Infor mazione auto **matica**

La disciplina **informatica**, più che scienza del computer (*computer science*), è da intendersi, alla francese (*information automatique*), come scienza per elaborare e scambiare informazione[1] in modo automatico.

Scienza/*e che si occupa/no dei* ***processi*** *e* ***tecnologie*** *che consentono il* ***trattamento*** *(****creazione, raccolta, elaborazione, memorizzazione, comunicazione****) automatico e razionale* ***dell'informazione*** *e della* ***progettazione degli strumenti*** *che* ***concretizzano tali funzioni***

Computer: strumento per risolvere problemi (prendere decisioni) e comunicare informazioni

Un computer moderno è un sistema (HardWare – *dispositivi e* SoftWare – programmi) con la funzione non solo di **elaborare** automaticamente **informazioni** per risolvere problemi ma anche con la funzione di facilitare il **reperimento** e lo **scambio** delle informazioni con una capillarità a livello mondiale (Internet).

Un Personal Computer:

- potrà essere usato in modalità *stand alone* sfruttando le capacità elaborative autonome
- se connesso in una *rete*[2] *locale* (LAN), potrà **condividere risorse locali** sia hardware (come la stampante o supporti magnetici di memorizzazione) che software (come programmi applicativi)
- se connesso ad INTERNET ('la rete delle reti"), potrà **interagire mediante scambio di messaggi al fine di condividere le risorse messe a disposizione da un sistema di comunicazione a livello mondiale**

[1] **Informare** significa arricchire di conoscenza il destinatario; si genera e trasmette informazione con il fine di una risoluzione dell'incertezza del destinatario; minore è la probabilità di emissione di un messaggio (quindi maggiore è l'incertezza), maggiore è il contenuto informativo (I = K * 1/P).

[2] **Rete**: due o più computer (**nodi** di elaborazione) connessi assieme per comunicare, condividere risorse ed interoperare (SO diversi)

Imparare l'informatica non è solo riuscire a far funzionare un calcolatore o arrivare ad un programma che "gira", ma è anche **organizzare con più razionalità il modo di affrontare i problemi** e di **ricercare** e **usare le conoscenze**.

Si trattano dati per risolvere problemi ma è l'informazione l'obiettivo cui tendiamo nella risoluzione dei problemi.

Un problema[3] è incertezza cioè mancanza di informazioni

Risolvere un problema significa ***costruire un elemento di conoscenza che ci mancava*** *per compiere una* ***scelta*** *tecnica, economica o politica rispetto alla quale ci sentivamo incerti e sguarniti e lo scopo stesso della comunicazione è arricchire di conoscenza il destinatario,* risolvendone l'incertezza ed aumentandone la comprensione.

Il signor A comunica al signor B un'informazione

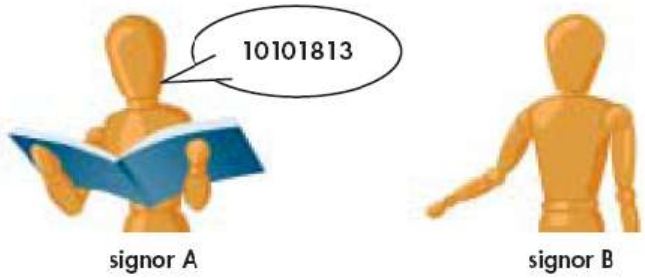
→ la conoscenza del signor B è aumentata se ora sa qualcosa di nuovo

La conoscenza aumenta se riceviamo un **dato** e la relativa **chiave di lettura**, ma non aumenta se manca uno di questi elementi:

Dato + significato = informazione

Un computer manipola **dati** (misure nel mondo reale) sotto forma di **simboli**

Es: la cifra 10101813 è un **dato** cioè una delle possibili **rappresentazioni** di un'informazione



Il signor B non è in grado di interpretare un dato se non ha la chiave di lettura.

Che cosa rappresenta "10101813"?

Un numero di telefono? Di chi?
Una data di nascita? Di chi?
Una somma di denaro? In dollari o in euro?

Si definisce il **dato** come la parte **estensionale** di un'informazione: la sola parte estensionale non fornisce conoscenza, *non è un'informazione*. Un dato preso al di fuori del suo contesto *non è interpretabile*, manca il suo significato, quindi non accresce la conoscenza.

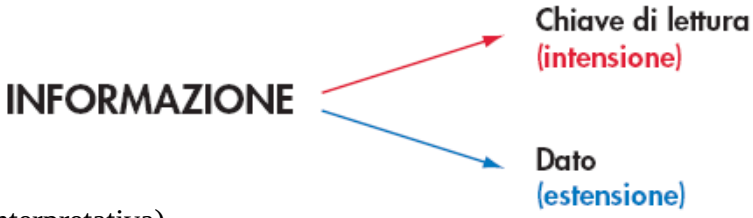
---

[3] **Problema**: quando non sappiamo affrontare "normalmente" la vita; di solito il nostro comportamento è meccanico, le nostre azioni sono frutto di abitudine ma nasce il problema quando un fatto imprevisto ci causa dubbio, disagio perché ci impone una scelta con esigenza di riflettere cioè ragionare analizzando dettagliatamente il pro e il contro delle decisioni al fine di sciogliere quel dubbio (*analysis* = "scioglimento, risoluzione").

Anche la sola chiave di lettura non consente al signor B di aumentare la sua conoscenza

Si definisce tale chiave di lettura come la parte **intensionale** di un'informazione: la sola parte intensionale non fornisce conoscenza, *non è un'informazione*

L'**interpretazione** del dato produce **incremento di conoscenza → informazione**



Noto il **contesto** (chiave interpretativa)

Es: 8 inserito nel registro di classe in relazione ad un nominativo

Si chiarisce **cosa** …. si voleva comunicare (***comprensione***)

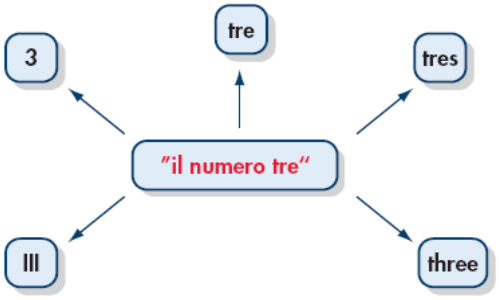
Es: una valutazione positiva dell'allievo

> Il **dato**, quindi, è una *conoscenza elementare* che, presa individualmente e fuori da un preciso contesto, non ha alcun valore.

> L'**informazione**, invece, è il *dato elaborato*, ossia l'incremento di conoscenza che deriva dall'*interpretazione di un dato.*

Un **dato** può rappresentare **più informazioni**

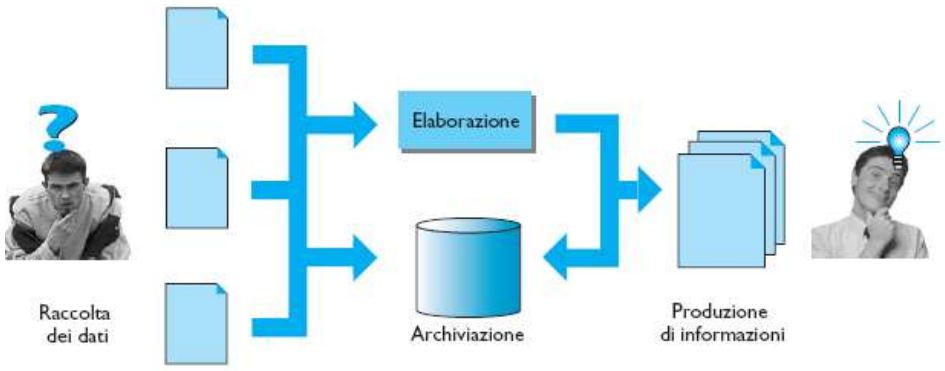
Una **informazione** può essere rappresentata da **più dati**



Il **computer non interpreta i dati**, non attribuisce loro significati precisi (come fa la mente umana): per una macchina elettronica o ottica sono solo semplici ***simboli***

> il computer tratta i dati, l'uomo tratta le informazioni.

> L'attività che si occupa del trattamento dei dati per trasformarli in informazioni prende il nome di elaborazione



Sono di fondamentale importanza, nel ***risolvere un problema***:

- Una precisa **riformulazione**[4] del problema che evidenzi *dati ("datum = fatto")* e *risultati*;
- Lo sviluppo di un modello delle relazioni tra dati e risultati, costruito sulle conoscenze che già si posseggono; … approssimazione → possibile errore → *validazione*
- L'adozione di un modello di riferimento che rappresenti l'*esecutore* o il sistema di regole entro cui dovrà svilupparsi la soluzione del problema.

L'attenzione alla **formalizzazione** è essenziale quando *risolutore* ed *esecutore sono diversi* e questo comporta:

- **Riformulare** per evitare ambiguità
- Chiarire lo **scopo** (soluzione)
- Evidenziare i **dati**: **disponibili** o da **assumere**

[4] A volte la riformulazione con un percorso logico inverso rispetto a quello dato può facilitare la soluzione.

I dati si presentano alla nostra osservazione in varie forme:

- **numeri** (il peso di una persona, il voto di un compito in classe);
- **caratteri alfabetici** (il titolo di un film, il nome di un cane);
- **caratteri alfanumerici** (il numero di una targa, un codice fiscale, una parola chiave per accedere a un videogioco);
- **immagini** (una foto, un manifesto);
- **grafici** (un istogramma, una piantina topografica, un elettrocardiogramma);
- **suoni** (una sirena, l'allarme di un'auto, il suono della sveglia);
- **luci** (un semaforo, un faro);
- **gesti** (un saluto, un abbraccio, un bacio).

