PIPE-CODING

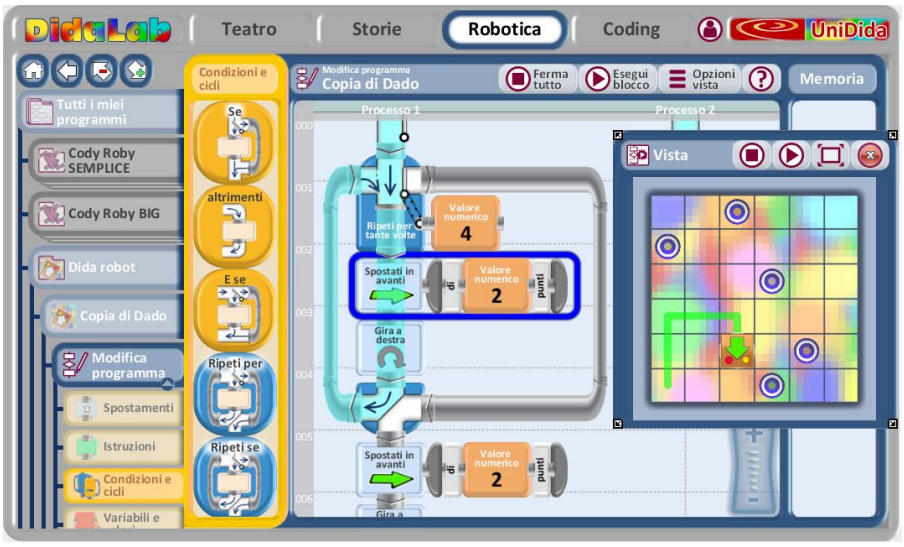
UN INEDITO APPROCCIO GRAFICO PER PROPORRE IL CODING PER SCOPI DIDATTICI A STUDENTI DI OGNI ORDINE E GRADO

Il *pipe-coding* è un approccio grafico concepito per presentare agli studenti i costrutti base della programmazione con modalità inedite. Il nuovo approccio è stato ideato da Daniele Costamagna e successivamente analizzato, in qualità di valutatore, dal prof. Alessandro Bogliolo dell'Università di Urbino, il quale ne ha riscontrato le peculiarità rispetto al panorama dell'offerta di sussidi didattici per il *coding* a livello internazionale (si veda `https://youtu.be/zmdBi2JNwVE`).

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

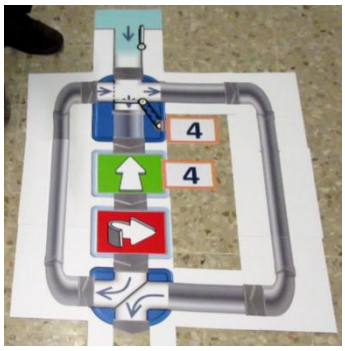
- flusso di programma;
- istruzioni condizionali (`if`);
- istruzioni di ripetizione (`for`).

L'approccio è stato implementato in un ambiente di sviluppo software, DidaLab, che è oggi disponibile per l'uso presso gli istituti scolastici.



La soluzione è adatta all'uso di bambini fin dalla scuola dell'infanzia, anche se esplica tutte le sue potenzialità con studenti del ciclo delle scuole superiori.

All'applicativo è stato affiancato uno strumento cartaceo, chiamato CodyMat, per coinvolgere i bambini più giovani, ma soprattutto per consentire l'esperienza anche a scuole senza dispositivi digitali. Esso riproduce le modalità di rappresentazione del *pipe-coding* con grosse schede (formato A4) che possono essere disposte sul pavimento per esercizi senza l'uso di tecnologie (*unplugged*).

