DAL PENSIERO COMPUTAZIONALE AL CODING

“I think everybody in this country should learn how to program a computer because it teaches you how to think”.

STEVE JOBS

Camera di Commercio
Roma

A cura della Struttura "Orientamento al lavoro e digitalizzazione"

SOMMARIO

Introduzione
Il pensiero computazionale
L'astrazione
Il problem solving
La ricorsione
La generalizzazione
La logica
La logica booleana
Gli algoritmi
Regole algoritmiche
I flow-chart
Il coding
Come si scrive un codice
I linguaggi di programmazione
Conclusioni

INTRODUZIONE

La recente riforma scolastica prevede che gli alunni siano istruiti sui fondamentali della scienza dei computer, inclusi l'astrazione, la logica, gli algoritmi e la presentazione dei dati.

Devono cioè poter analizzare i problemi in termini computazionali e risolverli attraverso l'esperienza diretta nella scrittura di programmi; devono inoltre conoscere e applicare le tecnologie informatiche, incluse quelle con cui non hanno familiarità. Devono avere infine un approccio creativo, autonomo e reattivo alle tecnologie informatiche.

In particolare, la riforma individua tra gli obiettivi formativi prioritari *"lo sviluppo delle competenze digitali degli studenti, con particolare riguardo al pensiero computazionale, all'utilizzo critico e consapevole dei social network e dei media nonché alla produzione e ai legami con il mondo del lavoro"*.

IL PENSIERO COMPUTAZIONALE

“Il pensiero computazionale comporta la formulazione di un problema e l’elaborazione di una soluzione in maniera che un umano o una macchina possano effettivamente portarli a termine”.

Al di là delle sue numerose definizioni, il pensiero computazionale è uno specifico approccio alla risoluzione dei problemi: ciò vuol dire saper pensare in termini “spiegabili” ad una macchina, analizzando un problema ed elaborando una soluzione.

Come un artigiano pensa al proprio lavoro prendendo in considerazione le caratteristiche dei propri strumenti, così il pensiero computazionale elabora una situazione o un problema in modo che sia gestibile attraverso la tecnologia informatica.

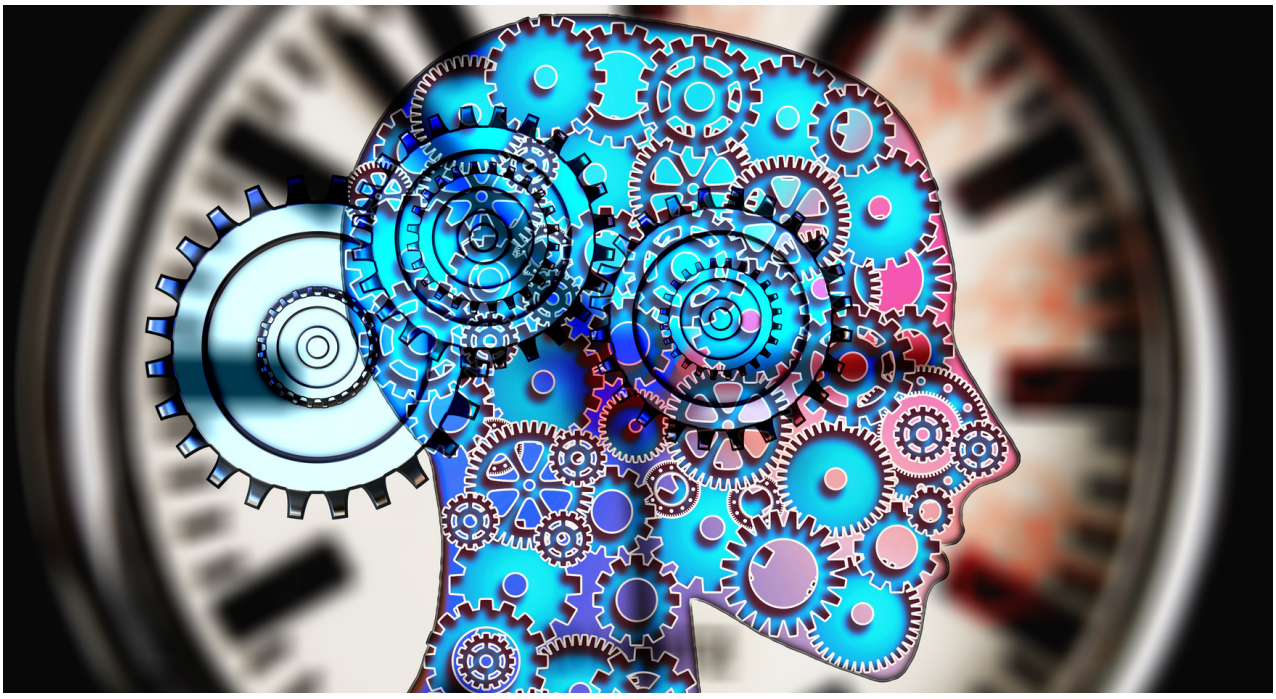
Per legare la parte computazionale alla procedura rigorosa di risoluzione di un problema serve un algoritmo, cioè un insieme di piccoli *step* da eseguire per portare a termine un compito preciso.

