CODING TATTILE con DidaLab, CodyMat e le attrezzature UniDida

GUIDA TEORICA E PRATICA
PER ATTIVITÀ IN CLASSE DI CODING E ROBOTICA
CON L'AUSILIO DEL PIPE-CODING®

VOLUME 1 – Aspetti metodologici e basi del digitale

SEZIONI SPECIFICHE PER:

- **SCUOLA DELL'INFANZIA**
- **SCUOLA PRIMARIA**
- **SCUOLA SECONDARIA 1°**
- **SCUOLA SECONDARIA 2°**

EDIZIONE UNIDIDA

Daniele Costamagna

PRESENTAZIONE

Il ***coding***, cioè l'informatica proposta agli studenti delle scuole italiane, non si è rivelato una moda passeggera, ma sta invece confermandosi come un solido contributo alla formazione scolastica.

UniDida, da sempre impegnata nell'ideazione e sviluppo di prodotti per la didattica digitale, propone agli insegnanti (e agli studenti della secondaria) questa pratica guida dedicata all'uso dei suoi applicativi software e dei sussidi per esperienze in classe. Primo fra tutti il *pipe-coding*®, il metodo inventato da UniDida per semplificare i linguaggi di programmazione mediante il paradigma dell'acqua che scorre nei tubi (*pipes*, in inglese).

Il termine stesso "*Coding Tattile*" è stato ideato da UniDida per rappresentare come il ***coding*** non sia assolutamente solo uno strumento per insegnare l'informatica e la programmazione nelle scuole. Esso è soprattutto un modo per stimolare il pensiero logico e analitico, attraverso un uso ***attivo*** delle tecnologie informatiche. Un metodo che, per adattarsi a tutti, fa uso di ***strumenti tattili***, come il CodyMat, oltre che digitali. Un modo nuovo e attuale per costruire, nei cittadini di domani, una migliore consapevolezza digitale, perché non **siano i dispositivi ad usare gli utenti ma viceversa**.

Questo primo volume si apre con un'ampia **sezione metodologica** per migliorare la consapevolezza dell'importante ruolo di ogni insegnante in questa nuova disciplina. Le esperienze di ***coding*** spesso vengono riservate a docenti di materie specialistiche, mentre si è rivelato sempre più come una conoscenza trasversale che migliora le abilità di fondo (*soft-skills*) degli studenti.

Affrontare argomenti a carattere tecnologico non è sempre facile per docenti che si trovano di fronte a bambini e ragazzi cresciuti con lo *smart-phone* in mano. Per questo **il testo contiene una amplissima raccolta di *discorsi* che possono essere direttamente raccontati o letti in classe**: essi consentono di introdurre tutti i principali concetti del mondo digitale con un linguaggio semplice e spesso divertente.

Il testo è pensato per ogni ordine e grado e riserva molte parti adatte già dalla scuola dell'infanzia, fino agli ultimi capitoli che sono senz'altro più che stimolanti per la secondaria di 2° grado.

Il volume 2, è tutto dedicato agli aspetti pratici con decine di attività pronte per l'uso in classe, in forma di gioco da tavolo, con il CodyPUZZLE, o con DidaLab su LIM e monitor *touch*, e in spazi più ampi con esperienze sul CodyMat, il tappeto morbido per il *coding*.

PREFAZIONE DI ALESSANDRO BOGLIOLO

Professore Ordinario di Sistemi di Elaborazione delle Informazioni presso l'Università degli studi di Urbino Carlo Bo e Europe Code Week Ambassador

Ho conosciuto il pipe-coding in treno, in metropolitana e per strada. Non è stato un caso, ma quasi. Daniele ed io avevamo partecipato ad un evento alla Fiera di Roma e stavamo rientrando in città. Avevamo solo il tempo di quel trasferimento sui mezzi pubblici che percorrevano brevi tratti condivisi prima che altri mezzi pubblici ci portassero verso destinazioni diverse.

La programmazione visuale a blocchi e il coding unplugged si stavano diffondendo come strumenti efficaci per divulgare l'informatica, per esercitare il pensiero computazionale e per offrire supporto alla didattica.

Nel portatile di Daniele, sballottato come noi dai mezzi pubblici di Roma, girava il pipe-coding. Un'interfaccia grafica permetteva di collegare pezzi di tubi e rubinetti attraverso i quali sarebbe fluita l'acqua dall'alto dello schermo. I componenti di quelle tubazioni non erano altro che istruzioni e costrutti che rendevano evidenti i propri collegamenti e la propria funzione sfruttando efficacemente l'analogia idraulica. Non appena si apriva l'acqua e la si lasciava scorrere, i componenti si animavano e producevano le azioni corrispondenti alle istruzioni che rappresentavano, mentre il flusso veniva deviato da costrutti condizionali e restava intrappolato in cicli.

Mentre l'interfaccia grafica del pipe-coding mostrava l'algoritmo in esecuzione e i suoi effetti, il motore che ci stava dietro mostrava ancora più chiaramente la passione, la professionalità e l'orgoglio di chi lo aveva sviluppato e me lo mostrava tenendolo in braccio.

Quello che Daniele mi chiedeva era un parere: "Che ruolo può avere il pipe-coding nel mondo del coding?". La mia risposta è stata "Non lo so, ma mi piace". A quanto pare questo è bastato a Daniele per decidere di continuarne lo sviluppo, il che significa che aveva già deciso di farlo indipendentemente da me...

A quel punto gli ho proposto di portare dentro al pipe-coding anche CodyRoby, il robottino unplugged che fino ad allora aveva vissuto in un mondo di carta.

Da lì a poco il pipe-coding ha visto la luce, dimostrando di avere la versatilità necessaria ad adeguarsi alle esigenze didattiche di ogni fascia di età, soprattutto grazie alla sensibilità e all'attenzione del suo sviluppatore.

Il pipe-coding unisce l'immediatezza della programmazione visuale a blocchi, all'astrazione dei diagrammi di flusso e al rigore dei linguaggi di programmazione testuali, offrendo a chi lo usa la libertà di decidere a quale livello lavorare.

Buona lettura, buon divertimento e buon lavoro.

Alessandro Bogliolo

L'AUTORE

Ingegnere e formatore nel settore della didattica, si è laureato nel 1991 in Ingegneria Elettronica, indirizzo in Automazione e Robotica, presso il Politecnico di Torino. Ha lavorato per circa 20 anni presso società che si occupano di sviluppo software per l'automazione (Comau Robotica e WindRiver System, tra queste).

È titolare di oltre 4 brevetti nazionali ed internazionali nel settore dell'automazione e della didattica.

Fonda nel 2010 **UniDida Srl** e sviluppa *Dida-Framework* che rappresenterà la piattaforma software sulla quale si baseranno tutti i successivi progetti: **Cliccolo**, **Plexy.IT** e, in parte, DidaLab.

Nel dicembre del 2015 avvia lo sviluppo di **DidaLab** che implementa un nuovo approccio alla programmazione grafica a blocchi con l'invenzione del *pipe-coding*®.

A partire dal dicembre 2016 collabora con Alessandro Bogliolo (prof. Università di Urbino e Ambassador per la Europe Code Week) per la sperimentazione di **DidaLab** e del *pipe-coding*®.

A gennaio 2017 inizia il giro d'Italia per il *pipe-coding*® che lo porta a conoscere centinaia di studenti ed insegnanti di ogni ordine e grado ed a sperimentare l'uso di DidaLab e dei laboratori sul coding in base al nuovo approccio.

Spinto dall'interesse riscontrato durante il viaggio, continua a fornire in tutt'Italia, durante il 2017, attività di formazione per docenti e laboratori per studenti sulle tematiche del coding, **S.T.E.M.** e uso delle tecnologie in classe.

A gennaio 2018 ripropone alle scuole d'Italia il Giro del Coding accompagnato da alcuni nuove proposte per la didattica: le novità della versione **3** di **DidaLab** (arricchito del teatro digitale), il CodyMat per sperimentare il coding a pavimento (soprattutto per scuola dell'infanzia e primi

anni primaria) ed il Il CronoPlano (l'ausilio didattico su carta per la storia del terzo e quarto anno della scuola primaria).

Ad aprile 2018 riceve il **primo premio del concorso "Ingegnere Innovativo"** dall'Ordine degli Ingegneri della Provincia di Torino, con il progetto dal titolo: *"Pipe-coding®: un inedito approccio grafico per proporre il coding per scopi didattici a studenti di ogni ordine e grado".*
Durante l'edizione 2018 di didacta (fiera della didattica di Firenze) presenta tre libri sul *pipe-coding®* (per la scuola primaria, secondaria 1° e 2°) con attività pratiche per usare tre applicativi derivati da **DidaLab**: **LollyBee**, **PlexyCode Junior** e **PlexyCode**.

Nell'edizione didacta del 2019 presenta CodyPUZZLE il sussidio didattico per il pipe-coding® in forma di gioco da tavolo adatto dai 5 anni fin oltre ai 15.

A partire dal 2021 sviluppa, in collaborazione con un produttore internazionale di robot industriali, una interfaccia utente che ripropone l'approccio del *pipe-coding®* in versione adatta e funzionale all'uso dei robot da parte di operatori con una minima formazione specialistica.

BENVENUTI

Questo sussidio didattico è dedicato a tutti gli insegnanti che intendono proporre ai loro studenti alcune esperienze di ***coding*** e ***robotica educativa*** attraverso un approccio che integra la manualità con l'uso dei dispositivi digitali. Nella sua componente ***manuale*** si usano ***sussidi tattili***, cioè costruiti con vari materiali (carta, gomma, plastica), e i prodotti sono **CodyPuzzle**, **CodyMat**, **CodyBlock**; nella sua componente ***digitale*** si propone l'uso di un applicativo software, **DidaLab**, nella sua versione 4.

