

# Algomotricità: manipolare i fondamenti dell'informatica

Carlo Bellettini      Violetta Lonati      Dario Malchiodi  
Mattia Monga  
Anna Morpurgo

Dipartimento di informatica, Università degli Studi di Milano  
<https://aladdin.di.unimi.it>

Il capitolo presenta una serie di attività e percorsi didattici di informatica intesa come scienza che studia i principi e i metodi per l'elaborazione automatica dell'informazione. I percorsi presentati nel capitolo toccano in particolare tre temi: la rappresentazione delle informazioni tramite codifiche digitali, il pensiero algoritmico per descrivere procedure e tracciare processi, la programmazione come strumento per automatizzare lo svolgimento di un compito. Le attività sono impostate con approccio socio-costruttivista e permettono agli alunni di esplorare un tema informatico indagandolo in prima persona, costruendo modelli interpretativi e facendo ipotesi che possano essere messe alla prova nel contesto guidato dell'attività.

## 1 Introduzione

L'**informatica** è la scienza che studia i principi, i metodi e le applicazioni dell'**elaborazione automatica dell'informazione**: l'informazione è trasformabile per dedurne — o indurne — di nuova e ciò può avvenire automaticamente, ovvero tramite *interpreti* che manipolano “meccanicamente” l'informazione sotto forma di rappresentazioni simboliche/digitali.

Le parole chiave usate nell'etimologia della parola *informatica*, che combina i termini *informazione* e *automatica*, riassumono bene i temi su cui vertono le proposte didattiche che presentiamo in questo articolo. L'attività '**Sacchi nell'ascensore**' descritta in §2 è centrata sul concetto di *algoritmo*, nella semplice accezione di descrizione di una procedura che possa essere eseguita meccanicamente da altri. Il tema della *rappresentazione dell'informazione tramite codifiche simboliche* è il cuore delle attività intitolate '**Wikipasta**' e '**Rettangoli**', presentate rispettivamente in §3 e §4. La possibilità di

*automatizzare* un compito, tramite la scrittura di programmi che possano essere eseguiti da interpreti automatici, è infine il tema dell'attività '**Robot umani**', oggetto del §5.

Tutti i percorsi proposti condividono lo stesso approccio metodologico, basato sulle teorie cognitive *socio-costruttiviste*, secondo le quali la conoscenza si *costruisce* attraverso l'esperienza e la riflessione sull'esperienza stessa; quindi è colui che apprende che crea la propria conoscenza, e questa creazione si basa fortemente su ciò che già sa e capisce. Sulla base di queste premesse, non è efficace cercare di trasferire agli alunni rappresentazioni della realtà "preconfezionate"; piuttosto, è utile aiutare la costruzione della conoscenza attraverso stimoli, metodi e strategie che favoriscano l'attivazione del processo di apprendimento.

Per un'introduzione alle teorie socio-costruttiviste e alla loro applicazione ai contesti didattici, si può fare riferimento al libro [6]. L'approccio costruttivista è particolarmente adatto nell'apprendimento dell'informatica e nello sviluppo del pensiero computazionale, in cui la capacità di mettere in campo le proprie risorse di creatività e comprensione per risolvere problemi in parte sempre nuovi (*problem solving*) riveste un ruolo centrale. Più che la conoscenza di ricette predefinite, serve sviluppare la capacità di affrontare in modo sistematico anche situazioni nuove o parzialmente sconosciute. L'**algotricità** è una metodologia, da noi sviluppata, che applica strategie costruttiviste ad argomenti informatici. Come suggerisce il nome che abbiamo scelto (un neologismo sincretico che combina le parole *algoritmo* e *motorio*), l'algotricità sfrutta attività cinestetiche o manipolatorie nelle quali gli alunni hanno l'occasione di confrontarsi in maniera informale con uno specifico tema informatico. Questo primo momento operativo è seguito da attività di gruppo che favoriscono un processo di astrazione nel quale gli alunni arrivano a costruire modelli mentali del concetto che si sta esplorando. Come ultima fase viene prevista un'attività basata sui computer per ricollegarsi a qualcosa che gli alunni riconoscono naturalmente come collegata con l'informatica e stabilire così la connessione con il mondo "reale".

I paragrafi che seguono sono dedicati alla presentazione delle proposte didattiche. Per ciascuna indichiamo l'età cui è destinata, qualche breve riferimento ai contenuti informatici relativi e gli obiettivi di apprendimento che mira a promuovere, come riportati nella Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella scuola dell'obbligo<sup>1</sup>, da qui in poi chiamata PINI. Forniamo inoltre una sintetica descrizione delle varie fasi di ciascuna attività, rimandando al sito <https://aladdin.di.unimi.it/materiali.html> per ulteriori dettagli utili per riproporre le attività in classe. Il capitolo si conclude con il §6 in cui suggeriamo alcuni elementi generali, validi per tutte le attività, che aiutano a dare un senso complessivo alle attività stesse.

## 2 L'ascensore

Il tema di questa attività è il concetto di **algoritmo**, inteso nella semplice accezione di *descrizione di una procedura che deve essere eseguita meccanicamente da altri*. 'Meccanicamente' e 'da altri' significano che la descrizione è sufficientemente precisa, priva

---

<sup>1</sup><https://www.consortio-cini.it/index.php/it/gdl-informatica-scuola>

di ambiguità e sottintesi, in modo che possa essere portata a termine senza interventi “intelligenti” da esecutori differenti.

Un algoritmo, tradotto in un opportuno linguaggio di programmazione, diventa un *programma*. Di solito i programmi sono eseguiti automaticamente da una macchina. Tuttavia un algoritmo può essere eseguito anche a mano, come in questa attività.

Nella PINI sono previsti alcuni traguardi e obiettivi di apprendimento che si possono promuovere con questa attività:

- comprendere che un algoritmo descrive una procedura che si presta a essere automatizzata in modo preciso e non ambiguo (T-P-1);
- utilizzare il ragionamento logico per spiegare il funzionamento di alcuni semplici algoritmi (T-P-4);
- esaminare il comportamento di programmi semplici anche al fine di correggerli (O-P5-P-1).

Viene inoltre stimolata la capacità di eseguire e tenere traccia dell’esecuzione di un algoritmo, che è un’abilità importante per raggiungere gli obiettivi indicati.

L’attività dura circa **due ore** e si rivolge ad alunni del secondo ciclo della scuola primaria (**8-11 anni**).

L’algoritmo proposto nell’attività è ispirato a un quesito della gara Bebras dell’Informatica<sup>2</sup> e descrive una procedura per trasportare in magazzino, usando un ascensore, alcuni sacchi di pesi diversi allineati in un corridoio. In questo caso l’algoritmo è descritto a parole (vedi Figura 1); inoltre non è descritta esplicitamente una sequenza di istruzioni da svolgere, ma queste sono determinate in modo preciso dall’ordine con cui vanno presi i sacchi, dalla regola che stabilisce con quale peso parte l’ascensore, ecc.

Gli alunni hanno a disposizione degli oggetti (o dei foglietti) che rappresentano i sacchi, con cui possono simulare l’esecuzione della procedura, e una scheda di lavoro che contiene la descrizione dell’algoritmo e alcune domande che consentono di soffermarsi di volta in volta su aspetti diversi. La scheda è fatta da tre parti, corrispondenti a fasi diverse dell’attività. L’intera attività si svolge a *piccoli gruppi*, in modo da favorire il confronto di idee e la verbalizzazione.