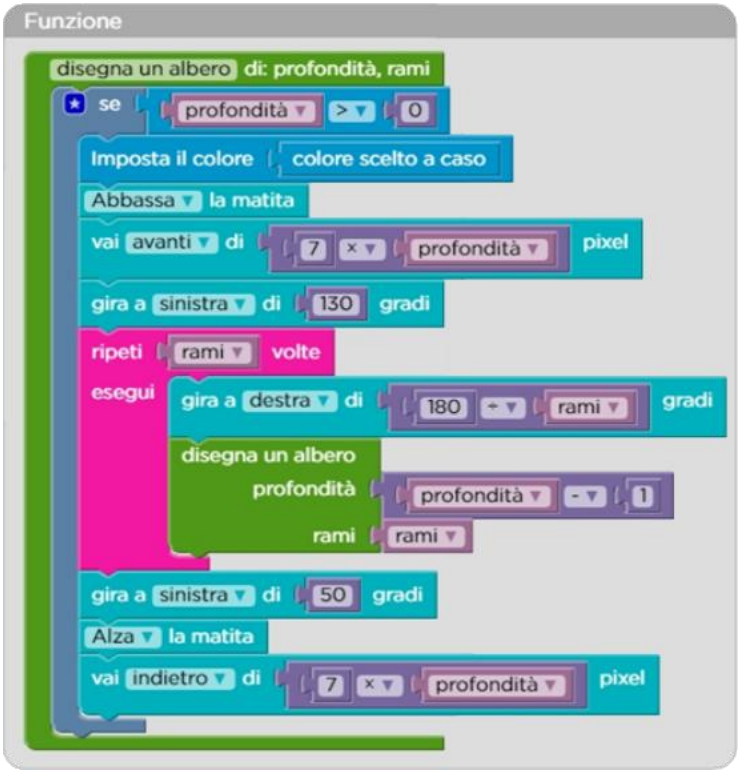


CONTESTO PRECEDENTE ALL'INNOVAZIONE

Il *coding* (un *inglesismo* usato dal MIUR per indicare la programmazione informatica proposta per scopi educativi) è diventato una priorità per la Scuola italiana a fine 2015 con la pubblicazione del Piano Nazionale per la Scuola Digitale (PNSD dell'ottobre 2015). La nuova attenzione per il mondo digitale era stata avviata, due anni prima negli U.S.A., con l'iniziativa *One Hour Of Code* e l'anno dopo, nel 2014, nell'Unione Europea con la *Europe Code Week*. Lo scopo è stato, fin dall'inizio, quello di alfabetizzare gli studenti ai concetti base della programmazione.

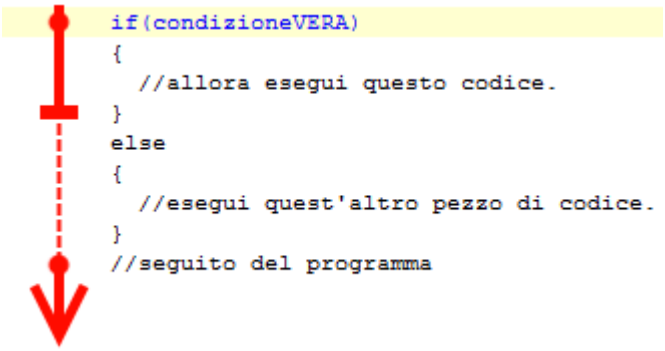
L'Italia in questi anni ha dimostrato una grande sensibilità verso le iniziative sul *coding*, posizionandosi spesso in testa per numero di attività partecipate da studenti e docenti su tutto il territorio nazionale.

Il panorama dell'offerta formativa di ausili didattici per il *coding* è molto ricco anche se, dal punto di vista tecnico, le soluzioni proposte sono molto simili. Si ispirano infatti tutte ad un ambiente di programmazione che è stato proposto in una versione del linguaggio *Scratch* sviluppato dal MIT (*Massachusetts Institute of Technology*). Tra le altre piattaforme disponibili, oltre a *Scratch*, vi sono alcuni siti internazionali come `code.org` e la sua versione italiana, `programmailfuturo.it`. Vi sono poi gli ambienti sviluppati da Google (`developers.google.com/blockly/`) ed alcuni prodotti con caratteristiche più specifiche per la comprensione della programmazione (`www.flowgorithm.org`).