Il testo è pensato prevalentemente per gli insegnanti, ma alcuni capitoli potranno anche essere lasciati in uso agli studenti a partire dagli ultimi anni della scuola primaria, per svolgere gli esercizi in laboratorio o a casa.

L'approccio è graduale ed inizia con un **inquadramento delle motivazioni** che rendono il ***coding*** uno strumento attuale ed efficace per crescere cittadini attivi e consapevoli anche nell'uso delle tecnologie digitali. In questa sezione si definiscono anche i termini fondamentali del mondo digitale.

Vi sono poi le **lezioni** proposte con la ***tecnica del copione***: in altre parole sono scritte per poter essere lette direttamente in classe, parlando agli studenti. Molte sono pensate per spiegare i termini che avrete trovato nella prima sezione in una forma adatta agli studenti.
Ovviamente, il linguaggio usato non è sempre adatto a qualsiasi età (e per alcune, in particolare, si usa un linguaggio pensato per l'infanzia o i primi anni della primaria) ma i contenuti sono universali e la loro trasmissione è parte integrante di questo percorso formativo. Sarà quindi compito dell'insegnante, adattare il linguaggio all'età degli studenti "complicando", se necessario, i concetti che vengono espressi in modo elementare per renderli adeguati ai loro studenti (è il caso, ad esempio, della *Storia sul coding*).

Infine vi sono le **attività pratiche** che sono vere e proprie esercitazioni che potrete proporre in classe con i vostri studenti. Esse iniziano presentando i sussidi didattici e le modalità con cui presentarli all'attenzione degli studenti. Quindi, il testo prende la forma di vera e propria guida pratica per l'esecuzione di attività in classe, presentando esercizi di ***coding*** e ***robotica*** progressivi: dalla scuola dell'infanzia, fino agli ordini superiori. Nella sua parte finale, infatti, si propongono

attività molto più impegnative, adatte ad interessare studenti delle scuole superiori e in particolare quelle dedicate alle scienze della formazione o alle scienze applicate.

Il metodo proposto in questo testo utilizza il **pipe-coding®**, l'innovativo approccio alla programmazione a blocchi usato per mostrare cosa sia un programma e come le istruzioni vengano eseguite dal *computer*.
Gli studenti creano i loro programmi attaccando, uno sotto l'altro, pezzi di tubo nei quali si immagina fluisca l'acqua per indicare l'evolversi del programma. Simpatici personaggi eseguono i comandi spostandosi su scacchiere o spazi liberi e raccogliendo oggetti, lettere o lasciando tracce colorate per comporre disegni geometrici.

Come si vedrà, le attività non si limitano al ***coding*** o alla ***robotica educativa*** ma spaziano dal semplice ***gioco di motricità***, alla ***cittadinanza digitale***. Inoltre, **DidaLab** contiene anche sezioni per fare esperienze con la ***pixel-art*** ed il ***teatro digitale***.

Il ***coding*** diventa, così, uno strumento nelle mani degli insegnanti per sviluppare abilità utili in molte discipline tra cui la letto-scrittura, il calcolo (con il robottino *gnam-gnam*), l'orientamento spazio-temporale (***esplora la tua scuola con il coding***), fino alla programmazione con un linguaggio professionale (***Java***).

Trattandosi di una guida rivolta agli insegnanti, non si è badato a limitare il numero di pagine e sono stati affrontati tutti gli argomenti utili a rispondere alle più diverse esigenze.
I contenuti quindi potranno essere utilizzati in modo sequenziale, oppure saltando alle attività pratiche più conformi ai bisogni specifici.

Si consulti il seguente indice per avere una visione d'insieme sulla varietà delle esperiente proposte.

Sommario

CONVENZIONI NEL TESTO

Nel testo si tengano presenti le seguenti considerazioni e convenzioni.

Glossario	Nel testo vi sono alcune parole il cui significato è riportato e spiegato nel glossario che si trova al fondo. Queste parole sono rappresentante con una sottolineatura tratteggiata come segue: *esempio voce di glossario*. Dopo averla usata per alcune volte in *forma sottolineata*, la parola viene scritta normalmente, lasciando intendere che il suo significato sia stato acquisito precedentemente.
Gli approfondimenti	Si troveranno spesso nel testo dei riquadri colorati. Questi riportano approfondimenti o commenti di tipo più tecnico. Non sono indispensabili dunque all'esposizione dei concetti fondamentali, ma aiutano a capire meglio il significato e l'importanza di alcuni passaggi. Questo è un esempio di testo scritto per approfondire o chiarire.
I dialoghi	Sono le parti scritte sotto forma di copione. Possono essere lette direttamente in classe o recitate. In genere sono scritte con un linguaggio adatto ai più piccoli, ma i contenuti sono generali ed è importante che possano essere esposti con modalità simili adattandole all'età. *I dialoghi del docente sono indicate con questa grafia.* *Le risposte suggerite per gli studenti sono indicate così.* *I commenti alle parti del dialogo sono indicate con questa grafia.*
Toccare o fare click	Nel corso della spiegazione delle esperienze che fanno uso di dispositivi digitali, quando ci si deve riferire ad un oggetto sullo schermo che possa essere selezionato *facendo click con il tasto sinistro del mouse*, normalmente si scrive *"toccando"*. Questo perché si presume che si stiano usando dispositivi *touch*, *LIM*, *monitor* o *tablet*, e quindi è più facile che l'utilizzatore capisca questo modo di dire. Chi usasse il mouse dovrà intendere, invece di "toccare", "fare click con il tasto sinistro del mouse".

SEZIONE METODOLOGICA

CODING: DOMANDIAMOCI IL PERCHÉ

Prima di tutto bisogna avere le idee chiarissime sul perché il ***coding*** sia diventato così importante per la scuola: in Italia e nel mondo. Nel corso **codeMOOC** del prof. Alessandro Bogliolo è stato chiaramente spiegato che lo scopo ultimo del *coding* non sia fare dei nostri studenti dei provetti programmatori (anche se ce ne sarebbe bisogno) ma **usare l'informatica come palestra per imparare a ragionare nelle più diverse discipline**.

Io cercherò di ripartire da queste motivazioni per approfondirle e per arrivare a **tracciare una diretta corrispondenza tra le basi del ragionamento umano e le loro rappresentazioni nell'informatica**.

È questa diretta ed evidente corrispondenza tra ***pensiero generale*** e ***pensiero computazionale orientato alla programmazione*** che sta alla base del grande interesse per il *coding* da parte del mondo dell'istruzione scolastica.

Un esempio per tutti: avrete probabilmente lavorato con i primi esercizi di `code.org` o con *Scratch*, e avrete dovuto inserire decine di volte le istruzioni di `vai_avanti`, `vai_a_destra`, `vai_a_sinistra` e così via. Qualcuno potrebbe essersi fatto l'idea che i programmatori scrivano istruzioni di questo tipo tutto il giorno e che il fondamento della programmazione sia il sapersi spostare in uno spazio geometrico con istruzioni simili. Qualcun altro potrebbe essersi fatto l'idea che si usino istruzioni di questo tipo perché hanno a che vedere con la robotica e che sono i programmatori del mondo della robotica ad usarle.
Nessuna di queste due ipotesi è dotata di un reale fondamento, ma esistono invece differenti e solide ragioni per avvicinarsi alla programmazione con esercizi di quel tipo.

Cercheremo di arrivare passo-passo a queste motivazioni e ad usarle per orientare le nostre attività didattiche al fine ultimo di stimolare il **pensiero computazionale: un metodo per ragionare, risolvere problemi e raggiungere obiettivi**.

LE BASI DEL RAGIONAMENTO UMANO

Le basi della capacità risolutiva umana: seguire un percorso verso un obiettivo

[illegible]

[illegible]

[illegible]

IL *CODING* NEL CONTESTO DIDATTICO

Il *coding*: palestra di pensiero risolutivo e creatività

È giunto il momento di mettere insieme le **sette componenti fondamentali del ragionamento umano**:

- l'essere umano riesce a raggiungere risultati compiendo sequenze di ***piccoli passi***;
- l'essere umano riesce a risolvere problemi dandone una ***rappresentazione astratta***;
- l'essere umano riesce a ottimizzare l'uso delle risorse ***ripetendo*** e ***riutilizzando***;
- l'essere umano può ***decidere*** per successive partizioni;
- l'essere umano riesce a pensare a realtà complesse con un metodo di ***classificazione gerarchica*** (da concetti più semplici a raggruppamenti più complessi);

- l'essere umano riesce a migliorare le proprie soluzioni con un ***metodo evolutivo***, spesso costoso;
- l'essere umano ***usa strumenti*** per applicare i propri ragionamenti al mondo fisico.

Arriviamo quindi al mondo dell'informatica. Essa è lo strumento con cui l'essere umano ha imparato a parlare con le cose (il *"linguaggio delle cose"*, direbbe Alessandro Bogliolo). Le cose con cui possiamo parlare grazie all'informatica sono piccole unità programmabili che oggi chiamiamo *microprocessori*. Le operazioni che un *microprocessore* può compiere sono molto poche e spesso molto semplici, ma l'essere umano è riuscito a rendere queste operazioni utili per scopi pratici costruendo strutture logiche sempre più complesse. Lo ha fatto: scrivendo sequenze di operazioni semplici, raggruppandole, impacchettandole, dando loro un nome e mettendole da parte. Lo ha fatto seguendo un metodo evolutivo: montando insieme i pacchetti, osservando il risultato, facendo errori e correggendoli.

I nostri calcolatori riescono a contare da `0` a `1` e ciononostante grazie a lunghissime file di istruzioni molto semplici, oggi, ci consentono di comunicare da una parte all'altra del mondo o scambiarci un testo o una fotografia in pochi istanti.

Lo possono fare grazie all'applicazione di questi sette metodi che sono alla base dell'informatica solo perché sono alla base di ogni ragionamento umano.