IL PENSIERO COMPUTAZIONALE

Tutti conoscono gli algoritmi: quando si comunica una ricetta di cucina o si danno delle indicazioni stradali, si trasmette ad un'altra persona un algoritmo.

Come tradurlo però in maniera semplice, rigorosa e comunicabile senza ambiguità?

Con un linguaggio di programmazione: in tal modo la procedura adottata per la risoluzione del problema non sarà comprensibile solo ad altri esseri umani, ma anche alle macchine, che in pochi secondi ne verificheranno il funzionamento.



L'ASTRAZIONE

Il pensiero computazionale complementa, estende e combina approcci già esistenti, quali il pensiero matematico e il pensiero ingegneristico.

L'astrazione è alla base del pensiero matematico; a sua volta l'informatica basa le sue definizioni formali sulla matematica. Diversamente dalle definizioni matematiche e logiche, le astrazioni create dal pensiero computazionale sono tangibili in quanto processabili.

Il pensiero computazionale ("pensare come un informatico") implica osservare il mondo ragionando per modelli astratti, creare astrazioni processabili e tangibili, avere la possibilità di testare le proprie astrazioni.
Questo è il pensiero computazionale: un processo di *problem solving* che include l'uso del *personal computer*.

IL PROBLEM SOLVING

Per *problem solving* si intende il complesso delle tecniche e delle metodologie necessarie all'analisi di una situazione problematica, allo scopo di individuare e mettere in atto la soluzione migliore.

Per risolvere un problema non c'è una ricetta infallibile, tuttavia utilizzare il giusto approccio è già parte della soluzione.

Al di là della creatività (che non solo non guasta, ma è parte necessaria del processo), la risoluzione dei problemi passa per quattro stadi fondamentali:

- **Definizione del problema**
- **Ricerca delle soluzioni**
- **Scelta della soluzione**
- **Attuazione e verifica**

IL PROBLEM SOLVING

Innanzitutto capire quale sia lo scopo che perseguiamo e cosa lo ostacoli. Questa è forse la parte più complessa del *problem solving*, perché questi aspetti potrebbero non esser affatto chiari.

È necessario analizzare la situazione da differenti prospettive, dividendo il problema in parti più semplici e comprensibili, senza lasciare nulla di implicito. Lo scopo perseguito e il problema devono essere espressi chiaramente.

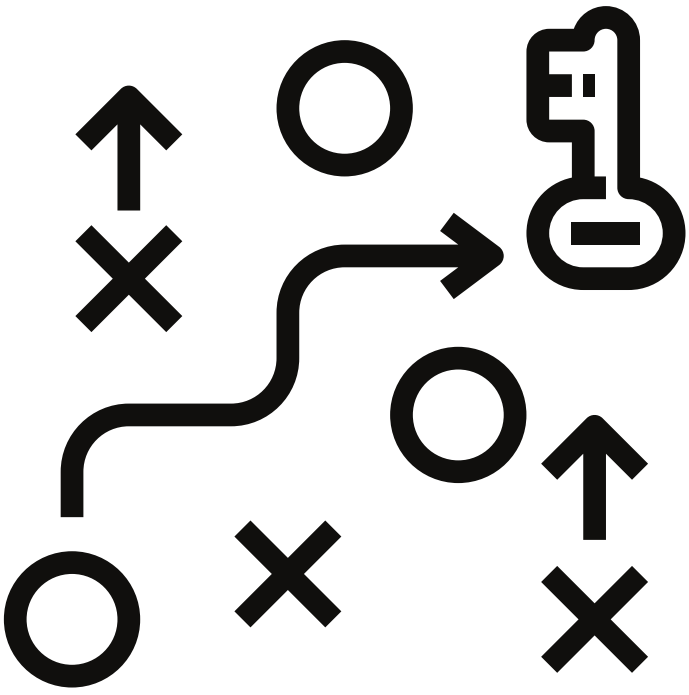
Possono utilizzarsi molte strategie, come il metodo empirico (prova/errore) o l'applicazione di soluzioni già sperimentate per casi simili (analogia).
A livello di pensiero computazionale, tuttavia, la "ricorsione" è l'approccio più utile allo scopo.