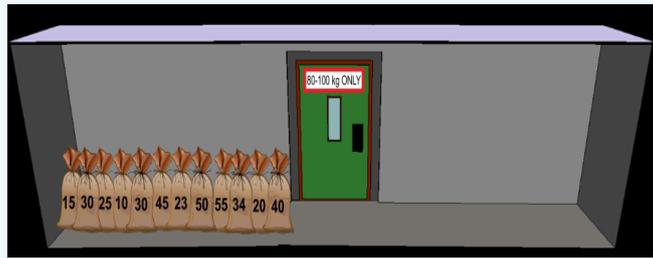
**Prima fase** Gli alunni devono leggere l’algoritmo e rispondere ad alcune domande iniziali; l’obiettivo è verificare che tutti abbiano compreso la regola secondo cui viene fatto partire l’ascensore.

**Seconda fase** Si consegnano agli alunni degli oggetti che rappresentano i sacchi, su ciascuno dei quali è indicato il peso corrispondente, e li si incoraggia a utilizzarli per fare pratica con la procedura per caricare l’ascensore. La possibilità di utilizzare dei

---

<sup>2</sup>Il Bebras dell’Informatica è un’iniziativa internazionale, che coinvolge oltre 50 Paesi, volta ad avvicinare all’informatica in modo divertente gli alunni di tutti gli ordini di scuola. In Italia il Bebras è organizzato da ALaDDIn e consiste in una gara non competitiva a squadre, cui le scuole partecipano tramite una piattaforma *online*, <https://bebras.it>.

Nel corridoio accanto all'ascensore ci sono alcuni sacchi allineati alla parete. Su ogni sacco è scritto il peso in kg.



L'ascensore serve per trasportare i sacchi in un magazzino. L'ascensore porta al massimo 100 kg e viene fatto partire non appena il suo carico supera gli 80 kg.

Per caricarlo, si prende il primo sacco della fila (quello più vicino all'ascensore) e lo si mette nell'ascensore, tranne quando il sacco fa superare il limite di 100 kg; in questo caso il sacco viene messo oltre l'ascensore a formare via via una nuova fila che parte dal fondo del lato opposto del corridoio.

Quando tutti i sacchi della fila sono stati spostati, si continua nello stesso modo considerando la fila formata dall'altro lato del corridoio.

Figura 1: L'algoritmo per trasportare i sacchi nell'ascensore

piccoli oggetti da spostare è molto utile per focalizzarsi concretamente sui vari passaggi da eseguire. In particolare gli studenti dovranno considerare i sacchi disegnati nella Figura 1 ed eseguire la procedura in quel caso specifico, per poi rispondere ad alcune domande: per esempio, quali sacchi sono stati spostati, in che ordine, quando, ecc.

Alla fine di questa fase viene annunciato che a breve non si potranno più usare gli oggetti che rappresentano i sacchi, e viene chiesto agli alunni di individuare un modo per tenere traccia (con degli appunti su un foglio) dei passaggi fondamentali della procedura, riflettendo su quali dati è importante annotare, quando e come. Il fatto che gli oggetti non siano più disponibili è fondamentale per favorire un passaggio di astrazione (dall'esecuzione pura e semplice, alla tracciatura dei dati principali).

**Terza fase** Gli alunni devono infine rispondere a domande simili a quelle della seconda fase, ma riferite a una situazione di partenza nuova (la disposizione iniziale dei sacchi non fa parte infatti dell'algoritmo, che si può applicare a situazioni diverse). Per farlo, dovranno simulare l'esecuzione della procedura e tenerne traccia usando il metodo ideato in precedenza.

Naturalmente non c'è un unico modo per tracciare l'esecuzione dell'algoritmo. Ad esempio, per stabilire quanti sacchi partono col primo carico si può procedere in questo modo: si esaminano i sacchi uno alla volta e li si cerchia o barra a seconda che vadano caricati nell'ascensore oppure spostati dall'altro lato del corridoio. Inoltre il modo in cui si annotano i dati è funzionale alla domanda cui interessa rispondere. Se si vuole tenere traccia di quali sacchi partono in ogni carico, si possono usare colori diversi per

ogni carico (e tutti i sacchi che partono nello stesso carico avranno lo stesso colore). In Figura 2 sono riportati alcuni esempi.



(a) Le 0 indicano i sacchi che vanno a destra, le X i sacchi che vanno direttamente in ascensore



(b) I colori delle X indicano i diversi viaggi in ascensore



(c) Ora anche le 0 sono colorate secondo il corrispondente viaggio in ascensore (il sacco da 30kg parte con i primi tre da sinistra)

Figura 2: Esempi di tracciamento

Anche se in generale non è necessario farlo, è possibile tenere traccia in maniera sistematica di tutti i dettagli che riguardano l'esecuzione dell'algoritmo. La Tabella 1 ne è un esempio, essa riporta passo passo qual è lo *stato* del sistema, espresso da: quali sacchi sono rimasti a sinistra dell'ascensore (colonne gialle), quali sacchi sono nell'ascensore (colonne azzurre), quali sacchi sono a destra nel corridoio (colonne verdi). La colonna rossa tiene inoltre traccia della somma dei pesi attualmente nell'ascensore. Lo spostamento di

un sacco corrisponde al passaggio da una riga alla successiva; le righe nere indicano che l'ascensore parte con il suo carico.

sacchi a sinistra												peso	sacchi in ascensore				sacchi a destra					
15	30	25	10	30	45	23	50	55	34	20	40	0					-	-	-			
15	30	25	10	30	45	23	50	55	34	20		40	40				-	-	-			
15	30	25	10	30	45	23	50	55	34			60	40	20			-	-	-			
15	30	25	10	30	45	23	50	55				94	40	20	34		-	-	-			
15	30	25	10	30	45	23	50					55	55				-	-	-			
15	30	25	10	30	45	23						55	55				50	50	50			
15	30	25	10	30	45							78	55	23			50	50	50			
15	30	25	10	30								78	55	23			45	50	50			
15	30	25	10									78	55	23			30	45	50			
15	30	25										88	55	23	10		30	45	50			
15	30											25	25				30	45	50			
15												55	25	30			30	45	50			
-												70	25	30	15		30	45	50			
-												100	25	30	15	30	45	50	50			
-												45	45				50	-	-			
-												95	45	50			-	-	-			

Tabella 1: Come evolve lo “stato” del sistema sacchi/ascensore

È chiaro che, avendo una tracciatura completa di questo tipo, è possibile ottenere tutte le informazioni necessarie a rispondere alle varie domande esemplificate sopra.

Alla fine di questa fase, è opportuno confrontare le strategie utilizzate dai gruppi per tracciare l’algoritmo. Le strategie potranno essere diverse, ma i dati annotati saranno gli stessi (a meno di errori che probabilmente saranno stati già individuati durante il confronto stesso). Questo è conseguenza del fatto che l’esecuzione rigorosa dell’algoritmo evolve necessariamente nella stessa maniera, se si parte dalla stessa situazione iniziale, e in particolare ciò succede quando l’esecutore è un agente meccanico: immaginate che i sacchi vengano spostati da un carrello con braccio meccanizzato che può sollevare, trasportare e depositare i sacchi seguendo proprio quanto prescritto dall’algoritmo.

Questa discussione finale è utile a mettere in evidenza altri elementi interessanti relativi all’attività svolta.

- Simulare a mano l’esecuzione di un algoritmo è uno dei modi per capire come l’algoritmo funziona o osservarne degli aspetti particolari.
- La capacità di *tracciare* un algoritmo si rivela utile sia per comprenderlo sia quando riscontriamo un malfunzionamento e vogliamo individuare il difetto che lo causa (il *bug* in inglese, spesso tradotto in italiano con *baco*<sup>3</sup>) nel processo che gli informatici chiamano *debugging*.

