# DAL PROBLEMA ALL'ALGORITMO

**Ing. Daniele Corti**




www.ingdanielecorti.it



Ver.1.0

# PREREQUISITI

- ✓ Linguaggi.
- ✓ Risolvere problemi di base nel campo matematico e fisico.

# OBIETTIVI

- ✓ Identificare un problema risolubile da un calcolatore automatico.
- ✓ Individuare le istruzioni adatte ad essere comprese dall'esecutore.
- ✓ Individuare lo stato iniziale e finale del processo risolutivo.
- ✓ Descrivere con sequenza di istruzioni non ambigue la risoluzione di un problema.
- ✓ Utilizzare le variabili in operazioni di assegnamento.
- ✓ Distinguere tra diversi tipi di strategie risolutive quella più efficiente (NO).

# ARGOMENTI

- ✓ Il problema.
- ✓ Risoluzione di un problema.
- ✓ Fasi per la risoluzione di un problema con il computer.
- ✓ Analisi.
- ✓ Progettazione.
- ✓ Realizzazione.
- ✓ Modello.
- ✓ Processi risolutivi.
- ✓ Processi risolutivi deterministici.
- ✓ Definizione di algoritmo.
- ✓ Caratteristiche dell'algoritmo.
- ✓ Stesura di un algoritmo.
- ✓ Variabili.
- ✓ L'operazione di assegnazione.

# CAP 2 – DAL PROBLEMA ALL'ALGORITMO

## IL PROBLEMA

Preparare una torta, risolvere un cruciverba, sommare 10 numeri, ordinare alfabeticamente dei nomi, prelevare contanti con il Bancomat, etc., sono alcuni esempi di **problemi**.
Alcuni possono apparire più semplici, altri più complessi, altri assurdi e altri ancora irrisolvibili.
Un problema, qualsiasi sia la sua natura (matematica, scientifica, informatica, umana, etc,) è uno stato di difficoltà o malessere che non permette di raggiungere un determinato obiettivo o soddisfare un certo bisogno o sviluppare delle idee.

**Un PROBLEMA è un quesito che attende una risposta, detta SOLUZIONE.**

Non sempre è facile trovare la soluzione di un problema.
In alcuni casi crediamo di averla trovata, ma dopo un'attenta **analisi** ci rendiamo conto che non fornisce i risultati che ci si aspettava o che tali risultati non sono coerenti con i dati di partenza.
Cerchiamo, allora, un'altra soluzione, spesso per **tentativi**. Se un tentativo non ha successo, si riprova con un altro, magari sfruttando i punti validi della soluzione precedente errata.

Il lavoro che svolgiamo per ricercare la soluzione di un problema si chiama **PROCESSO RISOLUTIVO**.

In questo capitolo vedremo come affrontare i problemi costruendo delle tecniche risolutive che portino alla risoluzione del problema attraverso l'uso dei mezzi informatici al fine di rendere automatico il processo stesso (**programmi**).

Ci poniamo, quindi, come **obiettivo** la risoluzione di un **problema**. Per poterlo risolvere occorre mettere in atto una **strategia**.


NB Tra tutte le possibili strategie che si possono mettere in atto, occorre prendere in esame quella che rende minimo un certo fattore (**ottimizzata del problema**). Per esempio, scegliere il mezzo di

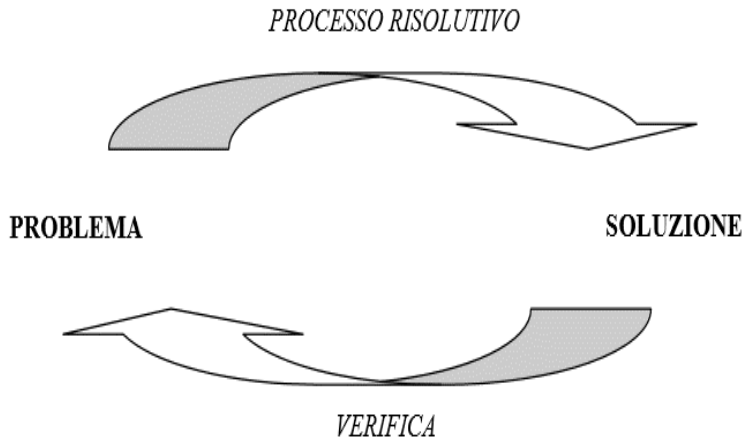
trasporto più economico per andare a Roma oppure quello con il quale si impiega meno tempo, o ancora quello che ci fa percorrere tragitti non autostradali (a pagamento).

# RISOLUZIONE DI UN PROBLEMA – PROBLEM SOLVING

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.

Dallo **stato iniziale**, in cui sono noti i dati del problema, si perviene a uno stato finale caratterizzato da una migliore sistemazione dei fatti già noti o dalla scoperta di nuovi fatti che completano il quadro in modo da renderlo logico e soddisfacente.

La soluzione viene poi sottoposta a verifica secondo una serie di **criteri**. La verifica deve individuare eventuali errori contenuti nel processo risolutivo al fine di ricercare il modo di eliminarli.




Il **problem solving** (soluzione di problemi) è l'insieme dei metodi formali per definire e trovare al soluzioni a problemi più o meno complessi, in diversi ambiti e situazioni.

Il problem solving non è soltanto un metodo usato dall'uomo per risolvere tutti i tipi di problemi (economici, statistici, giuridici, etc.) ma anche una tecnica utilizzata per **sviluppare il software**.