LA RICORSIONE

Dovendo allo stesso tempo capire un problema complesso e “spiegarlo” ad una macchina, bisogna pensare in piccolo e **scomporre il problema in problemi più piccoli, provando a risolverli uno alla volta** o portandoli a un livello di grandezza approcciabile in base alle nostre capacità.

Così facendo, il problema iniziale assume una struttura ad albero [problema -> sotto problemi -> sotto-sotto problemi, etc).

A quel punto, dobbiamo capire quali sono i sentieri e quali percorrere per arrivare alla radice del problema.



LA RICORSIONE

Ci sono altre strategie che possiamo adottare oltre alla **ricorsione:**

- Pensare criticamente, testando sistematicamente le proprie teorie. Non dare nulla per scontato

- Risolvere problemi concreti: un approccio eccessivamente astratto, a volte, non è il migliore possibile. Scomporre un problema astratto fino a giungere a un livello concretamente risolvibile, invece, può essere una buona chiave per trovare la soluzione.

- Esaminare le soluzioni già sperimentate per problemi simili e adattarle al problema concreto (ricordando che simili non significa uguali).

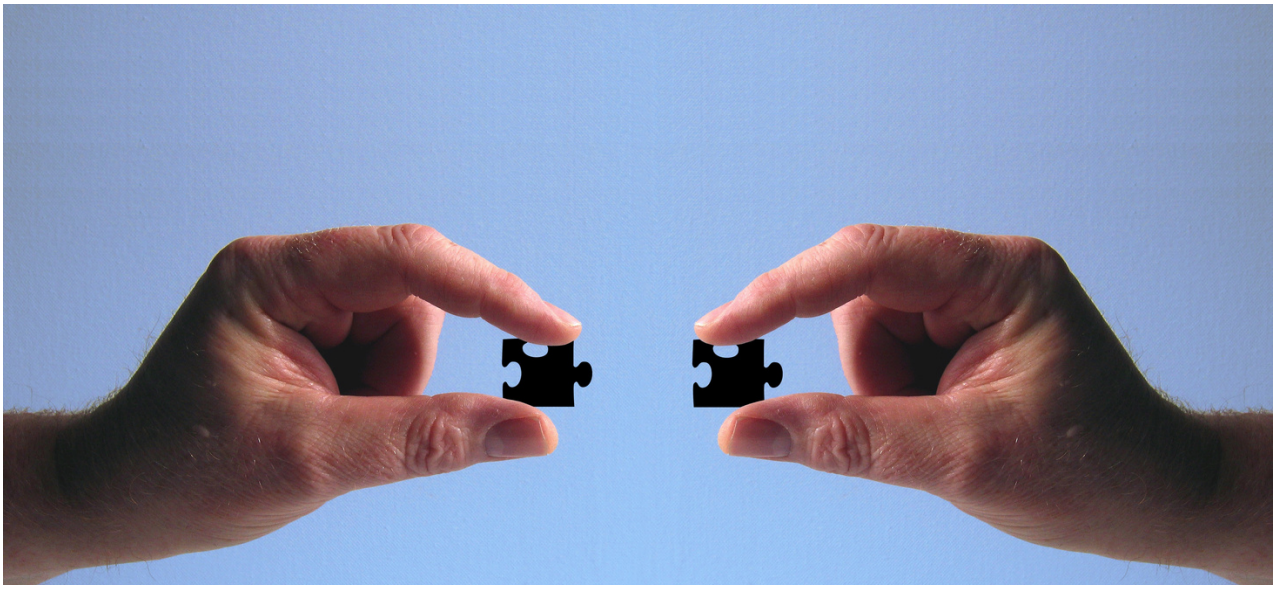
- Lavorare al contrario, cioè ripercorrendo a ritroso ogni passo, dall'obiettivo finale al punto di partenza, deducendo quanto sia necessario per arrivarci.

LA GENERALIZZAZIONE

Un metodo apparentemente contrario a quello ricorsivo è **la generalizzazione.**

Nella generalizzazione si prendono in considerazione gli elementi che abbiamo scomposto per poi considerarli nel loro insieme al fine di desumerne schemi e regole ricorrenti,
In tal modo si ottimizza la soluzione elaborata, rendendola più semplice ed eliminando le ridondanze.

La generalizzazione è una sorta di semplificazione: in un certo senso, possiamo parlare di astrazione.