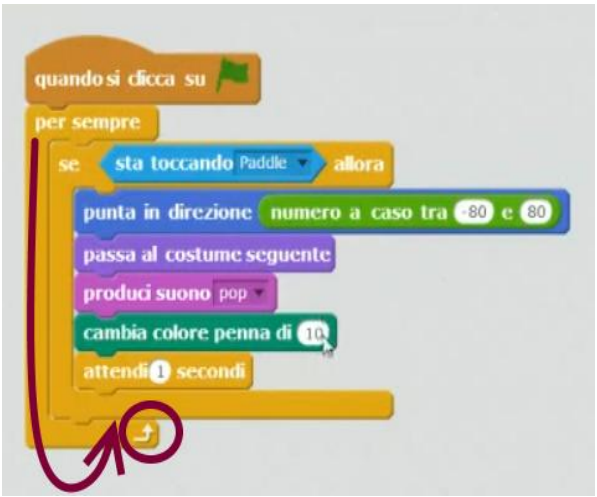


Gli applicativi per il *coding* didattico hanno quasi tutti un approccio simile, con blocchetti colorati che si incastrano per formare la sequenza delle istruzioni di programmazione. Lo scopo è consentire ai giovani studenti di ottenere risultati in poco tempo senza dover conoscere i dettagli di un linguaggio di programmazione. Quando i programmi non sono banali, questi applicativi consentono di introdurre anche i due costrutti fondamentali per la gestione del flusso del programma: le ***istruzioni condizionali***, che consentono di scegliere se eseguire un tratto di codice in base al verificarsi di una condizione, ed il ***ciclo***, che consente di ripetere un pezzo di codice per un certo numero di volte o fino al verificarsi di una condizione. Nella presentazione di questi costrutti, la gran parte delle interfacce degli applicativi sopracitati, mantengono sostanzialmente la struttura di queste istruzioni nella forma in cui sono abituati a scriverla i programmatori che è simile in moltissimi linguaggi di programmazione strutturata: `if()… then… else` per le condizioni e `for()` per i cicli. Questa modalità di scrittura è compatta e risulta comoda per programmatori esperti, ma non è immediatamente leggibile per uno studente inesperto. Questo avviene per ragioni distinte.

- L'***istruzione condizionale*** non è evidente perché viola un assunto che si spiega sempre nelle fasi iniziali di introduzione alla programmazione e cioè il fatto che *il codice viene eseguito rigorosamente dall'alto verso il basso*; se la condizione non è verificata l'istruzione condizionale viola questo assunto perché il codice in essa racchiuso non viene affatto eseguito determinando, in effetti, un salto; in presenza di istruzione `else`, inoltre, il flusso deve saltare un pezzo per eseguirne un altro che è più sotto. Questo aspetto è ovvio per tutti i programmatori che sanno che l'istruzione `if` è, in effetti, un *salto condizionato*. Tuttavia non lo è per gli studenti che hanno appena iniziato ad immaginare un programma come ***un flusso ininterrotto di istruzioni***.