L'informatica quindi è una della tante palestre in cui l'essere umano può cimentarsi in una delle sue attività più elevate: ***pensare e creare***.

Ma non è solo *una delle palestre*, è soprattutto ***una delle migliori palestre***. E non perché sia complicata e quindi dia molta soddisfazione riuscire a risolvere problemi di programmazione. Lo è per una fondamentale ragione: ***nel coding fare errori costa molto poco***.

Siccome la programmazione è un'operazione del tutto intellettuale, ciò che si costruisce si può smontare in poco tempo e ricostruirlo in un modo diverso per verificare se la nuova soluzione sia migliore o peggiore.

Questa è la ***chiave delle potenzialità formative della programmazione***: ***la possibilità di sperimentare metodi di pensiero e creazione***, in un contesto in cui il risultato è subito evidente e l'errore ha, solitamente, piccole conseguenze.

La programmazione è una buona palestra per imparare a pensare, creare, raggiungere risultati.

Il significato letterale della parola *coding*

Un buon modo per iniziare a parlare di ***coding*** è stabilire il significato della parola.

La prima parte del discorso che segue può essere proposta anche agli studenti e, infatti, la si ritroverà più avanti nella *Storia del coding*. Il seguito invece aiuta a completare, in modo schematico, il discorso che abbiamo appena concluso.

La parola ***coding*** è parte del più ampio discorso che il MIUR ha proposto alle scuole italiane con il ***Piano Nazionale Scuola Digitale*** (https://www.istruzione.it/scuola_digitale/allegati/Materiali/pnsd-layout-30.10-WEB.pdf).

È una parola inglese ed ha un significato letterale molto preciso.

Ma come mai un ministero italiano ha dovuto scegliere una parola straniera per indicare una attività strategica per l'innovazione del proprio sistema scolastico?

Apparentemente sembra trattarsi di un atteggiamento di esterofilia nel preferire parole inglesi per settori tecnologici. Ma esiste almeno una buona ragione per usare un termine a cui la maggior parte delle persone non sappia attribuire un significato preciso: consentire di affiancare alla *parola* contenuti molto più ampi del suo significato letterale.

Coding ha una desinenza in ***–ing*** e ricorda il tempo verbale che nell'inglese si usa per indicare un'azione continuativa nel tempo (*present continuous*). Ma, la lingua inglese usa la desinenza ***–ing*** anche per sostantivare i verbi. *Coding* è infatti il sostantivo del verbo *to code*. Ed è per questo che è corretto dire, in italiano, *il coding*.

Ma cosa significa *to code*? Se cercherete sul dizionario di inglese scoprirete che il significato letterale del termine è *codificare*. Quindi *the coding* significa: ***la codifica***!

Questa è un'altra buona ragione per usare una parola straniera: se vi avessero detto che *la codifica* era diventata una priorità per la Scuola italiana, vi sareste come minimo molto stupiti. Invece, dire *coding* ha tutto un altro suono!

Ma cosa si intende per *codifica*?

Nella sezione dedicata alle ***lezioni*** troverete una trascrizione del discorso più ampio che è possibile fare agli studenti per spiegarlo. Qui ne darò una definizione sintetica.

La *codifica* è una parte del lavoro del programmatore; quella in cui, dopo il lavoro preparatorio per progettare l'applicazione, egli scrive le istruzioni in una forma adatta ad essere eseguita dai calcolatori. In pratica deve usare un linguaggio comprensibile per la macchina. Il testo che scrive si chiama ***codice***.

Una prima estensione del significato della parola *coding*

Il ***coding***, la *codifica*, è dunque una parte del lavoro del programmatore, ma il suo significato può essere allargato all'insieme delle operazioni che si devono compiere per far funzionare i computer cioè:

la programmazione.

Fare ***coding*** a scuola dunque significa ***insegnare a programmare i computer***. Ma se il ***coding*** fosse solo questo, sarebbe un po' strano che esso sia proposto alle scuole italiane come strumento didattico fondamentale per lo sviluppo degli studenti. In fondo la programmazione dei computer è una professione che riguarda una ristrettissima porzione della popolazione ed è improbabile che anche solo uno studente per classe possa avviarsi a questo mestiere.

Questo è vero e, come vedremo, ci sono più solide ragioni per fare esperienze di ***coding*** in classe. Ma queste prime considerazioni si prestano per fare un discorso utile ai fini della ***cittadinanza digitale***, un altro caposaldo del ***Piano Nazionale Scuola Digitale***.

In un contesto globale in cui le tecnologie digitali impegnano una crescente parte del nostro tempo e sono in grado di condizionare le nostre opinioni ed il nostro modo di vivere, ***è molto importante che il sapere sui sistemi informatici non sia concentrato nella mani di un numero troppo ristretto di persone***. È anzi importante che ci sia un crescente numero di cittadini attivi che sia in grado di capire queste tecnologie, contribuire a controllarle e svilupparle in direzioni che non incoraggino ulteriormente la concentrazione delle informazioni e dei servizi *online* ma che, invece, aiutino a renderli disponibili in modo aperto e condiviso.
Per dirla in modo aulico, il fatto che, anche nel nostro Paese, via sia un sufficiente numero di esperti in informatica è una ***questione di democrazia***. Averne troppo pochi sarebbe un ***rischio***.

Ma anche a queste condizioni, non sarebbe giustificato avviare alla programmazione un così elevato numero di studenti a discapito di altre conoscenze e professioni altrettanto essenziali.

Ma allora come mai il ***coding*** può essere così importante?
A causa del ***pensiero computazionale***.

Il *coding* ed il pensiero computazionale

Per poter interagire con le macchine programmabili è necessario acquisire una mentalità adatta ad esse: questa mentalità è detta ***pensiero computazionale***. **Letteralmente significa saper pensare in modo da risolvere problemi attraverso calcoli**.

Come anticipato, questo modo di pensare non è molto lontano da ogni nostro ragionamento che abbia la finalità di compiere un'operazione o risolvere un problema complesso: *operazioni* e *problemi* non necessariamente di tipo informatico, anzi, normalmente molto lontani dal mondo delle macchine.
La ragione risiede nel fatto che l'essere umano, quando ha avuto bisogno di macchine che potessero ripetere i nostri ragionamenti, le ha create in modo da poter seguire il nostro metodo e ha quindi trasferito in esse la capacità di rappresentare la realtà, seguire percorsi (i programmi), ripetere azioni, fare delle scelte ed agire sul mondo esterno.
Ma le macchine, i computer in questo caso, non lo fanno nel nostro stesso modo.

Agli studenti spesso racconto questa situazione:

- *Se io chiedo ad ognuno di voi di mettere una mano sulla spalla di un compagno, è molto probabile che riusciate a farlo.*
- *Per un computer sarebbe difficile perché il comando è molto vago: non ho precisato quale mano, quale spalla, di quale compagno.*
- *Non sono stato preciso, ma voi siete riusciti ad eseguire il comando comunque.*
- *Questo perché il nostro cervello si è sviluppato per integrare le informazioni mancanti ed arrivare ad una soluzione soddisfacente anche in assenza di indicazioni precise e rigorose.*
- *Il cervello del computer no. È una macchina e ha dentro tutta e sola l'intelligenza che le abbiamo inserito traferendo i nostri ragionamenti all'interno di essa.*
- *Ha un'intelligenza, ma non è la sua. È la nostra, trasferita in modo meccanico in esso.*

Il fatto che il computer sia così poco elastico è una fatica per il programmatore, ma uno stimolo per gli studenti che debbano iniziare a lavorare con esso: la programmazione ci aiuta a migliorare il nostro metodo di ragionamento perché ***le macchine ci costringono a seguire procedure rigorose*** e cioè ad essere molto precisi nel dare istruzioni senza commettere errori e molto chiari nel darle. Ci costringono a ragionare nel migliore dei modi con cui l'essere umano è in grado di risolvere problemi.

L'applicazione del pensiero computazionale alla programmazione delle macchine è l'esercizio che si può fare con il *coding* nella "*palestra informatica*".

Una piccola nota storica.

C'è chi disse: "Il computer era uno specchio dei tuoi pensieri. Ti insegnava a ragionare. Credo sia la più alta forma di apprendimento. Tutti dovrebbero imparare a programmare un computer, perché è una cosa che insegna a pensare. E' come la facoltà di legge: non tutti devono fare l'avvocato, ma imparare la legge è utile perché insegna a pensare in un certo modo. Allo stesso modo, la programmazione informatica insegna a pensare in modo diverso. Ecco perché la considero un'arte liberale." https://youtu.be/4HdvBoAAWfE#t=19m35s

Era il 1995. Era Steve Jobs, il co-fondatore di Apple.

Dunque il contesto si è già notevolmente ampliato, a partire dal nucleo centrale rappresentato da una semplice parola di significato oscuro: ***coding***.

Ma c'è di più.

Qualcosa di meglio del *pensiero computazionale*

Quando affronteremo insieme le prime attività pratiche di programmazione per il mondo della scuola, scoprirete che di calcoli e di numeri se ne vedono ben pochi.

E allora dov'è tutto questo *pensiero computazionale* da applicare nella programmazione?

In effetti c'è, e risulta evidente ad ogni studente che si voglia addentrare nella programmazione con un linguaggio professionale. Ma questo metodo non viene proposto alle prime esperienze con gli studenti. Nelle attività di ***coding*** in classe si usano applicativi software e sussidi tattili che applicano i principi della programmazione in modo semplificato: attraverso una rappresentazione grafica molto colorata e procedure guidate.

Le strutture che caratterizzano il *pensiero computazionale* sono tutte presenti, ma liberate dalla loro componente più strettamente numerica.

È più corretto dire, quindi, che l'applicazione delle metodologie del ***coding*** in classe non sviluppa strettamente il *pensiero computazionale* ma, più in generale il ***pensiero logico ed analitico***.