LA GENERALIZZAZIONE

L’astrazione è un modo di esprimere un’idea in un contesto specifico, escludendo tutti gli elementi non rilevanti allo scopo. Il contesto gioca un ruolo chiave perché è in base a questo che viene determinato cosa l’astrazione deve contenere e di cosa possa fare a meno.

Il processo di astrazione conduce all’elaborazione di un modello (si parla di "modellazione"), cioè una rappresentazione di qualcosa di “reale”, pulita degli aspetti che non interessano; lo scopo non sarà quello di ottenere un modello assolutamente fedele alla realtà, ma che funzioni allo stesso modo.

Non si tratta di qualcosa di nuovo legato all’informatica, quanto di un processo mentale che l’uomo pratica dalla notte dei tempi, usando l’immaginazione e lavorando attraverso ipotesi.

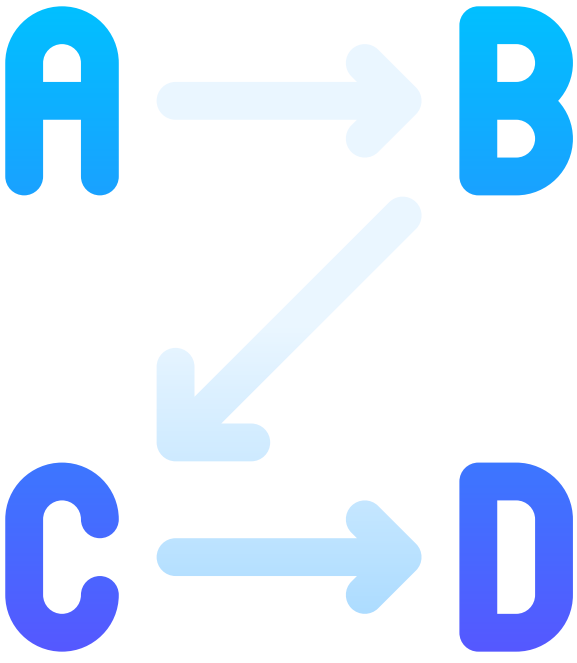
LA LOGICA

Le categorie mentali essenziali del pensiero computazionale sono la Logica e gli Algoritmi.

La logica è utile per capire se un argomento sia valido o meno, cioè se A > B e B > C, allora A > C.
Si tratta di questo ed è questo che una macchina è in grado di valutare.

Si possono distinguere due procedimenti: logico deduttivo e logico induttivo.

Logico-deduttivo in cui la conclusione è già inclusa nelle premesse (se A > B e B > C, allora A > C),

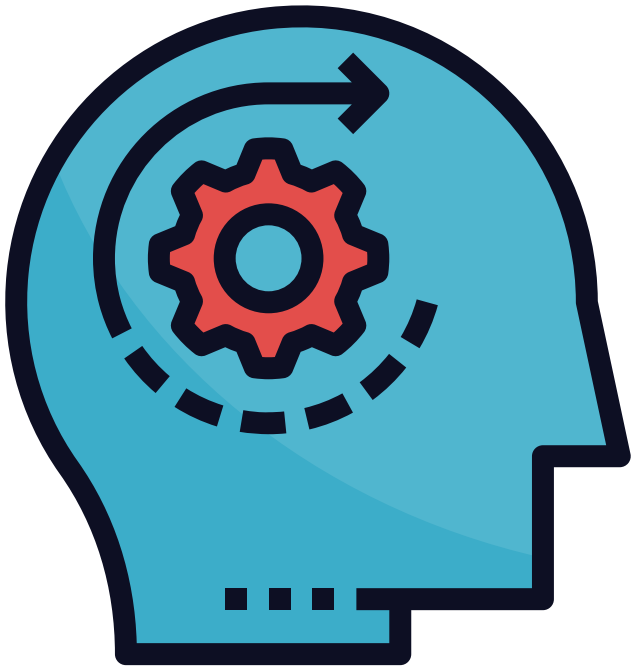


LA LOGICA

Logico-induttivo che coinvolge la probabilità e non la certezza delle premesse per giungere ad una conclusione generalmente valida.

(Es: se un sacco contiene 99 palle rosse e una nera; se 100 persone hanno estratto una palla; se Valeria è tra quelle 100 persone ha probabilmente preso una palla rossa).

Nel nostro quotidiano difficilmente possiamo davvero affidarci ad argomenti deduttivi, perché la complessità ci impedisce di avere delle certezze, perciò tendiamo a ragionare induttivamente, un aspetto che le macchine (e non solo loro) fanno fatica a cogliere.