Numeri e caratteri rappresentano i **dati semplici** (*primitivi*), perché sono relativamente semplici da raccogliere e analizzare; gli altri costituiscono i **dati complessi**, in quanto derivanti dalla fusione di più dati semplici.

In particolare, dati complessi quali suoni, animazioni e filmati, prendono il nome di **dati multimediali** (infatti le informazioni vengono rappresentate con diversi mezzi di comunicazione o, in latino, *media*) e con tale tipologia di dati si costruiscono ipermedia, consentendo un accesso alle informazioni non lineare.

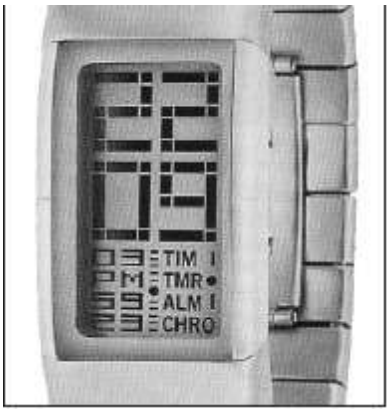
Rappresentazioni di un'informazione: dati analogici e dati digitali

La rappresentazione **analogica** di un'informazione si basa su un insieme **continuo** di valori. I dati da elaborare e trasmettere sono rappresentati fisicamente da grandezze fisiche che, in analogia col tempo, assumono infiniti valori (segnali continui o analogici).

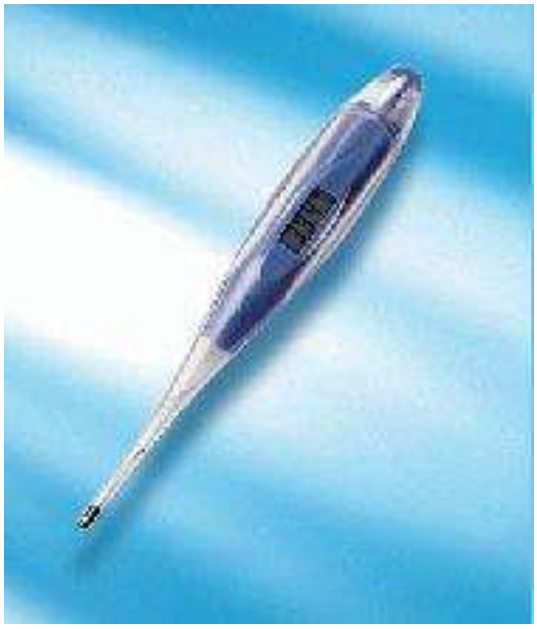
Esempi: un orologio analogico (con le lancette) rappresenta, senza brusche variazioni, tutti gli istanti di tempo; il termometro a mercurio rappresenta il valore della temperatura in base alla lunghezza della colonnina di mercurio.

In teoria non ci sono limiti alla precisione delle rappresentazioni analogiche: ad esempio, ad ogni minima variazione di temperatura, si ottiene (o si dovrebbe ottenere) un'analoga variazione della dimensione della colonnina di mercurio.

La rappresentazione **digitale** si basa su un insieme **discreto** di valori che presentano brusche variazioni.



In un **sistema digitale** i segnali da elaborare e trasmettere (segnali discreti o digitali) non assumono tutti i valori di un intervallo e sono rappresentabili con simboli in numero finito.



Esempi: un orologio digitale rappresenta il tempo a salti di minuti o di secondi o di frazioni più piccole; un termometro digitale rappresenta la temperatura in gradi e decimi di grado e la visualizza su un display.

Un termometro di questo tipo non è in grado di misurare variazioni di temperatura inferiori al decimo di grado poiché la sua rappresentazione si basa su un insieme discreto composto da multipli di un'unità di base (**digit**), nel nostro esempio decimi di grado.

Il numero di valori è **finito (elaborabile da un esecutore discreto)** ed in prima analisi si potrebbe affermare che le rappresentazioni digitali risultano più approssimate di quelle analogiche, ma in realtà la situazione è diametralmente opposta infatti la "precisione" digitale (dimensione del digit) può essere **modificata** mentre c'è un problema legato alla effettiva valutazione di una rappresentazione analogica: i limiti dell'operatore umano.

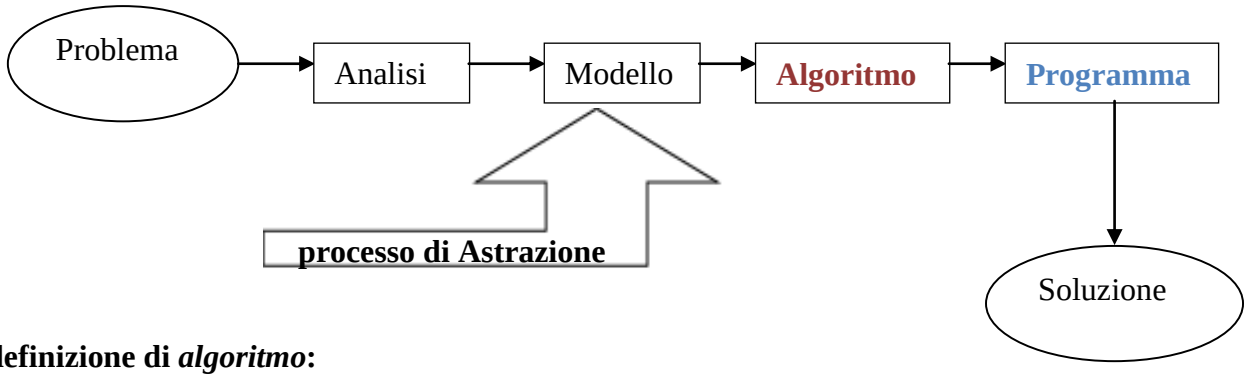
Nell'esempio, se le due colonnine di mercurio hanno misurato temperature differenti di pochi centesimi di grado, è difficile capire quale delle due misurazioni è più alta e quale più bassa

Nella storia si sono usate diverse **forme** di comunicazione.

In realtà, si va verso un "**mondo digitale**" e le informazioni di qualunque tipo vengono con sempre maggiore frequenza rappresentate in modo digitale, anzi **binario**.

Non vi è quindi alcuna differenza sostanziale tra di esse ed è possibile **integrare i servizi** offerti agli utenti consentendo loro durante uno stesso collegamento di parlare, scambiare dati o testi, inviare immagini sia fisse sia in movimento ecc., cioè consentire una "comunicazione multimediale".

In un contesto informatico, la soluzione di un problema può richiedere le **fasi** seguenti:



definizione di *algoritmo*:

elenco **finito** di *direttive* che un esecutore **discreto** interpreta **univocamente** (senza ambiguità) e può portare a **termine** in un numero finito di passi, producendo un risultato per ogni ingresso (**complessità limitata** delle istruzioni)

Quindi un algoritmo è la soluzione astratta di un problema, espresso in linguaggio formale più o meno rigoroso con caratteristiche di:

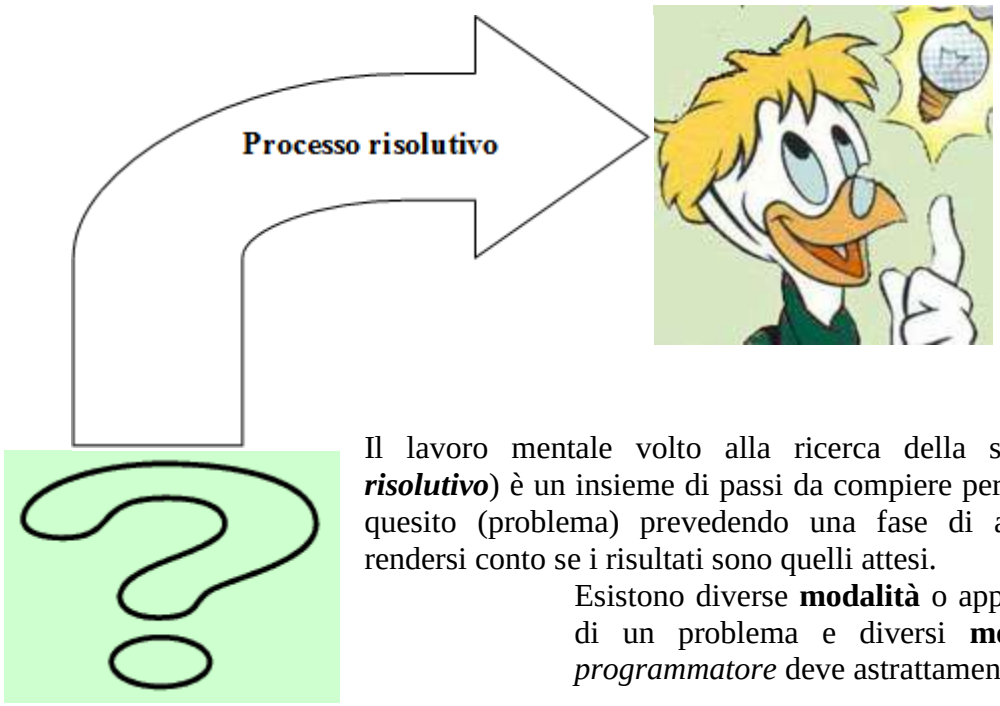
- *Finitezza* (se il tempo fosse infinito, la soluzione sarebbe praticamente irraggiungibile)
- *Univocità* (se eseguite da più esecutori devono condurre allo stesso risultato)
- *Complessità limitata delle istruzioni* (effettivamente eseguibili: realmente meccanizzabili, eseguite in tempi non nulli)
- Adatto per *esecutore* che opera in modo *discreto* (istruzioni discrete cioè distinte perché se le differenze diventassero infinitesime si ricadrebbe nell'elenco infinito)

definizione di *programma:*

una **sequenza di istruzioni**, scritte in un **linguaggio comprensibile al calcolatore** che le esegue per ottenere i risultati richiesti

definizione di *problema risolvibile:*

qualsiasi quesito la cui *soluzione* possa essere rappresentata da una macchina (*automa: sistema discreto* cioè caratterizzato da un avanzamento per passi successivi dove ogni ***passo*** produce nell'automa un cambiamento di *stato;* inoltre ogni stato è descrivibile mediante una stringa finita di *simboli* tratti da un insieme prefissato) o, equivalentemente, da un *programma* per un calcolatore.



