M.C.D. fra due interi M, N

- **Algoritmo 1**
  - Leggi *M* e *N*
  - Calcola l'insieme *A* dei divisori di *M*
  - Calcola l'insieme *B* dei divisori di *N*
  - Calcola l'insieme *C* dei divisori comuni = $A \cap B$
  - Calcola il massimo Max dell'insieme *C*
  - Stampa *Max*

- **Algoritmo 2 (di Euclide)**

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

ALGORITMI EQUIVALENTI (4)

ESEMPIO: calcolo del M.C.D. fra due interi M, N

Algoritmo 2 (di Euclide)

- Leggi *M* e *N*
- Finché $M \neq N$:
  - se $M>N$, sostituisci al corrente valore di *M* il valore *M-N*
  - altrimenti sostituisci al corrente valore di *N* il valore *N-M*

- Il Massimo Comune Divisore è il valore finale ottenuto quando M = N.
- Stampa M (o N).

Esempio:
MCD fra 18 e 12 -> (18-12) e 12 -> 6 e 12 -> 6 e 6 OK risultato 6

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