Per esempio, vediamo le fasi per il seguente problem solving:

- **Problema**: prelevare contanti in banca.
- **Analisi**: si possono prelevare contanti in diversi modi: sportello bancomat, assegno, prestito, finanziamento, etc.
- **Soluzioni**: se si decide per il bancomat, occorre descrivere i passi operativi e le azioni per prelevare con il bancomat.
- **Elaborazione**: vengono eseguite le azioni.
- **Risultati**: i contanti.

# FASI PER LA RISOLUZIONE DI UN PROBLEMA CON IL COMPUTER

Fra tutti i possibili problemi da risolvere noi siamo interessati ai problemi risolvibili da una macchina, ovvero da un elaboratore automatico, problemi la cui procedura risolutiva contiene istruzioni comprensibili (non ambigue) da parte del calcolatore.

Lo sviluppo del software per un calcolatore (macchina esecutrice automatica) è un'attività di "risoluzione dei problemi" secondo il metodo del problem solving.

La risoluzione di un problema, in uno specifico ambito reale, attraverso l'utilizzo di un calcolatore, viene realizzata attraverso le seguenti fasi distinte e successive:

1. **ANALISI DEL PROBLEMA**

   **Serve a formalizzare il problema.**

   1.1. **Interpretazione**: individuazione delle caratteristiche fondamentali e degli elementi che entrano in gioco. In questa fase occorre:

      1.1.1. Interpretare il problema da risolvere, focalizzando gli obiettivi.

      1.1.2. Individuare i dati espliciti e impliciti:

      - I **dati noti** (**Input**) e
      - I **dati da trovare**, o **risultati** (**Output**).

      1.1.3. Individuare le risorse a disposizione:

      - Logiche (tabelle, schemi logici, …),
      - Fisiche o hardware (calcolatori, Personal Computer, …).

      1.1.4. Individuare le regole e la soluzione da adottare, ovvero le procedura da utilizzare per passare dai dati noti ai risultati (da IN ad OUT).

      1.1.5. Individuare gli aspetti più importanti.

      1.1.6. Eliminare i dettagli inutili ed ambigui.

   1.2. **Modellizzazione** della realtà da studiare. Dopo aver analizzato il problema da risolvere, si **formalizza** il problema attraverso una rappresentazione semplificata della realtà, che ne evidenzia solo gli aspetti più importanti per la risoluzione del problema ed, eliminando ambiguità. Si costruisce quindi un **modello**.

   1.3. **Stesura del processo risolutivo** in un linguaggio naturale, un linguaggio più simile a quello usato dall'uomo, che porta dai dati al risultato attraverso una sequenza di **azioni**.

2. **PROGETTAZIONE**

2.1. **Stesura dell'algoritmo**. In questa fase il processo risolutivo scritto in un linguaggio naturale viene tradotto in un linguaggio ancora naturale ma **formale**: l'**ALGORITMO**.

2.2. **Esecuzione e testing dell'algoritmo**. L'algoritmo viene eseguito e testato al fine di individuare eventuali errori che portino a risultati errati.

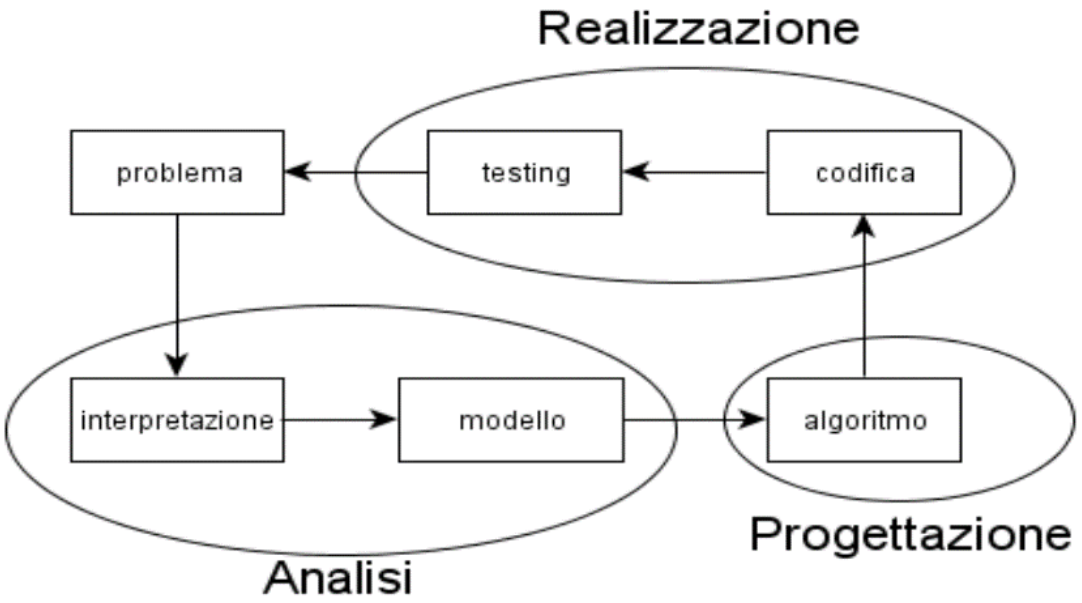
**3. REALIZZAZIONE**

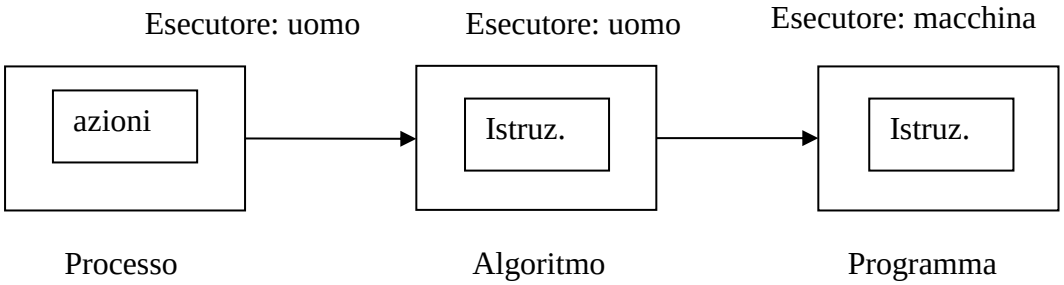
3.1. **Stesura del programma**. Implementazione (**codifica)** dell'algoritmo nel programma, scritto in un linguaggio formale di **programmazione ad alto livello**.