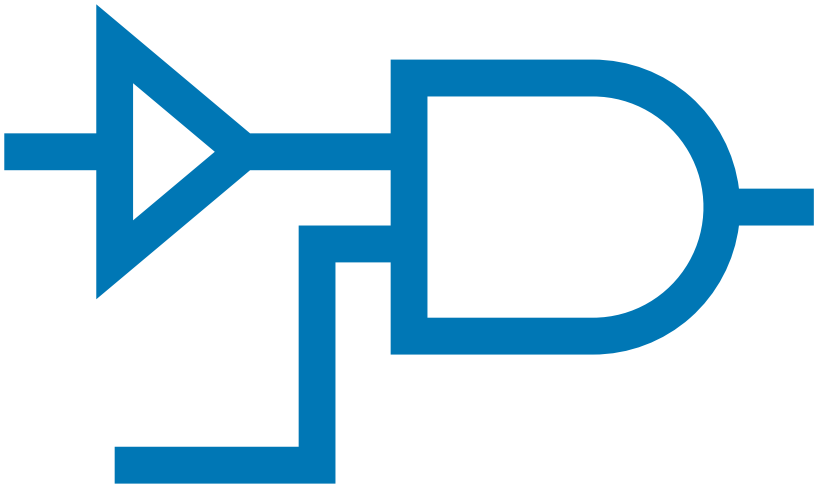
LA LOGICA BOOLEANA

Nella logica booleana, le proposizioni possono essere solo vere o false, 1 o 0. La logica booleana ha alcune **proprietà** tra cui:

- Una proposizione non può essere vera e falsa allo stesso tempo. Non può esserci ambiguità.

- Una proposizione non può avere significati ambigui e le “qualità” devono avere parametri oggettivi di riferimento. (Velocità 40 kmh. Se uomo = Altissima. Se Razzo = Lentissima).

- È possibile combinare più proposizioni individuali in una più complessa attraverso operatori logici, in modo da avere più parametri di valutazione prima di giungere ad una conclusione.



OPERATORI LOGICI

Gli operatori logici booleani sono:

AND	$\wedge$	come la congiunzione “e”, per cui le proposizioni congiunte sono tutte vere per la conclusione
OR	$\vee$	come “o”, per cui almeno una delle proposizioni deve essere vera per la conclusione
NOT	$\neg$	come negazione che non cambia la catena di proposizioni, ma la pone in negativo
IMPLIES	$\rightarrow$	come correlazione tra due proposizioni (vicinanza, non causa-effetto vera e propria)
IF AND ONLY IF	$\leftrightarrow$	come relazione di bicondizionalità, cioè di relazione esclusiva (se e solo se)

OPERATORI LOGICI

Se ragioniamo in termini binari (cioè di 1 = vero e 0 = falso) e giochiamo con qualche esempio possiamo effettuare delle operazioni di logica e vedere il risultato prendendo delle proposizioni A e B.

A	B	A∧B	A∨B	¬A	A →B*	A↔ B
1	1	1	1	0	1	1
1	0	0	1	0	0	0
0	1	0	1	1	1	0
0	0	0	0	1	1	1

**considerando A come antecedente e B come conseguente, in caso di falsità della prima non c'è contraddizione con la verità della seconda.*

GLI ALGORITMI

L’algoritmo è **una sequenza finita e ordinata di operazioni elementari e non ambigue che permettono di risolvere, in maniera deterministica, un problema in un tempo finito.**

L'algoritmo ha sempre un “termine”: si tratta di un procedimento definito con precisione e secondo le regole della logica.

L’algoritmo è un procedimento astratto che ci porta ad una soluzione, ma per fare in modo che questo si adegui a quegli aspetti della realtà che invece possono cambiare deve tener conto delle variabili, cioè dei contenitori in cui vengono conservati determinati valori di cui l’algoritmo deve tener conto per la sua esecuzione.

Ciò detto, parliamo di "pensiero algoritmo" come dell’approccio mentale alla risoluzione dei problemi attraverso algoritmi. Aver chiaro cosa sia un algoritmo però non basta: bisogna sapere come funziona.

REGOLE ALGORITMICHE

- È fondamentale che **le istruzioni siano precise e non ambigue** – considerato che sono dirette ad una macchina e che questa non ha facoltà di scegliere quale significato attribuire loro.