Vediamo il perché.

Pensiero logico perché i dispositivi digitali sono macchine e devono necessariamente seguire strutture di esecuzione prestabilite e conseguenti alle caratteristiche di progetto.

Pensiero analitico perché i dispositivi digitali seguono logiche molto semplici e quindi non è possibile risolvere problemi complessi se non si esegue una analisi, cioè una loro *scomposizione* in problemi più semplici. La scomposizione del problema in elementi via via più semplici è una necessità connaturata allo strumento che si deve usare per ottenere la soluzione: macchine affatto intelligenti e molto semplici.

Dunque, il *coding* è una palestra per esercitare il pensiero logico ed analitico.

In pratica è un modo per fare pratica di quel metodo che viene richiesto in quasi tutte le discipline di insegnamento della scuola.

- Potrà sicuramente servire per le materie matematiche e scientifiche.
- Ma non si può dire che non serva per imparare ad esprimersi in una lingua dato che, per scrivere un testo, è necessario seguire regole sintattiche. Non solo, per esprimersi è necessario avere una rappresentazione chiara di ciò che si vuole descrivere ed essere in grado di scomporlo (analisi) in argomenti, per poter poi scomporre ciascuno in frasi a cui applicare regole.
- E per le competenze geografiche? Vedremo che le esperienze di robotica educativa richiedono di sapersi orientare in uno spazio e scegliere i percorso per unire due punti, come le località su una mappa geografica.

- E per capire un testo di storia è necessario avere una chiara idea di cosa significa *il prima* e *il dopo*.
  E che ad una causa corrisponde un effetto: nel programma come nelle civiltà.
- Per non parlare del disegno tecnico che applica principi geometrici facilmente ottenibili con gli applicativi dedicati al ***coding*** didattico.
- Infine la motricità: con i sussidi tattili, come il **CodyMat**, gli studenti sono chiamati a spostarsi in ampi spazi in classe o in palestra e a migliorare la loro capacità di coordinazione.

L'aspetto interessante di questo nuovo strumento metodologico è che ***non ha limiti di età***. Le esperienze di ***coding*** possono essere proposte a bambini anche della scuola dell'infanzia per i quali saranno uno dei molti modi *giocosi* in cui sviluppare le diverse intelligenze. Ma le competenze acquisite, già in giovane età, non mancheranno di dare i loro frutti nella scuola primaria dove verranno applicate a contesti sempre più complessi e variegati sfruttando però capacità ben consolidate nei giochi d'infanzia.

Cosa manca?

Beh, naturalmente la ***creatività***!

Il *coding* come palestra di pensiero creativo

Certo, l'abbiamo detto: i *computer* sono macchine. Seguono schemi meccanici realizzati con circuiti elettrici e logiche rigorose per eseguire i comandi che vengono loro impartiti attraverso i programmi.

Non possono quindi certo essere macchine creative!

Nota. Preferisco non addentrarmi nel discorso dell'***intelligenza artificiale*** e della capacità di alcuni algoritmi di manifestare, ai nostri occhi, una attitudine creativa nel comporre musica o realizzare disegni.

Per il momento possiamo limitarci a dire che esiste una ***creatività naturale***, manifestata essenzialmente nell'essere umano per effetto di fenomeni biologici, ed una ***creatività artificiale***, creata dall'Uomo attraverso il trasferimento, in dispositivi digitali, di algoritmi in grado di produrre risultati originali costruiti sulla base di un numero così vasto di fattori da non poterne ricostruire l'origine prettamente meccanica.

Ma bisogna porsi la domanda giusta. **La domanda non è se i computer siano essi stessi creativi, ma se siano strumenti in grado di stimolare la creatività umana**.

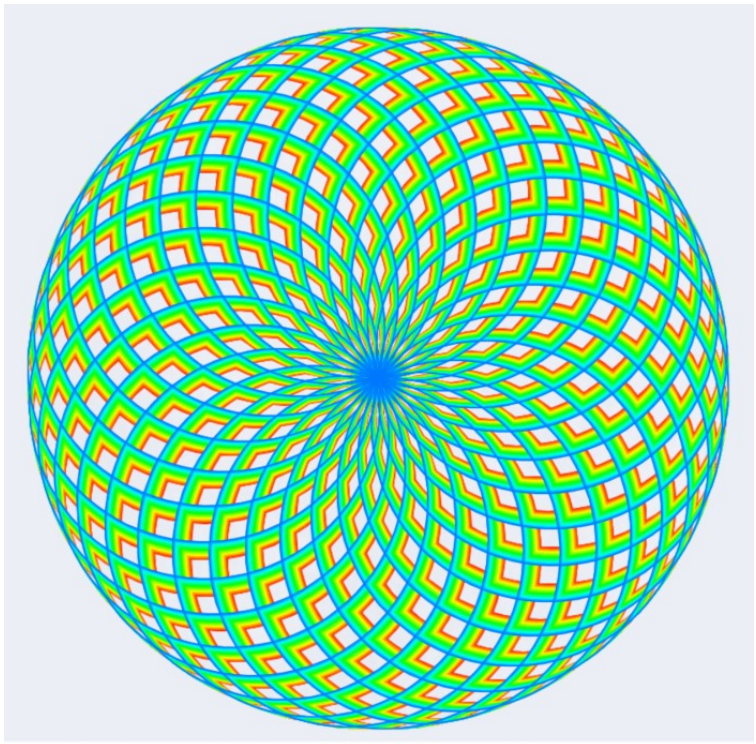
La risposta a questa domanda è sicuramente affermativa. Declassare il computer a mero strumento di fredda esecuzione di procedure prefissate, equivale a considerare una scatola di pastelli colorati come sterili pezzi di legno con un'anima di cera.
Ovviamente, non è importante se lo strumento sia creativo in sé, ma se offra all'essere umano l'opportunità di esercitare la propria creatività.
E allora, non c'è strumento che possa considerarsi più creativo di un computer, anche quando lo si usasse solo per scrivere programmi.

La programmazione è un'attività creativa come poche altre, perché, prima di esercitarla, il computer è una inutile macchina di calcolo, mentre dopo svolge una funzione che prima non esisteva e che, dunque, è stata creata dalla mente del programmatore.

Anche l'idea che la programmazione sia una disciplina *fredda* perché non lascia spazio alla creatività è inesatta. Affermare che un programma che effettua una certa funzione possa essere scritto in un solo modo, o con pochi margini di libertà, equivale a dire che esistano pochi modi per raccontare una storia di due promessi sposi perseguitati dalla protervia di un potente. Alessandro Manzoni, in questo caso, non avrebbe esercitato più creatività di un programmatore. Al contrario, tutti e due hanno dovuto fare innumerevoli scelte per arrivare al risultato in modo assolutamente unico e personale.

Non me ne vogliano i letterati ed appassionati di discipline umanistiche: posso riconoscere che nello scrivere opere letterarie gli autori abbiano avuto margini di liberta maggiori a quelle di un programmatore. Ma il concetto è che non esiste un solo modo di scrivere una frase (complessa), così come non esiste un solo modo di implementare un algoritmo (complesso).

Si può ancora discutere sul fatto che il risultato della programmazione non sia *artistico* quanto lo scrivere un romanzo o che scrivere un programma che consente agli studenti di fare ***coding*** con blocchi colorati non sia affascinante quanto creare la vetrata di una chiesa. Ma, anche qui, bisogna riconoscere che, attraverso i *computer*, è possibile ottenere risultati che la nostra sensibilità umana percepisce come *artistici*.



Un disegno realizzato con DidaLab in onore di Seymour Papert, inventore del linguaggio LOGO

Dunque, siamo arrivati al termine del nostro viaggio di arricchimento del contenuto della parola ***coding***.

Sarebbe stato difficile scegliere una parola migliore per condensare tutti questi significati.

Il fatto che sia stata scelta una parola straniera aiuta a non attribuirle una accezione prevalente nel momento in cui la si ascolta per la prima volta e induce quel senso di stupore e curiosità che incoraggia ad approfondire e sperimentare.

Dovrebbe essere ora chiaro che proporre esperienze di *coding* non può che migliorare le capacità di ragionamento e le competenze dei nostri studenti di ogni ordine e grado.

I DISPOSITIVI DIGITALI: MALATTIA E CURA

Le nuove carenze cognitive nell'era digitale

CONCLUSIONI

Siamo arrivati al termine di questa prima parte dedicata all'inquadramento metodologico sull'uso didattico di questi nuovi strumenti chiamati ***coding*** e ***robotica educativa***.

Si sono approfondite le motivazioni per cui valga la pena affrontare lo sforzo di entrare in questo *nuovo mondo dei dispositivi digitali*. Si è tuttavia anche cercato di dimostrare come l'informatica non sia un mondo a sé, ma sia invece strettamente legata all'antica natura dell'essere umano ed alla sua capacità di trasferire su macchine costruite le attività ritenute ***faticose*** e ***ripetitive***.

Proprio queste due parole, ***faticoso*** e ***ripetitivo***, apriranno la prossima sezione dedicata alla trascrizione di lezioni in forma di dialogo. La teoria non è finita e, come vedrete, le lezioni approfondiranno molti dei concetti che sono fin qui stati solo accennati.

Ma ora, alla vostra lettura, tutto dovrebbe poter poggiare su solide basi.

LE LEZIONI IN DIALOGO

LA STORIA SUL *CODING*

Perché usare la storia

Cercando una traccia delle ragioni che mi hanno spinto, fin dalle prime lezioni con i più giovani, a raccontare una storia sul *coding* ai bambini più piccoli, mi sono imbattuto in questo pezzo scritto da Gianni Marconato [http://www.giannimarconato.it/info/].

Una partecipante ci racconta una storia di didattica: una lezione da lei tenuta poche ore prima, su di un tema trattato più volte in precedenza, solo che questa volta (?) non era riuscita ad agganciare il gruppo (giovani apprendisti).

