

# L'ELABORATORE ELETTRONICO

---

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

# 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



# 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?

# 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 (problema di Fermat)*
  - *Ordinare una lista di elementi*
  - *Calcolare il massimo comun 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 molti anni (problema di Fermat)
  - per alcuni problemi è stato dimostrato che **non esiste** un metodo risolutivo automatizzabile

# 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

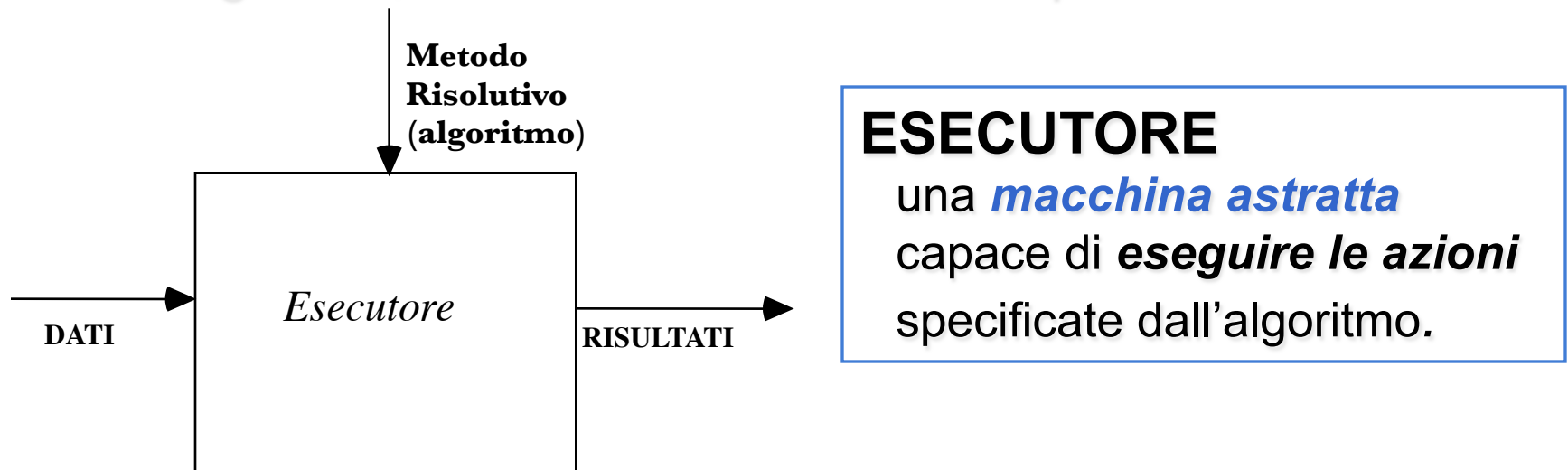
---

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





# ALGORITMI: PROPRIETÀ

---

- **Eseguibilità**: ogni azione dev'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**.

# PROGRAMMA

---

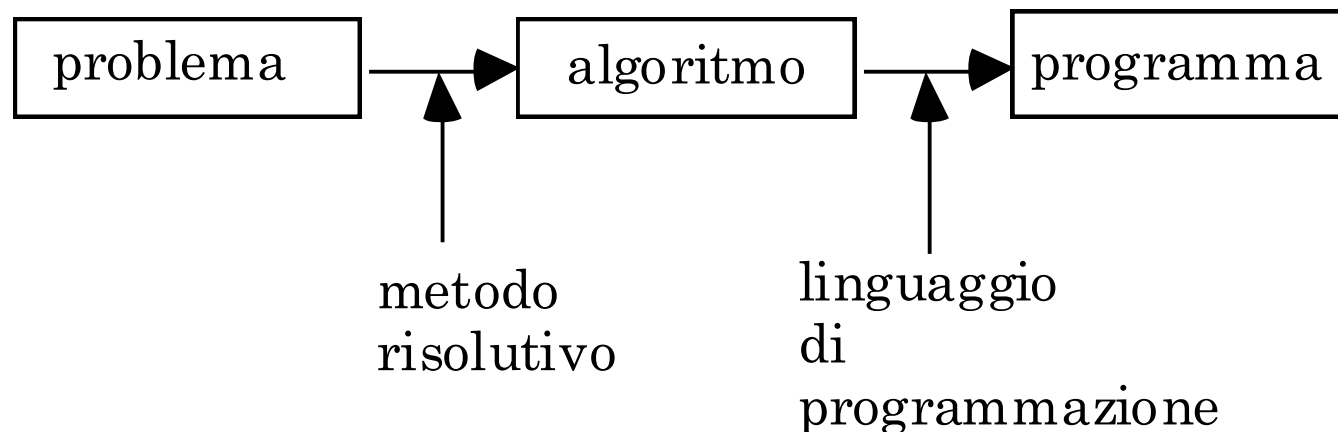
Un **programma** è un **testo** scritto in accordo alla **sintassi** e alla **semantica** di un linguaggio di programmazione.