<sup>3</sup>Il termine in realtà fa riferimento alla presenza di “insetti” che potrebbero compromettere il funzionamento delle macchine. Vedi per esempio la falena (<http://www.computerhistory.org/t dih/ september/9/>), di cui parla Grace Hopper, una delle pioniere dell’informatica moderna.

### 3 Wikipasta

Questa attività e la successiva (§4) vertono sulla **rappresentazione delle informazioni tramite codifiche simboliche**. Affinché sia possibile elaborare in maniera automatica dei dati, di qualunque tipo essi siano, questi devono infatti essere “rappresentati” in modo che l’elaboratore possa manipolare la rappresentazione per produrre la rappresentazione della nuova informazione trasformata dall’elaborazione. In informatica si preferiscono rappresentazioni tramite *simboli*, dette anche *digitali*, perché per comodità i tecnici usano generalmente simboli numerici, cifre (in inglese *digit*): in pratica quindi la rappresentazione consiste in una serie di simboli che rappresentano l’informazione iniziale; essi vengono manipolati dall’elaboratore che produce una nuova rappresentazione simbolica da interpretare nuovamente come informazione elaborata. Si noti che in questo contesto (e in generale in informatica) l’espressione “rappresentazione di un dato” sta a indicare una qualunque descrizione formale del dato stesso. Un *sistema di codifica* consente di passare da un certo dato alla sua rappresentazione simbolica, e viceversa di interpretare una rappresentazione simbolica per ricostruire il dato rappresentato; si parla dunque di *codifica* del dato oppure di *dato codificato*. Nella rappresentazione binaria si usano solo i simboli 0 e 1, ma in generale può essere usato un insieme qualsiasi di simboli. È importante sottolineare che l’uso di rappresentazioni binarie è un dettaglio tecnologico, (è infatti facile costruire dispositivi in grado di avere due stati stabili, utili quindi a rappresentare i simboli 0 e 1), ma la scrittura di parole con le lettere dell’alfabeto è già un ottimo esempio di sistema di codifica (e dunque “digitalizzazione”) delle parole pronunciate.

Entrambe le attività si prestano a promuovere vari traguardi e obiettivi di apprendimento previsti nella PINI:

- esplorare la possibilità di rappresentare dati di varia natura mediante formati diversi, anche arbitrariamente scelti (T-P-6);
- definire l’interpretazione degli oggetti utilizzati per rappresentare l’informazione (legenda) (O-P3-D-2);
- utilizzare combinazioni di simboli per rappresentare informazioni familiari complesse (O-P5-D-1);
- utilizzare simboli per rappresentare semplici informazioni strutturate (O-P5-D-2);
- riconoscere se due rappresentazioni alternative semplici della stessa informazione sono intercambiabili per i propri scopi (O-M-D-1).

L’attività dura circa **due ore** e si rivolge ad alunni del secondo ciclo della scuola primaria (**8-11 anni**).

Gli alunni, *lavorando a coppie*, si trovano ad affrontare il problema di come descrivere e formalizzare con dei simboli l’aspetto tipografico di un testo. Giocando con pasta, legumi e altri piccoli oggetti, sono condotti alla scoperta del paradigma di marcatura del testo usato dalle pagine Wikipedia e in generale nel Web (dove è prevalente l’uso del linguaggio di marcatura chiamato HTML, *HyperText Markup Language*).

**Prima fase** Ad ogni coppia di alunni vengono dati un foglio con stampato a grandi lettere una frase e un foglietto con la stessa frase, come illustrato in Figura 3.

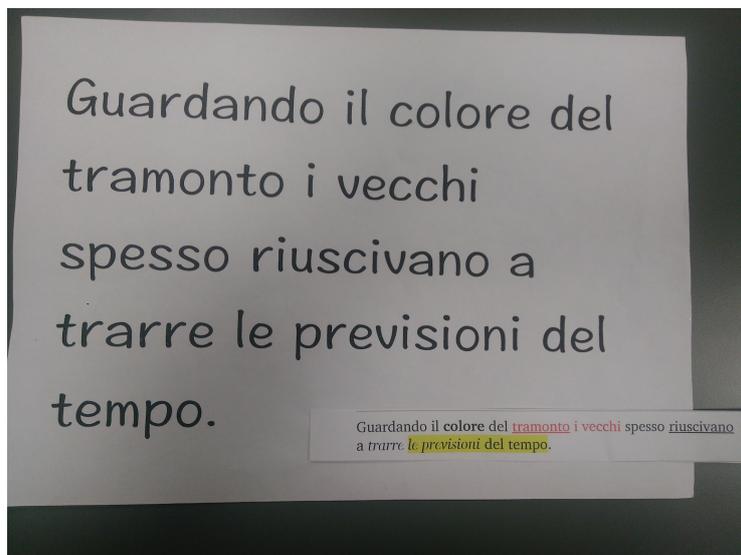


Figura 3: Testo semplice e testo formattato

Inoltre vengono messi a disposizione vari oggetti (pasta di vari formati, legumi di vario tipo, pezzetti di carta colorata, fermagli, perline, ecc.), oltre a carta e penna per supportare il ragionamento con annotazioni.

Agli alunni si chiede di usare gli oggetti per *decorare* il testo non formattato, in modo da rendere l'idea di come vada formattato per riprodurre l'aspetto del testo nel foglietto. In questa fase, gli alunni usano generalmente gli oggetti in maniera evocativa, cercando di riprodurre l'effetto visivo della formattazione (ovvero in maniera *analogica*, cioè "in analogia" con l'aspetto tipografico): gli spaghetti sono usati tipicamente per sottolineare, la pastina gialla per evidenziare, le perline per cambiare colore; tuttavia si osservano di frequente anche usi particolarmente creativi dei vari oggetti (vedi Figura 4(a)).

**Seconda fase** Vengono introdotti dei costi per gli oggetti e si sfidano le coppie a ottenere lo stesso risultato "spendendo" meno possibile. I costi sono studiati in modo da scoraggiare l'uso degli oggetti considerati più *utili*, come gli spaghetti o le perline rosse; gli oggetti più economici sono quelli apparentemente meno adatti alla situazione, come la pastina o le lenticchie. In questa fase gli alunni iniziano a usare gli oggetti in maniera più astratta, simbolica (cioè *digitale*, anziché *analogica*) attribuendo loro un significato basato su una convenzione arbitraria. Gli alunni realizzano soluzioni che prediligono oggetti economici, che riutilizzano gli oggetti meno costosi in diverse posizioni o modi, che usano due delimitatori/marcatori, uno di inizio e uno di fine, quando la porzione di testo da formattare è lunga. Alcuni alunni non riescono a uscire dall'approccio analogico finché non viene spiegato loro che "possono mettersi d'accordo con il tipografo"; ci si

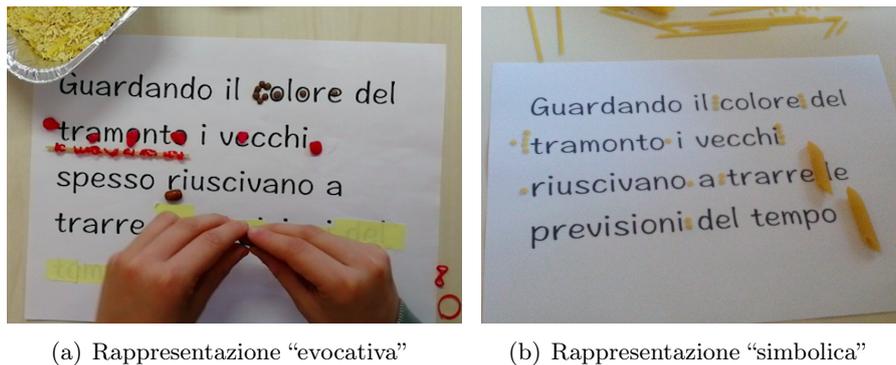


Figura 4: Testi decorati.

avvicina quindi a un'idea di "legenda" o di "regole" che definiscono come vadano interpretati gli oggetti. Cercando di ridurre i costi e attivando delle strategie di codifica più complesse, è possibile che nascano dei problemi di ambiguità delle codifiche (ad esempio qualora lo stesso oggetto venisse usato per indicare sia il corsivo che il grassetto), e questo non sempre sarà evidente a tutti gli alunni. In questo caso è utile proporre degli esempi o contro-esempi che mostrano gli eventuali problemi nell'interpretazione univoca del significato degli oggetti.