Il lavoro mentale volto alla ricerca della soluzione (***processo risolutivo***) è un insieme di passi da compiere per dare risposta ad un quesito (problema) prevedendo una fase di attenta verifica per rendersi conto se i risultati sono quelli attesi.

Esistono diverse **modalità** o approcci alla soluzione di un problema e diversi **modelli** ai quali un *programmatore* deve astrattamente uniformarsi.

A seconda della **metodologia** si definiscono **diversi paradigmi**[5] **di programmazione:**

- **procedurale** se affrontando i problemi in **modo tradizionale** si seguono le tre fasi seguenti:
  - **Conoscenza** e studio della *realtà* (o contesto) da cui è tratto il problema
  - **Comprensione** del problema ("**cosa**" si vuole)
  - **Ricerca** di una strategia (o idea) risolutiva per il problema ("**come**") e sintesi della procedura risolutiva (*algoritmo* e/o programma) per un esecutore

- **non procedurale** (programmazione **dichiarativa**) che prevede invece di:
  - **Specificare la conoscenza** della *realtà* e il problema in un *linguaggio di **specifica***
  - **Affidare all'interprete del linguaggio** la **soluzione**

definizione di *paradigma di programmazione*: modello ("chiave di interpretazione", "punto di vista") cui la mente di chi legge o scrive un programma deve idealmente conformarsi.

[5] **Paradigma** termine derivante dal greco *paradeigma* che significa «modello» (o «progetto») ed «esempio»; altra accezione del termine è prossima a quella di «analogia» (in Platone si trova usato in tutte le accezioni). Le idee sono infatti modelli o termini di paragone assoluti, conoscendo i quali è possibile decidere se qualcosa sia o non sia conforme ad essi: per esempio conoscendo che cosa è la santità, si può giudicare di un'azione se sia o non sia santa. Col significato di «esempio» o «caso esemplare», Socrate si considera un *paradeigma* di cui il dio si è servito per illustrare la condizione umana riguardo al sapere, che è appunto di non-sapere. Aristotele assume *paradeigma* come termine tecnico della logica e della retorica col significato di «argomento fondato su un esempio» o «induzione retorica», dal particolare al particolare; è l'argomento che si trae da un caso noto per illustrare, grazie alla pregnanza dell'esempio che si è scelto, un caso meno noto o affatto ignoto. http://www.riflessioni.it/enciclopedia/paradigma.htm

Strutture di controllo del flusso: tecniche di rappresentazione

Per realizzare un **algoritmo strutturato** nell'approccio **tradizionale** (programmazione imperativa) si usano le seguenti strutture di controllo del flusso:

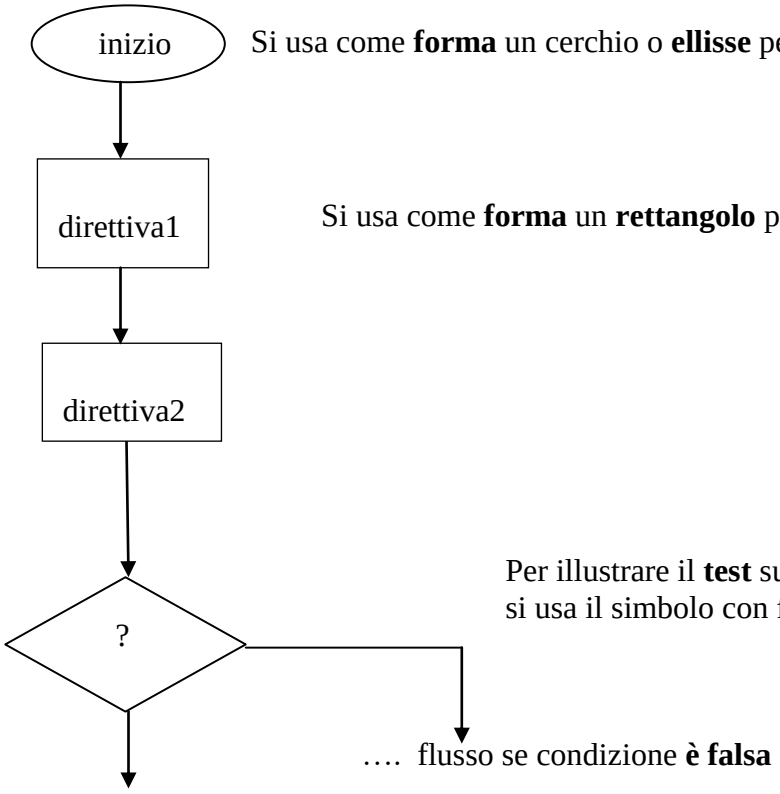
- la **sequenza**,
- la selezione o **alternativa**,
- l'iterazione o **ripetizione**.

Tali costrutti logici si possono illustrare con due tecniche di rappresentazione: è preferibile usare un **flow-chart** (diagramma di flusso) per algoritmi semplici, facilmente traducibili in linguaggio macchina, con richiesta di efficienza cioè velocità.

Si usa, invece, lo **pseudocodice** per algoritmi complessi, facilmente traducibili in linguaggi strutturati, senza richiesta di efficienza cioè velocità

- Le strutture di controllo del flusso si possono illustrare con la tecnica **flow-chart** (o diagramma di flusso):

Sequenza:



Si usa come **forma** un cerchio o **ellisse** per indicare *inizio* oppure *fine*

Si usa come **forma** un **rettangolo** per indicare una *direttiva*[6]

Per illustrare il **test** su una **condizione** si usa il simbolo con forma di rombo

…. flusso se condizione **è falsa**

... flusso se condizione **è vera**

[6] Con più dettaglio, le *direttive* che implicano operazioni di lettura o scrittura si possono indicare usando come forma un parallelepipedo.

- Utilizzando il simbolo **{** col significato di *inizio* ed il simbolo **}** col significato di *fine* si possono illustrare le strutture di controllo del flusso con il seguente **pseudocodice**:

Sequenza

```
{
  direttiva 1
  direttiva 2
   ……
}
```

Alternativa binaria:

```
{
  se (condizione)
      direttiva_Vera
 altrimenti
      direttiva_Falsa
}
```

Ripetizione con ***controllo in testa***:

```
{
  mentre (condizione)
     direttiva
}
```

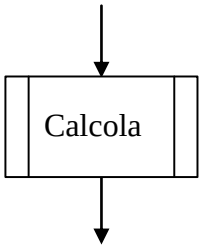
Ripetizione con ***controllo in coda***:

```
{
  fai
      direttiva
  mentre (condizione)
 }
```

NB: si noti l'***indentazione*** (margine rientrato) per evidenziare il comando o blocco di comandi da eseguire in base al verificarsi o meno di una condizione.

NB: Possiamo riferirci mediante un ***nome*** ad blocco di comandi (*direttive*) e proporre una soluzione "modulare" più chiara. Ad esempio si usa sia lo pseudocodice sia il flow-chart per indicare un'elaborazione col nome **Calcola** che analizzeremo dopo, individuando i singoli passi:

```
{
    Calcola
}
```



Strategie risolutive

Una tipica strategia risolutiva nello studio di sistemi dinamici prevede:

- La definizione dell'obiettivo
- L'identificazione del sistema (scomponendo in eventuali sottosistemi)
- La definizione dei vincoli (limiti reali e del modello)
- Il modello[7] di massima e rigoroso
- La simulazione che, attraverso modelli noti, produce risultati che rappresentano una o più possibili storie (evoluzioni) e permette di fare **previsioni**; spesso si usa il computer (simulazione *numerica*) che, per diverse eccitazioni, fornisce un numero molto elevato di possibili risposte
- La scelta del risultato cercato si risolve, poi, nel *vagliare* l'insieme dei risultati, alla ricerca di quello che verifica una prefissata condizione (*criteri di scelta*)

Possibili differenti strategie di ricerca della soluzione con sviluppo di un algoritmo dopo aver riformulato il problema in termini informatici e fissato il modello del sistema di esecuzione:

- metodo simbolico o **diretto** che produce il risultato con uso di formula risolutiva (modello matematico),
- **per enumerazione** quando il numero dei risultati è limitato ed è noto un criterio per individuare correttamente la soluzione: potrebbe essere il modo con cui individuare la chiave di casa nel mazzo che teniamo in tasca (il criterio di accertamento della soluzione è il confronto vincente con la serratura).
- per **tentativi** : si procede per approssimazioni successive restringendo sempre più il campo delle possibili soluzioni (molto usato nel calcolo numerico approssimato ad esempio l'algoritmo di Newton[8] per il calcolo della radice quadrata di un numero N: si sceglie un primo valore approssimato della radice con approssimazione anche grossolana, passo dopo passo si calcolano approssimazioni migliori finché la differenza tra due radici approssimate successive o la differenza tra il quadrato della radice ed N non diventano minori di una prefissata quantità).