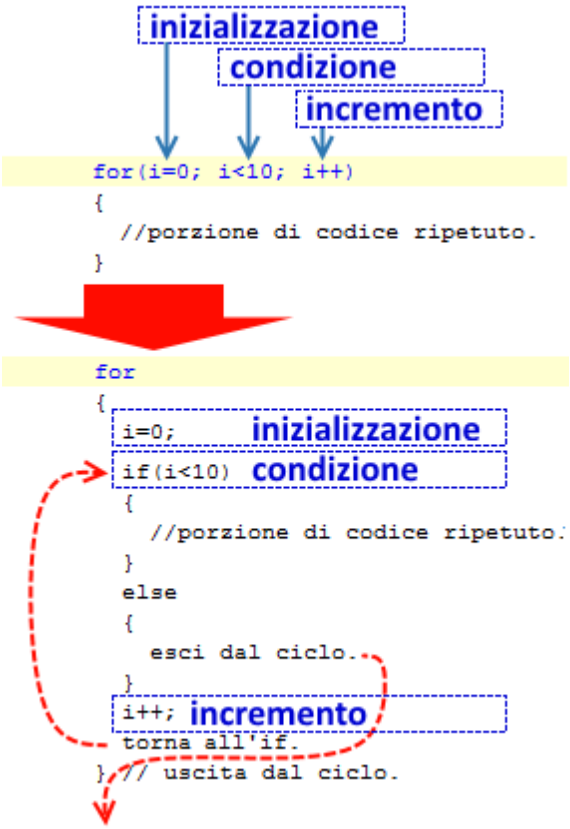
- Il ***ciclo*** è complesso per due ragioni: da una parte impone che l'esecuzione torni indietro ad un punto prestabilito per ripetersi e questo richiede di individuare il punto da cui si debba ricominciare; dall'altra, normalmente, il ciclo richiede una locazione di memoria (detta *variabile*) che consenta di contare il numero di ripetizioni per limitarsi a quanto prescritto dal programmatore, e questo viene scritto solitamente in un modo così conciso da non essere immediatamente comprensibile.



Nell'immagine qui sopra, si riporta come viene rappresentato il *ciclo* in uno dei linguaggi che si ispirano a *Scratch*. Si vede come la freccia verso l'alto, a fine ciclo, suggerisca, erroneamente, a chi legge che il flusso di codice debba essere eseguito al contrario dal basso verso l'alto. Anche interpretando correttamente la freccia (e cioè che *devi tornare all'inizio*) non è chiaramente indicato al lettore la via per il ritorno non equivoco al punto di partenza.

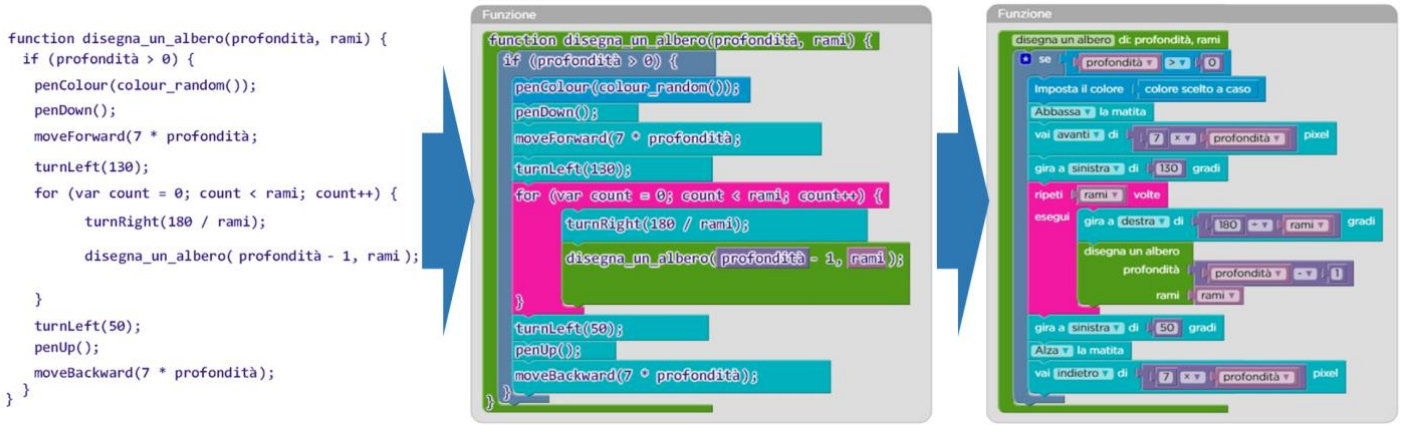
L'immagine a lato spiega come il ciclo `for` venga scritto nella maggior parte dei linguaggi di programmazione oggi utilizzati. I componenti dell'istruzione sono tre: **inizializzazione**, **condizione** ed **incremento**. Essi sono scritti sulla stessa linea, ma vengono in realtà eseguiti in tempi completamente diversi.