Per incentivare la scoperta di soluzioni via via più ingegnose è utile lanciare una gara a chi spende di meno, annunciando i progressi delle varie coppie a tutta la classe. Gli alunni tenderanno a sbirciare le soluzioni dei compagni e a farle proprie, anche piuttosto rapidamente, modificandole per i propri scopi. Non è il caso di valutare questi fenomeni come copiatore, perché funzionano anch'essi da stimolo a comprendere e migliorare. È però importante evitare che la competizione tra coppie diventi eccessiva: da un certo punto in poi, non si possono più ridurre i costi a meno di usare regole troppo specifiche o ambigue, e l'attività comincia a perdere senso.

Alla fine di questa fase si confrontano a classe intera le soluzioni trovate, mettendo in evidenza in particolare l'utilità di usare degli oggetti per delimitare l'inizio e la fine della porzione di testo da trasformare, quando questa è lunga (vedi Figura 4(b)). Capita spesso che qualche alunno contesti la correttezza dell'uso dei marcatori e/o delle regole usate da altre coppie: questa è una buona occasione per sottolineare il fatto che è necessario definire delle regole precise e non ambigue. Grazie al confronto tra le varie soluzioni, gli alunni si rendono conto infine che sono accettabili soluzioni differenti, ciascuna delle quali attribuisce agli oggetti significati diversi, pur usando approcci sostanzialmente simili; le regole usate non sono altro che "convenzioni": vanno tutte bene purché ci si metta d'accordo.

È interessante far notare che questi approcci ideati dagli alunni, e in particolare l'uso di *delimitatori* per racchiudere le parti di testo da formattare, sono effettivamente usati per memorizzare ed elaborare testi in formato digitale. Per esempio, questo è il modo con cui sono gestite le pagine di Wikipedia: dopo aver aperto nel browser una pagina di Wikipedia e cliccato su "Modifica wikitesto", è possibile vedere che, oltre al testo della



Figura 5: Interfaccia del software usato nella fase finale di Wikipasta

pagina, sono presenti dei simboli speciali (come \* ' =) che fanno da delimitatori a porzioni di testo, proprio come la pastina o le lenticchie nel caso nella nostra attività (cliccando su “Aiuto” si può avere inoltre una sintesi delle regole principali usate).

**Terza fase** Wikipasta si conclude con una fase finale al computer, in cui agli alunni è richiesto di formattare un testo usando delle regole di tipo “Wiki”, utilizzando un’applicazione ad-hoc<sup>4</sup>. In questo caso il ruolo dei delimitatori è svolto da simboli non utilizzati nel testo e le regole sono fissate dal *software* stesso (e mostrate in modo schematico sulla destra, vedi Figura 5), che fornisce un *feedback* immediato dell’effetto prodotto. Gli alunni di solito si adeguano senza resistenze né difficoltà alla sintassi formale prevista dal sistema *software*; hanno infatti compreso che si tratta di una convenzione che vale tanto quanto un’altra, ma che è necessaria per poter far “comprendere” al sistema come deve essere formattato il testo e ottenere automaticamente la visualizzazione corretta. La soddisfazione di vedere apparire automaticamente l’effetto desiderato ripaga così della fatica e della piccola frustrazione sentite da alcuni nelle fasi iniziali dell’attività.

Quest’ultima fase può essere propedeutica all’introduzione dell’HTML per sviluppare semplici pagine web, ad esempio di presentazione della scuola o della classe: la sintassi si fa più articolata, ma è basata sullo stesso concetto di delimitatore usato dai linguaggi di tipo Wiki, e anche qui il risultato molto concreto può motivare allo sforzo richiesto.

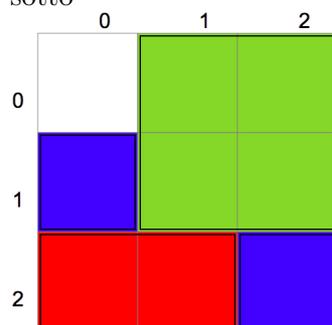
## 4 Rettangoli

Anche l’attività presentata in questo paragrafo, come Wikipasta (vedi §3), verte sul tema della **rappresentazione delle informazioni tramite codifiche simboliche**, dura circa **due ore** e si rivolge ad alunni del secondo ciclo della scuola primaria (**8-11 anni**).

<sup>4</sup><https://aladdin.di.unimi.it/sw/wikipasta/index.html>

A differenza che nel caso di Wikipasta, qui non si deve inventare/stabilire un sistema di codifica (come si era fatto usando lenticchie e pasta), né si devono applicare regole di codifica date esplicitamente (come nel software wiki). Invece, in questa attività gli alunni devono *scoprire* come funziona un sistema di codifica, partendo da esempi e usando uno strumento *software* per fare esperimenti. Naturalmente l'obiettivo non è imparare questo specifico sistema di codifica ma, di nuovo, comprendere come sia possibile rappresentare informazioni anche complesse attraverso l'uso di simboli e regole precise applicate rigorosamente.

Tuttavia, per consentire la comprensione di quanto segue, descriviamo qui nel dettaglio come funziona il sistema di codifica oggetto dell'attività. Il *sistema di codifica* permette di dare una descrizione simbolica di una immagine composta da riquadri colorati, ed è basato sulla scomposizione delle immagini in porzioni quadrate o rettangolari; l'immagine è infatti descritta da una serie di righe di simboli, una in corrispondenza di ciascuna delle porzioni rettangolari che formano l'immagine. Ciascuna porzione è caratterizzata dal suo colore, dalla sua dimensione e dalla sua posizione all'interno della figura. Questi elementi vengono usati nella codifica nel modo seguente: il colore è identificato da una lettera minuscola (ad esempio, c vuol dire giallo); la dimensione è descritta da due numeri, di cui il primo indica l'altezza e il secondo la larghezza (misurate usando come riferimento la quadrettatura sullo sfondo); la posizione è descritta da una coppia di numeri tra parentesi, come (1,2), dove il primo numero indica il numero di riga (partendo dall'alto e contando da 0) e il secondo indica la colonna (partendo da sinistra e contando da 0) del quadrato in alto a sinistra della porzione rettangolare. Ad esempio l'immagine qui sotto



viene codificata come segue:

a 2 2 (0,1)  
 b 1 1 (1,0)  
 d 1 2 (2,0)  
 b 1 1 (2,2)

Lo strumento *software*<sup>5</sup> mostra un'immagine accompagnata da simboli (lettere, cifre, punteggiatura) che descrivono l'immagine stessa. L'applicazione, su richiesta, modifica in modo casuale una porzione dell'immagine, permettendo di osservare come il cambiamento si riflette nella descrizione con i simboli dell'immagine stessa. Non è prevista la possibilità