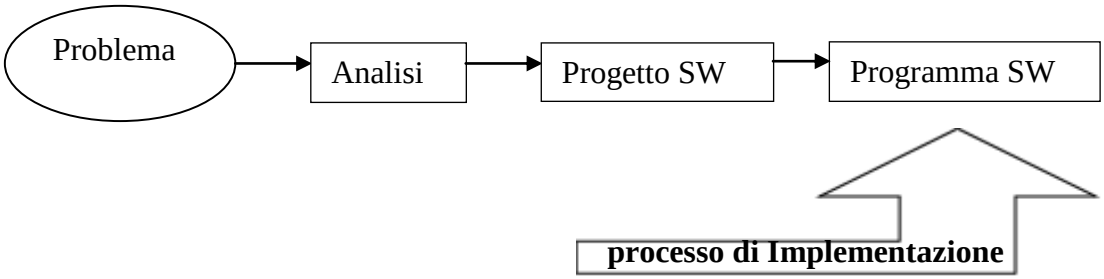
TRACCIA DEI PASSAGGI PER N=25

R_i	$R_{i+1}=\frac{1}{2}\cdot(R_i+N/R_i)$
$R_0=25$	$R_1=\frac{1}{2}\cdot(25+25/25)=13$
$R_1=13$	$R_2=\frac{1}{2}\cdot(13+25/13)=7.462$
$R_2=7.462$	$R_3=\frac{1}{2}\cdot(7.462-25/7.462)=5.406$
$R_3=5.406$	$R_4=\frac{1}{2}\cdot(5.406+25/5.406)=5.015$
$R_4=5.015$	$R_5=\frac{1}{2}\cdot(5.015+25/5.015)=5.000$
⋮	⋮

[7] Tale *descrizione dinamica* di un sistema *non è mai priva di errori* perché i modelli sono delle approssimazioni.
[8] $R_{i+1} = ½ * (R_i + N/R_i)$

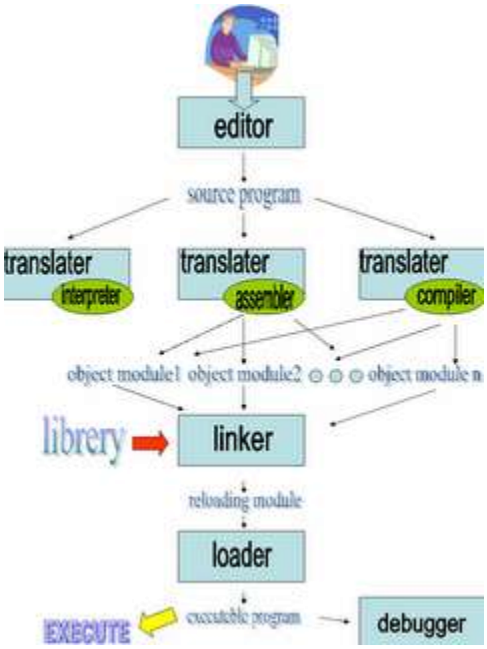
Ciclo di vita del software

L'espressione *ciclo di vita del software* si riferisce al modo di realizzare un **prodotto SW** (con associata documentazione) che, nel contesto dell'ingegneria del software, prevede la successione delle fasi illustrate:



- **analisi** del problema (con produzione documento: specifiche dei requisiti)
- progettazione con definizione della **struttura dati** e delle **interfacce**[9] (con produzione documento di progetto)
- programmazione cioè **codifica** (con produzione di listato commentato)

Nel caso di programmi *compilati*[10] si succedono le seguenti **fasi nella produzione di SW** (*ciclo di vita del programma*):



1. la prima fase consiste nell'**editing** cioè la scrittura del programma ***sorgente***: testo ASCII
2. la seconda nella **compilazione** cioè la *traduzione* con *controllo sintattico* producendo un file ***oggetto*** (con estensione .obj) in linguaggio esadecimale interpretabile da una particolare CPU
3. la terza è la fase di **building** (costruzione) o ***linker***: collegamento con librerie con produzione del file ***eseguibile*** (con estensione .exe) per caricamento (***loader***) in RAM ad un determinato indirizzo fisico
4. esecuzione

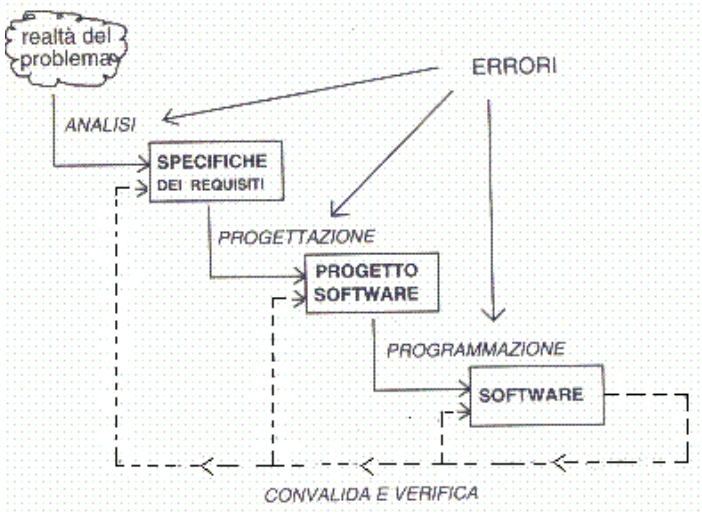
Nel caso di programmi codificati in linguaggio ***interpretato*** la traduzione avviene al momento dell'esecuzione o *run time*. Nei linguaggi semicompilati come Java (architettura interpiattaforma) si realizza una fase di precompilazione e produzione di ***bytecode*** compatibile con architetture diverse.

---

[9] Software modulare

[10] L'approccio tradizionale alla programmazione prevede l'esecuzione di sequenze di istruzioni. I linguaggi con questa impostazione vengono definiti *procedurali* (un problema è scomposto in una serie di procedure di tipo algoritmico per individuare "come" risolvere il problema) ed il più noto, tra questi, è il *paradigma imperativo* (linguaggi ***compilati*** come il Pascal o il C e linguaggi ***interpretati*** come il Basic). On-line una classificazione del SW e dei linguaggi.

Poiché ogni fase può aver introdotto errori (specialmente errori di *omissione* durante l'analisi) si prevedono *più percorsi di convalida e verifica* oltre alla verifica sull'uscita di ogni fase.

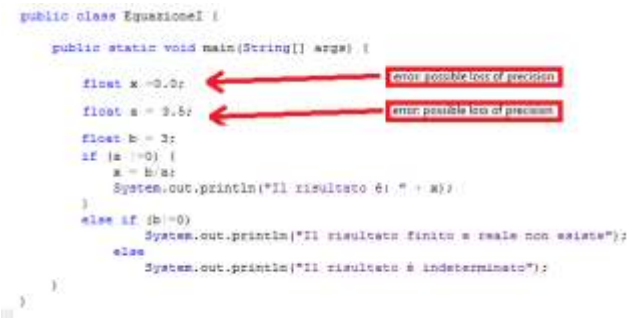


Collaudo: dettagliato il procedimento risolutivo di un problema, si ha la necessità di saggiarne la correttezza rispetto all'obiettivo che si vuole con esso raggiungere.

Il modo più facile di verificare la funzionalità di un algoritmo è quello di tradurlo in programma per un sistema di esecuzione reale e provarlo su calcolatore.

In alcuni ambienti di sviluppo[11] esistono facilitazioni per effettuare tale test come la possibilità di avere una traccia *passo-passo* delle elaborazioni svolte e dei valori assunti dalle variabili.

Il debugger: trovare gli errori



Curiosità: Sulla parola "bug" ("insetto") c'è un aneddoto divertente.

Il 9 settembre del 1945 Grace Murray Hopper (ufficiale e matematica di gran valore) che prestava servizio in Virginia presso la marina militare degli Stati Uniti stava cercando di trovare l'errore che inceppava il computer basato su un sistema Harvard Mark II, quando trovò un insetto che girovagava allegramente in mezzo ai circuiti e che era la causa del malfunzionamento.

Da allora il termine "bug" entrò nell'informatica per indicare un errore di programmazione. Potete leggere tutta la vicenda cercando la Storia di Grace Murray Hopper.

Aneddoto raccontato anche da http://it.wikipedia.org/wiki/Bug_(informatica) con maggiori dettagli

Dopo aver rimosso l'insetto (alle ore 15.45), il tenente incollò la falena rimossa sul registro del computer e annotò: «1545. Relay #70 Panel F (moth) in relay. First actual case of bug being found».

Questo registro è conservato presso lo Smithsonian National Museum of American History.