Analizziamo il suo racconto, che sollecito essere molto dettagliato, soprattutto relativamente ai primi minuti di lezione (si trattava del primo incontro con quel gruppo).

Riesaminiamo il caso e ad un certo punto mi "scappa" l'analogia: sono forse mancati i preliminari … cognitivi? Enfatizzando, per chi non lo avesse capito (e c'era chi non lo aveva) la questione dei preliminari … e, attivato il pensiero analogico, la discussione parte … Ben focalizzata, perché è chiaro di cosa si stia parlando.

E si "scopre" che senza adeguati "preliminari", questa volta cognitivi, è difficile avere una buona e convinta partecipazione dei partner. E se i partner (nell'apprendimento) non partecipano, la relazione (educativa) sarà insoddisfacente. Per tutti.

Troppo spesso le nostre lezioni assomigliano ad un saltare addosso ai nostri studenti, sommergendoli di contenuti a loro estranei, senza aver fatto, preliminarmente, alcun tentativo di portarli dentro a quello che noi intendiamo fare con loro: offriamo (o imponiamo?) il nostro sapere che resterà sempre a loro estraneo. Alcuni lo rifiuteranno platealmente e respingeranno il nostro focoso assalto (messaggeranno, leggeranno – on-line – la gazzetta sportiva …); altri, educatamente, fingeranno partecipazione con qualche segno di presenza (un cenno del capo, un si, qualche assente risposta ad una domanda …) ma ascolteranno passivamente e tutto sarà come non fosse successo.

> *Sarà stata un po' irrituale la mia analogia, ma pare abbia funzionato. Abbiamo, infatti, lungamente parlato sull'attivazione, come primo passo di una nuova sessione didattica. Non puoi "avere" i tuoi studenti se prima non li …scaldi ….*
> *[http://www.giannimarconato.it/2013/04/perche-dovremo-fare-i-preliminari-cognitivi/]*

E infatti, se non avessi trovato questo brano, avrei iniziato come segue.

Quando vogliamo iniziare a parlare di ***coding*** ai nostri bambini, la domanda da porsi è: come faccio ad introdurre l'argomento? Come faccio ad interessarli al ***coding***? Come faccio a spiegare cosa è un *computer* quando i bambini credono di sapere tutto su di essi?

La storia del coding è un modo: introduce i concetti fondamentali in maniera indipendente dal contesto dell'informatica allo scopo da allontanare i bambini da ciò che credono di sapere per presentare i concetti sotto una nuova luce.

Come usare la storia

Di solito, senza le varianti, impiego meno di un'ora per raccontarla. Ma a vederla trascritta sembra più lunga. Siccome non è detto che la vorrete raccontare alla stessa mia velocità, potrete decidere di dividerla in due o più puntate. Per questo la presento a voi, già divisa in capitoli.

I contenuti della storia sono adatti all'intero ciclo della primaria: piace moltissimo sia ai più piccoli, sia ai più grandi. Ma il linguaggio usato per scriverne il testo è quello adatto ai primi anni della primaria o addirittura infanzia. Per i più grandi si potrà forse solo sostituire *bimbi* con *ragazzini* e usare qualche altro accorgimento stilistico.

Nel corso della storia vi sono diverse brevi attività pratiche, presentate sotto forma di gioco, che possono essere ampliate o ridotte a seconda del tempo a disposizione. In generale è preferibile svolgerle tutte e lasciare a ciascuna il tempo necessario perché i bambini ne capiscano il significato. I giochi possono anche essere ripetuti in lezioni successive non tanto per migliorare le abilità, quanto per consolidare il ricordo dell'attività svolta e dei contenuti che i giochi richiamano alla mente.

Per la scuola secondaria 1° i contenuti sarebbero adattissimi, ma non lo è la forma. La durata stessa, che non è un problema con intere classi di prima o seconda, risulta lunga per il livello di attenzione che è possibile constatare nella secondaria.

Per loro c'è un capitolo apposta, più avanti. Se anche la storia non verrà loro proposta, se ne consiglia comunque la lettura ai docenti che potranno trarre spunto da essa per le loro lezioni.

Narrazione della storia

Ecco dunque la storia.

■ Introduzione

- *Buongiorno bambini.*
- *Mi chiamo Daniele e sapete perché sono qui?*
- *Sono qui per parlarvi di computer.*
- *Ma voi sapete cos'è un computer?*

Le risposte sono le più varie…

- *Ne vedete uno qui in classe?*
- *Sì?*
- *In effetti quello è un computer.*
- *Ma il computer è intelligente?*
- *Siii!*
- *Ma sapete cosa vuole dire intelligente?*

Silenzio…

- *Deriva dalla lingua degli antichi romani, il latino. Essere intelligente vuole dire… capire!*
- *E voi capite cosa sto dicendo?*
- *Siii!*
- *Allora significa che siete intelligenti.*

Ora rivolgendosi al computer in classe…

- *Ehi, computer! Capisci quello che dico? … Computer!*

Silenzio…

- *Vedete? Non mi capisce! Il computer non è intelligente, perché non capisce.*
- *Ma ho un'altra domanda per voi. Il computer ragiona?*

Silenzio…

- *Ma se non è intelligente, forse non sa neanche ragionare.*
- *E, in effetti, non è capace di ragionare… Eppure ragiona!*
- *Com'è possibile?!*
- *Ve lo voglio raccontare con una storia.*
- *Vi piacciono ancora le storie?*

Di solito, per tutto il ciclo della primaria, i bambini rispondono con un entusiastico "sì". Se non lo fanno, potete chiedere: "Siete troppo grandi per le storie? Ma questa è una storia per grandi! I piccoli non la potrebbero capire!"

Questa è la storia…

- *Questa è una storia breve, ma lunga allo stesso tempo.*
- *Pensate che inizia tantissimi anni fa, quando ancora non c'erano i computer. Ma non c'erano neanche le automobili, i telefoni e neanche le biciclette. Non era ancora stato inventato quasi nulla.*
- *Solo, si sapevano costruire delle capanne.*
- *Ma è una storia lunga perché finisce ai giorni nostri. Anzi, per dirla tutta, racconta cose successe fino a poco fa, quando sono entrato e mi sono presentato.*
- *Siete pronti?*
- *Siii!*
- *Allora partiamo.*

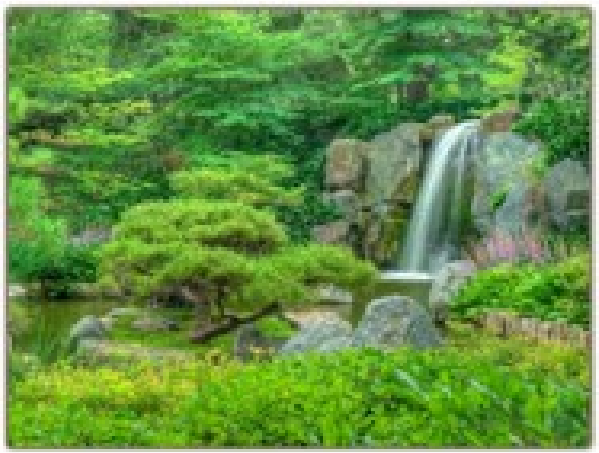
QUESTA È LA STORIA…

Il contesto parallelo di fantasia…

- *Questa è la storia di una famigliola: c'erano il papà, la mamma, e due bimbi più o meno della vostra età. Un maschietto ed una femminuccia.*

Questa famigliola viveva in un bel posto.

Più o meno come questo.



Proiettare slide...

C'era tanto verde: alberi, cespugli e, dietro, non si vede, ma c'era anche un bel prato e campi.

C'era anche tanta acqua, con una bella cascata.

Si erano costruiti una casetta ed avevano tutto quanto serve per vivere bene.

Ma dov'è la casetta? Non si vede!

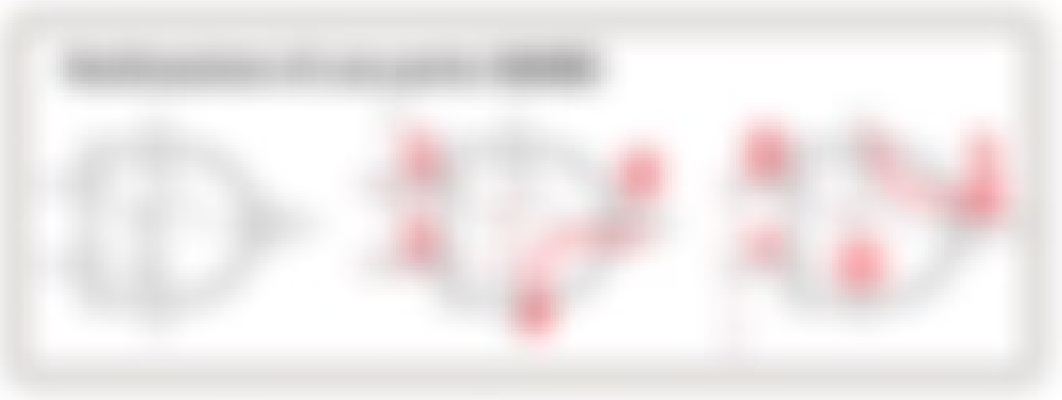
Non si vede perché questa è la vista che si aveva uscendo dalla casetta. La casetta è dietro!