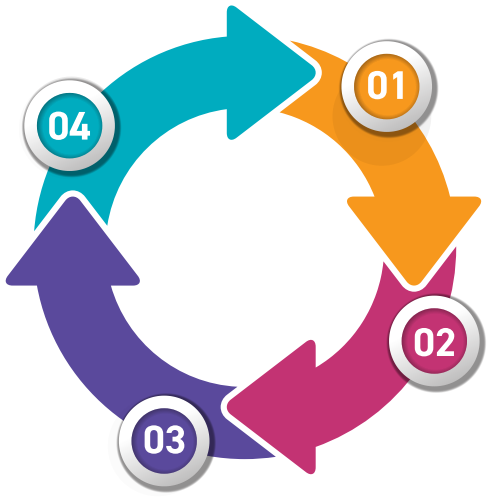
- Saper **usare gli operatori logici** è un altro requisito essenziale – a dispetto del senso comune che le persone possono attribuire alla congiunzione "e", il linguaggio logico ha uno specifico significato che può contraddire le intenzioni ["quelli con il cognome che inizia per B e C vengono selezionati per primi" -> ERRORE: una parola può iniziare con una lettera o un'altra, non entrambe].

- Infine, quando si tratta di definire una correlazione come "se-allora", bisogna tener conto dell'ipotesi "altrimenti" **(if-then-else)**, definendo il valore da assegnare se la proposizione è falsa e, nel caso di condizionalità più complesse, considerare il preciso ordine di precedenza delle operazioni logiche, in modo simile alle operazioni matematiche.

I FLOW-CHART

L'algoritmo può essere rappresentato in vari modi, grafici o testuali. Uno dei metodi grafici più usati e conosciuti è il cosiddetto **diagramma di flusso**, ciascun componente del quale ha un significato ben determinato. Nell'esempio illustrato sotto, sono riportati i vari elementi di un diagramma di flusso e il relativo significato.

È importante sottolineare la differenza tra i diversi elementi. Intanto c'è sempre un punto di inizio e uno di fine. I blocchi rettangolari rappresentano una operazione generica descritta al suo interno, mentre il rombo costituisce sempre una "scelta": in base alla valutazione della condizione racchiusa, viene seguito un certo percorso invece di un altro.
Esistono poi altri blocchi per l'ingresso di dati e informazioni contenuti nelle variabili.

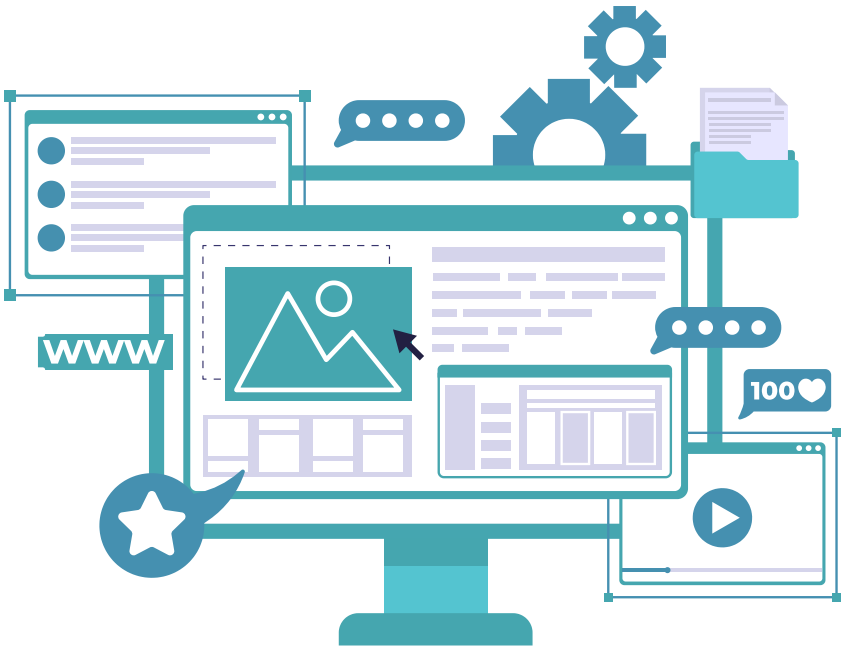


IL CODING

“**Una macchina fa quello che gli dici, non quello che vuoi”**.

Imparare il *coding* significa spiegare alle macchine cosa dovrebbero fare per noi. Per riuscirci, bisogna prima pensare in termini computazionali e, successivamente, apprendere un linguaggio di programmazione che ci permetta di "parlare" con i dispositivi.

Fondamentalmente, **le macchine fanno bene soltanto due cose: conservare dati e compiere operazioni con essi**.



IL CODING

Il *coding* può sembrarci più o meno logico oppure qualcosa di completamente astratto: molto dipende da quanto a fondo sono radicati nel nostro modo di ragionare alcuni concetti fondamentali. La pratica, in tal senso, non è affatto opzionale.