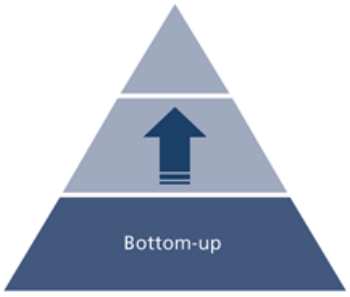
In realtà il termine *bug* nel senso di problema tecnico è stato usato almeno dal 1870: è citato in questo senso in una lettera di Edison del 1878, probabilmente questo uso del termine è di origine scozzese.

[11] Ambiente di sviluppo di tipo **I**ntegrated **D**evelopment **E**nvironment con disponibilità di programmi **debugger**

Approcci nella soluzione di un problema

Bottom -Up: è un approccio nella soluzione di un problema che a partire dal considerare oggetti più semplici o dati empirici (***dal particolare***), con processo di astrazione (cioè trascurando i particolari inessenziali), per *induzione*[12], giunge a conclusioni di carattere ***generale*** con formulazione di proprietà generali.

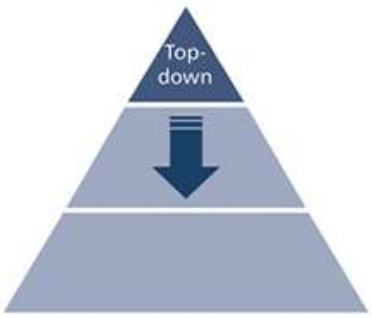
Un esempio di tecnica bottom-up in ambito informatico: dal particolare (semplici istruzioni), con processo di integrazione, si realizzano moduli più complessi fino alla soluzione del problema (algoritmo oppure programma)



Top-Down: è una metodologia di trattamento dei problemi per *scomposizione* o, equivalentemente, di sviluppo degli algoritmi per *raffinamenti successivi*.

E' un'analisi dall'alto al basso cioè dal vertice alla base, dal generale al particolare, dal risultato finale alle azioni elementari componenti la sua elaborazione. E' ricorrente nella comune attività mentale.

L'indagine scientifica ne fa largo uso quando, procedendo in modo analitico, presuppone di poter scomporre un fenomeno in aspetti tra loro indipendenti per concentrare l'attenzione sull'evoluzione di poche variabili alla volta.



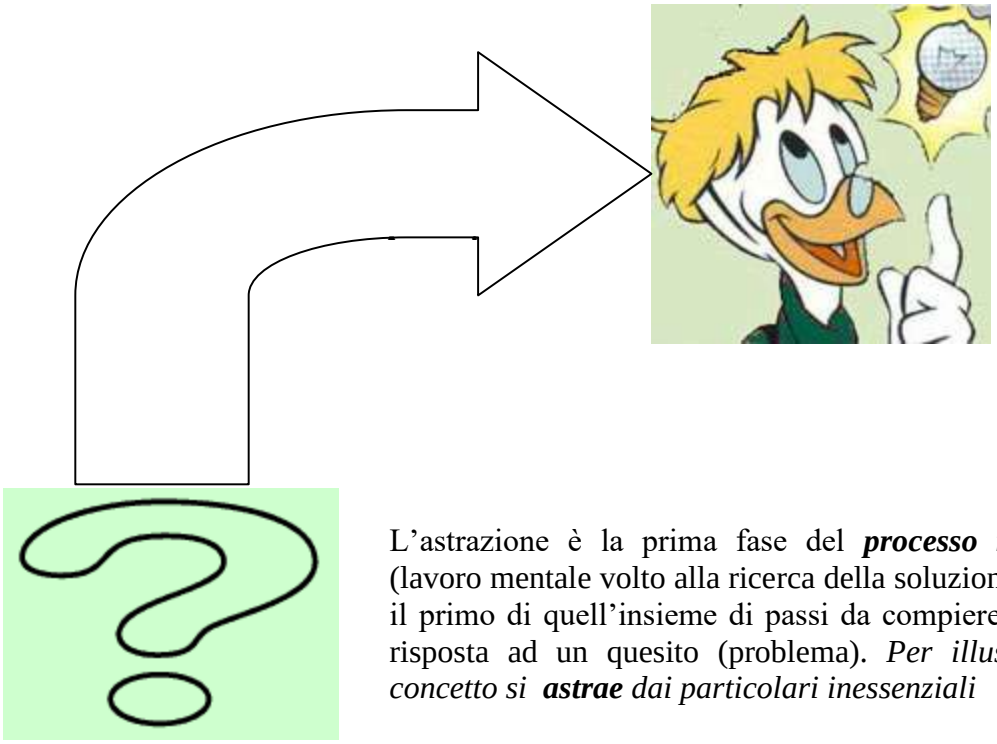
Essa si presenta quindi come un modo naturale di ridurre la complessità di un problema in quanto il numero di dati e di relazioni all'interno di un sottoproblema costituiscono sicuramente un sottoinsieme più facilmente dominabile del problema complesso.

[12] Induzione: inferenze (passaggi da un argomento all'altro) che partendo **dal particolare** giungono ad affermazioni di **carattere generale** ; ragionamenti con conclusioni probabili, attendibili fino a prova contraria.

Astrazione e Implementazione

Termini molto usati in Informatica sono "*astrazione*" e "*implementazione*".

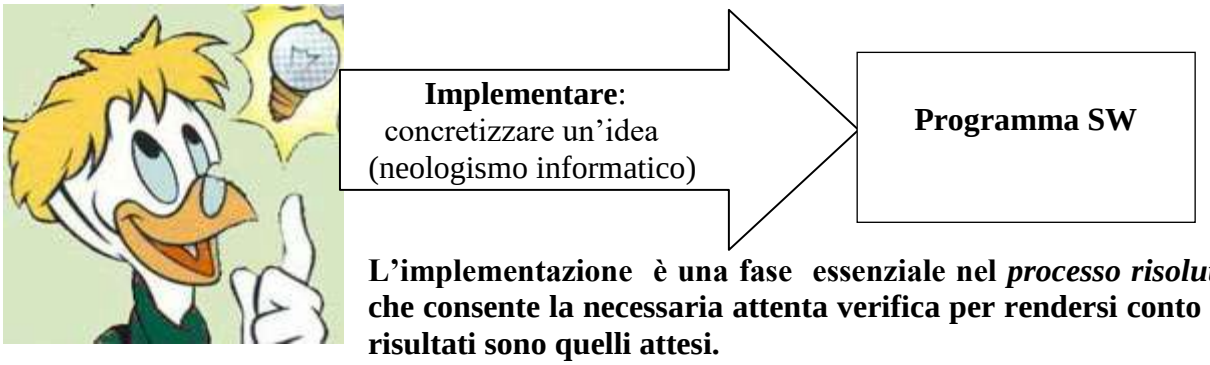
- Il primo ("*astrazione*") fa riferimento alla necessaria estrazione delle **caratteristiche essenziali** (dati significativi e funzionalità caratteristiche) del sistema da analizzare *astraendo* dai particolari inessenziali. E' una **descrizione**[13] **idealizzata** di un oggetto (entità concreta).



L'astrazione è la prima fase del ***processo risolutivo*** (lavoro mentale volto alla ricerca della soluzione) cioè è il primo di quell'insieme di passi da compiere per dare risposta ad un quesito (problema). *Per illustrare un concetto si* ***astrae*** *dai particolari inessenziali*

" Il ***processo di astrazione*** *è tipico del metodo scientifico: a partire dalla ricerca e raccolta di* ***dati empirici*** *(tratti da esperienze concrete) si giunge* ***alla generalizzazione*** *formulando leggi"*

- Il secondo ("*implementazione*") è la **realizzazione concreta** del concetto idealizzato (si realizza un'entità concreta con le **stesse funzionalità** dell'entità astratta). Implementazione è sia l'attività di realizzazione di una astrazione sia il prodotto di questa attività cioè l'entità concreta.



L'implementazione è una fase essenziale nel *processo risolutivo* che consente la necessaria attenta verifica per rendersi conto se i risultati sono quelli attesi.

[13] "**Descrizione** (a voce o per iscritto): rappresentazione che indichi le caratteristiche più importanti per dare un'idea dell'oggetto il più precisa possibile"

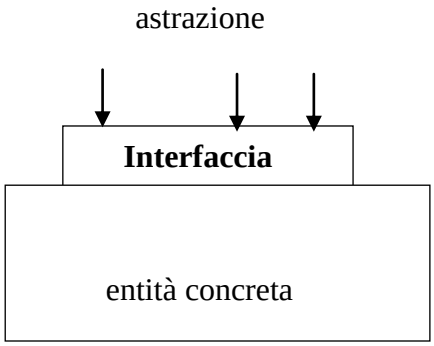
Interfacce SW

Nella costruzione del software di qualità i dettagli inessenziali presenti nelle implementazioni si nascondono alla vista dell'utente.

Tale "occultamento" di informazioni (*information hiding*) ha un duplice scopo:

- **Semplificare al massimo l'uso** dell'entità astratta
- **Impedire usi non previsti** dalle funzionalità dell'entità astratta

L'utente deve servirsi di apposita ***interfaccia d'uso*** per accedere (indirettamente) all'entità concreta.