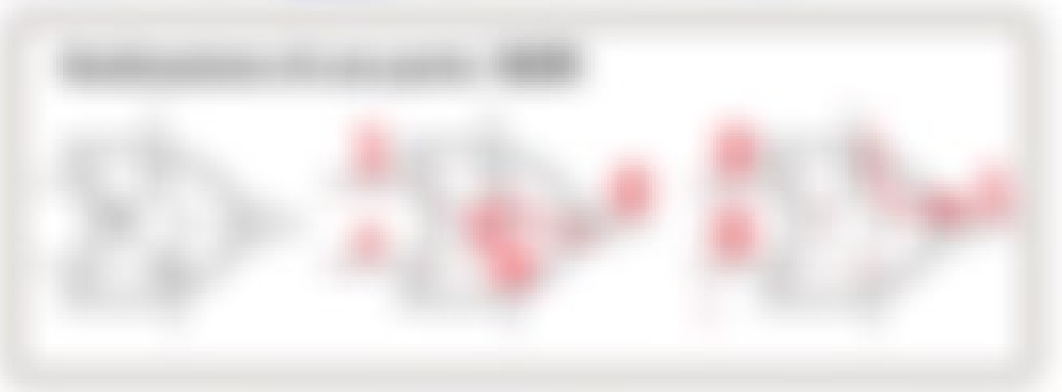
Ma vediamo un po'... Cosa gli serviva per vivere?

Tempo lasciato ai bambini per rispondere...

L'aria..., da mangiare..., l'acqua...

Sì in effetti è tutto giusto! In quanto all'aria ne avevano in quantità e non era neanche inquinata. Era tanto tempo fa!





Il fascino del digitale

- *Siamo arrivati al termine di un lungo percorso che ci ha portato a scoprire alcune caratteristiche fondamentali di tutti i dispositivi digitali.*
- ***Non abbiamo capito tutto di come siano fatti i computer, ma sappiamo come funzionano i mattoncini che li compongono**.*
- *Adesso vi invito a scoprire qual è l'aspetto affascinante dell'elettronica digitale.*
- ***I computer non sono macchine complicate ma complesse**: non sono fatte da centinaia di meccanismi diversi tra loro come le prime macchine calcolatrici meccaniche ed elettromeccaniche.*
- ***Nel tempo siamo riusciti a costruire macchine più semplici** e, per questo, **efficientissime**: consumano pochissimo rispetto ai primi computer e hanno una potenza di calcolo elevatissima.*
- ***I computer sono complessi sistemi di elementi molto semplici**: ciò che rende i computer così affascinanti è il fatto che sono costruiti a strati e, in ciascun livello, sono composti da pochissimi elementi semplici connessi a formare strutture sempre più complesse.*
- *Alla fine di questo nostro ragionamento possiamo vedere ciascun livello e verificare se questa affermazione sia vera.*
  - C'è il ***livello costruttivo***, l'elettronica: e qui si può dire che il suo nucleo, il *microprocessore*, sia costituito praticamente ***un solo componente: il transistor***.
  - C'è il ***livello rappresentativo***, lo strato che consente all'elettronica di assumere stati utili a rappresentare le informazioni in forma di numeri: e qui abbiamo visto che l'elettricità viene usata nel modo più semplice possibile, *acceso/spento*, segnale *alto/basso*. Due soli segnali per attribuire loro due soli significati numerici elementari, ***zero ed uno***.
  - C'è il ***livello logico***, lo strato che consente di eseguire operazioni sui numeri: qui si osserva che, a causa della semplicità della rappresentazione numerica elementare, ***solo zeri ed uno***, le operazioni logiche possibili sono pochissime e possono essere tutte ottenute dalla connessione di una ***sola operazione logica, ad esempio NAND (o NOR)***. Una sola, di nuovo!
- *Dunque direi che abbiamo dimostrato a sufficienza la nostra affermazione sulla natura dei dispositivi digitali.*

DALLA CALCOLATRICE AL COMPUTER

- *Abbiamo visto come facciano i computer a calcolare con l'elettricità, ma, se fossero solo questo, avremmo solo inventato delle* **calcolatrici***! Un po' più efficienti (molto più efficienti) di quelle elettromeccaniche, ma pur sempre solo delle calcolatrici.*
- *Dove sta la vera potenza dei computer?*
- *In due aspetti che abbiamo già anticipato, ma che ora cerchiamo di approfondire:*
  - la possibilità di memorizzare dati;
  - la possibilità di eseguire operazioni programmate.
- *Vediamo questi due aspetti uno dopo l'altro.*

La memoria di un computer

- *Un computer è in grado di conservare dati all'interno della propria memoria.*
- *E indovinate un po'? Cosa usa per memorizzare i dati?*
- *I* transistor*!*
- *Esatto, le memorie dei nostri smartphone sono fatte da* transistor *(generalmente MOSFET) e cosa pensate che possano memorizzare?*
- *Numeri!*
- *Esatto! E questi numeri in quale forma sono memorizzati?*
- *Sotto forma di* byte *che sono formati, ciascuno, da 8* bit*.*
- *E cosa si può memorizzare mediante i* byte*?*
- *Di tutto: numeri, caratteri di testi, immagini, suoni, filmati, ecc.*
- *Bravissimi!*
- *Ora voglio solo aggiungere un concetto importante per capire cosa sia una memoria.*
- *Quando abbiamo parlato di operazioni sui numeri binari e di* porte logiche *abbiamo sempre parlato di corrente che passa o non passa e di segnali che potevano essere alti o bassi (tensione più alta,* ***V+****, o più bassa,* terra*).*
- *Per capire le memorie di un computer vorrei parlarvi di* carica elettrica*.*

- *La carica elettrica delle memorie di un computer non è molto diversa da quella che riuscite a produrre strofinando un palloncino gonfio con un panno di lana. In quel caso scoprite che il palloncino strofinato attira i vostri capelli e respinge un altro palloncino anch'esso carico.*
- *Come mai?*
- *Perché alcune cariche elettriche sono rimaste imprigionate sulla superficie del palloncino e, siccome la gomma non consente alle cariche di spostarsi facilmente, esse rimangono bloccate fino a che non si scarica con il tempo o perché avete toccato il palloncino con le mani.*

Ho preferito non distinguere in questo contesto tra cariche elettriche negative (elettroni) e positive (ioni) per evitare dettagli. Rimane vero, tuttavia, che, nel caso del palloncino strofinato, gli elettroni vengono strappati dal panno al palloncino e quindi, tecnicamente, sul palloncino rimangono cariche positive dovute alla carenza di elettroni (carichi negativamente) che rimangono sul panno.

- *La memoria di un computer è fatta più o meno così: ci sono tante minuscole cellette che sono in grado di bloccare alcune cariche elettriche al loro.*
- *E sapete cosa può memorizzare una di queste cellette?*

In teoria dovrebbero riuscire a capire che si tratta di un bit, cioè di una singola cifra binaria `0` o `1` perché la cella può assumere solo due stati: carica o non carica.

- *Un bit .*
- *Esatto: ogni celletta di memoria può solo essere in due situazioni: o contiene cariche elettriche o non le contiene; cella carica o non carica.*
- *Se diamo a questa due situazioni un significato numerico potremmo dire che la cella carica rappresenta un `1` e la cella scarica rappresenta uno `0`.*
- *E così abbiamo proprio rappresentato un bit, cioè una cifra binaria (**BI**nary digi**T**).*
- *Per rappresentare un byte, ovviamente, avremo bisogno di 8 di queste cellette: 8 bit.*
- *E come si fa a scrivere o leggere da una memoria fatta così?*
- *Beh, per scrivere un numero in memoria ci sono circuiti elettrici che sono in grado di mettere e togliere queste cariche elettriche dalle cellette. Mentre per leggere ci sono circuiti elettrici che riescono a capire se in ciascuna celletta vi siano cariche oppure non ve ne siano.*
- *E il tutto non è molto più complicato di così. Semplice, no?*

I computer: macchine programmabili

- *Ora che abbiamo la memoria all'interno di un computer potremo memorizzare dei numeri e poi fare delle operazioni su di essi, salvando il risultato in memoria.*
- *Ma se potessimo fare solo questo, i computer non sarebbero macchine molto interessanti.*
- *Come abbiamo visto nei primi capitoli,* ***i computer sono stati costruiti per ripetere in modo automatico le operazioni necessarie per risolvere un problema dopo che, con il nostro ragionamento, avevamo capito il procedimento corretto****.*
- *Serve quindi un posto dove andare a scrivere questa sequenza di operazioni in modo che il computer le possa ripetere automaticamente.*
- *E qual è questo posto?*

Tempo per pensare…

- *La memoria, naturalmente!*
- *Quindi* ***nella memoria di un computer possiamo conservare i numeri che servono per fare i nostri calcoli, ma anche le istruzioni che gli dicono cosa debba fare****: passo dopo passo.*
- *Per questo* ***i computer sono macchine programmabili****: siamo riusciti a scrivere in forma numerica ogni singola operazione che vogliamo che il computer esegua e le abbiamo memorizzate in byte esattamente come tutti gli altri dati.*
- *Abbiamo dato un nome a queste operazioni: le abbiamo chiamate istruzioni.*
- *Vedremo nei prossimi capitoli che tutto quello che fa un computer è stato scritto come lunga lista di istruzioni che esso esegue automaticamente.*
- *Siamo finalmente arrivati a fare* **coding***!*
- *Vi ricordate cosa vuole dire la parola* **coding***?*
- Programmazione dei computer*!*
- *Esatto:* **coding** *è l'attività che faremo per istruire i computer a svolgere i compiti richiesti.*
- *Istruzione per istruzione, in un linguaggio che la macchina possa capire.*
- *Lo faremo scrivendo direttamente i numeri che rappresentano questi comandi?*
- *Nooo!!!*
- *Infatti, non scriveremo molti numeri.*

- *Useremo invece dei linguaggi di programmazione che abbiamo inventato per comunicare con i computer in modo più vicino al nostro. Potrebbero essere sequenze di parole, numeri e simboli, come fanno di solito i programmatori professionisti.*
- *Ma non saremo costretti neanche a fare questo (anche se, dopo un po' sarebbe più comodo). Lo faremo invece usando dei programmi che ci consentono di rappresentare le istruzioni in modo grafico: attaccando dei blocchetti colorati tra loro per formare catene che rappresenteranno i programmi in una forma più semplice e diretta.*
- *Li chiameranno linguaggi di programmazione grafici a blocchi.*
- *E vedrete che riusciremo a diventare tutti programmatori nel modo più lineare possibile.*
- *E scoprirete che riusciremo a divertirci anche un po': faremo muovere robot sullo schermo, faremo disegni bellissimi con poche istruzioni, impareremo a ragionare in modo da poter trasferire i nostri ragionamenti nei computer perché li possano ripetere velocemente e, forse, ci verrà voglia di saperne sempre di più.*

GLI STRUMENTI PER IL CODING

IL *CODING* E LE SUE RAPPRESENTAZIONI

Ci stiamo avvicinando alle attività pratiche, quelle in cui gli studenti potranno mettersi in gioco in prima persona per rendere i *computer* docili macchine al loro servizio.