3.2. **Esecuzione e testing del programma**. Il programma viene eseguito e testato al fine di individuare eventuali errori che portino a risultati errati. Il programma viene tradotto in un linguaggio, linguaggio macchina, comprensibile dal computer.

3.3. **Comunicazione dei risultati**.

3.4. **Documentazione, manutenzione e aggiornamento del programma**.

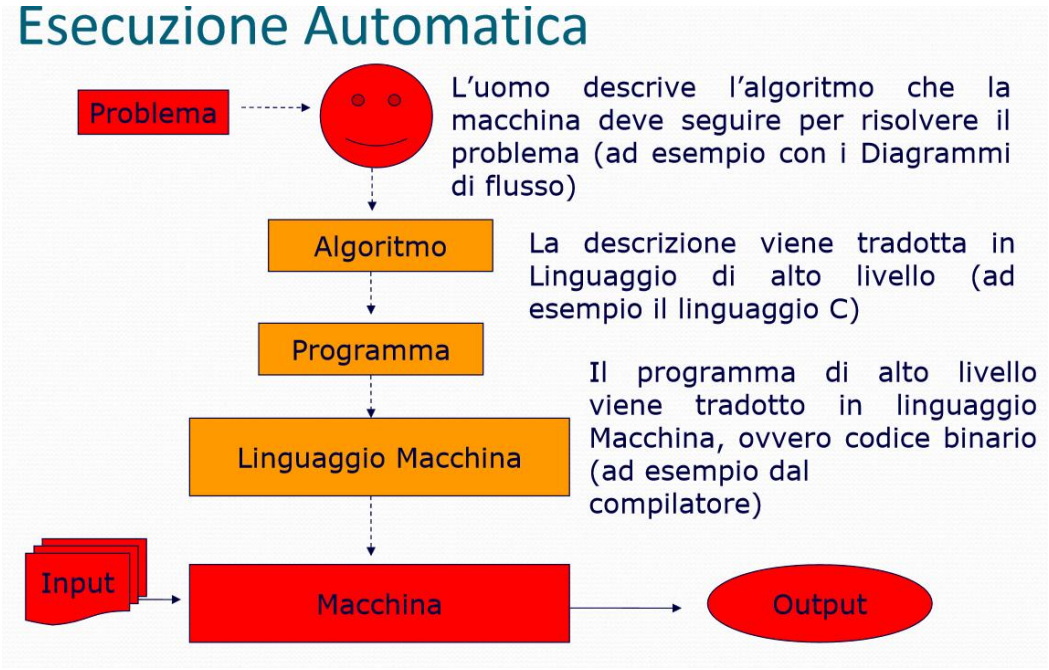



Come si vede, per risolvere un problema, dopo l'analisi e la formalizzazione del problema, il programmatore deve individuare i dati disponibili e la tecnica risolutiva stendendo il processo risolutivo in un linguaggio naturale (**Processo**). I processo sarà costituito da una sequenza di **azioni** elementari comprensibili all'esecutore (uomo). Ma la tecnica adottata non è ancora formale e

comprensibili ad una macchina, pertanto, viene successivamente costruita la risoluzione del problema attraverso la stesura della procedura risolutiva e formale (l'**Algoritmo**) attraverso l'utilizzo di schemi grafici (diagrammi a blocchi, **flowchart**). Infine, traduce l'algoritmo in un **Programma** scritto in un linguaggio informatico come il C++.

Naturalmente, un programmatore esperto potrebbe saltare la fase di realizzazione del Processo e addirittura quella dell'Algoritmo e realizzare il Programma direttamente in un linguaggio ad alto livello.




# MODELLO

È una rappresentazione semplificata della realtà osservata. Un sistema può essere deterministico (lo stato del sistema in un determinato istante può essere derivato dalle condizioni in cui si trovava all'istante precedente) o stocastico (casuale, quando non si può prevedere il comportamento di un sistema in presenza di determinate condizioni iniziali), chiuso (non ha interazioni con l'ambiente esterno nel quale è inserito) o aperto, reale (quando si fa riferimento a oggetti fisici che interagiscono fra loro) o astratti (quando si vogliono studiare strutture organizzate, oppure logiche o matematiche). Lo studio della realtà da descrivere include anche la determinazione dei dati (dati di input) che consentono la risoluzione del problema. Risolvere un problema, attraverso una tecnica risolutiva, significa elaborare i dati al fine di ottenere il risultato (dati di output).

## VARIABILI E COSTANTI E AZIONI

I nomi che si danno alle variabili e alle costanti all'interno di un modello si dicono **identificatori**.
Le variabili sono dati che assumono valori che possono cambiare durante il processo risolutivo. Le costanti, viceversa, rimangono invariate per tuta la durata del processo.
Le azioni sono le operazioni da applicare ai dati al fine di ottenere i risultati desiderati.