Per scrivere un programma dobbiamo aver chiaro cosa vogliamo che la macchina faccia per noi, ma a volte pensiamo troppo in grande e vogliamo saltare dal problema alla soluzione direttamente.

In questo caso, dobbiamo utilizzare l'approccio chiamato "Ricorsione" che, come già detto, permette di scomporre il problema in problemi più piccoli per provare a risolverli uno alla volta o per portarli ad un livello di grandezza che sia approcciabile in base alle nostre capacità.

IL CODING

Così facendo la situazione viene scomposta in più passaggi, come fosse una ricetta di cucina. Raramente però verremo a contatto con una realtà univoca e immutabile, quindi è necessario stabilire anche delle condizioni per dei “piani di riserva”.

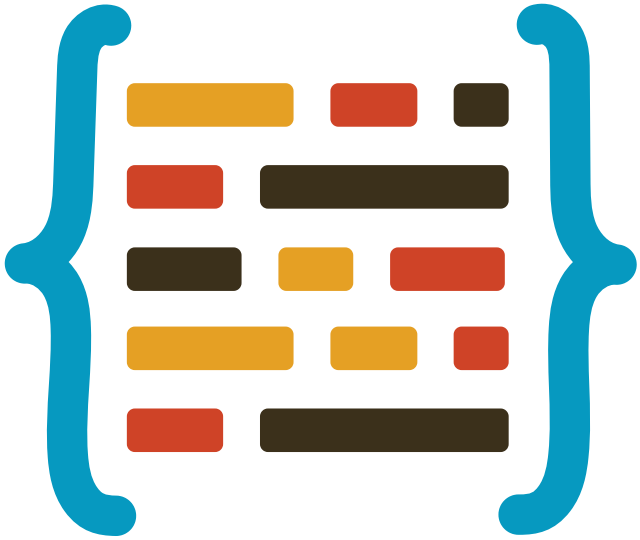
Non è un caso che molti libri che parlano di *software* siano chiamati “libri di cucina” o "*cookbooks*"; da questa prospettiva, infatti, un algoritmo non è altro che una ricetta, un insieme di istruzioni da eseguire in sequenza per risolvere un problema.

Spesso, nel passaggio dal pensiero umano a quello computazionale, gli algoritmi vengono scritti in pseudo codice, cioè in maniera rapida e informale ma comunque con la descrizione dei passaggi necessari; successivamente lo pseudocodice viene tradotto nel linguaggio che la macchina può capire ed eseguire.

IL CODING

Possiamo applicare la ricorsione anche al *coding*, per cui dobbiamo:

- Prima creare un algoritmo che descriva la procedura che dovrà effettuare il dispositivo;
- Tradurlo in codice di programmazione, creando il nostro codice sorgente;
- Avviare il programma e seguire le istruzioni impartite.



IL LINGUAGGIO DI PROGRAMMAZIONE

Un linguaggio di programmazione è un linguaggio speciale, creato espressamente per impartire istruzioni ad un dispositivo; saper programmare vuol dire quindi esprimere quelle istruzioni in modo comprensibile per la macchina. Come per qualunque linguaggio, ci sono due aspetti che è necessario padroneggiare:

La sintassi – cioè come devono essere scritte le istruzioni;

La semantica – cioè il significato delle istruzioni.

Così come esistono molte lingue, ognuna con la propria sintassi e la propria semantica, così esistono diversi linguaggi di programmazione, con caratteristiche e scopi molto diversi.

Per questo motivo è importante imparare il pensiero computazionale, perché è una forma mentale che si applica indistintamente a tutti i linguaggi.

IL LINGUAGGIO DI PROGRAMMAZIONE

Quando si scrive un codice si ha a che fare con valori che cambiano di volta in volta, **le variabili.**

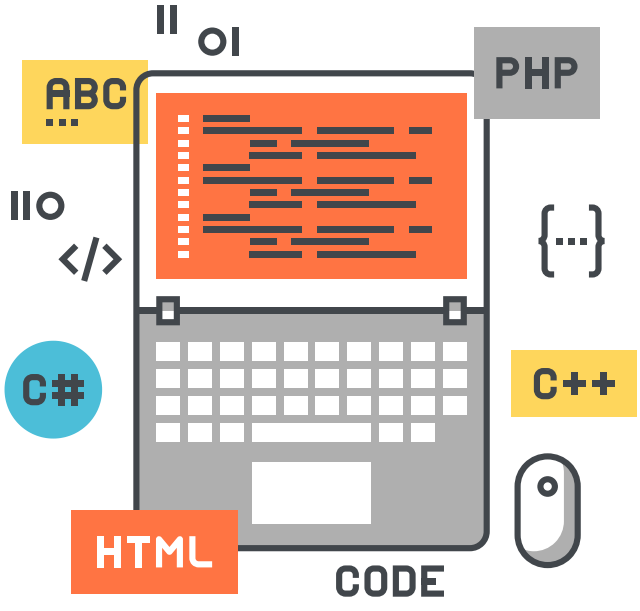
Indipendentemente dallo specifico linguaggio, le variabili vengono conservate all'interno del codice assegnando a ciascuna un valore, spesso determinato dall'utente che si troverà ad interagire col programma.