Gli ambienti di *coding* presenti prima dell'introduzione del *pipe-coding* non esplicitano questa complessità lasciando all'esperienza dello studente la comprensione della logica di funzionamento dell'istruzione.



In estrema sintesi, il *pipe-coding* è nato dall'idea che le rappresentazioni grafiche a blocchi esistenti consistessero per lo più in una *colorazione* delle istruzioni dei linguaggi di programmazione evoluti, mantenendone però intatte le complessità.

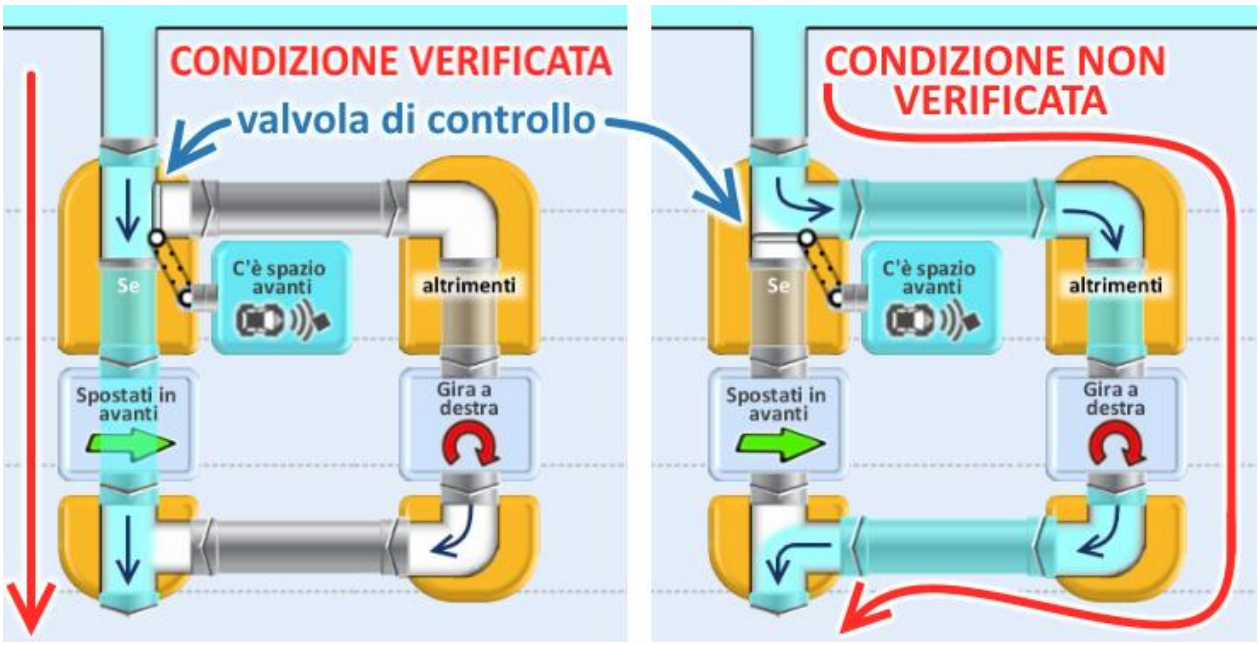
La seguente figura illustra come alcune soluzioni oggi esistenti si limitino a cambiare l'aspetto delle istruzioni senza aiutarne la comprensione.