# PROCESSO RISOLUTIVO

Insieme formato dalle operazioni da svolgere in sequenza e dai dati che vengono elaborati durante queste operazioni per svolgere il compito assegnato. Le caratteristiche fondamentali di un processo sono:

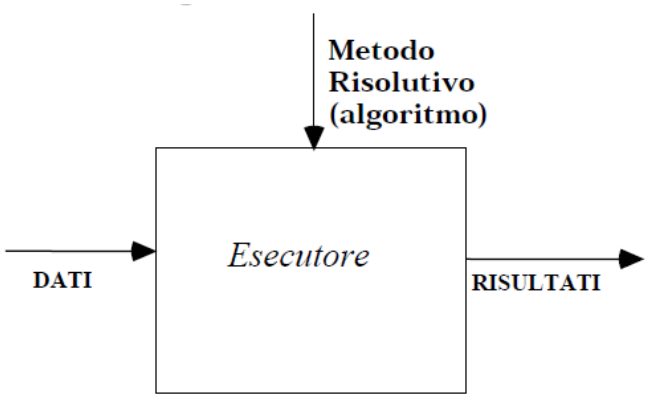
- Il processo evolve nel tempo.
- Il processo è sequenziale.

Nel processo risolutivo sono coinvolte tre figure:

- Il **risolutore**, o **programmatore**, è colui che progetta il processo risolutivo per risolvere il problema e, successivamente l'algoritmo. Il risolutore, nell'indicare la successione delle operazioni da eseguire, deve sapere quale sarà l'esecutore.
- L'**esecutore**, è c**olui che esegue il processo risolutivo**. L'esecutore, ovvero la "macchina astratta", può essere l'uomo o il computer.
- Il **cliente** colui che ha il problema.

Per risolvere un problema occorre individuare questi elementi:

- Dati dello stato iniziale (input), quelli forniti dall'esterno per poter risolvere il problema.
- Metodo risolutivo, che fornisce la soluzione al problema.
- Risultati dello stato finale (output), ovvero quelli da comunicare all'esterno.



NB Un'**azione** è un evento che si compie in un intervallo di tempo finito e che produce un risultato, un effetto, previsto e ben determinato. Per descrivere le azioni è necessario disporre di un linguaggio o di una notazione formale (flow-chart, pseudo-codice). Le descrizioni delle azioni sono dette **istruzioni**. Le istruzioni possono essere elementari o composte, un'istruzione elementare è un'istruzione che non può essere scomposta in istruzioni più semplici. Con il termine esecutore si indica colui che esegue l'algoritmo o meglio le istruzioni indicate in esso. Un **processo** non è altro che un **algoritmo in esecuzione**.

NB Il procedimento risolutivo deve essere tale per tutti i possibili valori assunti dai dati: i dati possono essere espressi in modo **parametrico.**

## ESEMPIO 1 procedura per la preparazione di un dolce

ANALISI

Abbiamo a disposizione una cucina, una serie di ingredienti e di utensili (dati d'ingresso), e attraverso la ricetta possiamo giungere alla soluzione del problema.

RISOLUTORE

Compilatore della ricetta.

ESECUTORE

Una persona in grado di eseguire alcune semplici azioni elementari (pesare, mescolare, etc.).

PASSO

Esecuzione di un'azione elementare.

DATI

INGRESSO: Ingredienti, utensili.

USCITA: Dolce.

PROCESSO

La ricetta è una sequenza di azioni, descrizione dettagliata di una successione finita di semplici azioni che ogni persona è in grado di compiere.

## ESEMPIO 2

**PROBLEMA: Copia di un file dal DVD-ROM all'unità di massa.**

**DATI IN**: DVD-ROM da inserire, file da copiare.

**DATI OUT**: memorizzazione del file in una determinata cartella del PC

**PROCEDURA RISOLUTIVA**

Apri il cassettino del driver dvd-rom.

Inserisci il dvd-rom.

Chiudi il cassettino del driver dvd-rom.

Apri Esplora Risorse.

Doppio Clic sulla Unità nella quale è stato inserito il dvd-rom.

Doppio clic sulla cartella contenete il file desiderato.

Selezionare il file desiderato.

Clic sul menu Modifica.

Clic sulla voce Copia.

Doppio clic sulla Unità C corrispondete all'unità di massa.

Doppio clic sulla cartella nella quale si vuole copiare il file.

Clic sul menu Modifica.

Clic sulla voce Incolla.

## ESEMPIO 3

**PROBLEMA: Copia di N file dal DVD-ROM all'unità di massa.**

Per ciascun file esegui le operazioni del precedente algoritmo.

## ESEMPIO 4

**PROBLEMA: procedura per realizzare una ricetta di cucina.**

# PROCESSI RISOLUTIVI DETERMINISTICI

**Una ricetta non è un algoritmo.**

Di solito fra gli ingredienti vi sono espressioni ambigue come “un pizzico di sale” o “quanto basta” quindi azioni lasciate alla soggettività.

I procedimenti di risoluzione visti negli esempi precedenti, come quello delle ricette, hanno i seguenti inconvenienti

- **Violazione del principio di determinismo**: la descrizione delle azioni non sono rigorose e devono essere interpretare (“fare un impasto base per torte”, “temperare il cioccolato”),
- **Violazione del principio di ripetitività**: ripetendo più volte la stessa azione, dallo stesso esecutore, non è detto che si ottengo gli stessi risultati. Questo dipende dal fatto che le azioni elementari sono **ambigue e non univocamente interpretabili**.