Un ***programma*** è la **formulazione testuale**, in un certo linguaggio di programmazione, di un **algoritmo** che risolve un dato *problema*.

# 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:**

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



# ALGORITMI: ESEMPI

---

- **Somma degli elementi dispari di un insieme**

- Detto INS l'insieme di elementi 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.

- **Somma di due numeri  $X$  e  $Y$**

- Incrementare il valore di  $Z$ , inizialmente posto uguale a  $X$  per  $Y$  volte.
  - poni  $Z = X$
  - poni  $U = 0$
  - finché  $U$  è diverso da  $Y$
- incrementa  $Z$  ( $Z := Z + 1$ )
- incrementa  $U$  ( $U := U + 1$ )
- Il risultato è  $Z$

*Si supponga di avere a disposizione come mossa elementare solo l'incremento e non la somma tra interi*

# ALGORITMI EQUIVALENTI

---

Due algoritmi si dicono *equivalenti* quando:

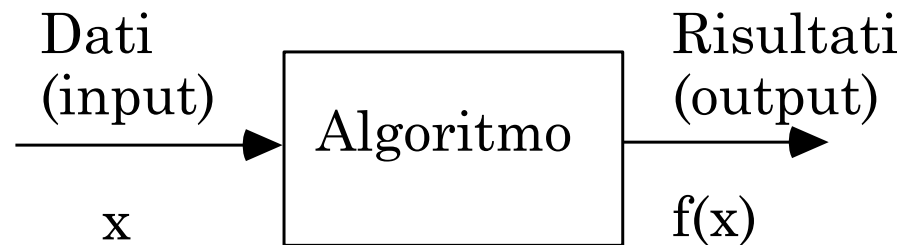
- hanno lo stesso **dominio di ingresso**;
- hanno lo stesso **dominio di uscita**;
- in corrispondenza degli **stessi valori del dominio di ingresso** ***producono gli stessi valori*** nel dominio di uscita.

# 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**

- Calcola l'insieme A dei divisori di M
- Calcola l'insieme B dei divisori di N
- Calcola l'insieme C dei divisori comuni =  $A \cap B$
- Il risultato è il massimo dell'insieme C

- **Algoritmo 2 (di Euclide)**

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

# ALGORITMI EQUIVALENTI (4)

---

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

- **Algoritmo 2 (di Euclide)**

Finché  $M \neq N$ :

- 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

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

## **ALGORITMI EQUIVALENTI (5)**

---

**Gli algoritmi 1 e 2 sono equivalenti...  
...ma hanno efficienza ben diversa!!**

# IL PROBLEMA DEL PROGETTO

---

- La descrizione del problema, in genere, non indica direttamente il modo per ottenere il risultato voluto (il procedimento risolutivo)
- Occorrono *metodologie* per affrontare il problema del progetto in modo sistematico

# IL PROBLEMA DEL PROGETTO

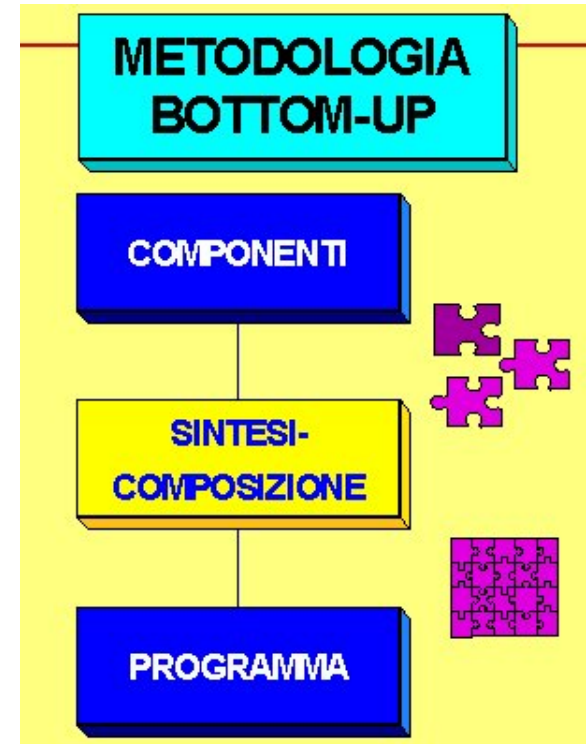
---

- **Due dimensioni progettuali:**
  - Programmazione in piccolo  
*(in-the-small)*
  - Programmazione in grande  
*(in-the-large)*
- procedere per livelli di astrazione
- garantire al programma *strutturazione e modularità*