(Per esempio, quando un programma chiede di inserire il nome, la data di nascita etc., il codice determinerà le variabili "nome" e "data di nascita" in base a quanto inserito dall'utente e le conserverà per le attività successive, del tipo ciao "nome" !).

IL LINGUAGGIO DI PROGRAMMAZIONE

Tutte le volte che dovremo riutilizzare quel valore, nel codice utilizzeremo il nome della variabile in cui è conservato.

È nella natura delle variabili il cambiamento e quando si tratta di variabili dipendenti, quando ne cambia una, bisogna spiegare alla macchina come considerare le altre.
Qui la matematica e, nello specifico, l'algebra ci aiutano un bel po'. All'interno del codice possiamo effettuare operazioni con le variabili, come se si trattasse di incognite algebriche.



COME SI SCRIVE UN CODICE

Normalmente, viene preparata una bozza attraverso un programma di elaborazione testuale, successivamente salvata in un file testuale, che può essere copiato direttamente nell'*editor* dedicato al linguaggio di programmazione.

Molti di questi *editor* (o **IDE – Integrated Development Environments**) dispongono di funzioni accessorie che permettono di correggere gli errori o di fornire indicazioni su come dovremmo compilare il codice, un po' come la correzione automatica di Word.

Scritto il codice, lo si affida ad un **interprete,** che effettua tutti i passaggi necessari per eseguire le istruzioni in esso contenute.

COME SI SCRIVE UN CODICE

Qui è il caso di aprire una parentesi e distinguere due tipologie di linguaggio: **linguaggio macchina** (detto "a basso livello") e **linguaggio di programmazione** (detto "ad alto livello").

A differenza del secondo, il linguaggio o codice macchina non è espresso con utilizzo di sintassi o caratteri comprensibili all'essere umano, ma in codice binario, cioè serie lunghissime di 0 e 1.

I linguaggi di programmazione sono quindi codici semplificati che vengono poi tradotti in codice macchina.

I LINGUAGGI DI PROGRAMMAZIONE

Conoscere i linguaggi di programmazione più richiesti è essenziale per sapere quali competenze accrescere, al fine di essere più competitivi nel mondo del lavoro.

Ecco i 10 linguaggi più richiesti dal mercato:

- **Python:** linguaggio estremamente versatile e completamente gratuito, ideale per essere usato nei settori dell'intelligenza artificiale e del *machine learning*;

- **Java:** molto diffuso anche perché è uno dei linguaggi più stabili, completi e affidabili per costruire sistemi complessi;

- **JavaScript:** linguaggio di programmazione comunemente utilizzato nella programmazione web lato *client* per la creazione di siti web;

- **C e C++:** linguaggi utilizzati prevalentemente nell'ambito dei *videogames*;

I LINGUAGGI DI PROGRAMMAZIONE

- **C#:** si contraddistingue per la sua versatilità;
- **TypeScript:** linguaggio di programmazione *open source* sviluppato da Microsoft, con cui si creano applicazioni di grandi dimensioni;
- **PHP:** sviluppato per la programmazione di pagine web interattive e dinamiche;
- **Perl:** linguaggio usato per una grande varietà di sviluppi;
- **Ruby:** linguaggio di programmazione *open source* utilizzato per realizzare *web app* e servizi web.

CONCLUSIONI

In conclusione, il pensiero computazionale è l'unione del pensiero umano e del sistema informatico e si traduce nella capacità di pensare come una macchina e di far sì che la stessa macchina pensi come noi.

Il *coding* invece non è altro che la codificazione di un programma. Cioè, è quella fase in cui una persona impartisce una serie di istruzioni ad un elaboratore, che a sua volta li trasforma in sequenza e li restituisce sotto forma di *software*.

La conoscenza del pensiero computazionale e del *coding* è quindi fondamentale per qualsiasi attività lavorativa proiettata nel prossimo futuro.

CAMERA DI COMMERCIO DI ROMA

Struttura "Orientamento e digitalizzazione"

via de' Burrò 147
www.rm.camcom.it

orientamentoedigitalizzazione@rm.camcom.it

Luglio 2022