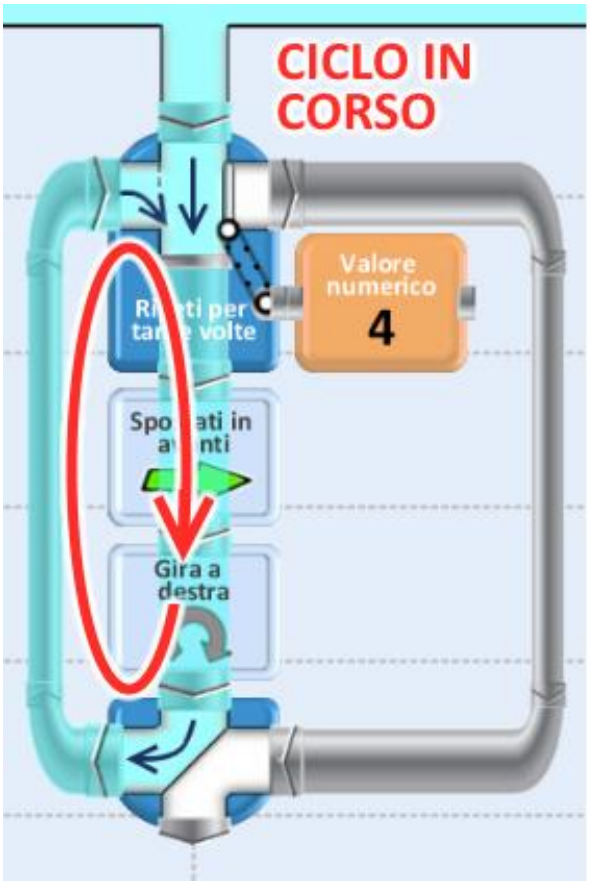
SCOPO E OBIETTIVI DELL'INNOVAZIONE

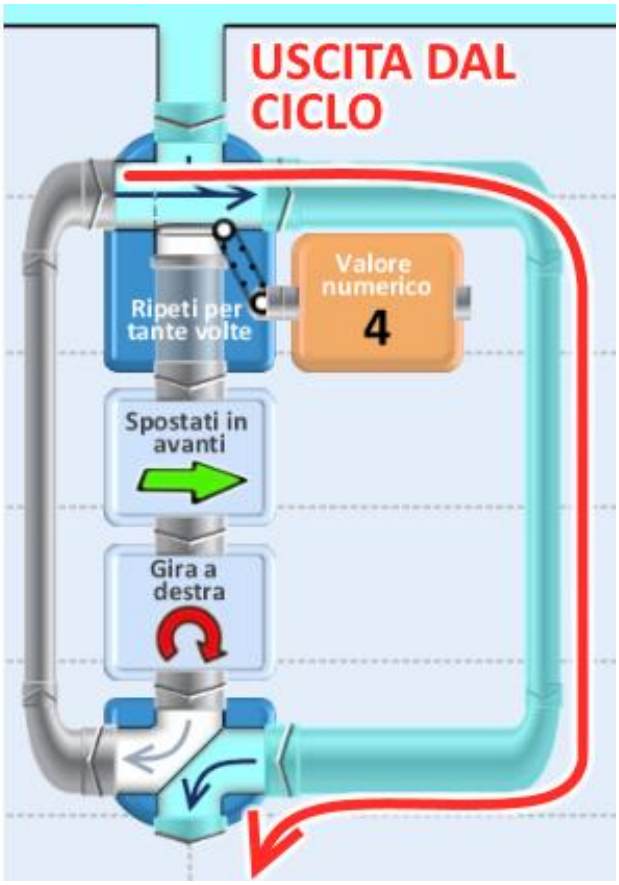
Il *pipe-coding* è stato concepito per dare una rappresentazione immediatamente leggibile dei due costrutti fondamentali per il cambiamento del flusso di un programma: l'***istruzione condizionale*** ed l'***istruzione di ripetizione***. Utilizza il paradigma del flusso dell'acqua all'interno di circuiti di tubi per consentire allo studente di seguire l'evolversi del programma muovendosi lungo i percorsi prestabiliti. Le istruzioni che controllano il flusso sono disegnate come *valvole* che aprono o chiudono biforcazioni permettendo l'evolversi del programma.