# METODOLOGIE DI PROGETTO

---



# METODOLOGIA TOP-DOWN

---

Procede per **decomposizione** del problema in sotto-problemi, per ***passi di raffinamento successivi***

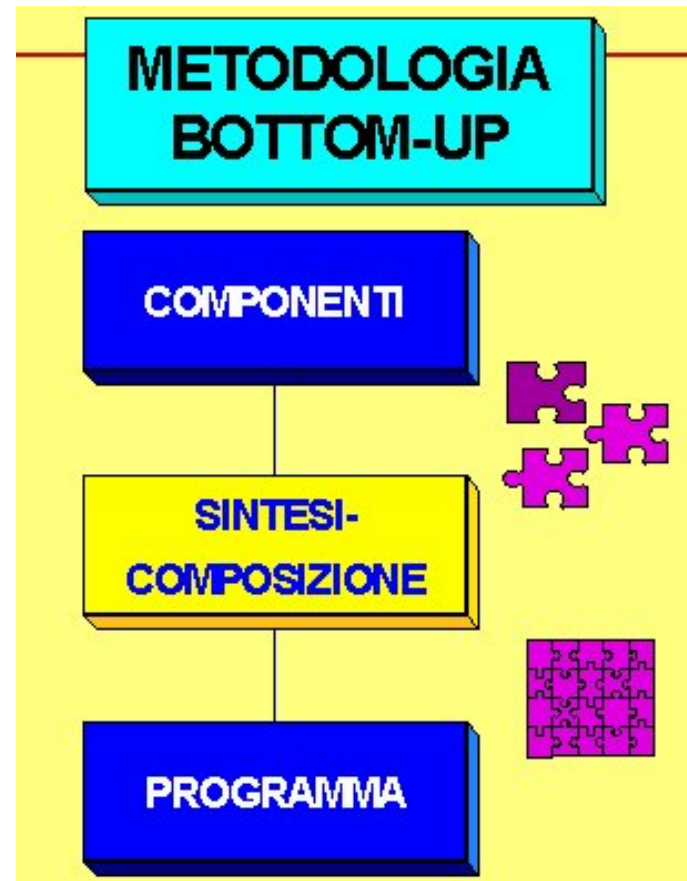
- Si scompone il problema in sottoproblemi
- Si risolve ciascun sottoproblema con lo stesso metodo, fino a giungere a sottoproblemi risolvibili con mosse elementari



# METODOLOGIA BOTTOM-UP

---

Procede per *composizione di componenti e funzionalità elementari*, fino alla **sintesi** dell'intero algoritmo (“*dal dettaglio all'astratto*”)



# IL PROBLEMA DEL PROGETTO

---

Dunque, dato un problema ***non si deve iniziare subito a scrivere il programma***

- così si scrivono *a fatica* programmi semplici
- spesso sono errati, e non si sa perché
- nessuno capisce cosa è stato fatto (dopo un po', nemmeno l'autore...)
- è necessario valutare la soluzione migliore tra tante
- è necessario scrivere programmi facilmente modificabili/estendibili

# IL PROBLEMA DEL PROGETTO

---

- La specifica della *soluzione* e la fase di *codifica* sono concettualmente distinte
- e tali devono restare anche in pratica!

# UN ESEMPIO

---

## Problema:

“Data una temperatura espressa in gradi Celsius, calcolare il corrispondente valore espresso in gradi Fahrenheit”

## Approccio:

- si parte dal **problema** e dalle **proprietà** note *sul dominio dei dati*

# UN ESEMPIO

---

## **Problema:**

“Data una temperatura espressa in gradi Celsius, calcolare il corrispondente valore espresso in gradi Fahrenheit”

## **Specifica della soluzione:**

$$c * 9/5 = f - 32$$

$$c = (f - 32) * 5/9 \text{ oppure } f = 32 + c * 9/5$$

# UN ESEMPIO

---

## L'Algoritmo corrispondente:

- Dato **c**
- calcolare **f** sfruttando la relazione

$$f = 32 + c * 9/5$$

## SOLO A QUESTO PUNTO

- si sceglie un linguaggio
- si *codifica* l'algoritmo in tale linguaggio