Interfaccia: software che nasconde l'implementazione

Nel senso generale del termine, un **interfaccia** è il punto, l'area o la superficie sulla quale due *entità* qualitativamente *differenti* si incontrano; tale termine viene spesso usato nelle discipline tecniche con il significato di dispositivo fisico (HW) o virtuale (SW) che permette la **comunicazione** fra due o più entità di tipo diverso; ogni entità espone una sua *faccia*, con il suo particolare protocollo di comunicazione e il dispositivo viene interposto fra di esse.

Con **interfaccia utente** si intendono le funzioni di *interpretazione* ed *esecuzione* dei linguaggi dei comandi.

I linguaggi dei comandi sono suddivisi, per grandi linee, in:

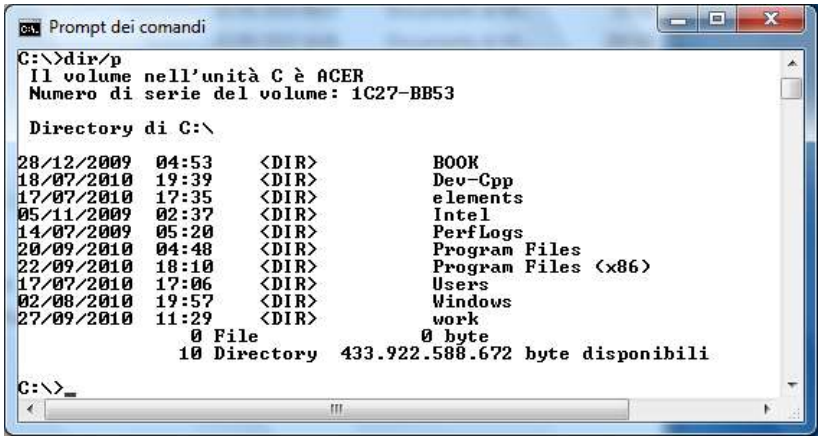
Il Linguaggio Comandi

• **testuali**: i comandi vengono impartiti al sistema attraverso una serie di **parole del linguaggio** scritte (da tastiera) su una riga di comando. Possono essere complessi da utilizzare.

• **a finestre**: le possibilità di esecuzione vengono mostrate in forma di **icone** e i comandi vengono impartiti al sistema attraverso **operazioni eseguite con il mouse** (click, trascinamento di icone). Sono più semplici da utilizzare ma meno espressivi dei linguaggi testuali.

Esempio di **G**raphical **U**ser **I**nterface:
tipica del Sistema Operativo Windows XP

Esempio di interfaccia a "**riga di comando**" (CLI) tipica del Sistema Operativo MS- DOS



Un insieme di **comandi** (parole chiave del **S**istema **O**perativo) vengono immessi da tastiera essenzialmente per gestire il file system (la gestione delle memorie a *supporto magnetico*).

Il **S. O.** organizza, infatti, i file[14] gestendo:
• la struttura delle cartelle (directory) e sottocartelle (subdirectory) organizzate ad albero
• la modalità di memorizzazione su memoria di massa e di recupero da memoria di massa

Seguendo, in ambiente Windows XX, il percorso:

Programmi → Accessori → Prompt dei comandi

ed usando un comando tipico del MS-DOS tra quelli elencati:

dir : mostra i file contenuti nella directory corrente
- **dir/w** per visualizzare il contenuto della directory corrente in forma compatta (videata)
- **dir/p** per visualizzare il contenuto della directory corrente a pagine con informazioni aggiuntive

cd : cambia la directory corrente
- **cd** per **risalire** alla directory principale (*root* o radice di un albero che si dirama in sottodirectory)
- **cd..** per **risalire** alla "directory padre" (Cambia Directory)

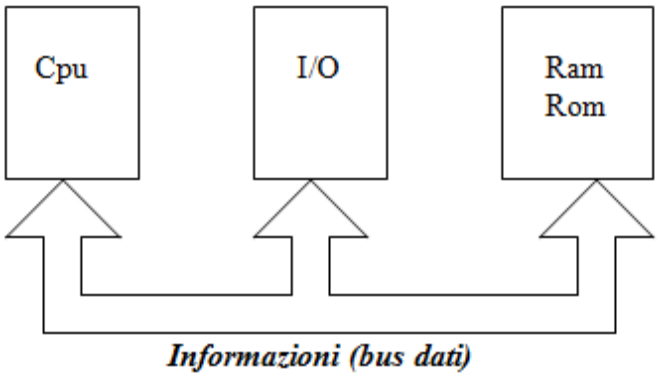
ren : cambia il nome di un file
copy : copia il file in un'altra directory
del : cancella un file
cls : cancella lo schermo

[14] Per **file** si intende un insieme di dati e/o istruzioni archiviati su *supporto magnetico* (memorie di massa)

Hardware: architettura base di un calcolatore

Nell'architettura di un **sistema a microprocessore** (modello logico-funzionale di Von Neumann) si distinguono tre grandi blocchi funzionali: CPU, I/O (periferiche) e memorie che **condividono un unico canale per scambiarsi informazioni** (il bus dati).

Tale contemporaneità si indica col termine architettura "*parallela*".



- **CPU** (*central processing unit*, o Microprocessore): con funzioni di elaborazione (operazioni aritmetico-logiche) e controllo
- **I/O** (input/output): dispositivi periferici[15] per consentire lo scambio di informazioni tra l'elaboratore e l'esterno Costituiscono l'interfaccia HardWare con l'operatore umano. Periferica standard di output è il video (monitor), quella di input la tastiera (keyboard).
- **Memoria di lavoro** o **centrale** (a semiconduttore): ***RAM*** *(Random Access Memory)* per memorizzare temporaneamente dati e programmi (volatile[16], con veloci tempi di accesso da parte della CPU, costosa). L'unità di misura è il Byte (8 bit).
- ***ROM*** *(Read Only Memory)* non modificabile, non volatile *(istruzioni incise su chip o* ***firmware*** *per gestire la fase di avvio all'accensione: esecuzione di test diagnostici sull'HardWare e bootstrap: lettura da disco con caricamento del Sistema Operativo d'avvio)*

Prende nome di **bus di sistema** il collegamento tra gli elementi funzionali illustrati; tale bus fornisce il supporto fisico per la trasmissione dei dati tra i vari elementi. Collega due unità funzionali alla volta: una trasmette e l'altra riceve. Il trasferimento avviene sotto il controllo della CPU. Su questo supporto (costituito da più linee) viaggiano dati, indirizzi e comandi. Tali linee si distinguono spesso logicamente[17] in:

- bus dati (**data** bus)
- bus indirizzi (**address** bus)
- bus controllo (**control** bus)

Bus dati: bidirezionale. Serve per lo scambio di informazioni cioè trasmettere dati dalla memoria al registro dati o viceversa.

Bus indirizzi: unidirezionale. Serve per individuare il dispositivo periferico o la cella di memoria che può accedere al bus dati per operazioni di lettura o scrittura.

Bus controllo: ogni linea è unidirezionale. Serve per gestire le diverse modalità nello scambio informativo: lettura o scrittura con la memoria, comando di stampa verso una periferica etc…
Se la dimensione (numero di bit) del bus dati è uguale alla dimensione della parola si può trasferire in parallelo un intero dato. Altrimenti occorrono più trasferimenti. Esistono vari tipi di bus di sistema ad esempio ISA, EISA, PCI.

---

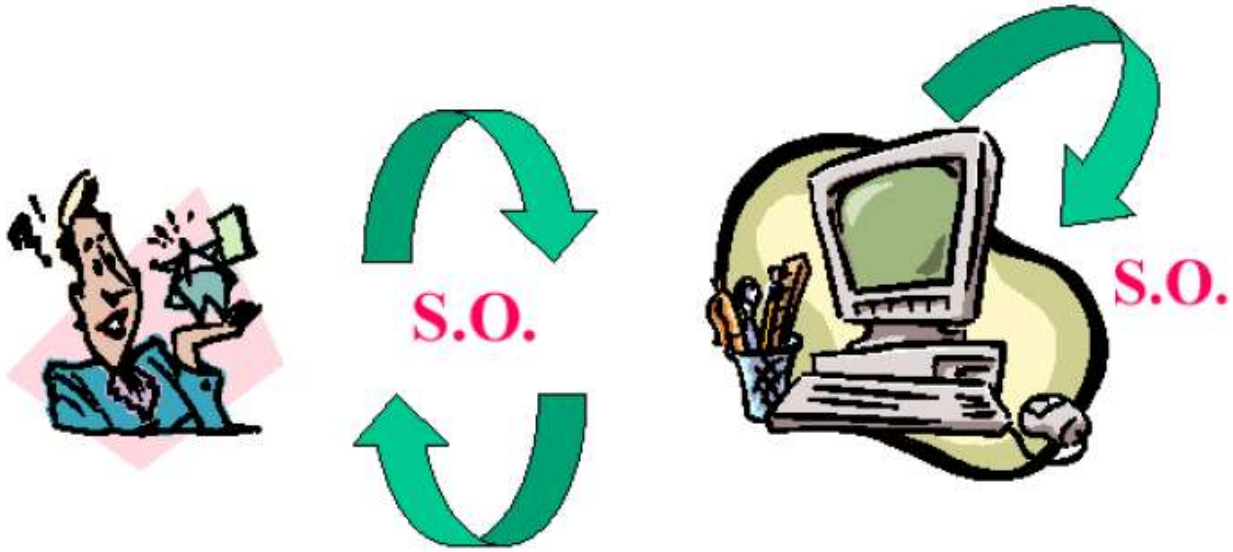
[15] Anche la memoria di massa è vista come dispositivo periferico.