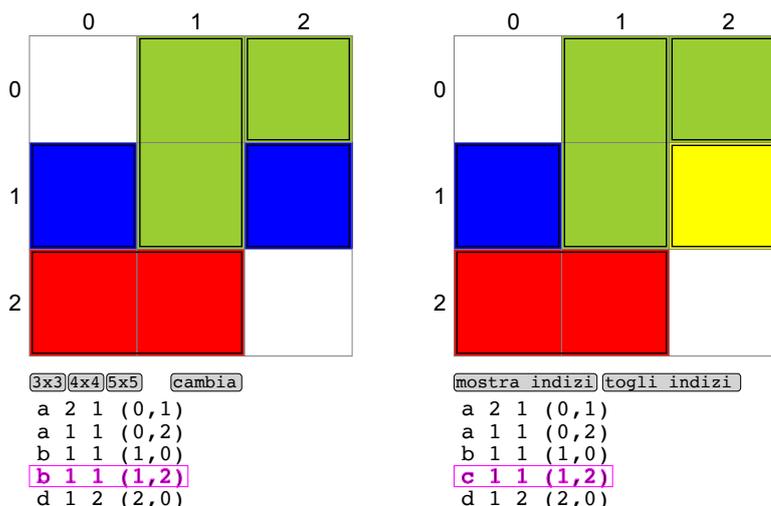
<sup>5</sup><https://aladdin.unimi.it/sw/square/square-primaria.html>

di modificare parti specifiche dell'immagine; in questo modo gli alunni sono costretti ad esaminare il sistema di codifica nel suo insieme e a organizzarsi per effettuare esperimenti significativi.

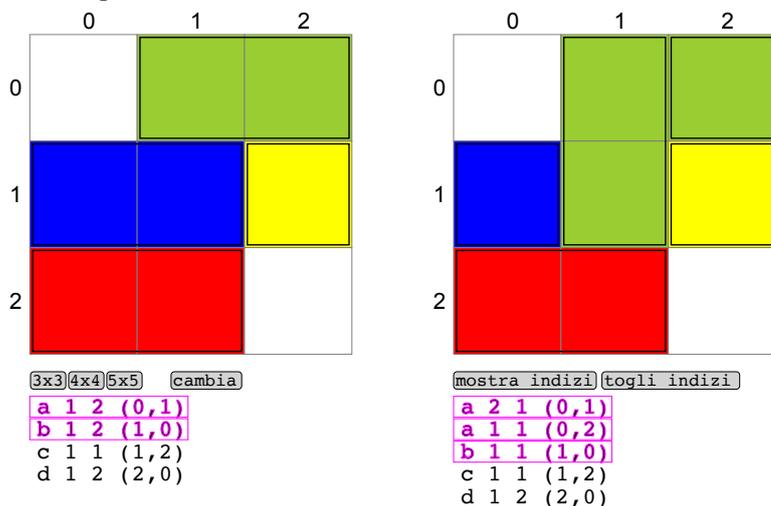
**Prima fase** Si presenta agli alunni lo strumento *software* e in particolare si fa notare che all'immagine raffigurata corrisponde una descrizione tramite simboli. È consigliabile fare lavorare gli alunni a coppie, in modo da favorire la verbalizzazione e il confronto tra le idee che emergono durante il lavoro. Gli alunni avranno 10–15 minuti per prendere confidenza con lo strumento, comprendere come funziona e qual è l'effetto dei vari bottoni. Si propone quindi una sfida: scoprire in che modo i simboli riportati sotto all'immagine la descrivono, ovvero come è possibile interpretare i simboli al fine di *ricostruire* l'intera immagine. Ovviamente non si deve fornire nessuna indicazione su come funziona il sistema di codifica e su come sono usati i simboli, poiché questo è proprio l'obiettivo del gioco.

**Seconda fase** Gli alunni sono invitati a rispondere alle domande della scheda. All'inizio gli alunni tenderanno ad usare lo strumento *software* in maniera casuale. Questa fase può essere un po' dispersiva ed è importante che il docente aiuti a ricordare qual è l'obiettivo del lavoro. Esplorando le funzionalità dello strumento, gli alunni si accorgeranno che fare delle prove non basta, ma che è necessario osservare attentamente gli effetti di queste prove, trarne delle deduzioni, formulare delle ipotesi sul ruolo dei diversi simboli nelle varie codifiche e fare nuovi esperimenti che confermino o meno le ipotesi fatte. Di fatto per poter rispondere alle domande della scheda si troveranno ad applicare il metodo scientifico, avvalendosi dello strumento *software* per fare esperimenti e generare nuove situazioni da osservare.

Ad esempio si renderanno conto che, quando si chiede di modificare l'immagine, è fondamentale scoprire cosa cambia tra il prima e il dopo. Questo sia nell'immagine, sia nella relativa codifica, cercando di capire che relazione c'è tra il cambiamento osservato nell'immagine e quello osservato nella codifica. Poiché un singolo cambiamento nell'immagine può produrre cambiamenti di entità differente nella codifica dell'immagine, è utile cominciare a ragionare su situazioni semplici, ad esempio quella che si ottiene quando il cambiamento del colore di un quadretto non modifica la decomposizione dell'immagine: in questo caso, infatti, nella codifica cambierà soltanto una lettera, come mostrato nell'esempio qui sotto:



Viceversa se, per effetto del cambiamento, si modifica la decomposizione in porzioni rettangolari, potranno comparire o sparire delle righe nella codifica, come nel caso qui sotto raffigurato:



Per questo motivo, può non essere opportuno fermarsi ad osservare gli effetti del primo cambiamento generato dallo strumento, ma è utile premere ripetutamente sul bottone 'cambia' finché si trova una situazione semplice da analizzare e su cui soffermarsi.

Tipicamente, dopo qualche prova e guidati dalle domande della scheda, gli alunni cominceranno ad intuire il significato di qualcuno dei simboli usati nella codifica, e faranno delle prove per confermare le loro intuizioni. Le domande della scheda aiutano gli alunni a concentrarsi singolarmente su ciascun dettaglio della codifica.

**Terza fase** L'obiettivo di questa fase è verificare che abbiamo compreso tutti gli aspetti del sistema di codifica. Agli alunni viene fornita la codifica di un'immagine e si chiede loro di ricostruirla disegnandola. Come si è già detto, la definizione o l'applicazione di regole non sono obiettivi dell'attività; tuttavia, ipotizzare delle regole di codifica e

provare ad applicarle è certamente un modo utile per convincersi di averle individuate correttamente, oppure per capire come modificarle. Infine viene chiesto agli alunni di spiegare in che modo funziona la codifica, mettendo per iscritto le regole con cui vanno interpretati i simboli, oppure la procedura da seguire per ricostruire l'immagine partendo dalla codifica, e viceversa.

È opportuno concludere questa fase (e così l'intera attività) con una parte dialogata a classe intera. Questa sarà anche l'occasione per sottolineare alcuni aspetti importanti, con cui gli alunni hanno dovuto certamente fare i conti nel loro processo di codifica, ma di cui forse non si sono resi conto esplicitamente.

- C'è una corrispondenza univoca tra l'immagine e la codifica corrispondente, ed è questo che consente di poter ricostruire sempre e senza incertezze l'immagine. Questa è una proprietà fondamentale di ogni codifica degna di questo nome: a dati diversi devono corrispondere codifiche diverse, e viceversa.
- I colori sono rappresentati da lettere minuscole, e l'associazione tra colori e lettere non ha nessuna "spiegazione" particolare: le lettere infatti sono usate in base ad una regola "astratta" e non perché abbiano un significato di per sé.
- Alcuni simboli sono usati in posizioni diverse all'interno della codifica e assumono significati diversi a seconda della posizione. Ad esempio i numeri possono indicare sia le dimensioni che la posizione delle porzioni rettangolari. Anche questa è una caratteristica tipica delle codifiche formali: il significato dei simboli è dato proprio dalle regole della codifica.