Noi siamo interessati a procedimenti risolutivi **DETERMINISTICI** ossia che producono lo stesso risultato a partire dagli stessi dati, tutte le volte che vengono eseguiti.

La ricetta di cucina non è un algoritmo, ovvero è costituita da una sequenza di azioni che spesso contiene degli elementi di **ambiguità** risolti da un esecutore intelligente. Per esempio, l’azione “sbattere le uova” è un’azione ambigua nel senso che non è detto che tutti gli esecutori umani sono in grado di comprenderla.

Oltre alla ambiguità si possono verificare dei **casi non specificati**. Per esempio, è chiaro che se c’è puzza di bruciato conviene spegnere il forno anche se la ricetta non lo specifica; si confida, quindi, nelle capacità deduttive dell’esecutore umano).

Esempi di problemi che ammettono procedimenti risolutivi di tipo deterministico:

- Dati tre numeri determinare il maggiore.
- Dato un elenco di nomi e relativi numeri di telefono, ricercare il numero di telefono di una determinata persona.
- Ordinare una lista di elementi.
- Stabilire se una persona viene alfabeticamente prima di un’altra.
- Dati i coefficienti a e b, risolvere l’equazione a x + b = 0.
- Risoluzione di una formula matematica.

## ESEMPIO 5

**PROBLEMA: determinare l'energia meccanica E posseduta da un corpo avente una determinata massa m che viaggia alla velocità della luce c.**

**Dati di Input**

Costanti: $c = 3*10^8$ [m/s]

Variabili: m [Kg]

**Dati di Outpit**

E [J]

**Procedura risolutiva (azioni)**

leggi m

assegna ad E $\leftarrow mc^2$

comunica il risultato E

## ESEMPIO 6

**PROBLEMA: determinare qual è il più grande fra due numeri.**

**Dati di Input**

Num1

Num2

**Dati di Outpit**

Max

**Procedura risolutiva (azioni)**

leggi Num1

leggi Num2

se Num1 è più grande di Num2 allora

    a Max assegna Num1

altrimenti

    a Max assegna Num2

comunica Max

## ESEMPIO 7

**PROBLEMA: determinare qual è il più grande fra tre numeri.**

**ESEMPIO 8**

**PROBLEMA: procedura per risolvere un'equazione di secondo grado.**

# DEFINIZIONE DI ALGORITMO

**L'algoritmo è la sequenza ordinata e finita di operazioni, definite con precisione e chiaramente comprensibili per l'esecutore, che l'esecutore deve eseguire per raggiungere un risultato utile, cioè che porti alla risoluzione del problema dato.**

L'**esecutore** può essere l'uomo o la macchina, cioè l'ente che esegue la procedura risolutiva, il procedimento. Il **procedimento** è una sequenza finita di istruzioni elementari.

Il termine **algoritmo** deriva dal nome del matematico arabo (persiano) Abū Jaʿfar Muhammad ibn Mūsā **al-Khwārizmī** vissuto nell'800 d.C., ritenuto l'ideatore della procedura risolutiva per il calcolo della moltiplicazione tra due numeri mediante l'incolonnamento delle cifre (quella che usiamo ancora oggi).

L'algoritmo è una sequenza finita di operazioni, come ad esempio una ricetta di cucina (supposto che l'esecutore riconosce tutte le azioni), le istruzioni per montare un mobile dell'Ikea, o le istruzioni di funzionamento di un DVD Recorder.
Possiamo immaginare l'algoritmo come un procedimento risolutivo che riceve dei dati in ingresso (input) esegue delle elaborazioni e restituisce il risultato della trasformazione (output).


Uno stesso problema può essere risolto in modi diversi. Ci possono essere, quindi, più algoritmi che risolvono lo stesso problema.

# CARATTERISTICHE DELL'ALGORITMO

Un algoritmo per poter essere eseguito automaticamente da un calcolatore deve rispettare le seguenti regole:

- **Finito**: la sequenza di istruzioni deve essere finita e portare ad un risultato avere un punto di *inizio*, dove si avvia l'esecuzione delle azioni, e un punto di *fine*, dove si interrompe l'esecuzione.
- **Eseguibile**: ogni istruzione deve poter essere eseguita materialmente dall'esecutore in un **tempo finito**.
- **Non ambiguo**: le istruzioni devono essere espresse in modo tale da **non essere ambigue**, cioè da poter essere interpretate da tutti allo stesso modo senza lasciare dubbi nella sua interpretazione.
- **Generale**: deve essere valido non solo per un particolare problema, ma per una classe di problemi.
- **Deterministico**: Ogni istruzione deve avere carattere deterministico, ovvero deve produrre il medesimo effetto a partire dalle stesse condizioni iniziali, indipendentemente dall'esecutore.
- **Completo**: deve contemplare tutti i casi che si possono verificare durante l'esecuzione.
- Ogni istruzione deve produrre un risultato **osservabile**.
- Le istruzioni devono essere **elementari**, cioè non ulteriormente scomponibili.

**NB** Non tutti i problemi sono risolvibili. Noi analizzeremo solo i problemi appartenenti alla classe dei problemi risolvibili.

**AMBIGUITÀ**

I linguaggi naturali (come per esempio l'italiano) sono spesso ambigui, cioè possono essere interpretati in modi diversi. Un esempio di frase ambigua è "La vecchia porta la sbarra", che può avere due significati completamente differenti.

In molti casi le istruzioni fornite in un linguaggio naturale sono inoltre imprecise, come per esempio quando in una ricetta c'è scritto "salare" senza specificare esattamente la quantità di sale.