[16] **Volatilità**: perdita del contenuto informativo in assenza di alimentazione

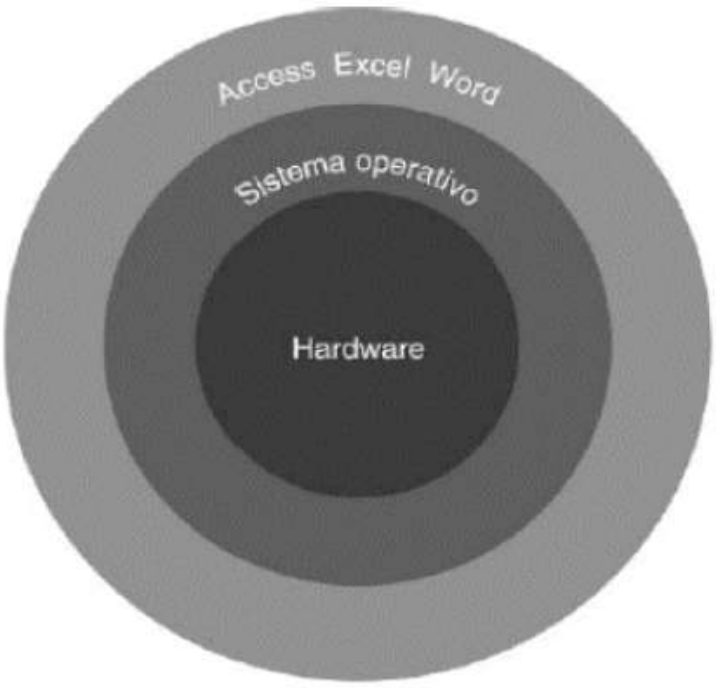
[17] Fisicamente le stesse linee possono essere usate, in tempi diversi, per funzioni diverse (ad esempio prima per indirizzare il dispositivo che accede al canale condiviso e poi per trasmettere dati)

SO

Un Sistema Operativo è un insieme complesso di programmi che, interagendo tra loro, devono svolgere una serie di funzioni per **gestire il comportamento** del computer e per agire come **intermediario** consentendo all'utente un utilizzo della macchina semplice ed economico.



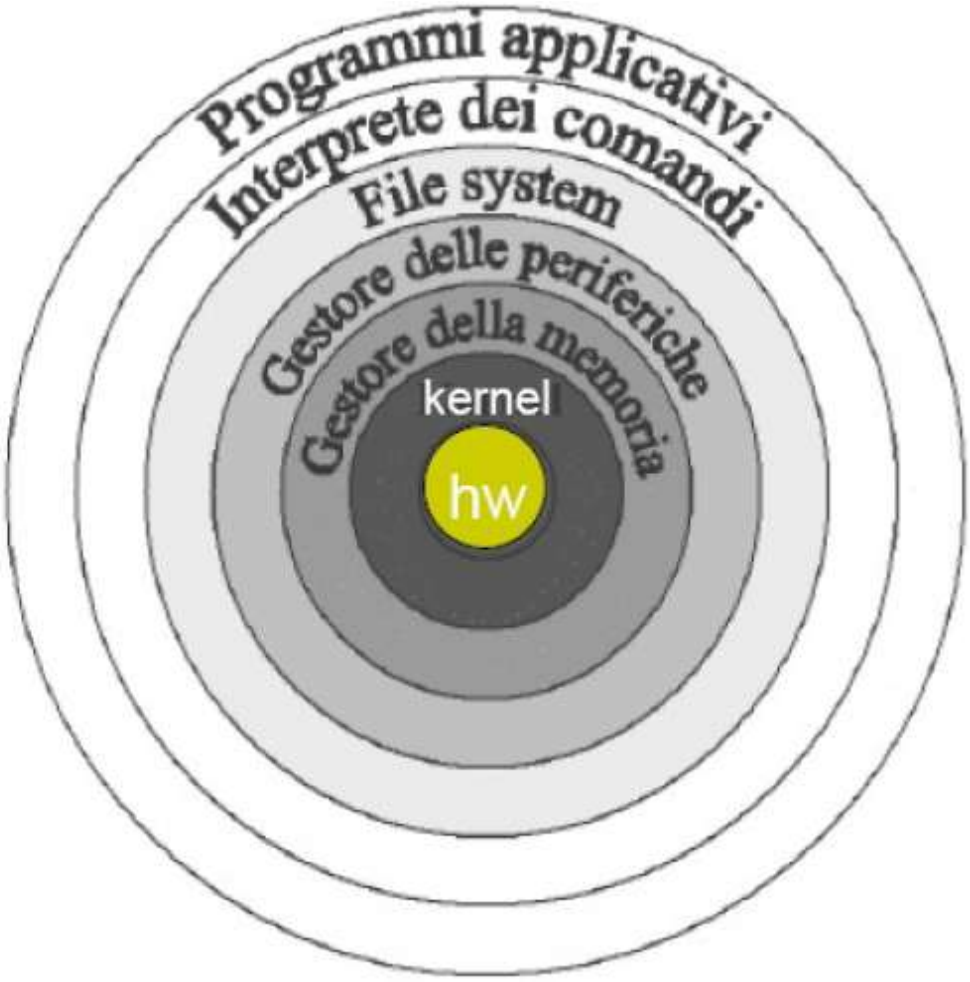
In un modello a strati il SO si pone come un guscio (shell) tra la macchina reale (HW) e le **applicazioni**[18] :



[18] Programmi usati per risolvere problemi in ambito specifico

Le funzioni di un SO si possono illustrare con una architettura a strati:

- La gestione della **CPU** (kernel)
- La gestione delle **memoria di lavoro** a semiconduttore (RAM)
- La gestione delle interfacce HW o **periferici** (tastiera, mouse, video, …)
- La gestione delle **memorie a supporto magnetico-ottico**
- L'**interpretazione** dei comandi



Ricordiamo alcuni termini

Algoritmo: elenco **finito** di direttive che un esecutore **discreto** interpreta **univocamente** (senza ambiguità) e può portare a **termine** in un numero finito di passi, producendo un risultato per ogni ingresso (**complessità limitata** delle istruzioni)

Astrarre: descrivere un oggetto (entità concreta) indicandone le caratteristiche più importanti per farsi o darne un'idea il più precisa possibile cioè "facendo in modo che l'entità astratta corrispondente esibisca la stesse funzionalità dell'entità concreta"

astrazione: idealizzazione di un oggetto descrivendone l'essenziale, le funzionalità caratteristiche
Citazioni:

*"In ambito informatico un'**astrazione** si dovrebbe usare esclusivamente mediante un'interfaccia cioè l'utente deve servirsi di un'apposita* interfaccia d'uso *per accedere (indirettamente) all'entità concreta.*
*Tutta la programmazione è permeata dal concetto di astrazione: un'**applicazione su calcolatore** può essere vista come una gerarchia di macchine astratte annidate una dentro l'altra: ad ogni macchina si fa corrispondere un linguaggio ed un **livello di astrazione**"*

Implementare: concretizzare un'idea
"realizzare un'entità astratta facendo in modo che l'entità concreta corrispondente esibisca la stesse funzionalità dell'entità astratta" (neologismo informatico)

implementazione: sia l'attività di realizzazione di un'astrazione sia il prodotto di questa attività cioè l'entità concreta.

Informatica:

> Scienza che si occupa dei processi e tecnologie che consentono il trattamento (creazione, raccolta, elaborazione, memorizzazione, comunicazione) automatico e razionale dell'informazione e della progettazione degli strumenti che concretizzano tali funzioni. Termine che deriva dal francese (*information automatique*),

Paradigma di programmazione: modello ("chiave di interpretazione", "punto di vista") cui la mente di chi legge o scrive un programma deve idealmente conformarsi

Problema risolvibile: qualsiasi quesito la cui *soluzione* possa essere rappresentata da una macchina (*automa: sistema discreto* cioè caratterizzato da un avanzamento per passi successivi dove ogni ***passo*** produce nell'automa un cambiamento di *stato;* inoltre ogni stato è descrivibile mediante una stringa finita di *simboli* tratti da un insieme prefissato) o, equivalentemente, da un *programma* per un calcolatore.

Programma (nel contesto dei linguaggi *imperativi*): una **sequenza di istruzioni**, scritte in un **linguaggio comprensibile al calcolatore** che le interpreta ed esegue per ottenere i risultati richiesti

Flusso di controllo: la descrizione a priori di tutte le possibili sequenze nell'esecuzione dei passi dell'***algoritmo***, in particolare di operazioni in alternativa e di operazioni da ripetere più volte ciclicamente (*strutture di controllo*)

Flusso di esecuzione: la sequenza di operazioni effettivamente seguita durante una particolare esecuzione del ***programma*** e che dipende dai particolari valori che i dati assumono in quell'esecuzione

Sistemi automatici

Definizione di concetti della **teoria elementare dei sistemi**:

1. **Sistema**: insieme di più elementi[19] che interagendo tra loro danno luogo ad una nuova entità con un determinato scopo (funzionalità).

2. **Modello**: descrizione del reale che evidenzia i parametri o le caratteristiche essenziali per ridurre la complessità nel raggiungere un determinato obiettivo. E' un'astrazione[20] del sistema.