Per rappresentare l'*istruzione condizionale* `if`, il *pipe-coding* propone uno sviluppo orizzontale piuttosto che verticale e questo consente il rigoroso rispetto dell'assioma: *il codice viene eseguito dall'alto verso il basso*.



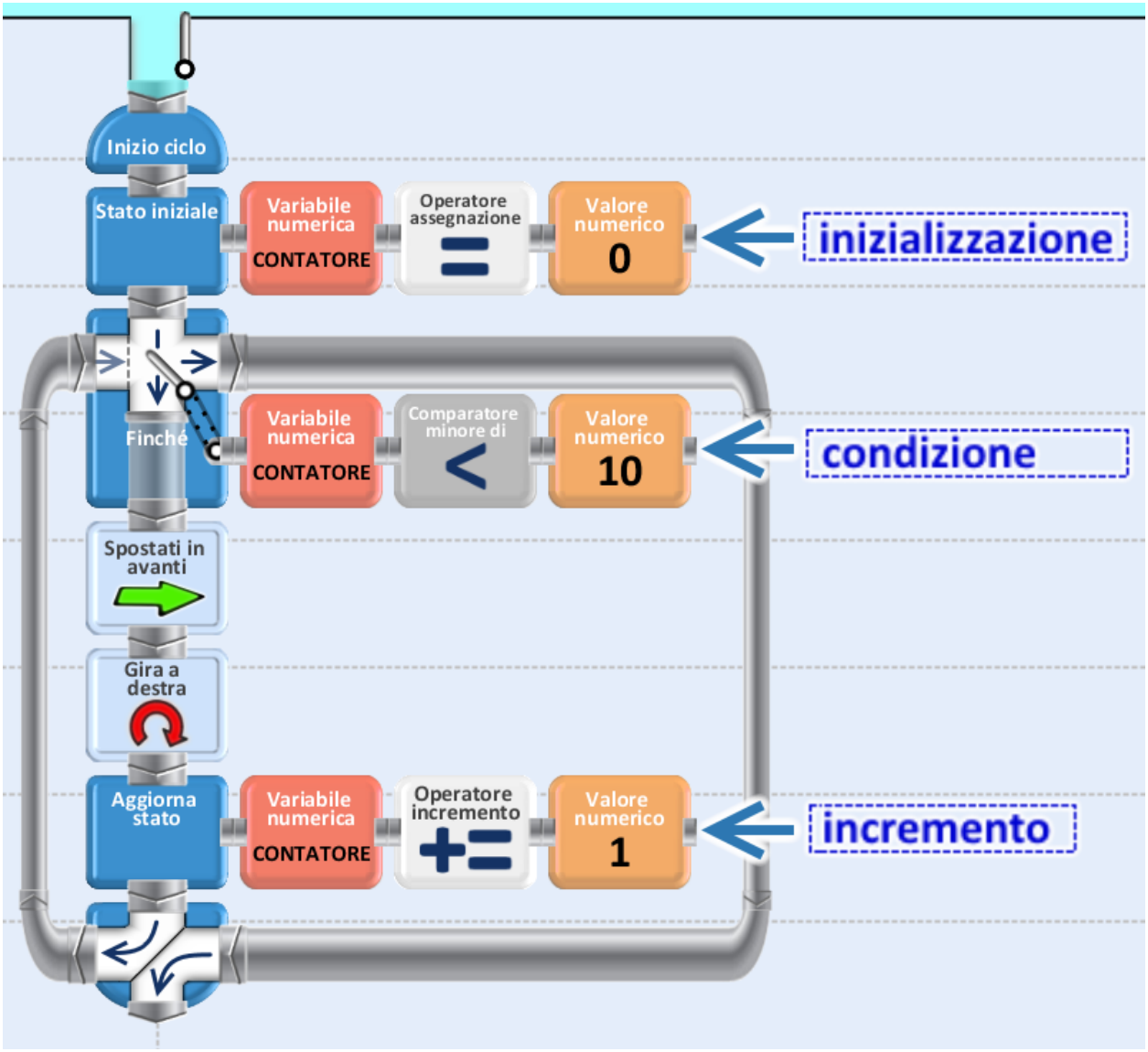
Per rappresentare il ciclo, il *pipe-coding* usa un circuito di tubazioni il cui ritorno è sulla sinistra mentre il percorso per uscire dal ciclo è sulla destra. Mentre il ciclo si ripete, l'acqua fluisce in senso orario all'interno del circuito e ne esce verso destra quanto la condizione non è più verificata. Questo stratagemma offre l'immediato vantaggio di evidenziare un cammino che lo studente può seguire direttamente per *vedere* il flusso che seguirà il programma durante la sua esecuzione.