Ambiguità e imprecisione non creano generalmente problemi nella comunicazione fra esseri umani dotati di intelligenza, ma non sono possibili dovendo comunicare con un computer (cioè una macchina). Per questa ragione i linguaggi di programmazione devono consentire di scrivere algoritmi in modo chiaro, senza imprecisioni e senza nessuna possibile ambiguità.

**ESEMPI DI ALGORITMI (RISOLUTORE = UOMO)**

Gli algoritmi li incontriamo e li eseguiamo quotidianamente, per esempio:

- Nel montaggio di un modellino.
- Nell'installazione di un software.
- Nel calcolo del McD di un insieme di numeri interi.

- Nel preparare una ricetta di cucina.
- Nel calcolo dell'area di un rettangolo.
- Nel effettuare il rifornimento ad un veicolo.
- Nella messa in moto dell'auto.
- Nella procedura per acquistare un biglietto di un concerto.

Vediamo per esempio l'algoritmo risolutivo del seguente problema:
Un barcarolo deve trasportare da una riva (A) all'altra (B) di un fiume una pecora, un lupo e un cavolo, ma dispone di un'unica barca con due soli posti (può trasportare uno solo dei tre alla volta). Occorre però fare attenzione a cosa combinano i due passeggeri rimasti soli mentre il barcarolo è in acqua con il terzo. Se il lupo venisse lasciato solo con la pecora la divorerebbe, e lo stesso farebbe la pecora col cavolo se fosse lasciata sola con tale ortaggio. Ricordiamo che il lupo non mangia il cavolo. Come può fare il contadino a portare tutti dall'altra parte?

**PROCEDURA**

**Esecutore: uomo**

1. Inizio.
2. Traghettare la pecora sulla sponda B, lasciando assieme il lupo ed il cavolo sulla sponda A.
3. Ritornare con la barca vuota sulla sponda A, lasciando la pecora da sola sulla sponda B.
4. Traghettare il cavolo sulla sponda B, lasciando il lupo da solo sulla sponda A.
5. Riportare la pecora sulla sponda A, lasciando il cavolo da solo sulla sponda B.
6. Traghettare il lupo sulla sponda B, lasciando da sola la pecora sulla sponda A
7. Ritornare con la barca vuota sulla sponda A.
8. Traghettare la pecora sulla sponda B.
9. Fine.

# STESURA DI UN ALGORITMO

**RIGA DI INTESTAZIONE**: nome dell'algoritmo

**SEZIONE DICHIARATIVA**: elenco dei dati (variabili in/out e costanti/variabili di lavoro) che utilizza l'algoritmo.

**SEZIONE ESECUTIVA**: compresa fra le parole INIZIO e FINE ed elenca le operazioni/istruzioni (dichiarazione, immissione, assegnazione, controllo, scrittura) che l'esecutore deve compiere.

## LA SEQUENZIALITA' DI UN ALGORITMO

La caratteristica fondamentale di un algoritmo è la sequenzialità: le operazioni vengono eseguite sequenzialmente, una dopo l'altra.

Le sequenze possono essere rappresentate graficamente attraverso i diagrammi di flusso (flow chart), come vedremo nel prossimo paragrafo. I flow chart sono le rappresentazioni più utilizzate e intuitive per la rappresentazione di un algoritmo. Esistono particolari programmi, per esempio ALGOBUILD che aiutano il programmatore nel disegnare i diagrammi e, non solo. Vedremo più avanti l'utilizzo di questo importante software di simulazione.

## ESEMPIO 1

Costruire un algoritmo per calcolare il prezzo di un prodotto sul quale è praticato uno sconto.

**DATI INPUT**: il prezzo del prodotto (**Prezzo**), la percentuale di sconto (**Percentuale**). La percentuale di sconto può essere considerata una costante (es 20%).

**DATI OUTPUT**: lo sconto (**Sconto**), e il prezzo scontato (**PrezzoScontato**).

**RISOLUZIONE**:

**INIZIO**

la formula per il calcolo dello sconto: Sconto = Prezzo * Percentuale /100

la formula per il calcolo del prezzo scontato: PrezzoScontato = Prezzo – Sconto

nell'algoritmo si deve prevedere:

1. L'acquisizione del prezzo del prodotto.
2. Il calcolo dello sconto.
3. Il calcolo del prezzo scontato.
4. La comunicazione del valore calcolato.

**FINE**

## ESEMPIO 2

**Problema**

Un automobilista si appresta ad attraversare un incrocio semaforico. Formalizzare il problema mettendo in evidenza i dati, le informazioni, le decisioni da intraprendere e quindi la procedura risolutiva del problema.

**Dati**

Colore del semaforo, limiti di velocità

**Informazione**

Attraversare l'incrocio/fermarsi allo stop

**Decisioni**

Se Verde posso attraversare l'incrocio rispettando i limiti di Velocità

Se Giallo devo rallentare e fermarmi allo stop

Se Rosso devo fermarmi allo stop

**Processo risolutivo**

Controllo il colore del semaforo

Se è verde accelero ed attraverso l'incrocio rispettando i limiti di velocità.

Se invece è giallo decelero e mi fermo allo stop.

Se invece è rosso mi fermo allo stop.

## ESEMPIO 3

Si scriva un algoritmo per cercare il numero di telefono di una persona noto il nome, il cognome e l'indirizzo usando l'elenco telefonico.

Suggerimenti:

- Usare una scomposizione per passi.
- Considerare anche il caso in cui la persona non sia in elenco.
- Provare a descrivere diverse procedure di ricerca.