## 5 Robot umani

Il tema di questa attività è la programmazione, cioè l'**automatizzazione di un compito tramite la scrittura di un programma**, in modo che il compito possa essere eseguito da un *interprete* "meccanico". La parte finale con i computer è simile a quelle spesso proposte associandole al termine '*coding*'<sup>6</sup>, ma le fasi iniziali impostano il problema in modo radicalmente differente: non si tratta semplicemente di programmare una soluzione (spesso predefinita), ma di riflettere sull'intero processo di automatizzazione (per una discussione più ampia sul tema, si veda [8]).

In particolare i concetti informatici su cui si focalizza l'attività sono quelli di: *istruzione primitiva*, nel senso di istruzione che l'interprete comprende e a cui corrisponde un'azione di base che l'interprete sa eseguire; *sequenza*, cioè insieme di azioni da eseguire in un dato ordine per produrre il risultato voluto; *struttura di controllo* come istruzione che permette di "controllare" il flusso di esecuzione delle azioni.

Nella PINI, sono previsti alcuni traguardi e obiettivi di apprendimento che si possono promuovere con questa attività:

---

<sup>6</sup>In inglese il termine è talvolta usato come sinonimo di '*programming*'. In italiano però la parola '*coding*' può essere facilmente fraintesa: tradurla con 'codifica', infatti, crea un'ambiguità con la traduzione di '*encoding*', la codifica necessaria in ogni rappresentazione simbolica, vedi §§ 3-4

- comprendere che un algoritmo descrive una procedura che si presta ad essere automatizzata in modo preciso e non ambiguo (T-P-1);
- comprendere come un algoritmo può essere espresso mediante un programma scritto usando un linguaggio di programmazione (T-P-2);
- spiegare usando il ragionamento logico perché un programma strutturalmente semplice raggiunge i suoi obiettivi (T-P-4);
- riconoscere gli elementi algoritmici in operazioni abituali della vita quotidiana (O-P3-A-1);
- rilevare eventuali malfunzionamenti in programmi semplici e intervenire per correggerli (O-P3-P-1);
- ordinare correttamente la sequenza di istruzioni (O-P3-P-2);
- utilizzare i cicli per esprimere sinteticamente la ripetizione di una stessa azione un numero prefissato di volte (O-P3-P-3);
- utilizzare la selezione ad una via per prendere decisioni all'interno di programmi semplici (O-P3-P-4).

L'attività dura almeno **due ore** e si rivolge ad alunni del secondo ciclo della scuola primaria (**8-11 anni**).

**Prima fase** Per iniziare occorre preparare un semplice percorso a L (o a S se si vuole rendere l'attività più impegnativa) per ogni gruppetto di 4–6 studenti. Per definire il percorso è sufficiente accostare due banchi rettangolari e usare una sedia per segnare l'arrivo; la sedia dovrebbe essere posta dirimpetto il percorso, in modo che arrivando, per sedersi, occorra girarsi nel verso opposto a quello d'arrivo. Serve inoltre predisporre un oggetto da raccogliere durante il tragitto. È opportuno che tutti i gruppi abbiano lo stesso tipo di percorso.

A ogni gruppo viene richiesto di identificare un componente che, opportunamente bendato, eseguirà pedissequamente le azioni definite dei compagni: sarà perciò denominato 'Robot umano'. Per evitare incidenti, un altro studente sarà incaricato di badare che il robot non inciampi o sbatta. Al gruppo viene poi spiegato bene l'obiettivo: il robot dovrà percorrere il tragitto previsto, raccogliendo l'oggetto a metà strada, arrivando a sedersi sulla sedia che segna l'arrivo. Il robot non dovrà mai agire di propria iniziativa, ma seguendo le azioni concordate dal gruppo.

I primi 5 minuti sono dedicati a prendere confidenza col percorso e fare delle prove. In questo momento non è ancora chiaro cosa significhi dare delle *istruzioni*, è quindi meglio usare il termine *azioni*. In questa fase gli studenti non dovrebbero cogliere nessuna particolare criticità.

**Seconda fase** Ai gruppi si distribuiscono un foglio protocollo, Post-it di quattro colori diversi e matite per scrivere. Si spiega quindi che le azioni che il robot dovrà compiere vanno scritte sui Post-it (in modo sintetico: per ciascuna azione sono permesse al massimo cinque parole); inoltre su Post-it di un determinato colore deve essere scritta sempre la stessa azione (non è quindi possibile descrivere più di quattro azioni differenti). Si possono usare tutti i Post-it a disposizione (si consiglia di fornirne almeno 4–5 per ogni colore): essi verranno attaccati al foglio protocollo e dovranno essere letti in sequenza.

Il conduttore dell'attività dovrà monitorare tutti i gruppi e sostenere in particolare quelli in difficoltà; quando la maggior parte dei gruppi ha individuato le 4 azioni da scrivere sui Post-it, si può passare alla fase successiva. Può essere necessario chiarire bene che la stessa azione può essere usata più volte, e in punti diversi della sequenza (non è necessario introdurre in questa fase il termine “programma” per la sequenza di azioni, lo si può fare nella fase al computer).

La discussione in gruppo dovrebbe favorire la formulazione di idee/domande/critiche: gli studenti hanno la necessità di spiegare ai compagni le proprie idee e perché queste possono funzionare. In questa fase gli studenti effettueranno una vera e propria sperimentazione: se si accorgono che le azioni scelte, o l'ordine con cui sono state messe sul foglio, non consentono al robot di percorrere correttamente il labirinto, inizieranno a domandarsi il perché, e a fare ipotesi su come possono migliorare la loro soluzione. Si noti in particolare che la disponibilità di solo quattro colori/azioni è un vincolo che costringe a sfruttare le azioni in modo creativo: per esempio, un gruppo potrebbe usare un'azione ‘gira a destra’ per... girare a sinistra (ripetendola più volte). Il Post-it dovrebbe inoltre dare concretezza a ciò che si intende con *istruzione*: non una generica descrizione di una volontà performativa, ma un comando “formalizzato”, non ambiguo, utilizzabile in contesti differenti con effetti identici.

Alcuni gruppi potrebbero utilizzare un approccio “interattivo”, leggendo la prossima azione solo quando il portavoce *vede* che il robot ha portato a termine correttamente l'azione precedente o quando è più opportuno eseguire la prossima (per esempio dando un comando di svolta proprio dove si trova l'angolo del tavolo). Per scoraggiare questa impostazione è opportuno che il portavoce non veda il robot durante la lettura delle azioni (ad esempio può voltare le spalle al percorso).

**Terza fase** Gli studenti dovrebbero a poco a poco accorgersi che una pura sequenza di azioni è una pianificazione molto rigida e quindi difficile da stendere *a priori*; inoltre le sequenze diventano presto piuttosto lunghe. In questa fase si aggiunge quindi la possibilità di sfruttare “superpoteri” che in un certo senso amplificano l'espressività delle istruzioni. Viene esposto e illustrato un cartellone con alcune *strutture di controllo* che semplificano la scrittura. Il loro uso però è vincolato da una *sintassi*, cioè una struttura o impalcatura in cui si “incastrano” i Post-it, anche più di uno se necessario:

1. SE ⟨condizione⟩ ESEGUI ⟨Post-it...⟩
2. RIPETI ⟨*n*⟩ VOLTE ⟨Post-it...⟩
3. RIPETI FINO A QUANDO ⟨condizione⟩: ⟨Post-it...⟩

Si noti che i “superpoteri” sono effettivamente un ampliamento delle potenzialità del robot: in particolare per la possibilità di descrivere *condizioni* che corrispondono alla verifica di una qualche proprietà dell’ambiente di esecuzione (e che nel caso di un robot non umano necessiterebbero la presenza di sensori opportuni e la gestione di elementi di “memoria”).