Lo faranno attraverso la programmazione dei *computer*, ma abbiamo già visto che non sarà per loro necessario imparare uno dei linguaggi di programmazione professionali. Useranno invece delle rappresentazioni che sono state concepite appositamente per scopi didattici in modo che sia possibile un approccio graduale fin dalla scuola dell'infanzia.

Abbiamo già esposto quali sono i costrutti fondamentali per scrivere un qualsiasi programma: si potranno costruire *sequenze di istruzioni* la cui esecuzione potrà essere controllata da *cicli* e *condizioni*; le informazioni verranno memorizzate in strutture chiamate *variabili* a cui ci si potrà riferire con un nome (*identificatore*) scelto opportunamente.

Non scriveremo le istruzioni in forma alfanumerica, ma useremo *linguaggi di programmazione grafici a blocchi*: sarà cioè possibile scrivere un programma attaccando blocchetti colorati in lunghe catene secondo logiche adatte a conseguire uno scopo.

Ma il modo di rappresentare questi elementi potrà cambiare a seconda del *sussidio* scelto per proporre le esperienze di ***coding***.

Sono disponibili diversi approcci alla programmazione a blocchi ma molti sono sostanzialmente simili. Tra questi si distingue il **pipe-coding®** a cui questo testo dedicherà ampio spazio perché si tratta di una esclusiva **UniDida**. Tutti i sussidi qui proposti, anche quelli *tattili*, faranno uso di questo approccio.

Ma la questione fondamentale è: ***che tipo di risultati cercheremo di ottenere con questi programmi***?

La programmazione, lo abbiamo visto, è un potente strumento per sfruttare ogni funzionalità dei *computer*. Ma non tutti i programmi sono altrettanto interessanti per degli studenti.

Per attrarre la loro attenzione quasi tutte le esperienze didattiche di programmazione iniziano con un insieme ristretto di istruzioni che consente di spostare personaggi sullo schermo, spesso su una scacchiera, lasciando eventualmente una traccia grafica del passaggio. Questo semplice

metodo permette di effettuare simulazioni di esplorazione in spazi strutturati, ma anche di tracciare splendidi disegni geometrici con programmi molto concisi.

Questo modo per avvicinare gli studenti alla programmazione fu inventato da Wally Feurzeig, Seymour Papert e Cynthia Solomon nel 1967 che concepirono ***LOGO***, un linguaggio di programmazione creato per scopi didattici fin dall'infanzia. Dei tre scienziati fu Seymour Papert a dare un inquadramento pedagogico a questo approccio grazie all'esperienza maturata con Jean Piaget. Papert introdusse il concetto di *costruzionismo* nelle teorie dell'apprendimento. Infatti, secondo Papert, il processo di apprendimento è un processo di costruzione di rappresentazioni più o meno corrette e funzionali del mondo con cui si interagisce.

L'interprete del ***LOGO*** è, infatti, uno strumento che consente ai bambini di utilizzare il computer per ottenere risultati concreti rapidamente, ma utilizzando principi matematici e logici rigorosi.

Si veda https://it.wikipedia.org/wiki/Seymour_Papert per altri dettagli.

Per ottenere questi risultati con un insieme molto ristretto di istruzioni si è semplicemente cambiato il punto di vista usato normalmente per disegnare: invece di mettersi dal punto di vista di chi disegna, ci si mette dal punto di vista della matita, cioè di chi traccia il segno sul foglio.

IL PIPE-CODING®: IL PARADIGMA DEL FLUSSO

DidaLab è originale perché usa un esclusivo approccio chiamato *pipe-coding* (pronunciato "*paip-codin*"): un modo di proporre la programmazione agli studenti in una forma nuova, tutta italiana.

Il *pipe-coding* (*pipe*=tubo e *coding*=programmazione) utilizza un paradigma grafico ispirato al **flusso dell'acqua in reti di tubi** per consentire agli studenti di apprendere e sperimentare i seguenti concetti:

- flusso di programma;
- istruzioni condizionali (`if`);
- istruzioni di ripetizione (`for`).

Nel caso di programmi molto semplici, il *pipe-coding* aiuta gli studenti a capire il significato di *flusso del programma* perché i blocchetti sono rappresentati come **pezzi di tubazione** che si incastrano e si attirano come se fossero magnetici.

Nella parte alta si trova la *vasca* dalla quale l'acqua può entrare nelle tubazioni del programma, aprendo la valvola: è sufficiente toccarla per aprire e simulare come l'acqua scorrerà nei tubi per eseguire le sequenze logiche necessarie.
L'acqua aiuta anche a capire che, in un programma, le istruzioni verranno eseguite dall'alto verso il basso e cioè nel verso che segue l'acqua cadendo dalla *vasca*.

Nel *pipe-coding* i blocchetti sono più grandi di altri programmi ed è più facile trascinarli su *dispositivi touch*.

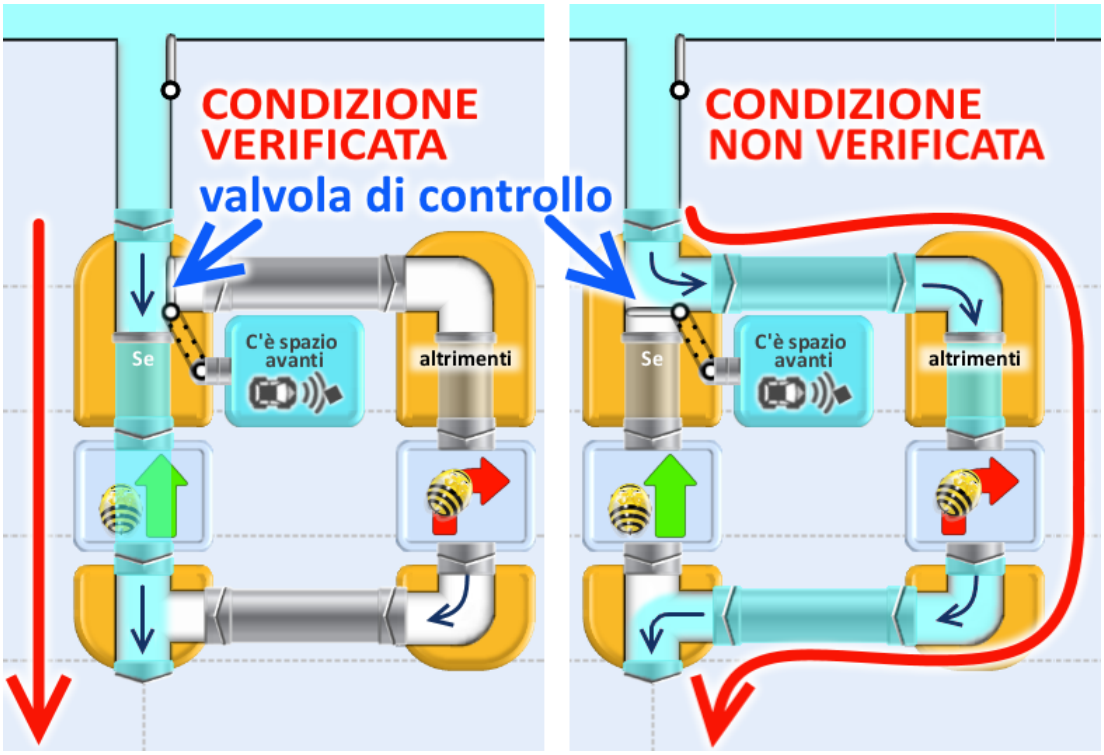
L'originalità del *pipe-coding* si coglie meglio quando si spiegano agli studenti i due concetti successivi: **le istruzioni condizionali** e **le ripetizioni**.
Queste istruzioni, pur nella loro semplicità, sono essenziali per capire come un programma possa raggiungere risultati molto interessanti e graficamente affascinanti con pochissime istruzioni (vedi in seguito la pittura robotica).

L'**istruzione di condizione**, ad esempio, consente di cambiare la direzione dell'acqua in base ad una domanda a cui si possa rispondere `SI` o `No`, oppure `VERO` o `FALSO`.
Nella rappresentazione proposta dal *pipe-coding* questa scelta viene disegnata come un blocchetto giallo che contiene una **valvola con due posizioni**:

- nella posizione aperta, il flusso dell'acqua prosegue verticalmente, andando a bagnare i blocchetti immediatamente sotto;
- nella posizione chiusa, il flusso viene deviato verso destra su una strada che può contenere blocchi diversi dai primi, per ottenere dal *computer* un comportamento diverso.