Per spiegare la seconda complessità implicita nel modo in cui il ciclo `for` viene normalmente scritto nei linguaggi di programmazione e la necessità di una *variabile* per il conteggio del numero di ripetizioni, il *pipe-coding* evidenzia le tre fasi, *inizializzazione-condizione-incremento*, mediante il disegno di appositi componenti grafici che ne identificano la funzione e l'ordine in cui essi vengono eseguiti. In questo modo, quando gli studenti dovessero, in seguito, venire in contatto con la forma sintattica concisa usata dai programmatori, potranno avvantaggiarsi dello schema del *pipe-coding* per ricordare l'ordine e la funzione dei tre blocchi di dati.

La seguente figura illustra come i tre blocchi siano ordinati, uno sotto l'altro, nella sequenza in cui verranno eseguiti. La forma dei circuito formata dai tubi indica chiaramente che l'inizializzazione avviene una sola volta, che la condizione controllata ad ogni ciclo e che l'incremento avviene al fondo, subito prima del ritorno alla condizione di controllo che può determinare, essa sola, l'uscita dal ciclo.



IL PIPE-CODING GIRA L'ITALIA

Dalla prima presentazione su siti specializzati, il *pipe-coding* coding ha attirato l'attenzione degli specialisti del settore, tra cui spicca il prof. Alessandro Bogliolo dell'Università di Urbino che è anche *Europe CodeWeek Ambassador* per l'Italia. Il prof. Bogliolo ha esaminato il nuovo approccio nel corso di una sperimentazione e lo ha confrontato con altri prodotti già presenti sul mercato evidenziandone i punti di vantaggio soprattutto in termini di comprensione dei concetti da parte degli studenti (si veda `https://youtu.be/zmdBi2JNwVE`). Trattandosi di un approccio nato per scopi didattici, questa valutazione è stata determinante per il successo dell'iniziativa nel momento in cui è stata proposta alle scuole.

Da gennaio a maggio 2018, Daniele Costamagnaha portato il *pipe-coding* negli istituti scolastici di tutta Italia. Sono state visitate decine di classi facendo laboratori con bambini e ragazzi dalla scuola dell'infanzia, alla secondaria di primo grado. Sono stati incontrati oltre 2000 studenti e 150 insegnanti ed il nuovo approccio ha dimostrato la sua validità come strumento didattico per ciascuna delle età. Secondo il parere della quasi totalità delle insegnanti, gli studenti hanno capito i costrutti fondamentali della programmazione attraverso il *pipe-coding* e, nella stragrande maggioranza dei casi, lo hanno preferito alle soluzioni che conoscevano precedentemente.

Oggi DidaLab viene distribuito gratuitamente per attività di sperimentazione e, per l'uso in classe, è concesso per un periodo di prova. Ha tuttavia dimostrato il suo valore commerciale perché è stato già venduto ai primi 10 istituti scolastici in pochi mesi. Questo ha dimostrato la capacità del *pipe-coding* di competere con gli altri prodotti, dal momento che tutti gli altri ambienti sono distribuiti **gratuitamente** sul web.

Anche la versione cartacea del *pipe-coding*, il CodyMat, è stato venduto nei primi esemplari dimostrando, ancora una volta, che è percepito come un ausilio di valore.