## ESEMPIO 4

**EFFETTUARE UNA TELEFONATA**

| | |
|---|---|
| 1. Sollevo il ricevitore. | Comando |
| 2. Compongo il numero. | Comando |
| 3. SE qualcuno risponde, ALLORA VADO al PUNTO 6. | Struttura di selezione |
| 4. Depongo il ricevitore. | Comando |

| | |
|---|---|
| 5. TORNO AL PUNTO 1 | Comando |
| 6. Effettuo la conversazione. | Comando |
| 7. Depongo il ricevitore. | Comando |

## ESEMPIO 5

Procedura risolutiva per il calcolo del prezzo di un prodotto scontato al 20%.

**Algoritmo** *ProdottoScontato*

**Dati costanti**:

dichiara Percentuale come numero intero di valore 20

**Dati variabili**:

dichiara Prezzo, Sconto, PrezzoScontato come numeri reali

**Inizio**:

immetti Prezzo

assegna a Sconto: Prezzo * Percentuale /100

assegna a PrezzoScontato: Prezzo – Sconto

scrivi a video PrezzoScontato

**Fine**.

## ESEMPIO 6

Algoritmo per la determinazione di una password.

**Algoritmo** *DeterminarePassword*

**Dati costanti**:

dichiara password come stringa di valore "dani"

**Dati variabili**:

dichiara pw come stringa

**Inizio**:

immetti pw

se pw=password allora

scrivi a video "Password indovinata"

altrimenti

scrivi a video “Password errata”

**Fine**.

## ESEMPIO 7

Procedura risolutiva per il calcolo dell’area di un rettangolo.

**Algoritmo** *AreaRettangolo*

**Dati variabili**:

dichiara BASE del tipo intero

dichiara ALTEZZA del tipo intero

dichiara AREA del tipo intero

**Inizio**:

leggi BASE

leggi ALTEZZA

assegna ad AREA il profotto fra BASE e ALTEZZA

stampa a video la frase “L’area del rettangolo è: “

restituisci AREA

**Fine**.

## VARIABILI – CONTENITORI DI DATI

Le variabili sono dei nomi simbolici usati negli algoritmi per denotare i dati. Le variabili possono contenere dati di tipologia differente:

- Numeri interi.
- Numeri in virgola mobile (reali).
- Singolo carattere.
- Stringhe di caratteri.

L’uso delle variabili semplifica la scrittura degli algoritmi, ma, soprattutto, li rende di validità generale. In altre parole, un algoritmo scritto usando le variabili, al posto dei valori numerici, potrà essere utilizzato per qualsiasi valore numerico. Viceversa, se i valori numerici fossero indicati esplicitatamene nell’algoritmo (in questo caso si parla di costanti per distinguerle dalle variabili), l’algoritmo sarebbe valido per un unico caso.

Possiamo immaginare la variabile come un contenitore, dotato di un **nome**, al cui interno è possibile memorizzare un **valore**.


Le variabili, anche in matematica vengono utilizzate con questo significato.
Per quanto riguarda i nomi, utilizzando le convenzioni usate normalmente dai linguaggi di programmazione, il nome di una variabile può essere composto da una sequenza di lettere (minuscole e/o maiuscole), dalle cifre da 0 a 9 e dal simbolo di under-score (_). Il primo carattere del nome deve essere sempre una lettera oppure il simbolo di sottolineatura e all'interno del nome non devono essere presenti spazi o punti.

Esempi di nomi di variabili validi sono: num, ris, area_cerchio, val2.

Esempi di nomi di variabili non corretti sono: %, 12a, primo val.

## L'OPERATORE DI ASSEGNAZIONE ← oppure =

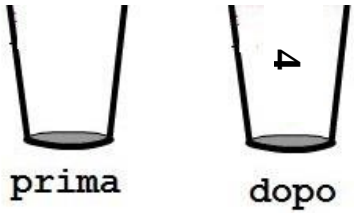
L'operazione fondamentale che si può eseguire su una variabile è l'**assegnazione**. L'assegnazione si indica con il simbolo di uguale (=) o con il simbolo di freccia sinistra (←).
L'operazione di assegnazione permette di attribuire ad una variabile un valore numerico.

nomeVariabile ← valore

Esempio:

X ← 4

Al contenitore (variabile) X viene assegnato il valore intero 4.




Ogni variabile può contenere solo un numero alla volta, perciò assegnazioni successive, eseguite sulla stessa variabile, cancellano i valori precedenti.

Per esempio se vengono eseguite in sequenze le seguenti tre assegnazioni:

X = 10

X = 15

X = 5

alla fine X conterrà il numero 5.

## RAM

La RAM, la memoria di lavoro, può essere immaginata come una sequenza di celle.

Ogni cella può contenere un dato grande un Byte (8 bit). La cella è individuata da un indirizzo numerico.

| Ind. | Cella |
|---|---|
| 0 | |
| 1 | |
| | …….. |
| n | …….. |

Ad una cella possiamo dare un nome simbolico (Variabile); in questo modo, l'istruzione di assegnazione X ← 3 indica che nella memoria RAM esiste una cella a cui è stato attribuito il nome X e in essa viene salvato il valore numerico 3.