In questa fase il conduttore deve assicurarsi che tutti i gruppi scrivano i loro programmi (ormai possiamo iniziare a chiamarli così) e li provino. Quando la maggior parte delle coppie ha provato il proprio programma almeno una volta e verificato che più o meno funziona, si può passare alla fase di discussione. Se un gruppo ha già definito una soluzione che “funziona” ma altri gruppi stanno ancora lavorando, si può cercare di stimolare il gruppo ad analizzare/migliorare il loro programma (es: far fare il robot a un altro componente del gruppo, far girare il portavoce di schiena, spostare l’oggetto da raccogliere, ecc).

**Quarta fase** I programmi vengono discussi a classe intera. Si fa eseguire il programma a ciascun gruppo, mentre gli altri gruppi osservano. Alla fine di ciascuna esecuzione, si chiede agli altri se hanno osservazioni da fare. Se c’è tempo si possono fare altre ripetizioni scambiando robot e/o portavoce. Alla fine si riassumono le caratteristiche delle varie soluzioni, in particolare evidenziando alcuni elementi che influiscono sulla generalità dei programmi, ossia la loro capacità di funzionare correttamente in ambienti diversi.

L’obiettivo di questa fase è far ragionare la classe sugli approcci proposti, partendo da un confronto tra le soluzioni proposte. È un passaggio delicato, cui dedicare un po’ di tempo, perché consente ai ragazzi di riflettere su quanto hanno fatto fino adesso. È molto importante il riferimento a esempi concreti che si riscontrano nelle proposte dei vari gruppi, incentivando le osservazioni da parte dei componenti degli altri gruppi. È probabile che in alcuni casi gli studenti contestino la correttezza delle soluzioni degli altri gruppi (il robot imbroglia, il portavoce modifica la lettura in base a quello che succede al robot, ecc.): questa è una buona occasione per sottolineare il fatto che è necessario definire delle azioni precise, non ambigue. Occorre però evitare che l’aspetto competitivo non prenda il sopravvento, allontanando l’attenzione dai concetti in discussione: in definitiva ascoltare le soluzioni dei compagni deve servire a mettere la propria in prospettiva, ampliando i propri orizzonti cognitivi.

**Quinta fase** Alla fine della discussione, per introdurre la fase successiva al computer, spiegare che ogni sistema informatico funziona proprio grazie a programmi formati da istruzioni. In particolare ora si userà un ambiente di programmazione (Scratch<sup>7</sup>) che consente di dare istruzioni a un “robot”, con l’obiettivo di uscire da un labirinto. L’attività al computer si svolge a coppie e dura come minimo una ventina di minuti, ma può essere prolungata per sviluppare maggiormente l’abilità nella programmazione.

---

<sup>7</sup><https://scratch.mit.edu/> Scratch è uno strumento molto versatile. Dal sito ALaDDIn (<https://aladdin.di.unimi.it/materiali.html>) è possibile scaricarlo una versione semplificata che ne riduce le potenzialità, evitando possibili confusioni iniziali. Sono inoltre disponibili tutti i labirinti descritti nel testo.

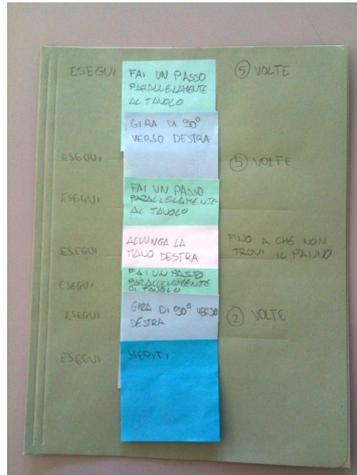


Figura 6: Un programma per un il robot umano

Quando le coppie si sono sistemate, occorre introdurre a grandi linee l'ambiente di programmazione Scratch (molto intuitivo), usando il labirinto di partenza (un percorso a L, vedi Figura), senza suggerire possibili soluzioni.

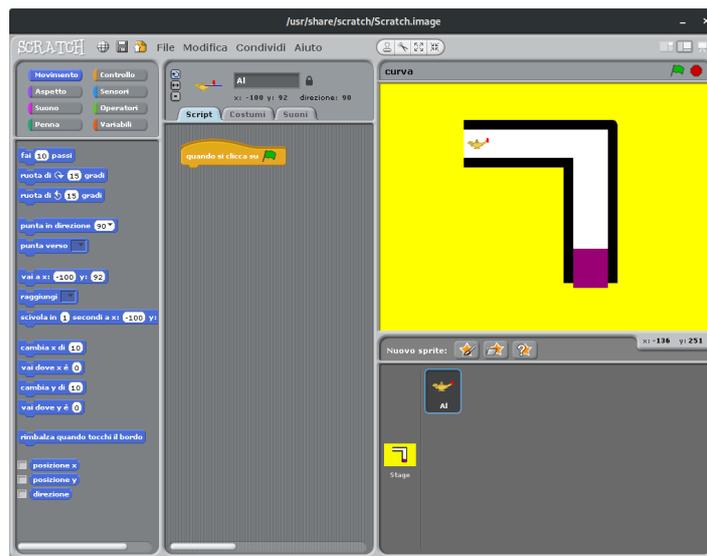
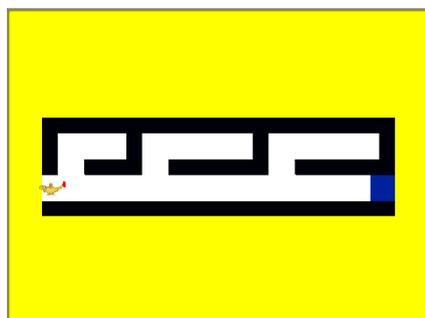
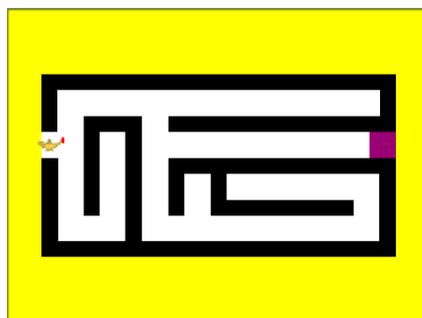


Figura 7: L'ambiente di programmazione Scratch con il labirinto di partenza

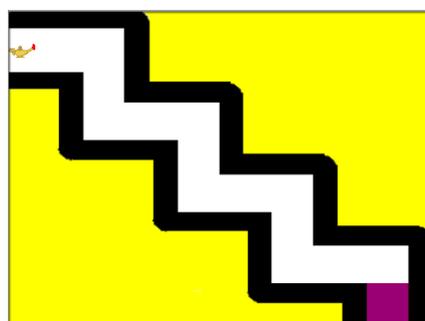
Quando tutti i gruppi hanno capito come lavorare, si chiede alle coppie di impraticarsi con i labirinti elementari (Figure 8(a) e 8(b)); acquisita dimestichezza è opportuno lanciare una piccola gara, in cui si celebra sulla lavagna la coppia che riesce a risolvere il labirinto con il programma "più corto", valutandone la lunghezza considerando solo i blocchetti delle istruzioni di movimento (blu) e non le strutture di controllo (o le condi-



(a) Labirinto elementare



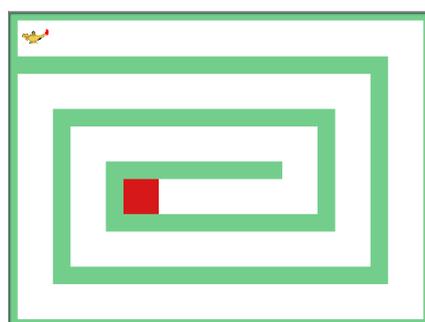
(b) Labirinto elementare



(c) Labirinto a scalini



(d) Labirinto a denti di sega



(e) Labirinto a spirale

Figura 8: Labirinti in ordine crescente di difficoltà

zioni). La gara serve a favorire l'uso delle strutture di controllo: per usarle efficacemente gli studenti devono identificare sottoparti ripetitive del labirinto o altre opportunità di generalizzazione. A questo scopo risultano particolarmente suggestivi i labirinti rappresentati nelle Figure 8(c), 8(d) e 8(e). A volte però (ad esempio quando si vedono solo programmi molto lunghi) può servire un piccolo suggerimento per fare osservare la struttura ripetitiva dei labirinti e proporre di sfruttarla per ridurre il numero delle istruzioni attraverso l'uso dei costrutti di controllo *se/ripeti*.