3. **Simulazione** (del comportamento del sistema): *strategia risolutiva* che, attraverso modelli noti, produce risultati che rappresentano una o più possibili storie (evoluzioni) di un sistema dinamico e permette di fare **previsioni**.

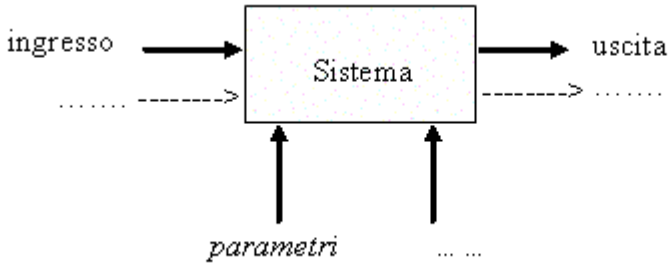
   Tale *strategia* prevede:

   - La definizione dell'obiettivo
   - L'identificazione del sistema (*scomponendo* in eventuali sottosistemi)
   - La definizione dei vincoli (limiti reali e del modello)
   - Modello di massima e rigoroso
   - Simulazione, spesso usando il computer (simulazione *numerica*), per ottenere per diverse eccitazioni un numero molto elevato di possibili risposte

   Tale descrizione dinamica di un sistema non è mai priva di errori perché i modelli sono della approssimazioni. La scelta del risultato cercato si risolve, poi, nel vagliare l'insieme dei risultati, alla ricerca di quello che verifica una prefissata condizione (*criteri di scelta*)

Rappresentazione *sistemica* (paradigma ingresso-uscita): descrizione a blocchi funzionali

Rappresentazione grafica che distingue tra variabili in ingresso *(grandezze su cui possiamo agire per introdurre modifiche)* e in uscita o risposte *(grandezze che risultano influenzate e possiamo osservare per studiare sperimentalmente l'andamento)* individuando gli eventuali *parametri costanti* :



NB: Nel caso di simulazione al computer, uno dei *parametri* è il tempo di discretizzazione cioè l'intervallo tra campioni da fissare poiché un **esecutore discreto** è in grado di distinguere solo un **numero finito di valori** e non consente di analizzare infiniti valori istantanei.

[19] Un sistema non è un oggetto ma la definizione di un *ambito* cioè dei **limiti di analisi**.

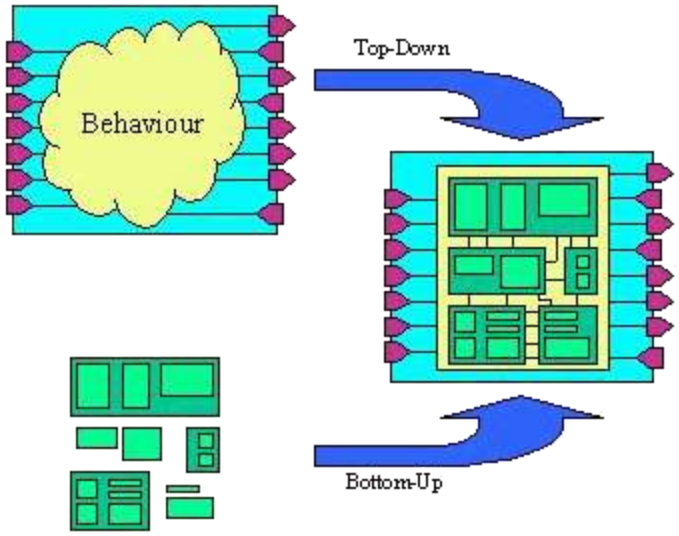
[20] **astrazione**: idealizzazione di un oggetto descrivendone l'essenziale, le funzionalità caratteristiche.

Approcci nel progetto di un sistema automatico

Top-Down:

una metodologia di impostazione di un progetto che parte dalla **chiara idea** delle **funzionalità** del **sistema** che si vogliono realizzare e per ***scomposizione*** in sottoproblemi ed ***approssimazioni successive*** progetta le relazioni tra blocchi funzionali di più semplice realizzazione.

Con **chiara idea** delle **funzionalità** della scheda



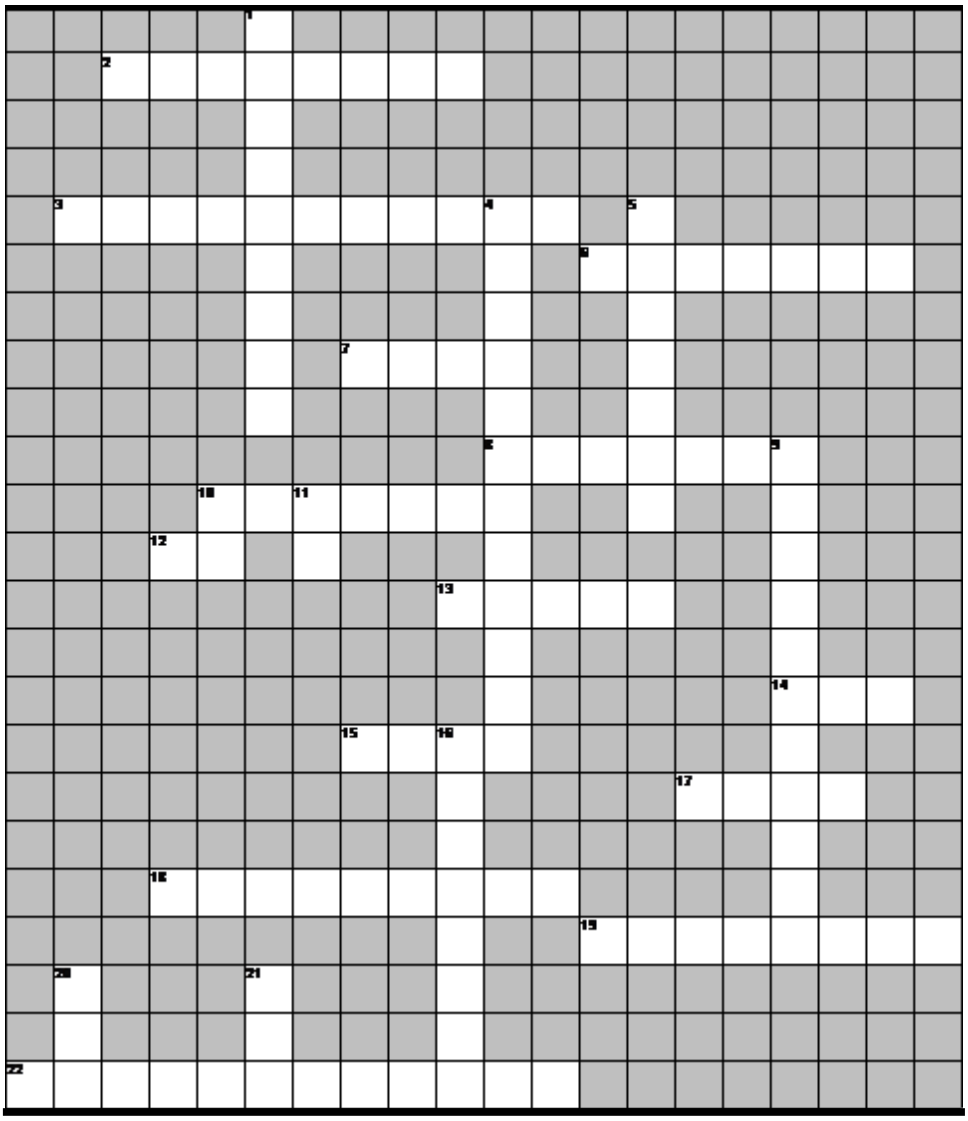
si progetta e realizza la scheda

Dalla **conoscenza** del funzionamento dei **singoli componenti**

Bottom-Up:

un approccio nel progetto di un sistema che a partire dalla conoscenza del funzionamento dei **singoli componenti,** con opportuna modellizzazione (cioè trascurando i particolari inessenziali) giunge alla realizzazione delle interconnesssioni che realizzano il sistema con le funzionalità volute

Verifica: Terminologia



Orizzontali (Across)

2. farsi un'idea per risolvere un problema
3. software che nasconde l'implementazione
6. descrizione semplificata che evidenzia caratteristiche considerate essenziali avendo individuato uno scopo
7. misura reale manipolata dal computer come simbolo
8. dispositivo, parte di un computer, dove immagazzinare dati e programmi
10. ambito di analisi o insieme di elementi che interagiscono per realizzare determinate funzionalità
12. acronimo per indicare i dispositivi che costituiscono un computer
13. periferica di output standard
14. acronimo per individuare l'unità che elabora dati e svolge funzioni di controllo nell'HW di un PC
15. computer connessi per comunicare e condividere risorse
17. unità di misura per i dispositivi di memoria
18. elenco finito di comandi espresso in linguaggio formale
19. termine inglese per riferirsi alla periferica di input standard
22. concretizzare un'idea (neologismo informatico)

Verticali (Down)

1. sequenza di istruzioni in linguaggio comprensibile al calcolatore
4. dato più significato
5. termine inglese per riferirsi alla periferica di output standard
9. programma usato per risolvere problema in ambito specifico
10. acronimo per individuare programmi che girano su PC
11. acronimo per l'insieme di programmi di base con la duplice funzione di gestire l'HW e permetterne uso semplice
16. periferica di input standard
20. acronimo per indicare una memoria di sola lettura
21. acronimo di dispositivo per memorizzare temporaneamente dati e programmi