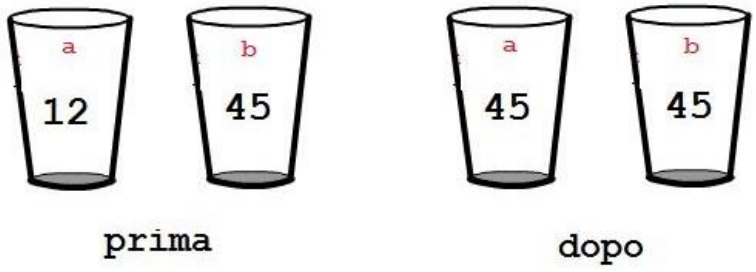
## ASSEGNAZIONE DI UNA VARIABILE AD UN'ALTRA VARIABILE

L'operazione di assegnazione può essere utilizzata anche per copiare il contenuto di una variabile in un'altra.

Per esempio:

a = b

Copia (assegna) il valore contenuto nella variabile b nella variabile a. Nella figura seguente è mostrato il risultato dell'assegnazione "a = b" supponendo che a contenesse inizialmente il numero 12 e b il numero 45:




Attenzione! Sarebbe stato diverso scrivere

b = a

In questo caso il valore di a sarebbe stato copiato in b e il risultato sarebbe stato il seguente:
Dunque l'uguale usato nell'assegnazione ha un verso: il valore (sorgente ovvero il contenuto della variabile) che si trova a destra viene copiato nella variabile che si trova a sinistra (destinazione).

## ASSEGNAZIONE DI UN ESPRESSIONE AD UNA VARIABILE

L'operazione di assegnazione permette anche di attribuire ad una variabile un valore ottenuto dal risultato di una qualsiasi espressione matematica.

nomeVariabile = espressione

Esempio:

Y ← X * 2

Assegno alla variabile Y il risultato dell'espressione X * 2.
Se X vale 3 allora Y assumerà il valore 6.
I simboli usati per scrivere le espressioni matematiche nei linguaggi di programmazione simili a quelli usati in matematica. L'operazione di moltiplicazione viene però indicata con l'asterisco * invece che con il x. Bisogna inoltre prestare attenzione all'uso delle parentesi (sempre e solo le parentesi tonde) per indicare la precedenza fra le operazioni.

Esempio:

ris = a + b * 5/(c + d)

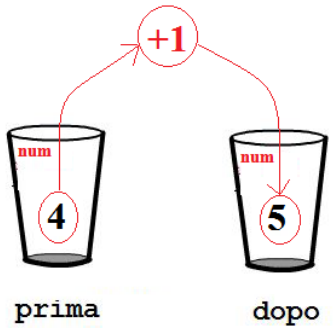
## ASSEGNAZIONE DI UNA VARIABILE A SE STESSA - VARIABILI DI ACCUMULO

Un aspetto molto interessante delle variabili è che è possibile assegnare una variabile a se stessa o, più precisamente, usare la stessa variabile a sinistra e a destra di una assegnazione.
Vediamo alcuni esempi significativi:

### L'AUTOINCREMENTO

```
num = num + 1
```

In questo caso la stessa variabile num compare sia a sinistra che a destra del simbolo di assegnazione e dunque è sia la sorgente che la destinazione di una assegnazione. Si presti particolare attenzione al significato di questa scrittura che, tradotta a parole, sarebbe: prendi il valore di num, aggiungi uno a tale valore e rimetti il risultato in num. In pratica se ad es. num valeva 4 prima dell'istruzione precedente, dopo l'istruzione il suo valore è diventato 5.




Questa semplice istruzione (che aumenta di uno il valore di una variabile) viene detta incremento (l'istruzione opposta, quella che diminuisce di uno, è detta decremento).

### SOMME PARZIALI

Supponiamo di dover calcolare la somma di MAX numeri, dome MAX è un numero intero indicante appunto il numero di numeri da sommare.
Per esempio, se MAX = 5 e i numeri da sommare sono 1, 3, 5, 7, 9, matematicamente si esegue la seguente addizione matematica:

somma = 1 + 3 + 5 + 7 + 9

dal punto di vista iterativo e informatico, la procedura adottata per il calcolo della somma non è efficiente, per vari motivi:

- L'espressione contiene MAX – 1 addizioni, e se quindi MAX è elevato, l'espressione stessa sarà molto lunga da scrivere e da calcolare.
- Tale addizione è valida solo per quei valori; non può essere utilizzata nel caso in cui si volesse generalizzare la situazione al calcolo di MAX numeri interi qualsiasi, scelti di volta in volta dall'utente, digitandoli da tastiera.

La tecnica risolutiva da adottare è quella che fa uso dell'iterazione, cioè dell'eseguire per MAX volte le seguenti azioni:

```
leggi num
somma = somma + num
```

Se questa istruzione vengono eseguite MAX volte, dove MAX ha un valore noto a priori, e se ogni volta ad num viene assegnato il valore digitato dall'utente da tastiera, ci ritroveremo alla fine ad avere nella variabile somma la somma di tutti i valori inseriti da tastiera.

**NB**. Occorre fare attenzione che la variabile somma deve essere inizializzata, cioè occorre attribuirgli valore 0 all'inizio.

**Esempio**

Un altro esempio un po' più complesso di variabile usata sia a destra che a sinistra dell'uguale di assegnazione è il seguente, che potrebbe servire per calcolare l'aumento della somma depositata su un conto corrente con un certo tasso di interesse:

somma = somma + tasso * somma

In questo caso la stessa variabile compare addirittura tre volte nella stessa espressione.

Quando una variabile, come in questi esempi, è sia la sorgente che la destinazione di una assegnazione, si dice che si tratta di una variabile di accumulo. Una variabile di accumulo, come suggerisce il nome, è una variabile nella quale vengono accumulati dei valori.

Sulle variabili di accumulo ci ritorneremo ancora quando affronteremo i cicli iterativi.