In alcuni casi gli studenti cercano di attraversare il percorso a pezzi, in modalità sostanzialmente interattiva: dispongono alcuni blocchi e li eseguono, poi li modificano e li eseguono, e così via fino all'uscita. In questo caso, come già discusso nelle fasi precedenti, bisogna chiarire che l'obiettivo invece è scrivere un programma intero che, una volta avviato, porti la lampada dalla partenza all'uscita senza ulteriori interventi del programmatore. Inoltre è opportuno che il conduttore si adoperi per evitare che i partecipanti perdano tempo modificando gli *sprite* o usando altre funzionalità di Scratch non pertinenti con gli obiettivi del laboratorio.

## 6 Riflessioni meta-cognitive finali

A tutte le attività dovrebbe seguire (anche in sessioni successive) una riflessione sul senso complessivo di ciò che si è fatto, che in definitiva è ragionare sul concetto di **informatica** come disciplina dell'elaborazione automatica dell'informazione. In particolare tutte le attività proposte qui ruotano intorno al problema di come si possa rappresentare l'informazione, di cosa voglia dire elaborarla in automatica tramite interpreti meccanici, anche sperimentando direttamente piccoli compiti di elaborazione.

Anche questa fase può essere condotta in modo interlocutorio, più adatta per far riflettere gli studenti su quello che essi hanno sperimentato; una lezione frontale rischia di fornire una “narrativa” dell'esperienza che corrisponde a fatica ai modelli mentali che i ragazzi si sono costruiti. La discussione può partire dalle tre componenti fondamentali: **elaborazione**, **esecutore automatico**, **informazione**. Si può partire da domande tipo: secondo voi quale (tipo di) informazione abbiamo preso in considerazione? Che cosa c'è di automatico in quello che abbiamo fatto? Si può dire che abbiamo fatto “informatica” oggi? Quando? Anche nella attività senza i computer? Anche se ognuna delle attività è focalizzata su un aspetto specifico (per esempio per l'attività di §2 è prevalente l'aspetto di elaborazione, nelle attività dei §§3 e 4 quello di informazione, nell'attività di §5 quello di esecutore automatico), sono sempre presenti tutti, in quanto intimamente legati: si rappresenta l'informazione proprio per poterla elaborare, tipicamente con un dispositivo in grado di farlo in maniera automatica. Per esempio nell'attività sui robot umani (§5) gli studenti dovrebbero essersi confrontati a fondo con il concetto di *esecutore automatico*: il compagno “robot” deve eseguire le istruzioni senza “metterci del suo”, addirittura sforzandosi di non percepire elementi dell'ambiente che non siano stati esplicitamente menzionati dai “programmatori”, cioè i progettisti delle attività del robot. E invece non c'è niente di automatico nel progettare e scrivere un programma che eseguito da “esecutori automatici” consenta di ottenere l'obiettivo desiderato, c'è anzi bisogno

di creatività e inventiva, seppure disciplinate dalla logica. Non meno importante è la *rappresentazione dell'informazione* da elaborare: sono in realtà in gioco almeno due tipi molto diversi di informazioni: da un lato c'è il programma, che deve essere descritto in termini “formali” (cioè rispettando regole di forma, come l'uso dei Post-it o l'opportuno incastro dei blocchetti di Scratch) con l'obiettivo di eliminare le ambiguità; dall'altro ci sono i dati sull'ambiente in cui si trova il robot che vengono rappresentati (e tipicamente semplificati/stilizzati) nel programma: per esempio, il fatto che nella stanza ci potesse essere un profumo, non entrava a far parte delle cose percepite dal robot (anche se la persona robot, lo avrebbe certamente percepito); alcune parti della realtà essenziali per l'elaborazione, invece, appaiono al robot in forma del tutto astratta: i muri che una persona percepisce col tatto, diventano, nell'attività proposta al computer, un confronto fra numeri che rappresentano i colori che appaiono sullo schermo. L'informazione viene quindi *elaborata* continuamente: il programma viene “eseguito” e le istruzioni agiscono e tengono conto delle informazioni rappresentate in ogni dato istante.

Se si riesce, anche in questa fase è utile cercare di valorizzare quanto emerge dalla classe. Ad esempio, se qualcuno si dichiara poco convinto di aver fatto informatica, è conveniente prendere l'occasione per cercare di spiegare meglio come si faccia informatica ogni volta che si analizzano problemi e studiano soluzioni che consentono poi una elaborazione automatica dell'informazione: anche sfruttare il bordo del tavolo per decidere quando girare o delimitare un parola con due lenticchie può essere una buona idea “informatica”. Buone idee che speriamo possano far parte degli strumenti di pensiero da portare con sé a prescindere dalle inclinazioni o opportunità professionali che verranno.

## Riferimenti bibliografici

- [1] C. Bellettini, V. Lonati, D. Malchiodi, M. Monga, and A. Morpurgo. Informatica e pensiero computazionale: una proposta costruttivista per gli insegnanti. In *Didamatica 2018*, pages 201–210. AICA, Apr. 2018.
- [2] C. Bellettini, V. Lonati, D. Malchiodi, M. Monga, A. Morpurgo, and M. Torelli. What you see is what you have in mind: constructing mental models for formatted text processing. In *Proceedings of ISSEP 2013*, pages 139–147, 2013.
- [3] C. Bellettini, V. Lonati, D. Malchiodi, M. Monga, A. Morpurgo, M. Torelli, and L. Zecca. Extracurricular activities for improving the perception of informatics in secondary schools. In *Proceedings of ISSEP 2014*, pages 161–172, 2014.
- [4] C. Bellettini, M. Monga, V. Lonati, A. Morpurgo, D. Malchiodi, and M. Torelli. Exploring the processing of formatted texts by a kynesthetic approach. In *Proceedings of WiPSCE 2012*, pages 143–144. ACM, 2012.
- [5] A. Calcagni, V. Lonati, D. Malchiodi, M. Monga, and A. Morpurgo. Promoting computational thinking skills: Would you use this Bebras task? In *Proceedings of ISSEP 2017*, volume 10696 of *Lecture Notes in Computer Science*, pages 102–113. Springer, 2017.

- [6] A. Carletti and A. Varani, editors. *Didattica costruttivista. Dalle teorie alla pratica in classe*. Erickson, 2005.
- [7] E. V. Glasersfeld. Idee costruttiviste. *Riflessioni Sistemiche*, (2):179–181, 2010.
- [8] V. Lonati, D. Malchiodi, M. Monga, and A. Morpurgo. Is coding the way to go? In A. Brodnik and J. Vahrenhold, editors, *Proceedings of ISSEP 2015*, pages 165–174, 2015.