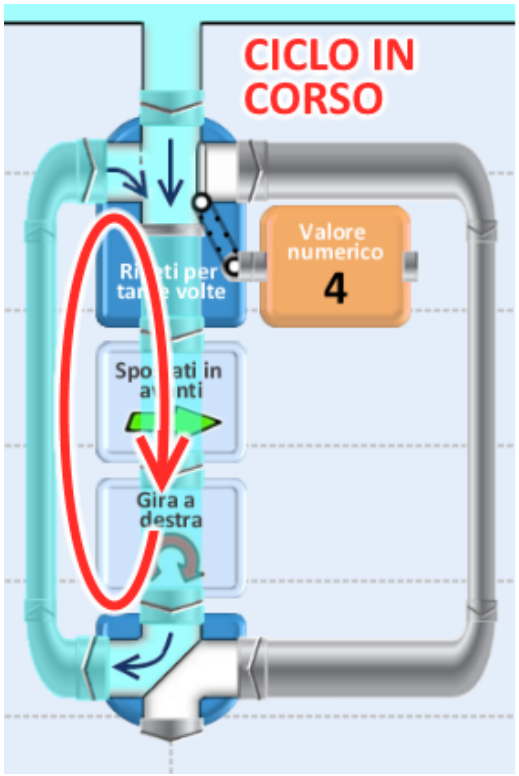
La **valvola** si apre quando la risposta alla domande è **`SI`** o **`VERO`**, cioè quando la condizione è verificata. Si chiude, invece, quando la risposta è **`NO`** o **`FALSO`**, cioè quando la condizione non è verificata.

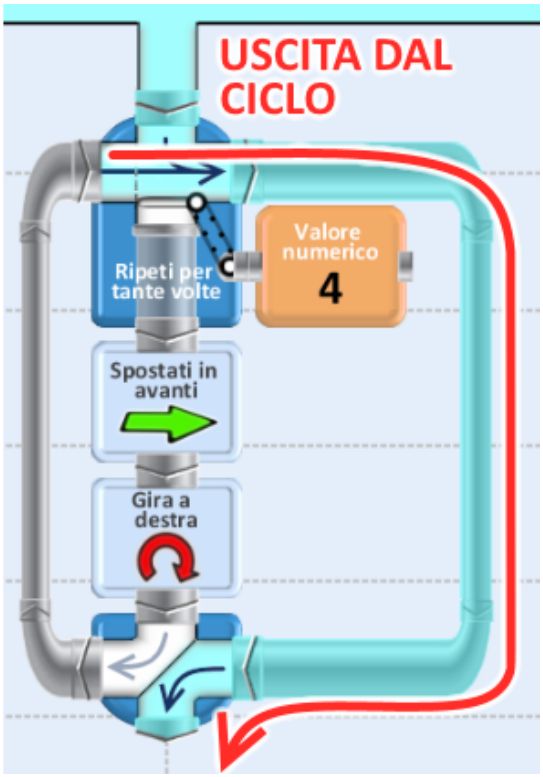
L'aspetto innovativo del *pipe-coding* è che l'avanzamento del flusso, pur spostandosi in orizzontale, non effettua *"salti"*, ma segue un percorso laterale che può essere facilmente seguito dagli studenti scorrendo il dito lungo il circuito.
Il flusso d'acqua si ricongiunge al fondo, per indicare che l'istruzione condizionale non può avere effetto sui blocchetti che seguono.



L'istruzione di **ripetizione**, nel *pipe-coding* ha anch'essa una rappresentazione sotto forma di circuito di tubi. Essa indica al programma di ritornare ad eseguire una sequenza di istruzioni per più volte.

L'evento che determina l'uscita dal ciclo di ripetizione è, ancora una volta, una condizione. Essa è del tutto simile alla domanda che apre e chiude la valvola nell'**istruzione condizionale** vista sopra. Questa volta, se la valvola è aperta (condizione vera), il flusso continua a circolare, mentre nel momento in cui si chiude, il flusso dell'acqua prosegue oltre.





Questi brevi cenni dovrebbero essere sufficienti ad esprimere come l'approccio alla programmazione di **DidaLab** sia caratterizzato in modo unico rispetto ad altri prodotti su internet .

Il *pipe-coding* è un'idea **UniDida** e, come è ovvio, è utilizzato in tutti i prodotti di **UniDida** sul coding.

Le note qui riportate saranno utili per svolgere gli esercizi nelle unità didattiche che verranno esposte nei successivi capitoli.

Nota: per avere maggiori dettagli sugli elementi distintivi del *pipe-coding*, rispetto agli altri linguaggi, si può consultare il documento: *"PipeCoding in breve"* scaricandolo dall'indirizzo: http://scarica.unidida.com/pipecoding.pdf

LA SOLUZIONE INTEGRATA PER IL CODING TATTILE E DIGITALE

Il presente testo propone una soluzione integrata tra *sussidi tattili* e *digitali* in modo che il ***coding*** non rimanga un'esperienza solo visiva, sullo schermo della *LIM* o dei *dispositivi individuali*, ma possa anche essere ***toccata*** e ***vissuta*** con modalità che possano coinvolgere l'intera classe anche in spazi aperti o in attività organizzate a piccoli gruppi.

Con il termine ***soluzione integrata*** si intende che gli *strumenti tattili* potranno essere affiancati da una *LIM* o *monitor touch* presente nel medesimo spazio in cui si fanno le attività per realizzare un continuo cambio di contesto e verificare con gli strumenti digitali quanto pensato sul sussidio tattile e viceversa.

Lo strumento digitale è **DidaLab**, un applicativo software appositamente realizzato per riprodurre su schermo tutto quanto è possibile fare con gli strumenti tattili.

La proposta per la scuola dell'infanzia ed i primi anni della primaria

Lo strumento chiave per la scuola dell'infanzia ed i primi anni della scuola primaria è il **CodyMat**, un tappeto gommoso, in materiale EVA certificato per l'utilizzo scolastico, pensato e progettato per svolgere molte diverse attività di ***coding*** con i bambini dai 3 fino ai 10 anni. È componibile e adattabile a molteplici attività.

CodyMat è composto da tessere ad incastro con appositi riquadri che consentono di realizzare una scacchiera colorata di dimensioni da 5x5 a 7x7 (dimensioni in metri: 3x3m - 4,2x4,2m). Sono disponibili diversi tipi di tessere che possono essere inserite all'interno dei riquadri per moltiplicare le esperienze possibili: *animali*, *figure geometriche*, *lettere dell'alfabeto*, *numeri* ed *operatori aritmetici*.

Si veda https://unidida.com/prodotti/codymat per dettagli.

L'aspetto originale dello strumento è la possibilità di comporre i programmi incastrando fisicamente delle speciali tessere colorate (in blu sulla destra della fotografia) che riproducono, in forma tattile, il paradigma del pipe-coding®, cioè la rappresentazione dei programmi informatici come flusso d'acqua in reti di tubi (esclusiva **UniDida**).

Il riquadro esterno (che porta le dimensioni della scacchiera da 5x5 a 7x7) permette di scegliere diverse tipologie di riferimento per identificare le righe e le colonne con lettere, numeri o simboli.

La proposta per la scuola primaria e secondaria

Il sussidio tattile più versatile è **CodyPuzzle** che, nelle sue varianti, può essere utilizzato già nella scuola dell'infanzia ma soprattutto per tutto il primo ciclo: primaria e secondaria 1°.

CodyPuzzle si presenta in una forma molto simile ad un gioco di società e, infatti, può anche essere semplicemente utilizzato come tale coinvolgendo da 2 a 4 giocatori.

È realizzato in cartone colorato ed il nome deriva dal fatto che i pezzi, come per il **CodyMat**, si incastrano a formare programmi di una complessità crescente: dalle semplici sequenza lineari di istruzioni ad algoritmi di una certa complessità con *cicli* e *condizioni*.

La scatola infatti contiene due scacchiere di gioco ed una grande varietà di inserti per realizzare sfide con diversi livelli di difficoltà. Le tessere per comporre o programmi sono però divise in due gruppi: una sufficiente per giocare e l'altra per esperienze di ***coding*** avanzato.

Nella sezione dedicata alle esperienze pratiche, verranno descritte alcune attività che prevedono l'utilizzo delle diverse tessere.

DIDALAB: DOVE IL CODING TATTILE INCONTRA IL SOFTWARE

DidaLab è una applicazione installata sulle *postazioni*. Questo significa che, normalmente, non ha bisogno di alcuna connessione ad internet durante l'uso.

L'unica fase in cui è necessaria una connessione è al **momento dell'attivazione della licenza**.

Nota: il fatto che **DidaLab** sia un'applicazione installata e non funzioni all'interno di un *browser* o collegato ad un *cloud* comporta alcuni vantaggi:

- funziona anche se la connessione ad internet non è sempre affidabile;
- non c'è nessuna possibilità per il *produttore* di rilevare a distanza le operazioni che fanno gli studenti su **DidaLab**;
- non c'è alcun modo per il *produttore* di ricondurre alcun dato personale all'uso del prodotto, tranne per quanto strettamente necessario all'attivazione della licenza.

DidaLab usa l'innovativo approccio del pipe-coding

DidaLab ha una caratteristica essenziale che lo differenzia da tutti gli altri metodi presenti.

Su internet le scuole hanno a disposizione diverse opportunità per fare esperienze sul coding in classe con i ragazzi, ma nessuno usa il metodo proposto in **DidaLab**.

Ecco un elenco dei principali siti internet:

http://programmailfuturo.it	È un sito internet realizzato dal MIUR che, in collaborazione con il CINI (*Consorzio Interuniversitario Nazionale per l'Informatica*), ha avviato questa iniziativa con l'obiettivo di fornire alle scuole una serie di strumenti per formare gli studenti ai concetti di base dell'informatica; il servizio è derivato da https://code.org/.
https://scratch.mit.edu/	È un sito dal quale è possibile accedere all'uso di un programma, chiamato **Scratch,** che propone la programmazione in forma grafica semplificata.
https://blockly-games.appspot.com/	È un sito internet dove vengono proposte alcune attività di programmazione a blocchi con modalità grafiche simili a quanto esposto sopra.

Nota: I siti internet sopra elencati usano tutti uno stesso modello grafico che è basato essenzialmente su blocchetti che si incastrano in modo da formare una sequenza di istruzioni per animare personaggi sullo schermo.